Abstract

This is the proof document of the IsarMathLib project version 1.29.0. IsarMathLib is a library of formalized mathematics for Isabelle2023 (ZF logic).

Contents

1 Introduction to the IsarMathLib project 11
   1.1 How to read IsarMathLib proofs - a tutorial . . . . . . . 11
   1.2 Overview of the project . . . . . . . . . . . . . . . . . . . . 13

2 First Order Logic 17
   2.1 Notions and lemmas in FOL . . . . . . . . . . . . . . . . . . 17

3 ZF set theory basics 20
   3.1 Lemmas in Zermelo-Fraenkel set theory . . . . . . . . . . . 20

4 Natural numbers in IsarMathLib 29
   4.1 Induction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
   4.2 Intervals . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37

5 Order relations - introduction 39
   5.1 Definitions . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39
   5.2 Intervals . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
   5.3 Bounded sets . . . . . . . . . . . . . . . . . . . . . . . . . . . 45

6 More on order relations 51
   6.1 Definitions and basic properties . . . . . . . . . . . . . . . . 52
   6.2 Properties of (strict) total orders . . . . . . . . . . . . . . . 54

7 Even more on order relations 56
   7.1 Maximum and minimum of a set . . . . . . . . . . . . . . . . 57
   7.2 Supremum and Infimum . . . . . . . . . . . . . . . . . . . . 64
   7.3 Strict versions of order relations . . . . . . . . . . . . . . . . 72
8 Functions - introduction
8.1 Properties of functions, function spaces and (inverse) images. 75
8.2 Functions restricted to a set .......................... 92
8.3 Constant functions ................................. 94
8.4 Injections, surjections, bijections etc. ............... 96
8.5 Functions of two variables .......................... 105

9 Semilattices and Lattices 109
9.1 Semilattices ........................................... 109

10 Order on natural numbers 116
10.1 Order on natural numbers .......................... 116

11 Binary operations 117
11.1 Lifting operations to a function space .............. 117
11.2 Associative and commutative operations .............. 119
11.3 Restricting operations ............................ 122
11.4 Compositions ....................................... 124
11.5 Identity function .................................... 126
11.6 Lifting to subsets .................................... 127
11.7 Distributive operations ............................. 132

12 More on functions 133
12.1 Functions and order ............................... 133
12.2 Functions in cartesian products ..................... 138
12.3 Induced relations and order isomorphisms .......... 140

13 Finite sets - introduction 147
13.1 Definition and basic properties of finite powerset . 147

14 Finite sets 156
14.1 Finite powerset ..................................... 156
14.2 Finite range functions .............................. 164

15 Finite sets 1 165
15.1 Finite vs. bounded sets ............................. 165

16 Finite sets and order relations 168
16.1 Finite vs. bounded sets ............................. 169
16.2 Order isomorphisms of finite sets .................. 171

17 Equivalence relations 177
17.1 Congruent functions and projections on the quotient 178
17.2 Projecting commutative, associative and distributive operations .......................... 184
17.3 Saturated sets .................................................. 186

18 Finite sequences .................................................. 189
  18.1 Lists as finite sequences ..................................... 189
  18.2 Lists and cartesian products .................................. 208

19 Formal languages .................................................. 211
  19.1 Introduction .................................................... 211
  19.2 Deterministic Finite Automata ................................. 213
  19.3 Operations on regular languages ............................... 230
  19.4 Non-deterministic finite state automata ...................... 240
  19.5 Equivalence of Non-deterministic and Deterministic Finite
       State Automata .................................................. 242

20 Inductive sequences ................................................ 245
  20.1 Sequences defined by induction ................................ 245
  20.2 Images of inductive sequences ................................ 254
  20.3 Subsets generated by a binary operation ...................... 255
  20.4 Inductive sequences with changing generating function ...... 257
  20.5 The Pascal’s triangle .......................................... 262

21 Enumerations ....................................................... 268
  21.1 Enumerations: definition and notation ....................... 268
  21.2 Properties of enumerations ................................... 269

22 Folding in ZF ........................................................ 271
  22.1 Folding in ZF .................................................. 272

23 Partitions of sets ................................................... 278
  23.1 Bisections ...................................................... 278
  23.2 Partitions ...................................................... 281

24 Quasigroups .......................................................... 282
  24.1 Definitions and notation ...................................... 283

25 Loops ................................................................. 285
  25.1 Definitions and notation ...................................... 286

26 Ordered loops ........................................................ 288
  26.1 Definition and notation ...................................... 288

27 Semigroups ........................................................... 295
  27.1 Products of sequences of semigroup elements .................. 295
  27.2 Products over sets of indices ................................ 299
  27.3 Commutative semigroups ...................................... 302
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>Commutative Semigroups</td>
<td>314</td>
</tr>
<tr>
<td>28.1</td>
<td>Sum of a function over a set</td>
<td>314</td>
</tr>
<tr>
<td>29</td>
<td>Monoids</td>
<td>318</td>
</tr>
<tr>
<td>29.1</td>
<td>Definition and basic properties</td>
<td>318</td>
</tr>
<tr>
<td>30</td>
<td>Summing lists in a monoid</td>
<td>323</td>
</tr>
<tr>
<td>30.1</td>
<td>Notation and basic properties of sums of lists of monoid elements</td>
<td>323</td>
</tr>
<tr>
<td>30.2</td>
<td>Multiplying monoid elements by natural numbers</td>
<td>327</td>
</tr>
<tr>
<td>31</td>
<td>Groups - introduction</td>
<td>328</td>
</tr>
<tr>
<td>31.1</td>
<td>Definition and basic properties of groups</td>
<td>329</td>
</tr>
<tr>
<td>31.2</td>
<td>Subgroups</td>
<td>341</td>
</tr>
<tr>
<td>31.3</td>
<td>Groups vs. loops</td>
<td>349</td>
</tr>
<tr>
<td>32</td>
<td>Groups 1</td>
<td>351</td>
</tr>
<tr>
<td>32.1</td>
<td>Translations</td>
<td>351</td>
</tr>
<tr>
<td>32.2</td>
<td>Odd functions</td>
<td>359</td>
</tr>
<tr>
<td>32.3</td>
<td>Subgroups and interval arithmetic</td>
<td>360</td>
</tr>
<tr>
<td>33</td>
<td>Groups - and alternative definition</td>
<td>365</td>
</tr>
<tr>
<td>33.1</td>
<td>An alternative definition of group</td>
<td>365</td>
</tr>
<tr>
<td>34</td>
<td>Abelian Group</td>
<td>367</td>
</tr>
<tr>
<td>34.1</td>
<td>Rearrangements formulae</td>
<td>367</td>
</tr>
<tr>
<td>35</td>
<td>Groups 2</td>
<td>379</td>
</tr>
<tr>
<td>35.1</td>
<td>Lifting groups to function spaces</td>
<td>380</td>
</tr>
<tr>
<td>35.2</td>
<td>Equivalence relations on groups</td>
<td>385</td>
</tr>
<tr>
<td>35.3</td>
<td>Normal subgroups and quotient groups</td>
<td>388</td>
</tr>
<tr>
<td>35.4</td>
<td>Function spaces as monoids</td>
<td>393</td>
</tr>
<tr>
<td>36</td>
<td>Groups 3</td>
<td>394</td>
</tr>
<tr>
<td>36.1</td>
<td>Group valued finite range functions</td>
<td>394</td>
</tr>
<tr>
<td>36.2</td>
<td>Almost homomorphisms</td>
<td>396</td>
</tr>
<tr>
<td>36.3</td>
<td>The classes of almost homomorphisms</td>
<td>404</td>
</tr>
<tr>
<td>36.4</td>
<td>Compositions of almost homomorphisms</td>
<td>407</td>
</tr>
<tr>
<td>36.5</td>
<td>Shifting almost homomorphisms</td>
<td>416</td>
</tr>
<tr>
<td>37</td>
<td>Direct product</td>
<td>417</td>
</tr>
<tr>
<td>37.1</td>
<td>Definition</td>
<td>417</td>
</tr>
<tr>
<td>37.2</td>
<td>Associative and commutative operations</td>
<td>418</td>
</tr>
</tbody>
</table>
47 Rings - Ideals
47.1 Ideals .............................................. 550
47.2 Ring quotient ...................................... 565

48 Rings - Ideals of quotient rings
48.1 Ring homomorphisms ................................. 571
48.2 Quotient ring with quotient map .................. 578
48.3 Quotient ideals ..................................... 582

49 Fields - introduction
49.1 Definition and basic properties .................... 591
49.2 Equations and identities ............................ 594
49.3 1/0=0 .................................................. 596

50 Modules
50.1 Definition and basic properties of modules ........ 597
50.2 Module axioms ....................................... 601

51 Vector spaces
51.1 Definition and basic properties of vector spaces ............................. 602
51.2 Vector space axioms .................................. 604

52 Ordered fields
52.1 Definition and basic properties ..................... 605
52.2 Inequalities .......................................... 609
52.3 Definition of real numbers ............................ 612

53 Integers - introduction
53.1 Addition and multiplication as ZF-functions. .... 613
53.2 Integers as an ordered group ......................... 619
53.3 Induction on integers ................................ 632
53.4 Bounded vs. finite subsets of integers .............. 635

54 Integers 1
54.1 Integers as a ring .................................... 638
54.2 Rearrangement lemmas ............................... 641
54.3 Integers as an ordered ring ........................... 647
54.4 Maximum and minimum of a set of integers .......... 658
54.5 The set of nonnegative integers .................... 662
54.6 Functions with infinite limits ....................... 668
54.7 Miscellaneous ...................................... 673

55 Division on integers
55.1 Quotient and reminder .............................. 675
68 Topology 2
68.1 Continuous functions ....................... 844
68.2 Homeomorphisms ........................... 851
68.3 Topologies induced by mappings ............. 853
68.4 Partial functions and continuity ............. 855
68.5 Product topology and continuity ............. 858
68.6 Pasting lemma ............................. 864

69 Rings - Zariski Topology - maps 867

70 Topology 3
70.1 The base of the product topology ............. 876
70.2 Finite product of topologies .................. 878

71 Topology 4
71.1 Nets ..................................... 887
71.2 Filters ................................... 890
71.3 Relation between nets and filters ............. 896

72 Topology and neighborhoods 907
72.1 Neighborhood systems ....................... 907
72.2 From a neighborhood system to topology .... 908
72.3 From a topology to a neighborhood system ... 910
72.4 Neighborhood systems are 1:1 with topologies 913
72.5 Set neighborhoods .......................... 915

73 Topology - examples 919
73.1 CoCardinal Topology ....................... 919
73.2 Total set, Closed sets, Interior, Closure and Boundary 921
73.3 Excluded Set Topology ...................... 927
73.4 Total set, closed sets, interior, closure and boundary 928
73.5 Special cases and subspaces ............... 932
73.6 Included Set Topology ..................... 933
73.7 Basic topological notions in included set topology 934
73.8 Special cases and subspaces ............... 937

74 More examples in topology 939
74.1 New ideas using a base for a topology ........ 939
74.2 The topology of a base ..................... 939
74.3 Dual Base for Closed Sets .................. 943
74.4 Partition topology ........................ 945
74.5 Partition topology is a topology ............. 947
74.6 Total set, Closed sets, Interior, Closure and Boundary 947
74.7 Special cases and subspaces ............... 953
74.8 Order topologies .......................... 955
1 Introduction to the IsarMathLib project

theory Introduction imports ZF.equalities

begin

This theory does not contain any formalized mathematics used in other theories, but is an introduction to IsarMathLib project.

1.1 How to read IsarMathLib proofs - a tutorial

Isar (the Isabelle’s formal proof language) was designed to be similar to the standard language of mathematics. Any person able to read proofs in a typical mathematical paper should be able to read and understand Isar proofs without having to learn a special proof language. However, Isar is a formal proof language and as such it does contain a couple of constructs whose meaning is hard to guess. In this tutorial we will define a notion
and prove an example theorem about that notion, explaining Isar syntax along the way. This tutorial may also serve as a style guide for IsarMathLib contributors. Note that this tutorial aims to help in reading the presentation of the Isar language that is used in IsarMathLib proof document and HTML rendering on the FormalMath.org site, but does not teach how to write proofs that can be verified by Isabelle. This presentation is different than the source processed by Isabelle (the concept that the source and presentation look different should be familiar to any LaTeX user). To learn how to write Isar proofs one needs to study the source of this tutorial as well.

The first thing that mathematicians typically do is to define notions. In Isar this is done with the \texttt{definition} keyword. In our case we define a notion of two sets being disjoint. We will use the infix notation, i.e. the string \texttt{is disjoint with} put between two sets to denote our notion of disjointness. The left side of the \equiv symbol is the notion being defined, the right side says how we define it. In Isabelle/ZF 0 is used to denote both zero (of natural numbers) and the empty set, which is not surprising as those two things are the same in set theory.

\texttt{definition}
\texttt{AreDisjoint \{infix \{is disjoint with\} \ 90\} where}
\texttt{A \{is disjoint with\} B \equiv A \cap B = 0}

We are ready to prove a theorem. Here we show that the relation of being disjoint is symmetric. We start with one of the keywords "theorem", "lemma" or "corollary". In Isar they are synonymous. Then we provide a name for the theorem. In standard mathematics theorems are numbered. In Isar we can do that too, but it is considered better to give theorems meaningful names. After the "shows" keyword we give the statement to show. The \leftrightarrow symbol denotes the equivalence in Isabelle/ZF. Here we want to show that "A is disjoint with B iff and only if B is disjoint with A". To prove this fact we show two implications - the first one that \texttt{A \{is disjoint with\} B} implies \texttt{B \{is disjoint with\} A} and then the converse one. Each of these implications is formulated as a statement to be proved and then proved in a subproof like a mini-theorem. Each subproof uses a proof block to show the implication. Proof blocks are delimited with curly brackets in Isar. Proof block is one of the constructs that does not exist in informal mathematics, so it may be confusing. When reading a proof containing a proof block I suggest to focus first on what is that we are proving in it. This can be done by looking at the first line or two of the block and then at the last statement. In our case the block starts with "assume \texttt{A \{is disjoint with\} B} and the last statement is "then have \texttt{B \{is disjoint with\} A}". It is a typical pattern when someone needs to prove an implication: one assumes the antecedent and then shows that the consequent follows from this assumption. Implications are denoted with the \rightarrow symbol in Isabelle. After we prove both implications we collect them using the "moreover" construct. The keyword
ultimately” indicates that what follows is the conclusion of the statements collected with "moreover". The "show" keyword is like "have", except that it indicates that we have arrived at the claim of the theorem (or a subproof).

**Theorem disjointness_symmetric:**

shows $A \{\text{is disjoint with}\} B \iff B \{\text{is disjoint with}\} A$

**Proof -**

have $A \{\text{is disjoint with}\} B \implies B \{\text{is disjoint with}\} A$

**Proof -**

{ assume $A \{\text{is disjoint with}\} B$

then have $A \cap B = 0$ using AreDisjoint_def by simp

hence $B \cap A = 0$ by auto

then have $B \{\text{is disjoint with}\} A$

using AreDisjoint_def by simp

} thus thesis by simp

qed

moreover have $B \{\text{is disjoint with}\} A \implies A \{\text{is disjoint with}\} B$

**Proof -**

{ assume $B \{\text{is disjoint with}\} A$

then have $B \cap A = 0$ using AreDisjoint_def by simp

hence $A \cap B = 0$ by auto

then have $A \{\text{is disjoint with}\} B$

using AreDisjoint_def by simp

} thus thesis by simp

qed

ultimately show thesis by blast

qed

1.2 Overview of the project

The Fol1, ZF1 and Nat_ZF.IML theory files contain some background material that is needed for the remaining theories. Order_ZF and Order_ZF_1a reformulate material from standard Isabelle’s Order theory in terms of non-strict (less-or-equal) order relations. Order_ZF_1 on the other hand directly continues the Order theory file using strict order relations (less and not equal). This is useful for translating theorems from Metamath.

In NatOrder_ZF we prove that the usual order on natural numbers is linear. The func1 theory provides basic facts about functions. func_ZF continues this development with more advanced topics that relate to algebraic properties of binary operations, like lifting a binary operation to a function space, associative, commutative and distributive operations and properties of functions related to order relations. func_ZF_1 is about properties of functions related to order relations.

The standard Isabelle’s Finite theory defines the finite powerset of a set as a certain "datatype" (?) with some recursive properties. IsarMathLib’s Finite1 and Finite_ZF_1 theories develop more facts about this notion.
These two theories are obsolete now. They will be gradually replaced by an approach based on set theory rather than tools specific to Isabelle. This approach is presented in Finite_ZF theory file.

In FinOrd_ZF we talk about ordered finite sets.

The EquivClass1 theory file is a reformulation of the material in the standard Isabelle’s EquivClass theory in the spirit of ZF set theory.

FiniteSeq_ZF discusses the notion of finite sequences (a.k.a. lists).

InductiveSeq_ZF provides the definition and properties of (what is known in basic calculus as) sequences defined by induction, i.e. by a formula of the form \( a_0 = x, \ a_{n+1} = f(a_n) \).

Fold_ZF shows how the familiar from functional programming notion of fold can be interpreted in set theory.

Partitions_ZF is about splitting a set into non-overlapping subsets. This is a common trick in proofs.

Semigroup_ZF treats the expressions of the form \( a_0 \cdot a_1 \cdot .. \cdot a_n \), (i.e. products of finite sequences), where \( \cdot \) is an associative binary operation.

CommutativeSemigroup_ZF is another take on a similar subject. This time we consider the case when the operation is commutative and the result of depends only on the set of elements we are summing (additively speaking), but not the order.

The Topology_ZF series covers basics of general topology: interior, closure, boundary, compact sets, separation axioms and continuous functions.

Group_ZF, Group_ZF_1, Group_ZF_1b and Group_ZF_2 provide basic facts of the group theory. Group_ZF_3 considers the notion of almost homomorphisms that is needed for the real numbers construction in Real_ZF.

The TopologicalGroup connects the Topology_ZF and Group_ZF series and starts the subject of topological groups with some basic definitions and facts.

In DirectProduct_ZF we define direct product of groups and show some its basic properties.

The OrderedGroup_ZF theory treats ordered groups. This is a surprisingly large theory for such relatively obscure topic.

Ring_ZF defines rings. Ring_ZF_1 covers the properties of rings that are specific to the real numbers construction in Real_ZF.

The OrderedRing_ZF theory looks at the consequences of adding a linear order to the ring algebraic structure.

Field_ZF and OrderedField_ZF contain basic facts about (you guessed it) fields and ordered fields.

Int_ZF_IML theory considers the integers as a monoid (multiplication) and an abelian ordered group (addition). In Int_ZF_1 we show that integers form a commutative ring. Int_ZF_2 contains some facts about slopes (almost homomorphisms on integers) needed for real numbers construction, used in
In the IntDiv_ZF_IML theory we translate some properties of the integer quotient and reminder functions studied in the standard Isabelle’s IntDiv_ZF theory to the notation used in IsarMathLib.

The Real_ZF and Real_ZF_1 theories contain the construction of real numbers based on the paper [2] by R. D. Arthan (not Cauchy sequences, not Dedekind sections). The heavy lifting is done mostly in Group_ZF_3, Ring_ZF_1 and Int_ZF_2. Real_ZF contains the part of the construction that can be done starting from generic abelian groups (rather than additive group of integers). This allows to show that real numbers form a ring. Real_ZF_1 continues the construction using properties specific to the integers and showing that real numbers constructed this way form a complete ordered field.

Cardinal_ZF provides a couple of theorems about cardinals that are mostly used for studying properties of topological properties (yes, this is kind of meta). The main result (proven without AC) is that if two sets can be injectively mapped into an infinite cardinal, then so can be their union. There is also a definition of the Axiom of Choice specific for a given cardinal (so that the choice function exists for families of sets of given cardinality). Some properties are proven for such predicates, like that for finite families of sets the choice function always exists (in ZF) and that the axiom of choice for a larger cardinal implies one for a smaller cardinal.

Group_ZF_4 considers conjugate of subgroup and defines simple groups. A nice theorem here is that endomorphisms of an abelian group form a ring. The first isomorphism theorem (a group homomorphism h induces an isomorphism between the group divided by the kernel of h and the image of h) is proven.

Turns out given a property of a topological space one can define a local version of a property in general. This is studied in the the Topology_ZF_properties_2 theory and applied to local versions of the property of being finite or compact or Hausdorff (i.e. locally finite, locally compact, locally Hausdorff). There are a couple of nice applications, like one-point compactification that allows to show that every locally compact Hausdorff space is regular. Also there are some results on the interplay between hereditarility of a property and local properties.

For a given surjection \( f : X \to Y \), where \( X \) is a topological space one can consider the weakest topology on \( Y \) which makes \( f \) continuous, let’s call it a quotient topology generated by \( f \). The quotient topology generated by an equivalence relation \( r \) on \( X \) is actually a special case of this setup, where \( f \) is the natural projection of \( X \) on the quotient \( X/r \). The properties of these two ways of getting new topologies are studied in Topology_ZF_8 theory. The main result is that any quotient topology generated by a function is homeomorphic to a topology given by an equivalence relation, so these two
approaches to quotient topologies are kind of equivalent.

As we all know, automorphisms of a topological space form a group. This fact is proven in Topology_ZF_9 and the automorphism groups for co-cardinal, included-set, and excluded-set topologies are identified. For order topologies it is shown that order isomorphisms are homeomorphisms of the topology induced by the order. Properties preserved by continuous functions are studied and as an application it is shown for example that quotient topological spaces of compact (or connected) spaces are compact (or connected, resp.)

The Topology_ZF_10 theory is about products of two topological spaces. It is proven that if two spaces are $T_0$ (or $T_1$, $T_2$, regular, connected) then their product is as well.

Given a total order on a set one can define a natural topology on it generated by taking the rays and intervals as the base. The Topology_ZF_11 theory studies relations between the order and various properties of generated topology. For example one can show that if the order topology is connected, then the order is complete (in the sense that for each set bounded from above the set of upper bounds has a minimum). For a given cardinal $\kappa$ we can consider generalized notion of $\kappa$-separability. Turns out $\kappa$-separability is related to (order) density of sets of cardinality $\kappa$ for order topologies.

Being a topological group imposes additional structure on the topology of the group, in particular its separation properties. In Topological_Group_ZF_1.thy theory it is shown that if a topology is $T_0$, then it must be $T_3$, and that the topology in a topological group is always regular.

For a given normal subgroup of a topological group we can define a topology on the quotient group in a natural way. At the end of the Topological_Group_ZF_2.thy theory it is shown that such topology on the quotient group makes it a topological group.

The Topological_Group_ZF_3.thy theory studies the topologies on subgroups of a topological group. A couple of nice basic properties are shown, like that the closure of a subgroup is a subgroup, closure of a normal subgroup is normal and, a bit more surprising (to me) property that every locally-compact subgroup of a $T_0$ group is closed.

In Complex_ZF we construct complex numbers starting from a complete ordered field (a model of real numbers). We also define the notation for writing about complex numbers and prove that the structure of complex numbers constructed there satisfies the axioms of complex numbers used in Metamath.

MMI_prelude defines the mmisar0 context in which most theorems translated from Metamath are proven. It also contains a chapter explaining how the translation works.

In the Metamath_interface theory we prove a theorem that the mmisar0 context is valid (can be used) in the complex0 context. All theories us-
ing the translated results will import the Metamath_interface theory. The Metamath_sampler theory provides some examples of using the translated theorems in the complex0 context.

The theories MMI_logic_and_sets, MMI_Complex, MMI_Complex_1 and MMI_Complex_2 contain the theorems imported from the Metamath’s set.mm database. As the translated proofs are rather verbose these theories are not printed in this proof document. The full list of translated facts can be found in the Metamath_theorems.txt file included in the IsarMathLib distribution. The MMI_examples provides some theorems imported from Metamath that are printed in this proof document as examples of how translated proofs look like.

end

2 First Order Logic

theory Fol1 imports ZF.Trancl

begin

Isabelle/ZF builds on the first order logic. Almost everything one would like to have in this area is covered in the standard Isabelle libraries. The material in this theory provides some lemmas that are missing or allow for a more readable proof style.

2.1 Notions and lemmas in FOL

This section contains mostly shortcuts and workarounds that allow to use more readable coding style.

The next lemma serves as a workaround to problems with applying the definition of transitivity (of a relation) in our coding style (any attempt to do something like using trans_def puts Isabelle in an infinite loop).

lemma Fol1_L2: assumes
A1: ∀ x y z. ⟨x, y⟩ ∈ r ∧ ⟨y, z⟩ ∈ r −→ ⟨x, z⟩ ∈ r
shows trans(r)

proof -
  from A1 have
  ∀ x y z. ⟨x, y⟩ ∈ r −→ ⟨y, z⟩ ∈ r −→ ⟨x, z⟩ ∈ r
  using imp_conj by blast
  then show thesis unfolding trans_def by blast
qed

Another workaround for the problem of Isabelle simplifier looping when the transitivity definition is used.

lemma Fol1_L3: assumes A1: trans(r) and A2: ⟨a, b⟩ ∈ r ∧ ⟨b, c⟩ ∈ r
shows \( (a, c) \in r \)

proof -
  from A1 have \( \forall x \ y \ z. \langle x, y \rangle \in r \rightarrow \langle y, z \rangle \in r \rightarrow \langle x, z \rangle \in r \)
  unfolding trans_def by blast
  with A2 show thesis using imp_conj by fast
qed

There is a problem with application of the definition of asymmetry for relations. The next lemma is a workaround.

lemma Fol1_L4:
assumes A1: antisym(r) and A2: \( (a, b) \in r \langle b, a \rangle \in r \)
shows a=b
proof -
  from A1 have \( \forall x \ y. \langle x, y \rangle \in r \rightarrow \langle y, x \rangle \in r \rightarrow x=y \)
  unfolding antisym_def by blast
  with A2 show a=b using imp_conj by fast
qed

The definition below implements a common idiom that states that (perhaps under some assumptions) exactly one of given three statements is true.

definition
Exactly_1_of_3_holds(p,q,r) \equiv
(p \lor q \lor r) \land (p \rightarrow \neg q \land \neg r) \land (q \rightarrow \neg p \land \neg r) \land (r \rightarrow \neg p \land \neg q)

The next lemma allows to prove statements of the form Exactly_1_of_3_holds(p,q,r).

lemma Fol1_L5:
assumes p \lor q \lor r
and p \rightarrow \neg q \land \neg r
and q \rightarrow \neg p \land \neg r
and r \rightarrow \neg p \land \neg q
shows Exactly_1_of_3_holds(p,q,r)
proof -
  from assms have
    \( (p \lor q \lor r) \land (p \rightarrow \neg q \land \neg r) \land (q \rightarrow \neg p \land \neg r) \land (r \rightarrow \neg p \land \neg q) \)
    by blast
  then show Exactly_1_of_3_holds (p,q,r)
    unfolding Exactly_1_of_3_holds_def by fast
qed

If exactly one of \( p, q, r \) holds and \( p \) is not true, then \( q \) or \( r \).

lemma Fol1_L6:
assumes A1: \( \neg p \) and A2: Exactly_1_of_3_holds(p,q,r)
shows q\lor r
proof -
  from A2 have
    \( (p \lor q \lor r) \land (p \rightarrow \neg q \land \neg r) \land (q \rightarrow \neg p \land \neg r) \land (r \rightarrow \neg p \land \neg q) \)
    unfolding Exactly_1_of_3_holds_def by fast
  hence p \lor q \lor r by blast


with A1 show \( q \lor r \) by simp

qed

If exactly one of \( p, q, r \) holds and \( q \) is true, then \( r \) can not be true.

**Lemma Fol1_L7:**

assumes A1: \( q \) and A2: Exactly_1_of_3_holds(p, q, r)

shows \( \neg r \)

proof -

from A2 have

\[
(p \lor q \lor r) \land (p \rightarrow \neg q \land \neg r) \land (q \rightarrow \neg p \land \neg r) \land (r \rightarrow \neg p \land \neg q)
\]

unfolding Exactly_1_of_3_holds_def by fast

with A1 show \( \neg r \) by blast

qed

The next lemma demonstrates an elegant form of the Exactly_1_of_3_holds\((p, q, r)\) predicate.

**Lemma Fol1_L8:**

shows Exactly_1_of_3_holds\((p, q, r)\) \(\iff\) \( (p \iff q \iff r) \land \neg (p \land q \land r) \)

proof

assume Exactly_1_of_3_holds\((p, q, r)\)

then have

\[
(p \lor q \lor r) \land (p \rightarrow \neg q \land \neg r) \land (q \rightarrow \neg p \land \neg r) \land (r \rightarrow \neg p \land \neg q)
\]

unfolding Exactly_1_of_3_holds_def by fast

thus \( (p \iff q \iff r) \land \neg (p \land q \land r) \) by blast

next assume \( (p \iff q \iff r) \land \neg (p \land q \land r) \)

hence

\[
(p \lor q \lor r) \land (p \rightarrow \neg q \land \neg r) \land (q \rightarrow \neg p \land \neg r) \land (r \rightarrow \neg p \land \neg q)
\]

by auto

then show Exactly_1_of_3_holds\((p, q, r)\)

unfolding Exactly_1_of_3_holds_def by fast

qed

A property of the Exactly_1_of_3_holds predicate.

**Lemma Fol1_L8A:** assumes A1: Exactly_1_of_3_holds\((p, q, r)\)

shows \( p \iff \neg (q \lor r) \)

proof -

from A1 have \( (p \lor q \lor r) \land (p \rightarrow \neg q \land \neg r) \land (q \rightarrow \neg p \land \neg r) \land (r \rightarrow \neg p \land \neg q) \)

unfolding Exactly_1_of_3_holds_def by fast

then show \( p \iff \neg (q \lor r) \) by blast

qed

Exclusive or definition. There is one also defined in the standard Isabelle, denoted \texttt{xor}, but it relates to boolean values, which are sets. Here we define a logical functor.

**Definition Xor** (infixl Xor 66) where

\[ p \text{ Xor } q \equiv (p \lor q) \land \neg (p \land q) \]
The "exclusive or" is the same as negation of equivalence.

```
lemma Fol1_L9: shows p Xor q ⟷ ¬(p ⟷ q)
  using Xor_def by auto
```

Constructions from the same sets are the same. It is surprising but we do have to use this as a rule in rare cases.

```
lemma same_constr: assumes x=y shows P(x) = P(y)
  using assms by simp
```

Equivalence relations are symmetric.

```
lemma equiv_is_sym: assumes A1: equiv(X,r) and A2: ⟨x,y⟩ ∈ r
  shows ⟨y,x⟩ ∈ r
proof -
  from A1 have sym(r) using equiv_def by simp
  then have ∀x y. ⟨x,y⟩ ∈ r ⟷ ⟨y,x⟩ ∈ r
    unfolding sym_def by fast
  with A2 show ⟨y,x⟩ ∈ r by blast
qed
```

3 ZF set theory basics

```
theory ZF1 imports ZF.Perm
begin

The standard Isabelle distribution contains lots of facts about basic set theory. This theory file adds some more.

3.1 Lemmas in Zermelo-Fraenkel set theory

Here we put lemmas from the set theory that we could not find in the standard Isabelle distribution or just so that they are easier to find.

A set cannot be a member of itself. This is exactly lemma mem_not_refl from Isabelle/ZF upair.thy, we put it here for easy reference.

```
lemma mem_self: shows x ∉ x by (rule mem_not_refl)
```

If one collection is contained in another, then we can say the same about their unions.

```
lemma collection_contain: assumes A⊆B shows ∪A ⊆ ∪B
proof
  fix x assume x ∈ ∪A
  then obtain X where x∈X and X∈A by auto
```
with assms show \( x \in \bigcup B \) by auto

qed

If all sets of a nonempty collection are the same, then its union is the same.

lemma ZF1_1_L1: assumes \( C \neq 0 \) and \( \forall y \in C. \ b(y) = A \)  
shows \( (\bigcup y \in C. \ b(y)) = A \) using assms by blast

The union of all values of a constant meta-function belongs to the same set as the constant.

lemma ZF1_1_L2: assumes \( A1: C \neq 0 \) and \( A2: \forall x \in C. \ b(x) \in A \) and \( A3: \forall x y. \ x \in C \land y \in C \rightarrow b(x) = b(y) \)  
shows \( (\bigcup x \in C. \ b(x)) \in A \) proof -  
  from A1 obtain \( x \) where D1: \( x \in C \) by auto  
with A3 have \( \forall y \in C. \ b(y) = b(x) \) by blast  
with A1 have \( (\bigcup y \in C. \ b(y)) \in A \) using ZF1_1_L1 by simp  
with D1 A2 show thesis by simp

qed

If two meta-functions are the same on a cartesian product, then the subsets defined by them are the same. This is similar to ZF1_1_L4, except that the set definition varies over \( p \in X \times Y \) rather than \( \langle x,y \rangle \in X \times Y \).

lemma ZF1_1_L4A: assumes \( A1: \forall x \in X. \forall y \in Y. \ a(\langle x,y \rangle) = b(x,y) \)  
shows \( \{a(x,y). \langle x,y \rangle \in X \times Y\} = \{b(x,y). \langle x,y \rangle \in X \times Y\} \) proof  
  fix z assume z \in \( \{a(x,y). \langle x,y \rangle \in X \times Y\} \)  
  with A1 show z \in \( \{b(x,y). \langle x,y \rangle \in X \times Y\} \) by auto

qed

proof

If two meta-functions are the same on a cartesian product, then the subsets defined by them are the same.

lemma ZF1_1_L4: assumes \( A1: \forall x \in X. \forall y \in Y. \ a(x,y) = b(x,y) \)  
shows \( \{a(x,y). \langle x,y \rangle \in X \times Y\} = \{b(x,y). \langle x,y \rangle \in X \times Y\} \) proof  
  fix z assume z \in \( \{a(x,y). \langle x,y \rangle \in X \times Y\} \)  
  then obtain \( p \) where D1: \( z=a(p) \) \( p \in X \times Y \) by auto  
  let x = fst(p) let y = snd(p)  
  from A1 D1 have z \in \( \{b(x,y). \langle x,y \rangle \in X \times Y\} \) by auto

qed
\{ \text{fix } z \text{ assume } z \in \{ b(x,y). \langle x,y \rangle \in X \times Y \} \text{ then obtain } x \text{ y where } D1: \langle x,y \rangle \in X \times Y \text{ z = a(p) by auto} \text{ let } p = \langle x,y \rangle \text{ from A1 D1 have } p \in X \times Y \text{ z = a(p) by auto} \} \text{ then show } \{ b(x,y). \langle x,y \rangle \in X \times Y \} \subseteq \{ a(p). p \in X \times Y \} \text{ by blast} \}

A lemma about inclusion in cartesian products. Included here to remember that we need the $U \times V \neq \emptyset$ assumption.

\text{lemma prod_subset: assumes } U \times V \neq \emptyset \text{ U \times V } \subseteq X \times Y \text{ shows } U \subseteq X \text{ and } V \subseteq Y \text{ using assms by auto}

A technical lemma about sections in cartesian products.

\text{lemma section_proj: assumes } A \subseteq X \times Y \text{ and } U \times V \subseteq A \text{ and } x \in U \text{ y } \in V \text{ shows } U \subseteq \{ t \in X. \langle t,y \rangle \in A \} \text{ and } V \subseteq \{ t \in Y. \langle x,t \rangle \in A \} \text{ using assms by auto}

If two meta-functions are the same on a set, then they define the same set by separation.

\text{lemma ZF1_1_L4B: assumes } \forall x \in X. a(x) = b(x) \text{ shows } \{ a(x). x \in X \} = \{ b(x). x \in X \} \text{ using assms by simp}

A set defined by a constant meta-function is a singleton.

\text{lemma ZF1_1_L5: assumes } X \neq 0 \text{ and } \forall x \in X. b(x) = c \text{ shows } \{ b(x). x \in X \} = \{ c \} \text{ using assms by blast}

Most of the time, auto does this job, but there are strange cases when the next lemma is needed.

\text{lemma subset_with_property: assumes } Y = \{ x \in X. b(x) \} \text{ shows } Y \subseteq X \text{ using assms by auto}

We can choose an element from a nonempty set.

\text{lemma nonempty_has_element: assumes } X \neq 0 \text{ shows } \exists x. x \in X \text{ using assms by auto}

In Isabelle/ZF the intersection of an empty family is empty. This is exactly lemma Inter_0 from Isabelle's equalities theory. We repeat this lemma here as it is very difficult to find. This is one reason we need comments before every theorem: so that we can search for keywords.

\text{lemma inter_empty_empty: shows } \bigcap \emptyset = \emptyset \text{ by (rule Inter_0)}
If an intersection of a collection is not empty, then the collection is not
empty. We are (ab)using the fact the the intersection of empty collection is
defined to be empty.

lemma inter_nempty_nempty: assumes \( \bigcap A \neq 0 \) shows \( A \neq 0 \)
using assms by auto

For two collections \( S, T \) of sets we define the product collection as the
collections of cartesian products \( A \times B \), where \( A \in S, B \in T \).

definition ProductCollection(T,S) \equiv \bigcup U \in T. \{U \times V. V \in S\}

The union of the product collection of collections \( S,T \) is the cartesian prod-
uct of \( \bigcup S \) and \( \bigcup T \).

lemma ZF1_1_L6: shows \( \bigcup \) ProductCollection(S,T) = \( \bigcup S \times \bigcup T \)
using ProductCollection_def by auto

An intersection of subsets is a subset.

lemma ZF1_1_L7: assumes A1: \( I \neq 0 \) and A2: \( \forall i \in I. P(i) \subseteq X \)
shows \( ( \bigcap i \in I. P(i) ) \subseteq X \)
proof -
from A1 obtain i_0 where i_0 \in I by auto
with A2 have \( ( \bigcap i \in I. P(i) ) \subseteq P(i_0) \) and \( P(i_0) \subseteq X \)
by auto
thus \( ( \bigcap i \in I. P(i) ) \subseteq X \) by auto
qed

Isabelle/ZF has a ”THE” construct that allows to define an element if there
is only one such that is satisfies given predicate. In pure ZF we can express
something similar using the indentity proven below.

lemma ZF1_1_L8: shows \( \bigcup \}x\} = x \) by auto

Some properties of singletons.

lemma ZF1_1_L9: assumes A1: \( \exists! x. x \in A \land \varphi(x) \)
shows
\( \exists a. \{x \in A. \varphi(x)\} = \{a\} \)
\( \bigcup \}x \in A. \varphi(x)\} \in A \)
\( \varphi(\bigcup \}x \in A. \varphi(x)\}) \)
proof -
from A1 show \( \exists a. \{x \in A. \varphi(x)\} = \{a\} \) by auto
then obtain a where I: \( \{x \in A. \varphi(x)\} = \{a\} \) by auto
then have \( \bigcup \}x \in A. \varphi(x)\} = a \) by auto
moreover
from I have a \( \in \}x \in A. \varphi(x)\} \) by simp
hence a \( \in A \) and \( \varphi(a) \) by auto
ultimately show \( \bigcup \}x \in A. \varphi(x)\) \( \in A \) and \( \varphi(\bigcup \}x \in A. \varphi(x)\}) \)
by auto
qed
A simple version of ZF1_1_L9.

**corollary singleton_extract:** assumes \( \exists ! \; x. \; x \in A \)
shows \( (\bigcup \; A) \in A \)

**proof** -
  from assms have \( \exists ! \; x. \; x \in A \land \text{True} \) by simp
  then have \( \bigcup \{x \in A. \; \text{True}\} \in A \) by (rule ZF1_1_L9)
  thus \( (\bigcup \; A) \in A \) by simp
qed

A criterion for when a set defined by comprehension is a singleton.

**lemma singleton_comprehension:**
assumes \( A1: \; y \in X \) and \( A2: \; \forall \; x \in X. \; \forall \; y \in X. \; P(x) = P(y) \)
shows \( (\bigcup \{P(x). \; x \in X\}) = P(y) \)

**proof** -
let \( A = \{P(x). \; x \in X\} \)
have \( \exists ! \; c. \; c \in A \)
  proof
  from \( A1 \) show \( \exists c. \; c \in A \) by auto
  next
  fix \( a \; b \) assume \( a \in A \) and \( b \in A \)
  then obtain \( x \; t \) where
      \( x \in X \; a = P(x) \) and \( t \in X \; b = P(t) \)
      by auto
  with \( A2 \) show \( a=b \) by blast
  qed
  then have \( (\bigcup \; A) \in A \) by (rule singleton_extract)
  then obtain \( x \) where \( x \in X \) and \( (\bigcup \; A) = P(x) \)
      by auto
  from \( A1 \; A2 \) \( x \in X \) have \( P(x) = P(y) \)
      by blast
  with \( \langle(\bigcup \; A) = P(x)\rangle \) show \( (\bigcup \; A) = P(y) \) by simp
  qed

Adding an element of a set to that set does not change the set.

**lemma set_elem_add:** assumes \( x \in X \)
shows \( X \cup \{x\} = X \)
using assms
  by auto

Here we define a restriction of a collection of sets to a given set. In romantic math this is typically denoted \( X \cap M \) and means \( \{X \cap A : A \in M\} \). Note there is also restrict\((f,A)\) defined for relations in ZF.thy.

**definition**
\( \text{RestrictedTo} \ (\text{infixl} \ \{\text{restricted to}\} \ 70) \ where \)
\( M \{\text{restricted to}\} X \equiv \{X \cap A : A \in M\} \)

A lemma on a union of a restriction of a collection to a set.

**lemma union_restrict:**
shows \( \bigcup (M \{\text{restricted to}\} X) = (\bigcup M) \cap X \)
using RestrictedTo_def by auto
Next we show a technical identity that is used to prove sufficiency of some condition for a collection of sets to be a base for a topology.

```
lemma ZF1_1_L10: assumes A1: \( \forall U \in C. \exists A \in B. U = \bigcup A \)
shows \( \bigcup \{ \bigcup \{ A \in B. U = \bigcup A \}. U \in C \} = \bigcup C \)
proof
  show \( \bigcup ( \bigcup \{ U \in C. \bigcup \{ A \in B. U = \bigcup A \} \}. U \in C \) \subseteq \bigcup C \) by blast
  show \( \bigcup C \subseteq \bigcup ( \bigcup \{ U \in C. \bigcup \{ A \in B. U = \bigcup A \} \}. U \in C \) \)
  proof -
    fix x assume x \in \bigcup C
    show x \in \bigcup ( \bigcup \{ U \in C. \bigcup \{ A \in B. U = \bigcup A \} \}. U \in C \)
    proof -
      from \( \langle x \in \bigcup C \rangle \) obtain U where U \in C \land x \in U by auto
      with A1 obtain A where A \in B \land U = \bigcup A by auto
      from \( \langle U \in C \land x \in U \rangle \) : \( \langle A \in B \land U = \bigcup A \rangle \) show x \in \bigcup ( \bigcup \{ U \in C. \bigcup \{ A \in B. U = \bigcup A \} \}. U \in C \)
    by auto
    qed
    qed
qed
```

Standard Isabelle uses a notion of \( \text{cons}(A,a) \) that can be thought of as \( A \cup \{a\} \).

```
lemma consdef: shows \( \text{cons}(a,A) = A \cup \{a\} \)
using cons_def by auto
```

If a difference between a set and a singleton is empty, then the set is empty or it is equal to the singleton.

```
lemma singl_diff_empty: assumes A - \{x\} = 0
  shows A = 0 \lor A = \{x\}
using assms by auto
```

If a difference between a set and a singleton is the set, then the only element of the singleton is not in the set.

```
lemma singl_diff_eq: assumes A1: A - \{x\} = A
  shows x \notin A
proof -
  have x \notin A - \{x\} by auto
  with A1 show x \notin A by simp
qed
```

Simple substitution in membership, has to be used by rule in very rare cases.

```
lemma eq_mem: assumes x \in A and y=x shows y \in A
using assms by simp
```

A basic property of sets defined by comprehension.

```
lemma comprehension: assumes a \in \{x \in X. p(x)\}
  shows a \in X and p(a) using assms by auto
```
A basic property of a set defined by another type of comprehension.

**Lemma comprehension_repl**: assumes $y \in \{p(x). x \in X\}$
shows $\exists x \in X. y = p(x)$ using assms by auto

The inverse of the comprehension lemma.

**Lemma mem_cond_in_set**: assumes $\varphi(c)$ and $c \in X$
shows $c \in \{x \in X. \varphi(x)\}$ using assms by blast

The image of a set by a greater relation is greater.

**Lemma image_rel_mono**: assumes $r \subseteq s$
shows $r(A) \subseteq s(A)$ using assms by auto

A technical lemma about relations: if $x$ is in its image by a relation $U$ and that image is contained in some set $C$, then the image of the singleton $\{x\}$ by the relation $U \cup C \times C$ equals $C$.

**Lemma image_greater_rel**: assumes $x \in U\{x\}$ and $U\{x\} \subseteq C$
shows $(U \cup C \times C)\{x\} = C$
using assms image_Un_left by blast

Reformulation of the definition of composition of two relations:

**Lemma rel_compdef**: shows $\langle x,z \rangle \in r \circ s \iff \exists y. \langle x,y \rangle \in s \land \langle y,z \rangle \in r$
unfolding comp_def by auto

Domain and range of the relation of the form $\bigcup\{U \times U : U \in P\}$ is $\bigcup P$:

**Lemma domain_range_sym**: shows domain($\bigcup\{U \times U. U \in P\}$) = $\bigcup P$
and range($\bigcup\{U \times U. U \in P\}$) = $\bigcup P$
by auto

An identity for the square (in the sense of composition) of a symmetric relation.

**Lemma symm_sq_prod_image**: assumes converse($r$) = $r$
shows $r \circ r = \bigcup\{(r\{x\}) \times (r\{x\}). x \in \text{domain}(r)\}$
proof
{ fix $p$ assume $p \in r \circ r$
then obtain $y z$ where $\langle y,z \rangle = p$ by auto
with $\langle p \in r \circ r \rangle$ obtain $x$ where $\langle y,x \rangle \in r$ and $\langle x,z \rangle \in r$
using rel_compdef by auto
from $\langle y,x \rangle \in r$ have $\langle x,y \rangle \in \text{converse}(r)$ by simp
with assms $\langle x,z \rangle \in r$ $\langle y,z \rangle = p$ have $\exists x \in \text{domain}(r). p \in (r\{x\}) \times (r\{x\})$
by auto
} thus $r \circ r \subseteq \bigcup\{(r\{x\}) \times (r\{x\}). x \in \text{domain}(r)\}$
by blast
{ fix $x$ assume $x \in \text{domain}(r)$
have $(r\{x\}) \times (r\{x\}) \subseteq r \circ r$
proof -

26
A reflexive relation is contained in the union of products of its singleton images.

**Lemma 1:** Assume $A \subseteq X \times X$ and $id(X) \subseteq A$ shows $A \subseteq \bigcup \{A \times (A \setminus \{x\}). x \in X\}$

**Proof:**

1. Fix $p$ assume $p \in A \times A$ with $assms(1)$ obtain $x$ $y$ where $x \in X$ $y \in X$ and $p = (x, y)$ by auto
2. with $assms(2)$ $\exists p \in A$. $p \in A \times A$ by auto

Thus thesis by auto.

qed

If the cartesian product of the images of $x$ and $y$ by a symmetric relation $W$ has a nonempty intersection with $R$ then $x$ is in relation $W \circ (R \circ W)$ with $y$.

**Lemma 2:** Assume $W = \text{converse}(W)$ and $(W \times (W \setminus \{x\}) \cap R \neq 0$ shows $(x, y) \in (W \circ (R \circ W))$

**Proof:**

1. From $assms(2)$ obtain $s$ $t$ where $s \in W \setminus \{x\}$ $t \in W \setminus \{y\}$ and $(s, t) \in R$ by blast
2. Then have $(x, s) \in W$ and $(y, t) \in W$ by auto
3. From $(x, s) \in W$ $(s, t) \in R$ have $(x, t) \in R \circ W$ by auto
4. From $(y, t) \in W$ have $(t, y) \in \text{converse}(W)$ by blast
5. With $assms(1)$ $(x, t) \in R \circ W$ show thesis by auto

qed

It’s hard to believe but there are cases where we have to reference this rule.

**Lemma 3:** Assume $x \in A$ $A = B$ shows $x \in B$ using $assms$ by simp

Given some family $\mathcal{A}$ of subsets of $X$ we can define the family of supersets of $\mathcal{A}$.

**Definition:**

$\text{Supersets}(X, \mathcal{A}) \equiv \{B \in \text{Pow}(X). \exists A \in \mathcal{A}. A \subseteq B\}$

The family itself is in its supersets.

**Lemma 4:** Assume $A \subseteq X$ $A \in \mathcal{A}$ shows $A \in \text{Supersets}(X, \mathcal{A})$
using assms unfolding Supersets_def by auto

This can be done by the auto method, but sometimes takes a long time.

lemma witness_exists: assumes \( x \in X \) and \( \varphi(x) \) shows \( \exists x \in X. \varphi(x) \)
using assms by auto

Another lemma that concludes existence of some set.

lemma witness_exists1: assumes \( x \in X \) \( \varphi(x) \) \( \psi(x) \) shows \( \exists x \in X. \varphi(x) \land \psi(x) \)
using assms by auto

The next lemma has to be used as a rule in some rare cases.

lemma exists_in_set: assumes \( \forall x. x \in A \rightarrow \varphi(x) \) shows \( \forall x \in A. \varphi(x) \)
using assms by simp

If \( x \) belongs to a set where a property holds, then the property holds for \( x \).
This has to be used as rule in rare cases.

lemma property_holds: assumes \( \forall t \in X. \varphi(t) \) and \( x \in X \)
serves \( \varphi(x) \) using assms by simp

Set comprehensions defined by equal expressions are the equal. The second
assertion is actually about functions, which are sets of pairs as illustrated in
lemma fun_is_set_of_pairs in func1.thy

lemma set_comp_eq: assumes \( \forall x \in X. p(x) = q(x) \)
serves \( \{p(x). x \in X\} = \{q(x). x \in X\} \) and \( \{(x,p(x)). x \in X\} = \{(x,q(x)). x \in X\} \)
using assms by auto

If every element of a non-empty set \( X \subseteq Y \) satisfies a condition then the set
of elements of \( Y \) that satisfy the condition is non-empty.

lemma non_empty_cond: assumes \( X \neq 0 \) \( X \subseteq Y \) and \( \forall x \in X. \varphi(x) \)
serves \( \{x \in Y. \varphi(x)\} \neq 0 \) using assms by auto

If \( z \) is a pair, then the cartesian product of the singletons of its elements is
the same as the singleton \( \{z\} \).

lemma pair_prod: assumes \( z = \langle x,y \rangle \) shows \( \{x\} \times \{y\} = \{z\} \)
using assms by blast

In Isabelle/ZF the set difference is written with a minus sign \( A - B \) because
the standard backslash character is reserved for other purposes. The next
abbreviation declares that we want the set difference character \( A \setminus B \) to be
synonymous with the minus sign.

abbreviation set_difference (infixl \ 65) where \( A \setminus B \equiv A-B \)

In ZF set theory the zero of natural numbers is the same as the empty set.
In the next abbreviation we declare that we want 0 and \( \emptyset \) to be synonyms
so that we can use \( \emptyset \) instead of 0 when appropriate.

28
4 Natural numbers in IsarMathLib

theory Nat_ZF_IML imports ZF.ArithSimp

begin

The ZF set theory constructs natural numbers from the empty set and the
notion of a one-element set. Namely, zero of natural numbers is defined
as the empty set. For each natural number \( n \) the next natural number is
defined as \( n \cup \{n\} \). With this definition for every non-zero natural number
we get the identity \( n = \{0, 1, 2, ..., n - 1\} \). It is good to remember that when
we see an expression like \( f : n \rightarrow X \). Also, with this definition the relation
"less or equal than" becomes "\( \subseteq \)" and the relation "less than" becomes "\( \in \)".

4.1 Induction

The induction lemmas in the standard Isabelle’s Nat.thy file like for example
nat_induct require the induction step to be a higher order statement (the
one that uses the \( \Rightarrow \) sign). I found it difficult to apply from Isar, which
is perhaps more of an indication of my Isar skills than anything else. Any-
way, here we provide a first order version that is easier to reference in Isar
declarative style proofs.

The next theorem is a version of induction on natural numbers that I was
thought in school.

theorem ind_on_nat:
  assumes A1: \( n \in \text{nat} \) and A2: \( P(0) \) and A3: \( \forall k \in \text{nat}. \; P(k) \Rightarrow P(\text{succ}(k)) \)
  shows \( P(n) \)
proof -
  note A1 A2
  moreover { fix \( x \)
    assume \( x \in \text{nat} \) \( P(x) \)
    with A3 have \( P(\text{succ}(x)) \) by simp }
  ultimately show \( P(n) \) by (rule nat_induct)
qed

A nonzero natural number has a predecessor.

lemma Nat_ZF_1_L3: assumes A1: \( n \in \text{nat} \) and A2: \( n \neq 0 \)
  shows \( \exists k \in \text{nat}. \; n = \text{succ}(k) \)
proof -
  from A1 have \( n \in \{0\} \cup \{\text{succ}(k). \; k \in \text{nat}\} \)
What is succ, anyway? It’s a union with the singleton of the set.

**Lemma succ_explained:** shows \( \text{succ}(n) = n \cup \{n\} \)

*Proof*

Using **succ_iff** by auto

The singleton containing the empty set is a natural number.

**Lemma one_is_nat:** shows \( \{0\} \in \text{nat} \)

*Proof*

\( \{0\} = \text{succ}(0) \) using **succ_explained** by simp

\( \{0\} \in \text{nat} \) by simp

\( \{0\} = 1 \) by blast

QED

If \( k \) is a member of \( \text{succ}(n) \) but is not \( n \), then it must be the member of \( n \).

**Lemma mem_succ_not_eq:** assumes \( k \in \text{succ}(n) \) \( k \neq n \)

shows \( k \in n \) using assms **succ_explained** by simp

Empty set is an element of every natural number which is not zero.

**Lemma empty_in_every_succ:** assumes A1: \( n \in \text{nat} \)

shows \( 0 \in \text{succ}(n) \)

*Proof*

- Note A1

  Moreover have \( 0 \in \text{succ}(0) \) by simp

  Moreover

  \{ fix \( k \) assume \( k \in \text{nat} \) and A2: \( 0 \in \text{succ}(k) \)

  then have \( \text{succ}(k) \subseteq \text{succ}(\text{succ}(k)) \) by auto

  with A2 have \( 0 \in \text{succ}(\text{succ}(k)) \) by auto

  \} then have \( \forall k \in \text{nat}. \ 0 \in \text{succ}(k) \implies 0 \in \text{succ}(\text{succ}(k)) \)

  by simp

  Ultimately show \( 0 \in \text{succ}(n) \) by (rule **ind_on_nat**)

QED

Various forms of saying that for natural numbers taking the successor is the same as adding one.

**Lemma succ_add_one:** assumes \( n \in \text{nat} \)

shows \( n + 1 = \text{succ}(n) \)

\( n + 1 \in \text{nat} \)

\( \{0\} + n = \text{succ}(n) \)

\( n + \{0\} = \text{succ}(n) \)

\( \text{succ}(n) \in \text{nat} \)

\( 0 \in n + 1 \)

\( n \subseteq n + 1 \)

*Proof*

- From assms show \( n + 1 = \text{succ}(n) \) \( n + 1 \in \text{nat} \) \( \text{succ}(n) \in \text{nat} \) by simp_all
moreover from assms have \{0\} = 1 and \( n \#+ 1 = 1 \#+ n \) by auto
ultimately show \{0\} \#+ n = succ(n) and \( n \#+ \{0\} = succ(n) \)
by simp_all
from assms \( \langle n \#+ 1 = succ(n) \rangle \) show \( 0 \in n \#+ 1 \) using empty_in_every_succ
by simp
from assms \( \langle n \#+ 1 = succ(n) \rangle \) show \( n \subseteq n \#+ 1 \) using succ_explained
by auto
qed

A more direct way of stating that empty set is an element of every non-zero natural number:

lemma empty_in_non_empty: assumes \( n \in\text{nat} \) \( n \neq 0 \)
shows \( 0 \in n \)
using assms Nat_ZF_1_L3 empty_in_every_succ by auto

If one natural number is less than another then their successors are in the same relation.

lemma succ_ineq: assumes \( A1: n \in\text{nat} \)
shows \( \forall i \in n. succ(i) \in succ(n) \)
proof -

note \( A1 \)
moreover have \( \forall k \in 0. succ(k) \in succ(0) \) by simp
moreover
\{ fix \( k \) assume \( A2: \forall i \in k. succ(i) \in succ(k) \)
\{ fix \( i \) assume \( i \in succ(k) \)
then have \( i \in k \lor i = k \) by auto
moreover
\{ assume \( i \in k \)
with \( A2 \) have \( succ(i) \in succ(k) \) by simp
hence \( succ(i) \in succ(succ(k)) \) by simp
moreover
\{ assume \( i = k \)
then have \( succ(i) \in succ(succ(k)) \) by auto \}
ultimately have \( \forall i \in succ(k). succ(i) \in succ(succ(k)) \)
by simp
\} then have \( \forall k \in\text{nat}. ( \forall i \in k. succ(i) \in succ(k)) \rightarrow (\forall i \in succ(k). succ(i) \in succ(succ(k))) \)
by simp
ultimately show \( \forall i \in n. succ(i) \in succ(n) \) by \( \text{rule ind_on_nat} \)
qed

For natural numbers if \( k \subseteq n \) the similar holds for their successors.

lemma succ_subset: assumes \( A1: k \in\text{nat} \) \( n \in\text{nat} \) and \( A2: k \subseteq\text{succ}(n) \)
shows \( \text{succ}(k) \subseteq\text{succ}(n) \)
proof -

from \( A1 \) have \( T: \text{Ord}(k) \) and \( \text{Ord}(n) \)
For any two natural numbers one of them is contained in the other.

**lemma nat_incl_total:** assumes A1: i ∈ nat  j ∈ nat  
shows i ⊆ j ∨ j ⊆ i  
proof -  
  from A1  
  then have T: Ord(i)  Ord(j) by auto  
  then have i≤j ∨ i=j ∨ j≤i using Ord_linear  
  moreover  
    { assume i≤j  
      then have i≤j ∨ j≤i by simp }  
  moreover  
    { assume i=j  
      then have i≤j ∨ j≤i by simp }  
  moreover  
    { assume j≤i  
      then have i≤j ∨ j≤i by simp }  
  ultimately show i ⊆ j ∨ j ⊆ i by auto  
qed

The set of natural numbers is the union of all successors of natural numbers.

**lemma nat_union_succ:** shows nat = (∪n ∈ nat. succ(n))  
proof -  
  from A1  
  then have succ(n) ⊆ (∪n ∈ nat. succ(n)) by auto  
  then show succ(n) ⊆ nat using nat_union_succ by simp  
qed

Successors of natural numbers are subsets of the set of natural numbers.

**lemma succnat_subset_nat:** assumes A1: n ∈ nat  
shows succ(n) ⊆ nat  
proof -  
  from A1  
  then show succ(n) ⊆ nat using nat_union_succ by simp  
qed
Element $k$ of a natural number $n$ is a natural number that is smaller than $n$.

**Lemma elem_nat_is_nat**: assumes $A1: n \in \text{nat}$ and $A2: k \in n$
shows $k < n \quad k \in \text{nat} \quad k \leq n \quad \langle k, n \rangle \in \text{Le}$

**Proof** -
from $A1$ A2 show $k < n$ using nat_into_Ord lt_def by simp
with $A1$ show $k \in \text{nat}$ using lt_nat_in_nat by simp
from $< k < n >$ show $k \leq n$ using leI by simp
with $A1$ $< k \in \text{nat} >$ show $\langle k, n \rangle \in \text{Le}$ using Le_def
by simp

**Qed**

A version of succ_ineq without a quantifier, with additional assertion using the $n \#+1$ notation.

**Lemma succ_ineq1**: assumes $n \in \text{nat} \quad i \in n$
shows $\text{succ}(i) \in \text{succ}(n) \quad i \#+1 \in n \#+1 \quad i \in n \#+1$
using assms succ_ineq succ_add_one(1,7) elem_nat_is_nat(2)
by auto

For natural numbers membership and inequality are the same and $k \leq n$ is the same as $k \in \text{succ}(n)$. The proof relies on lemmas in the standard Isabelle’s Nat and Ordinal theories.

**Lemma nat_mem_lt**: assumes $n \in \text{nat}$
shows $k < n \quad k \in n$ and $k \leq n \quad k \in \text{succ}(n)$
using assms nat_into_Ord Ord_mem_iff_lt by auto

The term $k \leq n$ is the same as $k < \text{succ}(n)$.

**Lemma leq_mem_succ**: shows $k \leq n \quad k < \text{succ}(n)$ by simp

If the successor of a natural number $k$ is an element of the successor of $n$ then a similar relations holds for the numbers themselves.

**Lemma succ_mem**: assumes $n \in \text{nat} \quad \text{succ}(k) \in \text{succ}(n)$
shows $k \in n$
using assms elem_nat_is_nat(1) succ_leE nat_into_Ord unfolding lt_def by blast

The set of natural numbers is the union of its elements.

**Lemma nat_union_nat**: shows $\text{nat} = \bigcup \text{nat}$
using elem_nat_is_nat by blast

A natural number is a subset of the set of natural numbers.

**Lemma nat_subset_nat**: assumes $A1: n \in \text{nat}$ shows $n \subseteq \text{nat}$

**Proof** -
from $A1$ have $n \subseteq \bigcup \text{nat}$ by auto
then show $n \subseteq \text{nat}$ using nat_union_nat by simp

**Qed**
Adding natural numbers does not decrease what we add to.

**Lemma add_nat_le**: assumes \( A1: n \in \mathbb{N} \) and \( A2: k \in \mathbb{N} \)

shows
\[
\begin{align*}
& n \leq n + k \\
& n \subseteq n + k \\
& n \subseteq k + n
\end{align*}
\]

**Proof**

- from \( A1 \) \( A2 \) have \( n \leq n + 0 \leq k \in \mathbb{N} \)
  using \( \text{nat_le_refl nat_0_le} \) by \( \text{auto} \)
- then show \( n \leq k + n \) using \( \text{add_0_right} \) by \( \text{simp} \)
- then show \( n \subseteq k + n \) using \( \text{add_commute} \) by \( \text{simp} \)

**QED**

Result of adding an element of \( k \) is smaller than of adding \( k \).

**Lemma add_lt_mono**: assumes \( k \in \mathbb{N} \) and \( j \in k \)

shows
\[
(n + j) < (n + k)
\]

**Proof**

- from assms have \( j < k \) using \( \text{elem_nat_is_nat} \) by \( \text{blast} \)
- ultimately show \( (n + j) < (n + k) \) \( (n + j) \in (n + k) \)
  using \( \text{add_lt_mono2 ltD} \) by \( \text{auto} \)

**QED**

A technical lemma about a decomposition of a sum of two natural numbers: if a number \( i \) is from \( m + n \) then it is either from \( m \) or can be written as a sum of \( m \) and a number from \( n \). The proof by induction w.r.t. to \( m \) seems to be a bit heavy-handed, but I could not figure out how to do this directly from results from standard Isabelle/ZF.

**Lemma nat_sum_decomp**: assumes \( A1: n \in \mathbb{N} \) and \( A2: m \in \mathbb{N} \)

shows \( \forall i \in m + n. \ i \in m \lor (\exists j \in n. \ i = m + j) \)

**Proof**

- moreover from \( A2 \) have \( \forall i \in m + 0. \ i \in m \lor (\exists j \in 0. \ i = m + j) \)
  using \( \text{add_0_right} \) by \( \text{simp} \)
- moreover have \( \forall k \in \mathbb{N}. \ (\forall i \in m + k. \ i \in m \lor (\exists j \in k. \ i = m + j)) \rightarrow (\forall i \in m + \text{succ}(k). \ i \in m \lor (\exists j \in \text{succ}(k). \ i = m + j)) \)
  using \( \text{add_lt_mono2 ltD} \) by \( \text{auto} \)

**QED**
moreover from A4 A3 have
\[ i \in m \#+ k \longrightarrow i \in m \lor (\exists j \in \text{succ}(k). i = m \#+ j) \]
by auto
ultimately have \[ i \in m \lor (\exists j \in \text{succ}(k). i = m \#+ j) \]
by auto
\}
then have \( \forall i \in m \#+ \text{succ}(k). i \in m \lor (\exists j \in \text{succ}(k). i = m \#+ j) \)
by simp
\}\then have \( (\forall i \in m \#+ k. i \in m \lor (\exists j \in k. i = m \#+ j)) \rightarrow \\
(\forall i \in m \#+ \text{succ}(k). i \in m \lor (\exists j \in \text{succ}(k). i = m \#+ j)) \)
by simp
\}\then show thesis by simp
qed
ultimately show \( \forall i \in m \#+ n. i \in m \lor (\exists j \in n. i = m \#+ j) \)
by (rule ind_on_nat)
qed
A variant of induction useful for finite sequences.

**lemma fin_nat_ind:** assumes A1: \( n \in \text{nat} \) and A2: \( k \in \text{succ}(n) \)
and A3: \( P(0) \) and A4: \( \forall j \in n. P(j) \rightarrow P(\text{succ}(j)) \)
shows \( P(k) \)

**proof** -
from A2 have \( k \in n \lor k=\text{n} \) by auto
with A1 have \( k \in \text{nat} \) using elem_nat_is_nat by blast
moreover from A3 have \( 0 \in \text{succ}(n) \rightarrow P(0) \) by simp
moreover from A1 A4 have
\( \forall k \in \text{nat}. (k \in \text{succ}(n) \rightarrow P(k)) \rightarrow (\text{succ}(k) \in \text{succ}(n) \rightarrow P(\text{succ}(k))) \)
using nat_into_Ord Ord_succ_mem_iff by auto
ultimately have \( k \in \text{succ}(n) \rightarrow P(k) \)
by (rule ind_on_nat)
with A2 show \( P(k) \) by simp
qed

Some properties of positive natural numbers.

**lemma succ_plus:** assumes \( n \in \text{nat} \ k \in \text{nat} \)
shows \( \text{succ}(n \#+ j) \in \text{nat} \)
\( \text{succ}(n) \#+ \text{succ}(j) = \text{succ}(\text{succ}(n \#+ j)) \)
using assms by auto

If \( k \) is in the successor of \( n \), then the predecessor of \( k \) is in \( n \).

**lemma pred_succ_mem:** assumes \( n \in \text{nat} \ n \neq 0 \ k \in \text{succ}(n) \) shows \( \text{pred}(k) \in n \)

**proof** -
from assms(1,3) have \( k \in \text{nat} \) using succnat_subset_nat by blast
\{ assume \( k \neq 0 \)
\with \( k \in \text{nat} \) obtain \( j \) where \( j \in \text{nat} \) and \( k = \text{succ}(j) \)
using Nat_ZF_1_L3 by auto
\with assms(1,3) have \( \text{pred}(k) \in n \) using succ_mem pred_succ_eq
by simp
\}

35
moreover
{ assume k=0
  with assms(1,2) have pred(k)∈n
    using pred_0 empty_in_non_empty by simp
} ultimately show thesis by blast
qed

For non-zero natural numbers pred(n) = n - 1.

lemma pred_minus_one: assumes n∈nat n≠0
  shows n #− 1 = pred(n)
proof -
  from assms obtain k where n=succ(k)
    using Nat_ZF_1_L3 by blast
  with assms show thesis
    using pred_succ_eq eq_succ_imp_eq_m1 by simp
qed

For natural numbers if j ∈ n then j + 1 ⊆ n.

lemma mem_add_one_subset: assumes n ∈ nat k ∈ n
  shows k #+ 1 ⊆ n
proof -
  from assms have k #+ 1 ∈ succ(n)
    using elem_nat_is_nat(2) succ_ineq1 succ_add_one(1) by simp
  with assms(1) show k #+ 1 ⊆ n using nat_mem_lt(2) le_imp_subset
    by blast
qed

For a natural n if k ∈ n + 1 then k + 1 ≤ n + 1.

lemma succ_ineq2: assumes n ∈ nat k ∈ n #+ 1
  shows k #+ 1 ≤ n #+ 1 and k≤n
proof -
  from assms show k≤n using succ_add_one(1) nat_mem_lt(2)
    by simp
  with assms(1) show k #+ 1 ≤ n #+ 1 using add_le_mono1 by blast
qed

A nonzero natural number is of the form n = m + 1 for some natural number m. This is very similar to Nat_ZF_1_L3 except that we use n + 1 instead of succ(n).

lemma nat_not0_succ: assumes n∈nat n≠0
  shows ∃m∈nat. n = m #+1
    using assms Nat_ZF_1_L3 succ_add_one(1) by simp

Adding and subtracting a natural number cancel each other.

lemma add_subctract: assumes m∈nat shows (m #+ n) #− n = m
  using assms diff_add_inverse2 by simp

A version of induction on natural numbers that uses the n + 1 notation instead of succ(n).
lemma ind_on_nat1:
assumes n∈nat and P(0) and ∀k∈nat. P(k)→P(k #+ 1)
shows P(n) using assms succ_add_one(1) ind_on_nat by simp

A version of induction for finite sequences using the n + 1 notation instead of succ(n):

lemma fin_nat_ind1:
assumes n∈nat and P(0) and ∀j∈n. P(j)→P(j #+ 1)
shows ∀k∈n #+ 1. P(k) and P(n)

proof -
{ fix k assume k∈n #+ 1
with assms have
n∈nat k∈succ(n) P(0) ∀j∈n. P(j) → P(succ(j))
using succ_add_one(1) elem_nat_is_nat(2) by simp_all
then have P(k) by (rule fin_nat_ind)
} thus ∀k∈n #+ 1. P(k) by simp
with assms(1) show P(n) by simp
qed

A simplification rule for natural numbers: if k < n then n−(k+1)+1 = n−k:

lemma nat_subtr_simpl0: assumes n∈nat k∈n
shows n #- (k #+ 1) #+ 1 = n #- k

proof -
from assms obtain m where m∈nat and n = m #+1
using nat_not0_succ by blast
with assms have succ(m) = m #+ 1 succ(m #- k) = m #- k #+ 1
using elem_nat_is_nat(2) succ_add_one by simp_all
moreover from assms(2) m∈nat n = m #+1 have
succ(m) #- k = succ(m #- k)
using diff_succ succ_ineq2(2) by simp
ultimately have m #- k #+ 1 = m #+ 1 #- k by simp
with n = m #+1 show thesis using diff_cancel2 by simp
qed

4.2 Intervals

In this section we consider intervals of natural numbers i.e. sets of the form {n + j : j ∈ 0..k − 1}.

The interval is determined by two parameters: starting point and length.

definition
NatInterval(n,k) ≡ {n #+ j. j∈k}

Subtracting the beginning af the interval results in a number from the length of the interval. It may sound weird, but note that the length of such interval is a natural number, hence a set.

lemma inter_diff_in_len:
assumes $A1: k \in \text{nat}$ and $A2: i \in \text{NatInterval}(n,k)$
shows $i \#- n \in k$
proof -
  from $A2$ obtain $j$ where $I: i = n \#+ j$ and $II: j \in k$
    using $\text{NatInterval_def}$ by auto
  from $A1$ $II$ have $j \in \text{nat}$ using $\text{elem_nat_is_nat}$ by blast
  moreover from $I$ have $i \#- n = \text{natify}(j)$ using $\text{diff_add_inverse}$
    by simp
  ultimately have $i \#- n = j$ by simp
  with $II$ show thesis by simp
qed

Intervals don’t overlap with their starting point and the union of an interval
with its starting point is the sum of the starting point and the length of the
interval.

lemma $\text{length_start_decomp}$: assumes $A1: n \in \text{nat}$ $k \in \text{nat}$
shows $n \cap \text{NatInterval}(n,k) = 0$
$n \cup \text{NatInterval}(n,k) = n \#+ k$
proof -
  \{
    fix $i$ assume $A2: i \in n$ and $i \in \text{NatInterval}(n,k)$
    then obtain $j$ where $I: i = n \#+ j$ and $II: j \in k$
      using $\text{NatInterval_def}$ by auto
    from $A1$ $I$ have $k \in \text{nat}$ using $\text{elem_nat_is_nat}$ by blast
    with $II$ have $j \leq i$ using $\text{add_nat_le}$ by simp
    moreover from $A1$ $A2$ have $i < n$ using $\text{elem_nat_is_nat}$ by blast
    ultimately have False using $\text{le_imp_not_lt}$ by blast
  \}
  thus $n \cap \text{NatInterval}(n,k) = 0$ by auto
  from $A1$ have $n \subseteq n \#+ k$ using $\text{add_nat_le}$ by simp
  moreover\{
    fix $i$ assume $i \in \text{NatInterval}(n,k)$
    then obtain $j$ where $III: i = n \#+ j$ and $IV: j \in k$
      using $\text{NatInterval_def}$ by auto
    with $A1$ $III$ have $j < k$ using $\text{elem_nat_is_nat}$ by blast
    with $A1$ $IV$ have $i \in n \#+ k$ using $\text{add_lt_mono2}$ $\text{ltD}$
      by simp\}
  ultimately have $n \cup \text{NatInterval}(n,k) \subseteq n \#+ k$ by auto
  moreover from $A1$ have $n \#+ k \subseteq n \cup \text{NatInterval}(n,k)$
    using $\text{nat_sum_decomp}$ $\text{NatInterval_def}$ by auto
  ultimately show $n \cup \text{NatInterval}(n,k) = n \#+ k$ by auto
qed

Some properties of three adjacent intervals.

lemma $\text{adjacent_intervals3}$: assumes $n \in \text{nat}$ $k \in \text{nat}$ $m \in \text{nat}$
shows $n \#+ k \#+ m = (n \#+ k) \cup \text{NatInterval}(n \#+ k,m)$
$n \#+ k \#+ m = n \cup \text{NatInterval}(n,k \#+ m)$
$n \#+ k \#+ m = n \cup \text{NatInterval}(n,k) \cup \text{NatInterval}(n \#+ k,m)$
5 Order relations - introduction

theory Order_ZF imports Fol1

begin

This theory file considers various notions related to order. We redefine the notions of a preorder, directed set, total order, linear order and partial order to have the same terminology as Wikipedia (I found it very consistent across different areas of math). We also define and study the notions of intervals and bounded sets. We show the inclusion relations between the intervals with endpoints being in certain order. We also show that union of bounded sets are bounded. This allows to show in Finite_ZF.thy that finite sets are bounded.

5.1 Definitions

In this section we formulate the definitions related to order relations.

A relation \( r \) is "total" on a set \( X \) if for all elements \( a, b \) of \( X \) we have \( a \) is in relation with \( b \) or \( b \) is in relation with \( a \). An example is the \( \leq \) relation on numbers.

**definition**

\[
\text{IsTotal (infixl \{ is total on \} 65) where}\\
\text{r \{is total on\} X \equiv (\forall a \in X. \forall b \in X. (a, b) \in r \lor (b, a) \in r)}
\]

A relation \( r \) is a partial order on \( X \) if it is reflexive on \( X \) (i.e. \( \langle x, x \rangle \) for every \( x \in X \)), antisymmetric (if \( \langle x, y \rangle \in r \) and \( \langle y, x \rangle \in r \), then \( x = y \)) and transitive \( \langle x, y \rangle \in r \) and \( \langle y, z \rangle \in r \) implies \( \langle x, z \rangle \in r \).

**definition**

\[
\text{IsPartOrder(X,r) \equiv refl(X,r) \land antisym(r) \land trans(r)}
\]

A relation that is reflexive and transitive is called a preorder.

**definition**

\[
\text{IsPreorder(X,r) \equiv refl(X,r) \land trans(r)}
\]

We say that a relation \( r \) up-directs a set if every two-element subset of \( X \) has an upper bound.

**definition**

\[
\text{UpDirects (_{up-directs} _90) where}\\
\text{r \{up-directs\} X \equiv X \neq 0 \land (\forall x \in X. \forall y \in X. \exists z \in X. \langle x, z \rangle \in r \land \langle y, z \rangle \in r)}
\]

39
Analogously we say that a relation $r$ down-directs a set if every two-element subset of $X$ has a lower bound.

**definition**

DownDirects $r \downarrow X \equiv X \neq 0 \land (\forall x \in X. \forall y \in X. \exists z \in X. \langle z, x \rangle \in r \land \langle z, y \rangle \in r)$

Typically the notion that is actually defined is the notion of a directed set or an upward directed set, rather than $r$ down-directs $X$ (or $r$ up-directs $X$). This is a nonempty set $X$ together with a preorder $r$ such that $r$ up-directs $X$. We set that up in separate definitions as we sometimes want to use an upward or downward directed set with a partial order rather than a preorder.

**definition**

IsUpDirectedSet($X, r$) $\equiv$ IsPreorder($X, r$) $\land$ ($r$ up-directs $X$)

We define the notion of a downward directed set analogously.

**definition**

IsDownDirectedSet($X, r$) $\equiv$ IsPreorder($X, r$) $\land$ ($r$ down-directs $X$)

We define a linear order as a binary relation that is antisymmetric, transitive and total. Note that this terminology is different than the one used the standard Order.thy file.

**definition**

IsLinOrder($X, r$) $\equiv$ antisym($r$) $\land$ trans($r$) $\land$ ($r$ is total on $X$)

A set is bounded above if there is that is an upper bound for it, i.e. there are some $u$ such that $\langle x, u \rangle \in r$ for all $x \in A$. In addition, the empty set is defined as bounded.

**definition**

IsBoundedAbove($A, r$) $\equiv$ ($A = 0$ $\lor$ ($\exists u. \forall x \in A. \langle x, u \rangle \in r$))

We define sets bounded below analogously.

**definition**

IsBoundedBelow($A, r$) $\equiv$ ($A = 0$ $\lor$ ($\exists l. \forall x \in A. \langle l, x \rangle \in r$))

A set is bounded if it is bounded below and above.

**definition**

IsBounded($A, r$) $\equiv$ (IsBoundedAbove($A, r$) $\land$ IsBoundedBelow($A, r$))

The notation for the definition of an interval may be mysterious for some readers, see lemma Order_ZF_2_L1 for more intuitive notation.

**definition**

Interval($r, a, b$) $\equiv$ $r$-{a} $\cap$ $r$-{b}

We also define the maximum (the greater of) two elements in the obvious way.
definition
GreaterOf(r,a,b) ≡ (if ⟨a,b⟩ ∈ r then b else a)

The definition a a minimum (the smaller of) two elements.

definition
SmallerOf(r,a,b) ≡ (if ⟨a,b⟩ ∈ r then a else b)

We say that a set has a maximum if it has an element that is not smaller that any other one. We show that under some conditions this element of the set is unique (if exists).

definition
HasAmaximum(r,A) ≡ ∃M ∈ A. ∀x ∈ A. ⟨x,M⟩ ∈ r

A similar definition what it means that a set has a minimum.

definition
HasAminimum(r,A) ≡ ∃m ∈ A. ∀x ∈ A. ⟨m,x⟩ ∈ r

Definition of the maximum of a set.

definition
Maximum(r,A) ≡ THE M. M ∈ A ∧ (∀x ∈ A. ⟨x,M⟩ ∈ r)

Definition of a minimum of a set.

definition
Minimum(r,A) ≡ THE m. m ∈ A ∧ (∀x ∈ A. ⟨m,x⟩ ∈ r)

The supremum of a set \(A\) is defined as the minimum of the set of upper bounds, i.e. the set \(\{u.\forall a ∈ A. ⟨a,u⟩ ∈ r\} = \bigcap_{a ∈ A} r\{a\}\). Recall that in Isabelle/ZF \(r-(A)\) denotes the inverse image of the set \(A\) by relation \(r\) (i.e. \(r-(A)=\{x : ⟨x,y⟩ ∈ r \text{ for some } y ∈ A\}\)).

definition
Supremum(r,A) ≡ Minimum(r,\bigcap_{a ∈ A. r(a)})

The notion of "having a supremum" is the same as the set of upper bounds having a minimum, but having it a a separate notion does simplify notation in some cases. The definition is written in terms of images of singletons \(\{x\}\) under relation. To understand this formulation note that the set of upper bounds of a set \(A ⊆ X\) is \(\bigcap_{x ∈ A}{\{y ∈ X|⟨x,y⟩ ∈ r\}}\), which is the same as \(\bigcap_{x ∈ A}{r(\{x\})}\), where \(r(\{x\})\) is the image of the singleton \(\{x\}\) under relation \(r\).

definition
HasASupremum(r,A) ≡ HasAminimum(r,\bigcap_{a ∈ A. r(a)})

The notion of "having an infimum" is the same as the set of lower bounds having a maximum.

definition
HasAnInfimum(r,A) ≡ HasAmaximum(r,\bigcap_{a ∈ A. r-{a}})

41
Infimum is defined analogously.

**definition**
\[
\text{Infimum}(r, A) \equiv \text{Maximum}(r, \bigcap_{a \in A} r - \{a\})
\]

We define a relation to be complete if every nonempty bounded above set has a supremum.

**definition**
\[
\text{IsComplete}(_{\text{is complete}}) \text{ where}\n\text{r \{is complete\}} \equiv \\
\forall A. \text{IsBoundedAbove}(A, r) \land A \neq 0 \implies \text{HasAminimum}(r, \bigcap_{a \in A} r \{a\})
\]

If a relation down-directs a set, then a larger one does as well.

**lemma** **down_dir_mono**: assumes \(r \{\text{down-directs}\} X \subseteq R\)
\[
\text{shows } R \{\text{down-directs}\} X \text{ using } \text{assms unfolding DownDirects_def by blast}
\]

If a relation up-directs a set, then a larger one does as well.

**lemma** **up_dir_mono**: assumes \(r \{\text{up-directs}\} X \subseteq R\)
\[
\text{shows } R \{\text{up-directs}\} X \text{ using } \text{assms unfolding UpDirects_def by blast}
\]

The essential condition to show that a total relation is reflexive.

**lemma** **Order_ZF_1_L1**: assumes \(r \{\text{is total on}\} X \text{ and } a \in X\)
\[
\text{shows } (a, a) \in r \text{ using } \text{assms IsTotal_def by auto}
\]

A total relation is reflexive.

**lemma** **total_is_refl**: 
\[
\text{assumes } r \{\text{is total on}\} X \text{ shows } \text{refl}(X, r) \text{ using } \text{assms Order_ZF_1_L1 refl_def by simp}
\]

A linear order is partial order.

**lemma** **Order_ZF_1_L2**: assumes \(\text{IsLinOrder}(X, r)\)
\[
\text{shows } \text{IsPartOrder}(X, r) \text{ using } \text{assms IsLinOrder_def IsPartOrder_def refl_def Order_ZF_1_L1 by auto}
\]

Partial order that is total is linear.

**lemma** **Order_ZF_1_L3**: 
\[
\text{assumes } \text{IsPartOrder}(X, r) \text{ and } r \{\text{is total on}\} X \text{ shows } \text{IsLinOrder}(X, r) \text{ using } \text{assms IsPartOrder_def IsLinOrder_def by simp}
\]

Relation that is total on a set is total on any subset.

**lemma** **Order_ZF_1_L4**: assumes \(r \{\text{is total on}\} X \text{ and } A \subseteq X\)
\[
\text{shows } r \{\text{is total on}\} A
\]
using assms IsTotal_def by auto

We can restrict a partial order relation to the domain.

lemma part_ord_restr: assumes IsPartOrder(X,r)
  shows IsPartOrder(X,r ∩ X×X)
  using assms unfolding IsPartOrder_def refl_def antisym_def trans_def
by auto

We can restrict a total order relation to the domain.

lemma total_ord_restr: assumes r {is total on} X
  shows (r ∩ X×X) {is total on} X
  using assms unfolding IsTotal_def by auto

A linear relation is linear on any subset and we can restrict it to any subset.

lemma ord_linear_subset: assumes IsLinOrder(X,r) and A ⊆ X
  shows IsLinOrder(A,r) and IsLinOrder(A,r ∩ A×A)
proof -
  from assms show IsLinOrder(A,r) using IsLinOrder_def Order_ZF_1_L4
  by blast
  then have IsPartOrder(A,r ∩ A×A) and (r ∩ A×A) {is total on} A
    using Order_ZF_1_L2 part_ord_restr total_ord_restr unfolding IsLinOrder_def
    by auto
  then show IsLinOrder(A,r ∩ A×A) using Order_ZF_1_L3 by simp
qed

If the relation is total, then every set is a union of those elements that are
nongreater than a given one and nonsmaller than a given one.

lemma Order_ZF_1_L5:
  assumes r {is total on} X and A ⊆ X and a ∈ X
  shows A = {x ∈ A. ⟨a,x⟩ ∈ r} ∪ {x ∈ A. ⟨x,a⟩ ∈ r}
  using assms IsTotal_def by auto

A technical fact about reflexive relations.

lemma refl_add_point:
  assumes refl(X,r) and A ⊆ B ∪ {x} and B ⊆ X and
  x ∈ X and ∀y∈B. ⟨y,x⟩ ∈ r
  shows ∀a∈A. ⟨a,x⟩ ∈ r
  using assms refl_def by auto

5.2 Intervals

In this section we discuss intervals.

The next lemma explains the notation of the definition of an interval.

lemma Order_ZF_2_L1:
  shows x ∈ Interval(r,a,b) ↔ ⟨a,x⟩ ∈ r ∧ ⟨x,b⟩ ∈ r
using Interval_def by auto

Since there are some problems with applying the above lemma (seems that simp and auto don’t handle equivalence very well), we split Order_ZF_2_L1 into two lemmas.

lemma Order_ZF_2_L1A: assumes x ∈ Interval(r,a,b) shows ⟨a,x⟩ ∈ r  ⟨x,b⟩ ∈ r using assms Order_ZF_2_L1 by auto

Order_ZF_2_L1, implication from right to left.

lemma Order_ZF_2_L1B: assumes ⟨a,x⟩ ∈ r  ⟨x,b⟩ ∈ r shows x ∈ Interval(r,a,b) using assms Order_ZF_2_L1 by simp

If the relation is reflexive, the endpoints belong to the interval.

lemma Order_ZF_2_L2: assumes refl(X,r) and a ∈ X b ∈ X and ⟨a,b⟩ ∈ r shows a ∈ Interval(r,a,b) b ∈ Interval(r,a,b) using assms refl_def Order_ZF_2_L1 by auto

Under the assumptions of Order_ZF_2_L2, the interval is nonempty.

lemma Order_ZF_2_L2A: assumes refl(X,r) and a ∈ X b ∈ X and ⟨a,b⟩ ∈ r shows Interval(r,a,b) ≠ 0 proof - from assms have a ∈ Interval(r,a,b) using Order_ZF_2_L2 by simp then show Interval(r,a,b) ≠ 0 by auto qed

If a, b, c, d are in this order, then [b, c] ⊆ [a, d]. We only need transitivity for this to be true.

lemma Order_ZF_2_L3: assumes A1: trans(r) and A2: ⟨a,b⟩ ∈ r  ⟨b,c⟩ ∈ r  ⟨c,d⟩ ∈ r shows Interval(r,b,c) ⊆ Interval(r,a,d) proof
fix x assume A3: x ∈ Interval(r, b, c)
note A1 moreover from A2 A3 have (a,b) ∈ r ∧ (b,c) ∈ r using Order_ZF_2_L1A by simp ultimately have T1: ⟨a,x⟩ ∈ r by (rule Fol1_L3)
note A1 moreover from A2 A3 have (x,c) ∈ r ∧ (c,d) ∈ r using Order_ZF_2_L1A by simp ultimately have ⟨x,d⟩ ∈ r by (rule Fol1_L3)
with T1 show x ∈ Interval(r,a,d) using Order_ZF_2_L1B

44
For reflexive and antisymmetric relations the interval with equal endpoints consists only of that endpoint.

**lemma** Order_ZF_2_L4:

assumes A1: refl(X,r) and A2: antisym(r) and A3: a∈X

shows Interval(r,a,a) = {a}

**proof**

from A1 A3 have ⟨a,a⟩ ∈ r using refl_def by simp

with A1 A3 show {a} ⊆ Interval(r,a,a) using Order_ZF_2_L2 by simp

from A2 show Interval(r,a,a) ⊆ {a} using Order_ZF_2_L1A Fol1_L4 by fast

qed

For transitive relations the endpoints have to be in the relation for the interval to be nonempty.

**lemma** Order_ZF_2_L5: assumes A1: trans(r) and A2: ⟨a,b⟩ /∈ r

shows Interval(r,a,b) = 0

**proof** -

{ assume Interval(r,a,b)≠0 then obtain x where x ∈ Interval(r,a,b) by auto

with A1 A2 have False using Order_ZF_2_L1A Fol1_L3 by fast

} thus thesis by auto

qed

If a relation is defined on a set, then intervals are subsets of that set.

**lemma** Order_ZF_2_L6: assumes A1: r ⊆ X×X

shows Interval(r,a,b) ⊆ X

using assms Interval_def by auto

5.3 Bounded sets

In this section we consider properties of bounded sets.

For reflexive relations singletons are bounded.

**lemma** Order_ZF_3_L1: assumes refl(X,r) and a∈X

shows IsBounded({a},r)

using assms refl_def IsBoundedAbove_def IsBoundedBelow_def IsBounded_def by auto

Sets that are bounded above are contained in the domain of the relation.

**lemma** Order_ZF_3_L1A: assumes r ⊆ X×X and IsBoundedAbove(A,r)

shows A⊆X using assms IsBoundedAbove_def by auto

Sets that are bounded below are contained in the domain of the relation.

**lemma** Order_ZF_3_L1B: assumes r ⊆ X×X
and IsBoundedBelow(A,r) shows A ⊆ X using assms IsBoundedBelow_def by auto

For a total relation, the greater of two elements, as defined above, is indeed greater of any of the two.

lemma Order_ZF_3_L2: assumes r {is total on} X and x ∈ X y ∈ X shows ⟨x,GreaterOf(r,x,y)⟩ ∈ r ⟨y,GreaterOf(r,x,y)⟩ ∈ r ⟨SmallerOf(r,x,y),x⟩ ∈ r ⟨SmallerOf(r,x,y),y⟩ ∈ r using assms IsTotal_def Order_ZF_1_L1 GreaterOf_def SmallerOf_def by auto

If A is bounded above by u, B is bounded above by w, then A ∪ B is bounded above by the greater of u, w.

lemma Order_ZF_3_L2B: assumes A1: r {is total on} X and A2: trans(r) and A3: u ∈ X w ∈ X and A4: ∀ x ∈ A. ⟨x,u⟩ ∈ r ∀ x ∈ B. ⟨x,w⟩ ∈ r shows ∀ x ∈ A ∪ B. ⟨x,GreaterOf(r,u,w)⟩ ∈ r proof
  let v = GreaterOf(r,u,w)
  from A1 A3 have T1: ⟨u,v⟩ ∈ r and T2: ⟨w,v⟩ ∈ r using Order_ZF_3_L2 by auto
  fix x assume A5: x ∈ A ∪ B show ⟨x,v⟩ ∈ r proof
  - { assume x ∈ A
      with A4 T1 have ⟨x,u⟩ ∈ r ∧ ⟨u,v⟩ ∈ r by simp
      with A2 have ⟨x,v⟩ ∈ r by (rule Fol1_L3) }
  moreover
  { assume x /∈ A
      with A5 A4 T2 have ⟨x,w⟩ ∈ r ∧ ⟨w,v⟩ ∈ r by simp
      with A2 have ⟨x,v⟩ ∈ r by (rule Fol1_L3) }
  ultimately show thesis by auto
  qed
qed

For total and transitive relation the union of two sets bounded above is bounded above.

lemma Order_ZF_3_L3: assumes A1: r {is total on} X and A2: trans(r) and A3: IsBoundedAbove(A,r) IsBoundedAbove(B,r) and A4: r ⊆ X×X shows IsBoundedAbove(A∪B,r) proof -
  { assume A=0 ∨ B=0
with A3 have IsBoundedAbove(A∪B, r) by auto } 
moreover
{ assume ¬ (A = 0 ∨ B = 0)
then have T1: A ≠ 0 B ≠ 0 by auto
with A3 obtain u w where D1: ∀ x∈A. ⟨x,u⟩ ∈ r ∀ x∈B. ⟨x,w⟩ ∈ r
using IsBoundedAbove_def by auto
let U = GreaterOf(r,u,w)
from T1 A4 D1 have u ∈ X w ∈ X by auto
with A1 A2 D1 have ∀ x∈A∪B. ⟨x,U⟩ ∈ r
using Order_ZF_3_L2 by blast
then have IsBoundedAbove(A∪B,r)
using IsBoundedAbove_def by auto
ultimately show thesis by auto
qed

For total and transitive relations if a set A is bounded above then A∪{a} is bounded above.

lemma Order_ZF_3_L4:
assumes A1: r {is total on} X and A2: trans(r) and A3: IsBoundedAbove(A,r) and A4: a ∈ X and A5: r ⊆ X×X
shows IsBoundedAbove(A∪{a},r)
proof -
from A1 have refl(X,r) using total_is_refl by simp
with assms show thesis using Order_ZF_3_L1 IsBounded_def Order_ZF_3_L3 by simp
qed

If A is bounded below by l, B is bounded below by m, then A∪B is bounded below by the smaller of u, w.

lemma Order_ZF_3_L5B:
assumes A1: r {is total on} X and A2: trans(r) and A3: l ∈ X m ∈ X and A4: ∀ x∈A. ⟨l,x⟩ ∈ r ∀ x∈B. ⟨m,x⟩ ∈ r
shows ∀ x∈A∪B. ⟨SmallerOf(r,l,m),x⟩ ∈ r
proof -
let k = SmallerOf(r,l,m)
from A1 A3 have T1: ⟨k,l⟩ ∈ r and T2: ⟨k,m⟩ ∈ r
using Order_ZF_3_L2 by auto
fix x assume A5: x ∈ A∪B show ⟨k,x⟩ ∈ r
proof -
{ assume x∈A
with A4 T1 have ⟨k,l⟩ ∈ r ∧ ⟨l,x⟩ ∈ r by simp
with A2 have ⟨k,x⟩ ∈ r by (rule Fol1_L3) }
moreover
{ assume x∉A
with A5 A4 T2 have ⟨k,m⟩ ∈ r ∧ ⟨m,x⟩ ∈ r by simp
with A2 have ⟨k,x⟩ ∈ r by (rule Fol1_L3) }
ultimately show thesis by auto

47
For total and transitive relation the union of two sets bounded below is bounded below.

**Lemma Order_ZF_3_L6:**

assumes
A1: \( r \) is total on \( X \) and
A2: trans\( (r) \)
and
A3: IsBoundedBelow\( (A,r) \) IsBoundedBelow\( (B,r) \)
and
A4: \( r \subseteq X \times X \)

shows IsBoundedBelow\( (A \cup B,r) \)

**Proof:**

- \{ assume \( A = 0 \) \( \lor \) \( B = 0 \) 
  with A3 have thesis by auto \}

moreover

- \{ assume \( \neg (A = 0 \lor B = 0) \) 
  then have T1: \( A \neq 0 \) \( B \neq 0 \) by auto 
  with A3 obtain \( \lambda \) \( \mu \) where D1: \( \forall x \in A. \ (\lambda ,x) \in r \) \( \forall x \in B. \ (\mu ,x) \in r \) 
    using IsBoundedBelow_def by auto 
  let L = SmallerOf\( (r,\lambda ,\mu ) \) 
  from T1 A4 D1 have T1: \( \lambda \in X \) \( \mu \in X \) by auto 
  with A1 A2 D1 have \( \forall x \in A \cup B. \ (L,x) \in r \) 
    using Order_ZF_3_L5B by blast 
  then have IsBoundedBelow\( (A \cup B,r) \) 
    using IsBoundedBelow_def by auto \}

ultimately show thesis by auto

**Qed**

For total and transitive relations if a set \( A \) is bounded below then \( A \cup \{a\} \) is bounded below.

**Lemma Order_ZF_3_L7:**

assumes
A1: \( r \) is total on \( X \) and
A2: trans\( (r) \)
and
A3: IsBoundedBelow\( (A,r) \) and
A4: \( a \in X \) and
A5: \( r \subseteq X \times X \)

shows IsBoundedBelow\( (A \cup \{a\},r) \)

**Proof:**

- from A1 have refl\( (X,r) \)
  using total_is_refl by simp 
  with assms show thesis using 
    Order_ZF_3_L1 IsBounded_def Order_ZF_3_L6 by simp

**Qed**

For total and transitive relations unions of two bounded sets are bounded.

**Theorem Order_ZF_3_T1:**

assumes \( r \) is total on \( X \) and trans\( (r) \)
and IsBounded\( (A,r) \) IsBounded\( (B,r) \)
and \( r \subseteq X \times X \)

shows IsBounded\( (A \cup B,r) \)

using assms Order_ZF_3_L3 Order_ZF_3_L6 Order_ZF_3_L7 IsBounded_def by simp
For total and transitive relations if a set $A$ is bounded then $A \cup \{a\}$ is bounded.

**Lemma Order_ZF_3_L8:**

assumes $r$ is total on $X$ and trans($r$) and IsBounded($A$, $r$) and $a \in X$ and $r \subseteq X \times X$

shows IsBounded($A \cup \{a\}$, $r$)

using assms total_is_refl Order_ZF_3_L1 Order_ZF_3_T1 by blast

A sufficient condition for a set to be bounded below.

**Lemma Order_ZF_3_L9:** assumes $A1: \forall a \in A. \langle 1, a \rangle \in r$

shows IsBoundedBelow($A$, $r$)

proof -
from $A1$ have $\exists l. \forall x \in A. \langle l, x \rangle \in r$
by auto
then show IsBoundedBelow($A$, $r$)
using IsBoundedBelow_def by simp
qed

A sufficient condition for a set to be bounded above.

**Lemma Order_ZF_3_L10:** assumes $A1: \forall a \in A. \langle a, u \rangle \in r$

shows IsBoundedAbove($A$, $r$)

proof -
from $A1$ have $\exists u. \forall x \in A. \langle x, u \rangle \in r$
by auto
then show IsBoundedAbove($A$, $r$)
using IsBoundedAbove_def by simp
qed

Intervals are bounded.

**Lemma Order_ZF_3_L11:** shows
IsBoundedAbove(Interval($r$, $a$, $b$), $r$) IsBoundedBelow(Interval($r$, $a$, $b$), $r$) IsBounded(Interval($r$, $a$, $b$), $r$)

proof -
{ fix $x$ assume $x \in \text{Interval}(r, a, b)$
then have $\langle x, b \rangle \in r \langle a, x \rangle \in r$
using Order_ZF_2_L1A by auto
} then have $\exists u. \forall x \in \text{Interval}(r, a, b). \langle x, u \rangle \in r$
$\exists l. \forall x \in \text{Interval}(r, a, b). \langle l, x \rangle \in r$
by auto
then show IsBoundedAbove(Interval($r$, $a$, $b$), $r$) IsBoundedBelow(Interval($r$, $a$, $b$), $r$) IsBounded(Interval($r$, $a$, $b$), $r$)
using IsBoundedAbove_def IsBoundedBelow_def IsBounded_def by auto
qed
A subset of a set that is bounded below is bounded below.

**Lemma Order_ZF_3_L12:** assumes \( A_1: \text{IsBoundedBelow}(A,r) \) and \( A_2: B \subseteq A \)
shows \( \text{IsBoundedBelow}(B,r) \)

**Proof** -

\[
\begin{align*}
\text{assume } A &= \emptyset \\
\text{with assms have } \text{IsBoundedBelow}(B,r) &\text{ using IsBoundedBelow_def by auto } \}
\end{align*}
\]

moreover

\[
\begin{align*}
\text{assume } A &\neq \emptyset \\
\text{with } A_1 \text{ have } \exists l. \forall x \in A. \langle l,x \rangle \in r &\text{ using IsBoundedBelow_def by simp } \\
\text{with } A_2 \text{ have } \exists l. \forall x \in B. \langle l,x \rangle \in r &\text{ by auto } \}
\end{align*}
\]

then have \( \text{IsBoundedBelow}(B,r) \) using \( \text{IsBoundedBelow_def by auto } \)

ultimately show \( \text{IsBoundedBelow}(B,r) \) by auto

**QED**

A subset of a set that is bounded above is bounded above.

**Lemma Order_ZF_3_L13:** assumes \( A_1: \text{IsBoundedAbove}(A,r) \) and \( A_2: B \subseteq A \)
shows \( \text{IsBoundedAbove}(B,r) \)

**Proof** -

\[
\begin{align*}
\text{assume } A &= \emptyset \\
\text{with assms have } \text{IsBoundedAbove}(B,r) &\text{ using IsBoundedAbove_def by auto } \}
\end{align*}
\]

moreover

\[
\begin{align*}
\text{assume } A &\neq \emptyset \\
\text{with } A_1 \text{ have } \exists u. \forall x \in A. \langle x,u \rangle \in r &\text{ using IsBoundedAbove_def by simp } \\
\text{with } A_2 \text{ have } \exists u. \forall x \in B. \langle x,u \rangle \in r &\text{ by auto } \}
\end{align*}
\]

then have \( \text{IsBoundedAbove}(B,r) \) using \( \text{IsBoundedAbove_def by auto } \)

ultimately show \( \text{IsBoundedAbove}(B,r) \) by auto

**QED**

If for every element of \( X \) we can find one in \( A \) that is greater, then the \( A \) can not be bounded above. Works for relations that are total, transitive and antisymmetric, (i.e. for linear order relations).

**Lemma Order_ZF_3_L14:**

assumes \( A_1: r \text{ (is total on) } X \)
and \( A_2: \text{trans}(r) \) and \( A_3: \text{antisym}(r) \)
and \( A_4: r \subseteq X \times X \) and \( A_5: X \neq 0 \)
and \( A_6: \forall x \in X. \exists a \in A. x \neq a \wedge \langle x,a \rangle \in r \)
shows \( \neg \text{IsBoundedAbove}(A,r) \)

**Proof** -

\[
\begin{align*}
\text{from } A_5 A_6 \text{ have } I \neq 0 &\text{ by auto } \\
\text{moreover assume } \text{IsBoundedAbove}(A,r) &\text{ using IsBounded_def IsBoundedAbove_def by auto } \\
\text{ultimately obtain } u \text{ where II: } &\forall x \in A. \langle x,u \rangle \in r
\end{align*}
\]

50
with A4 I have u∈X by auto
with A6 obtain b where b∈A and III: u≠b and ⟨u,b⟩ ∈ r
by auto
with II have ⟨b,u⟩ ∈ r ⟨u,b⟩ ∈ r by auto
with A3 have b=u by (rule Fol1_L4)
with III have False by simp
} thus ¬IsBoundedAbove(A,r) by auto
qed

The set of elements in a set A that are nongreater than a given element is bounded above.

lemma Order_ZF_3_L15: shows IsBoundedAbove({x∈A. ⟨x,a⟩ ∈ r},r)
using IsBoundedAbove_def by auto

If A is bounded below, then the set of elements in a set A that are nongreater than a given element is bounded.

lemma Order_ZF_3_L16: assumes A1: IsBoundedBelow(A,r)
shows IsBounded({x∈A. ⟨x,a⟩ ∈ r},r)
proof -
{ assume A=0
  then have IsBounded({x∈A. ⟨x,a⟩ ∈ r},r)
  using IsBoundedBelow_def IsBoundedAbove_def IsBounded_def
  by auto }
moreover
{ assume A≠0
  with A1 obtain l where I: ∀x∈A. ⟨l,x⟩ ∈ r
  using IsBoundedBelow_def by auto
  then have ∀y∈{x∈A. ⟨x,a⟩ ∈ r}. ⟨l,y⟩ ∈ r by simp
  then have IsBoundedBelow({x∈A. ⟨x,a⟩ ∈ r},r)
  by (rule Order_ZF_3_L9)
  then have IsBounded({x∈A. ⟨x,a⟩ ∈ r},r)
  using Order_ZF_3_L15 IsBounded_def by simp }
ultimately show thesis by blast
qed

end

6 More on order relations

theory Order_ZF_1 imports ZF.Order ZF1
begin

In Order_ZF we define some notions related to order relations based on the nonstrict orders (≤ type). Some people however prefer to talk about these notions in terms of the strict order relation (< type). This is the case for the standard Isabelle Order.thy and also for Metamath. In this theory file we repeat some developments from Order_ZF using the strict order relation as
a basis. This is mostly useful for Metamath translation, but is also of some general interest. The names of theorems are copied from Metamath.

6.1 Definitions and basic properties

In this section we introduce some definitions taken from Metamath and relate them to the ones used by the standard Isabelle Order.thy.

The next definition is the strict version of the linear order. What we write as $R$ Orders $A$ is written $ROrdA$ in Metamath.

definition
StrictOrder (infix Orders 65) where
$R$ Orders $A \equiv \forall x \ y \ z. (x \in A \land y \in A \land z \in A) \rightarrow$
$(\langle x,y \rangle \in R \iff \neg (x=y \lor \langle y,x \rangle \in R)) \land$
$(\langle x,y \rangle \in R \land \langle y,z \rangle \in R \rightarrow \langle x,z \rangle \in R)$

The definition of supremum for a (strict) linear order.

definition
Sup(B,A,R) \equiv \bigcup \{x \in A. (\forall y \in B. \langle x,y \rangle \notin R) \land$
$(\forall y \in A. \langle y,x \rangle \in R \rightarrow (\exists z \in B. \langle y,z \rangle \in R))\}

Definition of infimum for a linear order. It is defined in terms of supremum.

definition
Infim(B,A,R) \equiv Sup(B,A,\text{converse}(R))

If relation $R$ orders a set $A$, (in Metamath sense) then $R$ is irreflexive, transitive and linear therefore is a total order on $A$ (in Isabelle sense).

lemma orders_imp_tot_ord: assumes A1: $R$ Orders $A$
shows
irrefl(A,R)
trans[A](R)
polr_deq(
linear(A,R)
tot_ord(A,R)
proof -
from A1 have I:
$\forall x \ y \ z. (x \in A \land y \in A \land z \in A) \rightarrow$
$(\langle x,y \rangle \in R \iff \neg (x=y \lor \langle y,x \rangle \in R)) \land$
$(\langle x,y \rangle \in R \land \langle y,z \rangle \in R \rightarrow \langle x,z \rangle \in R)$

unfolding StrictOrder_def by simp
then have $\forall x \in A. \langle x,x \rangle \notin R$ by blast
then show irrefl(A,R) using irrefl_def by simp
moreover
from I have
$\forall x \in A. \forall y \in A. \forall z \in A. \langle x,y \rangle \in R \rightarrow \langle y,z \rangle \in R \rightarrow \langle x,z \rangle \in R$
by blast

52
then show trans[A](R) unfolding trans_on_def by blast
ultimately show part_ord(A,R) using part_ord_def
by simp
moreover
from I have
\[ \forall x \in A. \forall y \in A. \langle x,y \rangle \in R \lor x=y \lor \langle y,x \rangle \in R \]
by blast
then show linear(A,R) unfolding linear_def by blast
ultimately show tot_ord(A,R) using tot_ord_def
by simp
qed

A converse of orders_imp_tot_ord. Together with that theorem this shows
that Metamath’s notion of an order relation is equivalent to Isabelle’s tot_ord
predicate.

lemma tot_ord_imp_orders: assumes A1: tot_ord(A,R)
shows R Orders A
proof -
from A1 have
I: linear(A,R) and
II: irrefl(A,R) and
III: trans[A](R) and
IV: part_ord(A,R)
using tot_ord_def part_ord_def by auto
from IV have asym(R \cap A \times A)
using part_ord_Imp_asym by simp
then have V: \[ \forall x y. \langle x,y \rangle \in (R \cap A \times A) \longrightarrow \neg(\langle y,x \rangle \in (R \cap A \times A)) \]
unfolding asym_def by blast
from I have VI: \[ \forall x \in A. \forall y \in A. \langle x,y \rangle \in R \lor x=y \lor \langle y,x \rangle \in R \]
unfolding linear_def by blast
from III have VII: \[ \forall x \in A. \forall y \in A. \forall z \in A. \langle x,y \rangle \in R \longrightarrow (\langle y,z \rangle \in R \longrightarrow (\langle x,z \rangle \in R) \]
unfolding trans_on_def by blast
\{ fix x y z 
assume T: x \in A y \in A z \in A
have \langle x,y \rangle \in R \longleftrightarrow \neg(x=y \lor \langle y,x \rangle \in R)
proof
assume A2: \langle x,y \rangle \in R
with V T have \neg((y,x) \in R) by blast
moreover from II T A2 have x\neq y using irrefl_def
by auto
ultimately show \neg(x=y \lor \langle y,x \rangle \in R) by simp
next assume \neg(x=y \lor \langle y,x \rangle \in R)
with VI T show \langle x,y \rangle \in R by auto
qed
moreover from VII T have
\langle x,y \rangle \in R \land (\langle y,z \rangle \in R \longrightarrow (\langle x,z \rangle \in R)
by blast
ultimately have ((\langle x,y \rangle \in R \longleftrightarrow \neg(x=y \lor \langle y,x \rangle \in R)) \land
\]
\[(x,y) \in R \land (y,z) \in R \rightarrow (x,z) \in R\]

by simp

\[
\begin{align*}
\forall x \ y \ z. \ (x \in A \land y \in A \land z \in A) & \rightarrow \\
((x,y) \in R & \leftarrow \neg(x=y \lor (y,x) \in R)) \land \\
((x,y) \in R \land (y,z) \in R & \rightarrow (x,z) \in R)
\end{align*}
\]

by auto

then show R Orders A using StrictOrder_def by simp
qed

6.2 Properties of (strict) total orders

In this section we discuss the properties of strict order relations. This continues the development contained in the standard Isabelle’s Order.thy with a view towards using the theorems translated from Metamath.

A relation orders a set iff the converse relation orders a set. Going one way we can use the the lemma tot_od_converse from the standard Isabelle’s Order.thy. The other way is a bit more complicated (note that in Isabelle for \(\text{converse}(\text{converse}(r)) = r\) one needs \(r\) to consist of ordered pairs, which does not follow from the StrictOrder definition above).

lemma cnvso: shows R Orders A \(\leftrightarrow\) converse(R) Orders A
proof
  let \(r = \text{converse}(R)\)
  assume R Orders A
  then have tot_ord(A,r) using orders_imp_tot_ord tot_ord_converse
  by simp
  then show r Orders A using tot_ord_imp_orders
  by simp
next
  let \(r = \text{converse}(R)\)
  assume r Orders A
  then have A2: \(\forall x \ y \ z. \ (x \in A \land y \in A \land z \in A) \rightarrow \)
  \(\left(\left(x,y\right) \in r \leftarrow \neg(x=y \lor (y,x) \in r)\right) \land \)
  \(\left((x,y) \in r \land (y,z) \in r \rightarrow (x,z) \in r\right)\)
  using StrictOrder_def by simp
  { fix x y z
    assume x\in A \land y\in A \land z\in A
    with A2 have
      I: \(\langle y,x \rangle \in r \leftrightarrow \neg(x=y \lor (x,y) \in r)) and\)
      II: \(\langle y,x \rangle \in r \land (z,y) \in r \rightarrow \langle z,x \rangle \in r\)
      by auto
    from I have \(\langle x,y \rangle \in R \leftrightarrow \neg(x=y \lor (y,x) \in R)\)
    by auto
    moreover from II have \(\langle x,y \rangle \in R \land (y,z) \in R \rightarrow (x,z) \in R\)
    by auto
    ultimately have \((x,y) \in R \leftrightarrow \neg(x=y \lor (y,x) \in R)) \land \)
    \((x,y) \in R \land (y,z) \in R \rightarrow (x,z) \in R) by simp
  } then have \(\forall x \ y \ z. \ (x \in A \land y \in A \land z \in A) \rightarrow\)
proof

Supremum is unique, if it exists.

lemma sup_eq: assumes A1: R Orders A and A2: x∈A and
A3: ∀y∈B. ⟨x,y⟩ ∉ R and A4: ∀y∈A. ⟨y,x⟩ ∈ R → ( ∃z∈B. ⟨y,z⟩ ∈ R)
shows
∃!x. x∈A ∧ (∀y∈B. ⟨x,y⟩ ∉ R) ∧ (∀y∈A. ⟨y,x⟩ ∈ R → ( ∃z∈B. ⟨y,z⟩ ∈ R))
proof

from A2 A3 A4 show
∃ x. x∈A ∧ (∀y∈B. ⟨x,y⟩ ∉ R) ∧ (∀y∈A. ⟨y,x⟩ ∈ R → ( ∃z∈B. ⟨y,z⟩ ∈ R))

by auto

next fix x_1 x_2

assume A5:
x_1 ∈ A ∧ (∀y∈B. ⟨x_1,y⟩ ∉ R) ∧ (∀y∈A. ⟨y,x_1⟩ ∈ R → ( ∃z∈B. ⟨y,z⟩ ∈ R))
x_2 ∈ A ∧ (∀y∈B. ⟨x_2,y⟩ ∉ R) ∧ (∀y∈A. ⟨y,x_2⟩ ∈ R → ( ∃z∈B. ⟨y,z⟩ ∈ R))

from A1 have linear(A,R) using orders_imp_tot_ord tot_ord_def
by simp

then have ∀x∈A. ∀y∈A. ⟨x,y⟩ ∈ R ∨ x=y ∨ ⟨y,x⟩ ∈ R

unfolding linear_def by blast

with A5 have ⟨x_1,x_2⟩ ∈ R ∨ x_1=x_2 ∨ ⟨x_2,x_1⟩ ∈ R by blast

moreover

{ assume ⟨x_1,x_2⟩ ∈ R
  with A5 obtain z where z∈B and ⟨x_1,z⟩ ∈ R by auto
  with A5 have False by auto }

moreover

{ assume ⟨x_2,x_1⟩ ∈ R
  with A5 obtain z where z∈B and ⟨x_2,z⟩ ∈ R by auto
  with A5 have False by auto }

ultimately show x_1 = x_2 by auto

qed

Supremum has expected properties if it exists.

lemma sup_props: assumes A1: R Orders A and
A2: ∃x∈A. (∀y∈B. ⟨x,y⟩ ∉ R) ∧ (∀y∈A. ⟨y,x⟩ ∈ R → ( ∃z∈B. ⟨y,z⟩ ∈ R))
shows

Sup(B,A,R) ∈ A
∀y∈B. ⟨Sup(B,A,R),y⟩ ∉ R
∀y∈A. ⟨y,Sup(B,A,R)⟩ ∈ R → ( ∃z∈B. ⟨y,z⟩ ∈ R )
proof -

let S = {x∈A. (∀y∈B. ⟨x,y⟩ ∉ R) ∧ (∀y∈A. ⟨y,x⟩ ∈ R → ( ∃z∈B. ⟨y,z⟩ ∈ R))

qed
from $A_2$ obtain $x$ where $x \in A$ and $(\forall y \in B. \langle x, y \rangle \notin R) \land (\forall y \in A. \langle y, x \rangle \in R \rightarrow (\exists z \in B. \langle y, z \rangle \in R))$

by auto

with $A_1$ have $I$:

$\exists! x. x \in A \land (\forall y \in B. \langle x, y \rangle \notin R) \land (\forall y \in A. \langle y, x \rangle \in R \rightarrow (\exists z \in B. \langle y, z \rangle \in R))$

by auto with $A_1$

have $I$:

$\exists! x. x \in A \land (\forall y \in B. \langle x, y \rangle \notin R) \land (\forall y \in A. \langle y, x \rangle \in R \rightarrow (\exists z \in B. \langle y, z \rangle \in R))$

by simp

then have $(\bigcup S) \in A$ by (rule ZF1_1_L9)

then show $\text{Sup}(B, A, R) \in A$ using $\text{Sup_def}$ by simp

from $I$ have $II$:

$(\forall y \in B. \langle \bigcup S, y \rangle \notin R) \land (\forall y \in A. \langle y, \bigcup S \rangle \in R \rightarrow (\exists z \in B. \langle y, z \rangle \in R))$

by (rule ZF1_1_L9)

hence $\forall y \in B. \langle \bigcup S, y \rangle \notin R$ by blast

moreover have $III$: $(\bigcup S) = \text{Sup}(B, A, R)$ using $\text{Sup_def}$ by simp

ultimately show $\forall y \in B. \langle \text{Sup}(B, A, R), y \rangle \in R \rightarrow (\exists z \in B. \langle y, z \rangle \in R)$

by blast

{ fix $y$ assume $A3$: $y \in A$ and $(y, \text{Sup}(B, A, R)) \in R$

with $III$ have $\langle y, \bigcup S \rangle \in R$ by simp

with $IV A3$ have $\exists z \in B. \langle y, z \rangle \in R$ by blast

} thus $\forall y \in A. \langle y, \text{Sup}(B, A, R) \rangle \in R \rightarrow (\exists z \in B. \langle y, z \rangle \in R)$

by simp

qed

Elements greater or equal than any element of $B$ are greater or equal than supremum of $B$.

lemma supnub: assumes $A1$: $R$ Orders $A$ and $A2$:

$\exists x \in A. (\forall y \in B. \langle x, y \rangle \notin R) \land (\forall y \in A. \langle y, x \rangle \in R \rightarrow (\exists z \in B. \langle y, z \rangle \in R))$

and $A3$: $c \in A$ and $A4$: $\forall z \in B. \langle c, z \rangle \notin R$

shows $\langle c, \text{Sup}(B, A, R) \rangle \notin R$

proof -

from $A1 A2$ have

$\forall y \in A. \langle y, \text{Sup}(B, A, R) \rangle \in R \rightarrow (\exists z \in B. \langle y, z \rangle \in R)$

by (rule sup_props)

with $A3 A4$ show $\langle c, \text{Sup}(B, A, R) \rangle \notin R$ by auto

qed

end

7 Even more on order relations

theory Order_ZF_1a imports Order_ZF

begin

This theory is a continuation of Order_ZF and talks about maximuma and minimum of a set, supremum and infimum and strict (not reflexive) versions
of order relations.

7.1 Maximum and minimum of a set

In this section we show that maximum and minimum are unique if they exist. We also show that union of sets that have maxima (minima) has a maximum (minimum). We also show that singletons have maximum and minimum. All this allows to show (in Finite\_ZF) that every finite set has well-defined maximum and minimum.

A somewhat technical fact that allows to reduce the number of premises in some theorems: the assumption that a set has a maximum implies that it is not empty.

**lemma** set\_max\_not\_empty: assumes HasAmaximum(r,A) shows A≠0
  using assms unfolding HasAmaximum_def by auto

If a set has a maximum implies that it is not empty.

**lemma** set\_min\_not\_empty: assumes HasAminimum(r,A) shows A≠0
  using assms unfolding HasAminimum_def by auto

If a set has a supremum then it cannot be empty. We are probably using the fact that ∩∅ = ∅, which makes me a bit anxious as this I think is just a convention.

**lemma** set\_sup\_not\_empty: assumes HasAsupremum(r,A) shows A≠0
proof -
  from assms have HasAminimum(r,∩a∈A. r\{a\}) unfolding HasAsupremum_def
    by simp
  then have (∩a∈A. r\{a\}) ≠ 0 using set\_min\_not\_empty by simp
  then obtain x where x ∈ (∩y∈A. r\{y\}) by blast
  thus thesis by auto
qed

If a set has an infimum then it cannot be empty.

**lemma** set\_inf\_not\_empty: assumes HasAnInfimum(r,A) shows A≠0
proof -
  from assms have HasAmaximum(r,∩a∈A. r-{a}) unfolding HasAnInfimum_def
    by simp
  then have (∩a∈A. r-{a}) ≠ 0 using set\_max\_not\_empty by simp
  then obtain x where x ∈ (∩y∈A. r-{y}) by blast
  thus thesis by auto
qed

For antisymmetric relations maximum of a set is unique if it exists.

**lemma** Order\_ZF\_4\_L1: assumes A1: antisym(r) and A2: HasAmaximum(r,A)
  shows ∃!M. M∈A ∧ (∀x∈A. ⟨x,M⟩ ∈ r)
proof
from A2 show \( \exists M. M \in A \land (\forall x \in A. \langle x, M \rangle \in r) \)
using HasAmaximum_def by auto
fix M1 M2 assume
  A2: M1 \in A \land (\forall x \in A. \langle x, M1 \rangle \in r) M2 \in A \land (\forall x \in A. \langle x, M2 \rangle \in r)
then have \( \langle M1, M2 \rangle \in r \langle M2, M1 \rangle \in r \) by auto
with A1 show M1=M2 by (rule Fol1_L4)
qed

For antisymmetric relations minimum of a set is unique if it exists.

lemma Order_ZF_4_L2: assumes A1: antisym(r) and A2: HasAminimum(r,A)
shows \( \exists! m. m \in A \land (\forall x \in A. \langle m, x \rangle \in r) \)
proof
  from A2 show \( \exists m. m \in A \land (\forall x \in A. \langle m, x \rangle \in r) \)
  using HasAminimum_def by auto
fix m1 m2 assume
  A2: m1 \in A \land (\forall x \in A. \langle m1, x \rangle \in r) m2 \in A \land (\forall x \in A. \langle m2, x \rangle \in r)
then have \( \langle m1, m2 \rangle \in r \langle m2, m1 \rangle \in r \) by auto
with A1 show m1=m2 by (rule Fol1_L4)
qed

Maximum of a set has desired properties.

lemma Order_ZF_4_L3: assumes A1: antisym(r) and A2: HasAmaximum(r,A)
shows Maximum(r,A) \in A \land (\forall x \in A. \langle x, Maximum(r,A) \rangle \in r)
proof -
  let Max = THE M. M \in A \land (\forall x \in A. \langle x, M \rangle \in r)
  from A1 A2 have \( \exists! M. M \in A \land (\forall x \in A. \langle x, M \rangle \in r) \)
    by (rule Order_ZF_4_L1)
  then have Max \in A \land (\forall x \in A. \langle x, Max \rangle \in r)
    by (rule theI)
  then show Maximum(r,A) \in A \land (\forall x \in A. \langle x, Maximum(r,A) \rangle \in r)
    using Maximum_def by auto
qed

Minimum of a set has desired properties.

lemma Order_ZF_4_L4: assumes A1: antisym(r) and A2: HasAminimum(r,A)
shows Minimum(r,A) \in A \land (\forall x \in A. \langle Minimum(r,A), x \rangle \in r)
proof -
  let Min = THE m. m \in A \land (\forall x \in A. \langle m, x \rangle \in r)
  from A1 A2 have \( \exists! m. m \in A \land (\forall x \in A. \langle m, x \rangle \in r) \)
    by (rule Order_ZF_4_L2)
  then have Min \in A \land (\forall x \in A. \langle Min, x \rangle \in r)
    by (rule theI)
  then show Minimum(r,A) \in A \land (\forall x \in A. \langle Minimum(r,A), x \rangle \in r)
    using Minimum_def by auto
qed

For total and transitive relations a union a of two sets that have maxima has a maximum.

lemma Order_ZF_4_L5:
assumes A1: r {is total on} (A∪B) and A2: trans(r)
and A3: HasAmaximum(r,A) HasAmaximum(r,B)
shows HasAmaximum(r,A∪B)

proof -
from A3 obtain M K where
  D1: M ∈ A ∧ (∀x∈A. ⟨x,M⟩ ∈ r) K ∈ B ∧ (∀x∈B. ⟨x,K⟩ ∈ r)
  using HasAmaximum_def by auto
let L = GreaterOf(r,M,K)
from D1 have T1: M ∈ A ∪ B K ∈ A ∪ B
  ∀x∈A. ⟨x,M⟩ ∈ r ∀x∈B. ⟨x,K⟩ ∈ r
  by auto
with A1 A2 have ∀x∈A∪B. ⟨x,L⟩ ∈ r by (rule Order_ZF_3_L2B)
moreover from T1 have L ∈ A∪B using GreaterOf_def IsTotal_def
  by simp
ultimately show HasAmaximum(r,A∪B) using HasAmaximum_def by auto
qed

For total and transitive relations A union a of two sets that have minima
has a minimum.

lemma Order_ZF_4_L6:
  assumes A1: r {is total on} (A∪B) and A2: trans(r)
  and A3: HasAminimum(r,A) HasAminimum(r,B)
  shows HasAminimum(r,A∪B)
proof -
from A3 obtain m k where
  D1: m ∈ A ∧ (∀x∈A. ⟨m,x⟩ ∈ r) k ∈ B ∧ (∀x∈B. ⟨k,x⟩ ∈ r)
  using HasAminimum_def by auto
let l = SmallerOf(r,m,k)
from D1 have T1: m ∈ A ∪ B k ∈ A ∪ B
  ∀x∈A. ⟨m,x⟩ ∈ r ∀x∈B. ⟨k,x⟩ ∈ r
  by auto
with A1 A2 have ∀x∈A∪B. ⟨l,x⟩ ∈ r by (rule Order_ZF_3_L5B)
moreover from T1 have l ∈ A∪B using SmallerOf_def IsTotal_def
  by simp
ultimately show HasAminimum(r,A∪B) using HasAminimum_def by auto
qed

Set that has a maximum is bounded above.

lemma Order_ZF_4_L7:
  assumes HasAmaximum(r,A)
  shows IsBoundedAbove(A,r)
using assms HasAmaximum_def IsBoundedAbove_def by auto

Set that has a minimum is bounded below.

lemma Order_ZF_4_L8A:
  assumes HasAminimum(r,A)
  shows IsBoundedBelow(A,r)
using assms HasAminimum_def IsBoundedBelow_def by auto
For reflexive relations singletons have a minimum and maximum.

**lemma** Order_ZF_4_L8: assumes refl(X,r) and a\in X
  shows HasAmaximum(r,\{a\}) HasAminimum(r,\{a\})
  using assms refl_def HasAmaximum_def HasAminimum_def by auto

For total and transitive relations if we add an element to a set that has a maximum, the set still has a maximum.

**lemma** Order_ZF_4_L9:
  assumes A1: r {is total on} X and A2: trans(r)
  and A3: A\subseteq X and A4: a\in X and A5: HasAmaximum(r,A)
  shows HasAmaximum(r,A\cup\{a\})
  proof -
  from A3 A4 have A\cup\{a\} \subseteq X by auto
  with A1 have r {is total on} (A\cup\{a\})
    using Order_ZF_1_L4 by blast
  moreover from A1 A2 A4 A5 have
    trans(r) HasAmaximum(r,A) by auto
  moreover from A1 A4 have HasAmaximum(r,\{a\})
    using total_is_refl Order_ZF_4_L8 by blast
  ultimately show HasAmaximum(r,A\cup\{a\}) by (rule Order_ZF_4_L5)
qed

For total and transitive relations if we add an element to a set that has a minimum, the set still has a minimum.

**lemma** Order_ZF_4_L10:
  assumes A1: r {is total on} X and A2: trans(r)
  and A3: r \subseteq X\times X
  and A4: \forall A. IsBounded(A,r) \land \ A\neq 0 \longrightarrow HasAminimum(r,A)
  shows HasAminimum(r,A\cup\{a\})
  proof -
  from A3 A4 have A\cup\{a\} \subseteq X by auto
  with A1 have r {is total on} (A\cup\{a\})
    using Order_ZF_1_L4 by blast
  moreover from A1 A2 A4 A5 have
    trans(r) HasAminimum(r,A) by auto
  moreover from A1 A4 have HasAminimum(r,\{a\})
    using total_is_refl Order_ZF_4_L8 by blast
  ultimately show HasAminimum(r,A\cup\{a\}) by (rule Order_ZF_4_L6)
qed

If the order relation has a property that every nonempty bounded set attains a minimum (for example integers are like that), then every nonempty set bounded below attains a minimum.

**lemma** Order_ZF_4_L11:
  assumes A1: r {is total on} X and 
  A2: trans(r) and 
  A3: r \subseteq X\times X 
  A4: \forall A. IsBounded(A,r) \land \ A\neq 0 \longrightarrow HasAminimum(r,A) and 
  A5: B\neq 0 and A6: IsBoundedBelow(B,r)
shows HasAminimum(r,B)

proof -
from A5 obtain b where T: b ∈ B by auto
let L = {x ∈ B. ⟨x, b⟩ ∈ r}
from A3 A6 T have T1: b ∈ X using Order_ZF_3_L1B by blast
with A1 T have T2: b ∈ L
  using total_is_refl refl_def by simp
then have L ≠ 0 by auto
moreover have IsBounded(L,r)
proof -
  have L ⊆ B by auto
  with A6 have IsBoundedBelow(L,r)
    using Order_ZF_3_L12 by simp
  moreover have IsBoundedAbove(L,r)
    by (rule Order_ZF_3_L15)
  ultimately have IsBoundedAbove(L,r) ∧ IsBoundedBelow(L,r)
    by blast
  then show IsBounded(L,r) using IsBounded_def by simp
qed
ultimately have IsBounded(L,r) ∧ L ≠ 0 by blast
with A4 have HasAminimum(r,L) by simp
then obtain m where I: m ∈ L and II: ∀ x ∈ L. ⟨m, x⟩ ∈ r
  using HasAminimum_def by auto
from I have m ∈ B by simp
moreover have ∀ x ∈ B. ⟨m, x⟩ ∈ r
proof
  fix x assume A7: x ∈ B
  from A3 A6 have B ⊆ X using Order_ZF_3_L1B by blast
  with A1 A7 T1 have x ∈ L ∪ {x ∈ B. ⟨b, x⟩ ∈ r}
    using Order_ZF_1_L5 by simp
  then have x ∈ L ∨ ⟨b, x⟩ ∈ r by auto
  moreover
  { assume x ∈ L
    with II have ⟨m, x⟩ ∈ r by simp }
  moreover
  { assume ⟨b, x⟩ ∈ r
    with A2 III have trans(r) and ⟨m, b⟩ ∈ r ∧ ⟨b, x⟩ ∈ r
      by auto
    then have ⟨m, x⟩ ∈ r by (rule Fol1_L3) }
  ultimately show ⟨m, x⟩ ∈ r by auto
qed
ultimately show HasAminimum(r,B) using HasAminimum_def by auto
qed

A dual to Order_ZF_4_L11: If the order relation has a property that every nonempty bounded set attains a maximum (for example integers are like
that), then every nonempty set bounded above attains a maximum.

**lemma** OrderZF4_L11A:
assumes A1: r (is total on) X and
A2: trans(r) and
A3: r ⊆ X×X and
A4: ∀A. IsBounded(A,r) ∧ A≠0 → HasAmaximum(r,A) and
A5: B≠0 and A6: IsBoundedAbove(B,r)
shows HasAmaximum(r,B)

**proof**
from A5 obtain b where T: b ∈ B by auto
let U = {x ∈ B. ⟨b,x⟩ ∈ r}
from A3 A6 T have T1: b ∈ X using OrderZF_3_L1A by blast
with A1 T have T2: b ∈ U using total_is_refl refl_def by simp
then have U ≠ 0 by auto
moreover have IsBounded(U,r)
proof -
  have U ⊆ B by auto
  with A6 have IsBoundedAbove(U,r)
  using OrderZF_3_L13 by blast
  moreover have IsBoundedBelow(U,r)
  using IsBoundedBelow_def by auto
  ultimately have IsBoundedAbove(U,r) ∧ IsBoundedBelow(U,r)
  by blast
  then show IsBounded(U,r) using IsBounded_def
  by simp
qed
ultimately have IsBounded(U,r) ∧ U ≠ 0 by blast
with A4 have HasAmaximum(r,U) by simp
then obtain m where I: m∈U and II: ∀x∈U. ⟨x,m⟩ ∈ r
  using HasAmaximum_def by auto
then have III: ⟨b,m⟩ ∈ r by simp
from I have m∈B by simp
moreover have ∀x∈B. ⟨x,m⟩ ∈ r
proof
  fix x assume A7: x∈B
  from A3 A6 have B⊆X using OrderZF_3_L1A by blast
  with A1 A7 T1 have x ∈ {x∈B. ⟨x,b⟩ ∈ r} ∪ U
  using OrderZF_1_L5 by simp
  then have x∈U ∨ ⟨x,b⟩ ∈ r by auto
  moreover
  { assume x∈U
    with II have ⟨x,m⟩ ∈ r by simp }
  moreover
  { assume ⟨x,b⟩ ∈ r
    with A2 III have trans(r) and ⟨x,b⟩ ∈ r ∧ ⟨b,m⟩ ∈ r
    by auto
    then have ⟨x,m⟩ ∈ r by (rule Fol1_L3) }
ultimately show ⟨x,m⟩ ∈ r by auto
If a set has a minimum and $L$ is less or equal than all elements of the set, then $L$ is less or equal than the minimum.

**lemma** Order_ZF_4_L12:
- assumes antisym(r) and HasAminimum(r,A) and $\forall a \in A. \langle L, a \rangle \in r$
- shows $\langle L, \text{Minimum}(r,A) \rangle \in r$
- using assms Order_ZF_4_L4 by simp

If a set has a maximum and all its elements are less or equal than $M$, then the maximum of the set is less or equal than $M$.

**lemma** Order_ZF_4_L13:
- assumes antisym(r) and HasAmaximum(r,A) and $\forall a \in A. \langle a, M \rangle \in r$
- shows $\langle \text{Maximum}(r,A), M \rangle \in r$
- using assms Order_ZF_4_L3 by simp

If an element belongs to a set and is greater or equal than all elements of that set, then it is the maximum of that set.

**lemma** Order_ZF_4_L14:
- assumes A1: antisym(r) and A2: $M \in A$ and A3: $\forall a \in A. \langle a, M \rangle \in r$
- shows $\text{Maximum}(r,A) = M$
- proof -
  from A2 A3 have I: HasAmaximum(r,A) using HasAmaximum_def by auto
  with A1 have $\exists ! M. M \in A \land (\forall x \in A. \langle x, M \rangle \in r)$
  using Order_ZF_4_L1 by simp
  moreover from A2 A3 have $M \in A \land (\forall x \in A. \langle x, M \rangle \in r)$ by simp
  moreover from A1 I have
  $\text{Maximum}(r,A) \in A \land (\forall x \in A. \langle x, \text{Maximum}(r,A) \rangle \in r)$
  using Order_ZF_4_L3 by simp
  ultimately show $\text{Maximum}(r,A) = M$ by auto
- qed

If an element belongs to a set and is less or equal than all elements of that set, then it is the minimum of that set.

**lemma** Order_ZF_4_L15:
- assumes A1: antisym(r) and A2: $m \in A$ and A3: $\forall a \in A. \langle m, a \rangle \in r$
- shows $\text{Minimum}(r,A) = m$
- proof -
  from A2 A3 have I: HasAminimum(r,A) using HasAminimum_def by auto
  with A1 have $\exists ! m. m \in A \land (\forall x \in A. \langle m, x \rangle \in r)$
  using Order_ZF_4_L2 by simp
moreover from A2 A3 have \( m \in A \land (\forall x \in A. \langle m, x \rangle \in r) \) by simp
moreover from A1 I have
Minimum(r,A) \in A \land (\forall x \in A. \langle \text{Minimum}(r,A), x \rangle \in r)
using Order_ZF_4_L4 by simp
ultimately show \( \text{Minimum}(r,A) = m \) by auto
qed

If a set does not have a maximum, then for any its element we can find one
that is (strictly) greater.

**lemma Order_ZF_4_L16:**
assumes A1: \( \text{antisym}(r) \) and
A2: \( r \text{ is total on } X \) and
A3: \( A \subseteq X \) and
A4: \( \neg \text{HasAmaximum}(r,A) \) and
A5: \( x \in A \)
shows \( \exists y \in A. \langle x, y \rangle \in r \land y \neq x \)

proof -
{ assume A6: \( \forall y \in A. \langle x, y \rangle \notin r \lor y=x \)
have \( \forall y \in A. \langle y, x \rangle \in r \)
proof
fix y assume A7: \( y \in A \)
with A6 have \( \langle x, y \rangle \notin r \lor y=x \) by simp
with A2 A3 A5 A7 show \( \langle y, x \rangle \in r \)
using IsTotal_def Order_ZF_1_L1 by auto
qed
with A5 have \( \exists x \in A. \forall y \in A. \langle y, x \rangle \in r \)
by auto
with A4 have False using HasAmaximum_def by simp
} then show \( \exists y \in A. \langle x, y \rangle \in r \land y \neq x \) by auto
qed

### 7.2 Supremum and Infimum

In this section we consider the notions of supremum and infimum a set.

Elements of the set of upper bounds are indeed upper bounds. Isabelle also
thinks it is obvious.

**lemma Order_ZF_5_L1:**
assumes u \( \in (\bigcap a \in A. r\{a\}) \) and a\( \in A \)
shows \( \langle a, u \rangle \in r \)
using asms by auto

Elements of the set of lower bounds are indeed lower bounds. Isabelle also
thinks it is obvious.

**lemma Order_ZF_5_L2:**
assumes l \( \in (\bigcap a \in A. r-\{a\}) \) and a\( \in A \)
shows \( \langle l, a \rangle \in r \)
using asms by auto

If the set of upper bounds has a minimum, then the supremum is less or equal
than any upper bound. We can probably do away with the assumption that
A is not empty, (ab)using the fact that intersection over an empty family is defined in Isabelle to be empty. This lemma is obsolete and will be removed in the future. Use sup_leq_up_bnd instead.

lemma Order_ZF_5_L3: assumes A1: antisym(r) and A2: A̸=0 and A3: HasAminimum(r,⋂a∈A. r{a}) and A4: ∀a∈A. ⟨a,u⟩ ∈ r
shows ⟨Supremum(r,A),u⟩ ∈ r

proof -
  let U = ⋂a∈A. r{a}
  from A4 have ∀a∈A. u ∈ r{a} using image_singleton_iff by simp
  with A2 have u∈U by auto
  with A1 A3 show ⟨Supremum(r,A),u⟩ ∈ r
    using Order_ZF_4_L4 Supremum_def by simp
qed

Supremum is less or equal than any upper bound.

lemma sup_leq_up_bnd: assumes antisym(r) HasAsupremum(r,A) ∀a∈A. ⟨a,u⟩ ∈ r
shows ⟨Supremum(r,A),u⟩ ∈ r

proof -
  let U = ⋂a∈A. r{a}
  from assms(3) have ∀a∈A. u ∈ r{a} using image_singleton_iff by simp
  with assms(2) have u∈U using set_sup_not_empty by auto
  with assms(1,2) show ⟨Supremum(r,A),u⟩ ∈ r
    unfolding HasAsupremum_def Supremum_def using Order_ZF_4_L4 by simp
qed

Infimum is greater or equal than any lower bound. This lemma is obsolete and will be removed. Use inf_geq_lo_bnd instead.

lemma Order_ZF_5_L4: assumes A1: antisym(r) and A2: A̸=0 and A3: HasAmaximum(r,⋂a∈A. r-{a}) and A4: ∀a∈A. ⟨l,a⟩ ∈ r
shows ⟨l,Infimum(r,A)⟩ ∈ r

proof -
  let L = ⋂a∈A. r-{a}
  from A4 have ∀a∈A. l ∈ r-{a} using image_singleton_iff by simp
  with A2 have l∈L by auto
  with A1 A3 show ⟨l,Infimum(r,A)⟩ ∈ r
    using Order_ZF_4_L3 Infimum_def by simp
qed

Infimum is greater or equal than any upper bound.

lemma inf_geq_lo_bnd: assumes antisym(r) HasAnInfimum(r,A) ∀a∈A. ⟨u,a⟩ ∈ r
shows ⟨u,Infimum(r,A)⟩ ∈ r

proof -
let $U = \bigcap_{a \in A} r\{a\}$
from assms(3) have $\forall a \in A. \ u \in r\{a\}$ using vimagesingletoniff by simp
with assms(2) have $u \in U$ using setinfnotempty by auto
with assms(1,2) show $\langle u, \text{inimum}(r, A) \rangle \in r$
unfolding HasAnInfimum_def Infimum_def using Order_ZF_4_L3 by simp
qed

If $z$ is an upper bound for $A$ and is less or equal than any other upper bound, then $z$ is the supremum of $A$.

lemma Order_ZF_5_L5: assumes A1: antisym(r) and A2: $A \neq 0$ and
A3: $\forall x \in A. \langle x, z \rangle \in r$
A4: $\forall y. \ (\forall x \in A. \langle x, y \rangle \in r) \rightarrow \langle z, y \rangle \in r$
shows
HasAminimum$(r, \bigcap_{a \in A} r\{a\})$
$z = \text{Supremum}(r, A)$
proof -
let $B = \bigcap_{a \in A} r\{a\}$
from assms(2,3,4) have I: $z \in B \ \forall y \in B. \langle z, y \rangle \in r$
by auto
then show HasAminimum$(r, \bigcap_{a \in A} r\{a\})$
using HasAminimum_def by auto
from A1 I show $z = \text{Supremum}(r, A)$
using Order_ZF_4_L15 Supremum_def by simp
qed

The dual theorem to Order_ZF_5_L5: if $z$ is an lower bound for $A$ and is greater or equal than any other lower bound, then $z$ is the infimum of $A$.

lemma inf_glb:
assumes antisym(r) refl(X,r) $z \in X$
shows HasAmaximum$(r, \bigcap_{a \in A} r\{-a\})$
$z = \text{Infimum}(r, A)$
proof -
let $B = \bigcap_{a \in A} r\{-a\}$
from assms(2,3,4) have I: $z \in B \ \forall y \in B. \langle y, z \rangle \in r$
by auto
then show HasAmaximum$(r, \bigcap_{a \in A} r\{-a\})$
unfolding HasAmaximum_def by auto
from assms(1) I show $z = \text{Infimum}(r, A)$
using Order_ZF_4_L14 Infimum_def by simp
qed

Supremum and infimum of a singleton is the element.

lemma sup_inf_singl: assumes antisym(r) refl(X,r) $z \in X$
shows
HasAsupremum$(r, \{z\})$ Supremum$(r, \{z\}) = z$ and
HasAnInfimum(r,{z}) \quad \text{Infimum}(r,{z}) = z

\text{proof} -
\begin{align*}
\text{from assms show Supremum}(r,{z}) = z \quad \text{and Infimum}(r,{z}) = z \\
\text{using inf_glb \quad Order_ZF_5_L5 unfolding refl_def by auto}
\end{align*}

\text{from assms show \quad HasAsupremum}(r,{z})
\begin{align*}
\text{using \quad Order_ZF_5_L5 unfolding HasAsupremum_def refl_def by blast}
\end{align*}

qed

\text{If a set has a maximum, then the maximum is the supremum. This lemma is obsolete, use max_is_sup instead.}

\text{lemma Order_ZF_5_L6:}
\begin{align*}
\text{assumes A1: \quad antisym(r) and A2: A\neq 0 and} \\
A3: \text{HasAmaximum}(r,A) \\
\text{shows} \\
\text{HasAminimum}(r,\bigcap a\in A. r\{a\}) \\
\text{Maximum}(r,A) = \text{Supremum}(r,A)
\end{align*}

\text{proof -}
\begin{align*}
\text{let M = Maximum(r,A)} \\
\text{from A1 A3 have I: M \in A and II: } \forall x\in A. \langle x,M \rangle \in r \\
\text{using Order_ZF_4_L3 by auto}
\end{align*}

\text{from I have III: } \forall y. \langle \forall x\in A. \langle x,y \rangle \in r \rangle \longrightarrow \langle M,y \rangle \in r
\begin{align*}
\text{by simp}
\end{align*}

\text{with A1 A2 II show HasAminimum(r,\bigcap a\in A. r\{a\})}
\begin{align*}
\text{by (rule Order_ZF_5_L5)}
\end{align*}

\text{from A1 A2 II III show M = Supremum(r,A)}
\begin{align*}
\text{by (rule Order_ZF_5_L5)}
\end{align*}

qed

\text{Another version of Order_ZF_5_L6 that: if a set has a maximum then it has a supremum and the maximum is the supremum.}

\text{lemma max_is_sup:}
\begin{align*}
\text{assumes antisym(r) A\neq 0 HasAmaximum(r,A) and Maximum(r,A) = Supremum(r,A) shows HasAsupremum(r,A) and Maximum(r,A) = Supremum(r,A)}
\end{align*}

\text{proof -}
\begin{align*}
\text{let M = Maximum(r,A)} \\
\text{from assms(1,3) have M \in A and I: } \forall x\in A. \langle x,M \rangle \in r \quad \text{using Order_ZF_4_L3 by auto}
\end{align*}

\text{with assms(1,2) have HasAminimum(r,\bigcap a\in A. r\{a\}) using Order_ZF_5_L5(1)}
\begin{align*}
\text{by blast}
\end{align*}

\text{then show HasAsupremum(r,A) unfolding HasAsupremum_def by simp}
\begin{align*}
\text{from assms(1,2) } M \in A \text{ I show M = Supremum(r,A) using Order_ZF_5_L5(2)}
\end{align*}

\text{by blast}

qed

\text{Minimum is the infimum if it exists.}
lemma min_is_inf: assumes antisym(r) A ≠ 0 HasAminimum(r,A)
shows HasAnInfimum(r,A) and Minimum(r,A) = Infimum(r,A)
proof -
let M = Minimum(r,A)
from assms(1,3) have M ∈ A and I: ∀ x ∈ A. ⟨M, x⟩ ∈ r using Order_ZF_4_L4
by auto
with assms(1,2) have HasAmaximum(r, ∩ a ∈ A. r-{a}) using inf_glb(1) by blast
then show HasAnInfimum(r,A) unfolding HasAnInfimum_def by simp
qed

For reflexive and total relations two-element set has a minimum and a maximum.

lemma min_max_two_el: assumes r {is total on} X x ∈ X y ∈ X
shows HasAminimum(r,{x,y}) and HasAmaximum(r,{x,y})
using assms unfolding IsTotal_def HasAminimum_def HasAmaximum_def by auto

For antisymmetric, reflexive and total relations two-element set has a supremum and infimum.

lemma inf_sup_two_el: assumes antisym(r) r {is total on} X x ∈ X y ∈ X
shows HasAnInfimum(r,{x,y}) Minimum(r,{x,y}) = Infimum(r,{x,y})
HasAsupremum(r,{x,y}) Maximum(r,{x,y}) = Supremum(r,{x,y})
using assms min_max_two_el max_is_sup min_is_inf by auto

A sufficient condition for the supremum to be in the space.

lemma sup_in_space:
assumes r ⊆ X × X antisym(r) HasAminimum(r, ∩ a ∈ A. r(a))
shows Supremum(r,A) ∈ X and ∀ x ∈ A. ⟨x, Supremum(r,A)⟩ ∈ r
proof -
from assms(3) have A ≠ 0 using set_sup_not_empty unfolding HasAsupremum_def by simp
then obtain a where a ∈ A by auto
with assms(1,2,3) show Supremum(r,A) ∈ X unfolding Supremum_def using Order_ZF_4_L4 Order_ZF_5_L1 by blast
from assms(2,3) show ∀ x ∈ A. ⟨x, Supremum(r,A)⟩ ∈ r unfolding Supremum_def using Order_ZF_4_L4 by blast
qed

A sufficient condition for the infimum to be in the space.

lemma inf_in_space: 
assumes r ⊆ X × X antisym(r) HasAmaximum(r, ∩ a ∈ A. r-{a})
shows $\infimum(r,A) \in X$ and $\forall x \in A. \langle \infimum(r,A), x \rangle \in r$
proof -
  from assms(3) have $A \neq 0$ using set_inf_not_empty unfolding HasAnInfimum_def
  by simp
  then obtain $a$ where $a \in A$ by auto
  with assms(1,2,3) show $\infimum(r,A) \in X$ unfolding Infimum_def
  using Order_ZF_4_L3 Order_ZF_5_L1 by blast
  from assms(2,3) show $\forall x \in A. \langle \infimum(r,A), x \rangle \in r$ unfolding Infimum_def
  using Order_ZF_4_L3 by blast
qed

Properties of supremum of a set for complete relations.

lemma Order_ZF_5_L7:
  assumes $A1: r \subseteq X \times X$ and $A2: \text{antisym}(r)$ and $A3: r \{\text{is complete}\}$ and $A4: A \neq 0$ and $A5: \exists x \in X. \forall y \in A. \langle y, x \rangle \in r$
  shows $\supremum(r,A) \in X$ and $\forall x \in A. \langle x, \supremum(r,A) \rangle \in r$
proof -
  from $A3$ $A4$ $A5$ have $\text{HasAminimum}(r, \bigcap a \in A. r \{a\})$
  unfolding IsBoundedAbove_def IsComplete_def by blast
  with $A1$ $A2$ show $\supremum(r,A) \in X$ and $\forall x \in A. \langle x, \supremum(r,A) \rangle \in r$
  using sup_in_space by auto
qed

Infimum of the set of infima of a collection of sets is infimum of the union.

lemma inf_inf:
  assumes $r \subseteq X \times X$ antisym($r$) trans($r$)
  $\forall T \in T . \text{HasAnInfimum}(r,T)$
  $\text{HasAnInfimum}(r, \{ \infimum(r,T) . T \in T \})$
  shows $\text{HasAnInfimum}(r, \bigcup T)$ and $\infimum(r, \{ \infimum(r,T) . T \in T \}) = \infimum(r, \bigcup T)$
proof -
  let $i = \infimum(r, \{ \infimum(r,T) . T \in T \})$
  note assms(2)
  moreover from assms(4,5) have $\bigcup T \neq 0$ using set_inf_not_empty by blast
  moreover have $\forall T \in T . \forall t \in T. \langle i, t \rangle \in r$
  proof -
    $\{$ fix $T$ t assume $T \in T$ $t \in T$
    with assms(1,2,4) have $\langle \infimum(r,T), t \rangle \in r$
    unfolding HasAnInfimum_def using inf_in_space(2) by blast
    moreover from assms(1,2,5) $T \in T$ have $\langle i, \infimum(r,T) \rangle \in r$
    unfolding HasAnInfimum_def using inf_in_space(2) by blast
    moreover note assms(3)
    ultimately have $\langle i, t \rangle \in r$ unfolding trans_def by blast
  $\}$ thus thesis by simp
qed
hence I: \( \forall t \in \bigcup T. \ (i,t) \in r \) by auto
moreover have J: \( \forall y. \ (\forall x \in \bigcup T. \ (y,x) \in r) \implies (y,i) \in r \)
proof -
  { fix y x assume A: \( \forall x \in \bigcup T. \ (y,x) \in r \)
    with assms(2,4) have \( \forall a \in \{\infimum(r,T).T \in T\}. \ (y,a) \in r \) using inf_geq_lo_bnd
      by simp
    with assms(2,5) have \( (y,i) \in r \) by (rule inf_geq_lo_bnd)
  } thus thesis by simp
qed
ultimately have HasAmaximum(r, \( \bigcap a \in \bigcup T. \ r-a \)) by (rule inf_glb)
then show HasAnInfimum(r, \( \bigcup T \)) unfolding HasAnInfimum_def by simp
from assms(2) \( \bigcup T \neq 0 \) I J show i = Infimum(r, \( \bigcup T \)) by (rule inf_glb) qed

Supremum of the set of suprema of a collection of sets is supremum of the union.

lemma sup_sup:
  assumes
    r \subseteq X \times X antisym(r) trans(r)
    \( \forall T \in T \). HasAsupremum(r,T)
    HasAsupremum(r,\{Supremum(r,T).T \in T\})
  shows
    HasAsupremum(r, \( \bigcup T \)) and Supremum(r, \( \{\text{Supremum}(r,T).T \in T\} \)) = Supremum(r, \( \bigcup T \))
proof -
  let s = Supremum(r, \( \{\text{Supremum}(r,T).T \in T\} \))
  note assms(2)
  moreover from assms(4,5) have \( \bigcup T \neq 0 \) using set_sup_not_empty by blast
  moreover have \( \forall T \in T. \forall t \in T. \ (t,s) \in r \)
  proof -
    { fix T t assume A: \( T \in T \). t \in T
      with assms(1,2,4) have \( (t,\text{Supremum}(r,T)) \in r \)
        unfolding HasAsupremum_def using sup_in_space(2) by blast
      moreover from assms(1,2,5) \( T \in T \) have \( (\text{Supremum}(r,T),s) \in r \)
        unfolding HasAsupremum_def using sup_in_space(2) by blast
      moreover note assms(3)
      ultimately have \( (t,s) \in r \) unfolding trans_def by blast
    } thus thesis by simp
  qed
  hence I: \( \forall t \in \bigcup T. \ (t,s) \in r \) by auto
  moreover have J: \( \forall y. \ (\forall x \in \bigcup T. \ (x,y) \in r) \implies (s,y) \in r \)
  proof -
    { fix y x assume A: \( \forall x \in \bigcup T. \ (y,x) \in r \)
      with assms(2,4) have \( \forall a \in \{\text{Supremum}(r,T).T \in T\}. \ (a,y) \in r \) using sup_leq_up_bnd
        by simp
      with assms(2,5) have \( (s,y) \in r \) by (rule sup_leq_up_bnd)
    } thus thesis by simp
  qed
ultimately have $\text{HasAminimum}(r, \bigcap a \in \bigcup T. r\{a\})$ by (rule \textbf{OrderZF5.L5}) then show $\text{HasASupremum}(r, \bigcup T)$ unfolding \textbf{HasASupremum_def} by simp from \textbf{assms}(2) $\bigcup T \neq 0$ I show $s = \text{Supremum}(r, \bigcup T)$ by (rule \textbf{OrderZF5.L5}) qed

If the relation is a linear order then for any element $y$ smaller than the supremum of a set we can find one element of the set that is greater than $y$.

**lemma** \textbf{OrderZF5.L8}:

\begin{align*}
\text{assumes A1: } r &\subseteq X \times X \text{ and A2: IsLinOrder}(X, r) \text{ and } \\
A3: r &\text{ (is complete) and } \\
A4: A &\subseteq X \ A \neq 0 \text{ and A5: } \exists x \in X. \ A \subseteq X \text{. } \forall y \in A. \ (y, x) \in r \text{ and } \\
A6: (y, \text{Supremum}(r, A)) &\in r \ y \neq \text{Supremum}(r, A) \\
\text{shows } &\exists z \in A. \ (y, z) \in r \wedge y \neq z
\end{align*}

\textbf{proof} -

\begin{align*}
\text{from A2 have } \text{I: antisym}(r) \text{ and } \\
\text{II: trans}(r) \text{ and } \\
\text{III: r } \text{(is total on) } X \\
\text{using } \text{IsLinOrder_def by auto} \\
\text{from A1 A6 have T1: } y \in X \text{ by auto} \\
\{ \text{ assume A7: } \forall z \in A. \ (y, z) \notin r \vee y = z \\
\text{ from A4 I have antisym}(r) \text{ and A} \neq 0 \text{ by auto} \\
\text{ moreover have } \forall x \in A. \ (x, y) \in r \\
\text{ proof} \\
\text{ fix } x \text{ assume A8: } x \in A \\
\text{ with A4 have T2: } x \in X \text{ by auto} \\
\text{ from A7 A8 have } (y, x) \notin r \vee y = x \text{ by simp} \\
\text{ with III T1 T2 show } (x, y) \in r \\
\text{ using } \text{IsTotal_def total_is_refl refl_def by auto} \\
\text{ qed} \\
\text{ moreover have } \forall u. \ (\forall x \in A. \ (x, u) \in r) \rightarrow (y, u) \in r \\
\text{ proof} - \\
\text{ \{ fix } u \text{ assume A9: } \forall x \in A. \ (x, u) \in r \\
\text{ from A4 A5 have IsBoundedAbove}(A, r) \text{ and } A \neq 0 \\
\text{ using IsBoundedAbove_def by auto} \\
\text{ with A3 A4 A6 I A9 have } \\
(y, \text{Supremum}(r, A)) \in r \wedge (\text{Supremum}(r, A), u) \in r \\
\text{ using } \text{IsComplete_def OrderZF5.L3 by simp} \\
\text{ with II have } (y, u) \in r \text{ by (rule Fol1_L3)} \\
\} \text{ then show } \forall u. \ (\forall x \in A. \ (x, u) \in r) \rightarrow (y, u) \in r \\
\text{ by simp} \\
\text{ qed} \\
\text{ ultimately have } y = \text{Supremum}(r, A) \\
\text{ by (rule OrderZF5.L5)} \\
\text{ with A6 have False by simp} \\
\} \text{ then show } \exists z \in A. \ (y, z) \in r \wedge y \neq z \text{ by auto} \\
\text{ qed}
\end{align*}
7.3 Strict versions of order relations

One of the problems with translating formalized mathematics from Meta-math to IsarMathLib is that Metamath uses strict orders (of the $<$ type) while in IsarMathLib we mostly use nonstrict orders (of the $\leq$ type). This doesn’t really make any difference, but is annoying as we have to prove many theorems twice. In this section we prove some theorems to make it easier to translate the statements about strict orders to statements about the corresponding non-strict order and vice versa.

We define a strict version of a relation by removing the $y = x$ line from the relation.

**definition**

\[
\text{StrictVersion}(r) \equiv r - \{(x,x) \mid x \in \text{domain}(r)\}
\]

A reformulation of the definition of a strict version of an order.

**lemma** def_of_strict_ver: shows 

\[
\langle x,y \rangle \in \text{StrictVersion}(r) \iff \langle x,y \rangle \in r \land x \neq y
\]

using StrictVersion_def domain_def by auto

The next lemma is about the strict version of an antisymmetric relation.

**lemma** strict_of_antisym: assumes A1: antisym(r) and A2: \(\langle a,b \rangle \in \text{StrictVersion}(r)\) shows \(\langle b,a \rangle \notin \text{StrictVersion}(r)\)

proof - 

{ assume A3: \(\langle b,a \rangle \in \text{StrictVersion}(r)\) with A2 have \(\langle a,b \rangle \in r \land \langle b,a \rangle \in r\) using def_of_strict_ver by auto with A1 have a=b by (rule Fol1_L4) with A2 have False using def_of_strict_ver by simp } then show \(\langle b,a \rangle \notin \text{StrictVersion}(r)\) by auto qed

The strict version of totality.

**lemma** strict_of_tot: assumes r is total on} X and a\in X b\in X a \neq b shows \(\langle a,b \rangle \in \text{StrictVersion}(r) \lor \langle b,a \rangle \in \text{StrictVersion}(r)\)

using assms IsTotal_def def_of_strict_ver by auto

A trichotomy law for the strict version of a total and antisymmetric relation. It is kind of interesting that one does not need the full linear order for this.

**lemma** strict_ans_tot_trich: assumes A1: antisym(r) and A2: r is total on} X and A3: a\in X b\in X and A4: s = StrictVersion(r) shows Exactly_1_of_3_holds(\(\langle a,b \rangle \in s, a=b,\langle b,a \rangle \in s\))
proof -
let \( p = \langle a, b \rangle \in s \)
let \( q = a=b \)
let \( r = \langle b, a \rangle \in s \)
from A2 A3 A4 have \( p \lor q \lor r \)
  using strict_of_tot by auto
moreover from A1 A4 have \( p \rightarrow \neg q \land \neg r \)
  using def_of_strict_ver strict_of_antisym by simp
moreover from A4 have \( q \rightarrow \neg p \land \neg r \)
  using def_of_strict_ver by simp
moreover from A1 A4 have \( r \rightarrow \neg p \land \neg q \)
  using def_of_strict_ver strict_of_antisym by auto
ultimately show Exactly_1_of_3_holds(p, q, r)
  by (rule Fol1_L5)
qed

A trichotomy law for linear order. This is a special case of strict_ans_tot_trich.

corollary strict_lin_trich: assumes A1: IsLinOrder(X,r) and
A2: a\in X b\in X and
A3: s = StrictVersion(r)
shows Exactly_1_of_3_holds(\langle a, b \rangle \in s, a=b, \langle b, a \rangle \in s)
using assms IsLinOrder_def strict_ans_tot_trich by auto

For an antisymmetric relation if a pair is in relation then the reversed pair
is not in the strict version of the relation.

lemma geq_impl_not_less:
  assumes A1: antisym(r) and A2: \langle a, b \rangle \in r
shows \langle b, a \rangle \notin StrictVersion(r)
proof -
  \{ assume A3: \langle b, a \rangle \in StrictVersion(r)
    with A2 have \langle a, b \rangle \in StrictVersion(r)
      using def_of_strict_ver by auto
    with A1 A3 have False using strict_of_antisym
      by blast
  \} then show \langle b, a \rangle \notin StrictVersion(r) by auto
qed

If an antisymmetric relation is transitive, then the strict version is also
transitive, an explicit version strict_of_transB below.

lemma strict_of_transA:
  assumes A1: trans(r) and A2: antisym(r) and
A3: s= StrictVersion(r) and A4: \langle a, b \rangle \in s \ \langle b, c \rangle \in s
shows \langle a, c \rangle \in s
proof -
  from A3 A4 have I: \langle a, b \rangle \in r \land \langle b, c \rangle \in r
    using def_of_strict_ver by simp
  with A1 have \langle a, c \rangle \in r by (rule Fol1_L3)
  moreover

\{ assume \( a = c \)
   with I have \( \langle a, b \rangle \in r \) and \( \langle b, a \rangle \in r \) by auto
   with A2 have \( a = b \) by (rule Fol1_L4)
   with A3 A4 have False using def_of_strict_ver by simp
\}
ultimately have \( a \neq c \) by auto

If an antisymmetric relation is transitive, then the strict version is also transitive.

**lemma** strict_of_transB:
assumes A1: trans\( (r) \) and A2: antisym\( (r) \)
show\( s = \text{StrictVersion}(r) \)
proof -
let \( s = \text{StrictVersion}(r) \)
from A1 A2 have \( \forall x y z. \langle x, y \rangle \in s \land \langle y, z \rangle \in s \rightarrow \langle x, z \rangle \in s \)
using strict_of_transA by blast
then show trans\( (\text{StrictVersion}(r)) \) by (rule Fol1_L2)
qed

The next lemma provides a condition that is satisfied by the strict version of a relation if the original relation is a complete linear order.

**lemma** strict_of_compl:
assumes A1: \( r \subseteq X \times X \) and A2: IsLinOrder\( (X, r) \) and
A3: \( r \) {is complete} and
A4: \( A \subseteq X \) A\( \neq 0 \) and A5: \( s = \text{StrictVersion}(r) \) and
A6: \( \exists u \in X. \forall y \in A. \langle y, u \rangle \in s \)
shows \( \exists x \in X. ( \forall y \in A. \langle x, y \rangle \notin s ) \land ( \forall y \in X. \langle y, x \rangle \in s \rightarrow (\exists z \in A. \langle y, z \rangle \in s)) \)
proof -
let \( x = \text{Supremum}(r, A) \)
from A2 have I: antisym\( (r) \) using IsLinOrder_def by simp
moreover from A5 A6 have \( \exists u \in X. \forall y \in A. \langle y, u \rangle \in r \)
using def_of_strict_ver by auto
moreover note A1 A3 A4
ultimately have II: \( x \in X \) \( \forall y \in A. \langle y, x \rangle \in r \)
using Order_ZF_5_L7 by auto
then have III: \( \exists x \in X. \forall y \in A. \langle y, x \rangle \in r \) by auto
from A5 I II have \( x \in X \) \( \forall y \in A. \langle x, y \rangle \notin s \)
using geq_impl_not_less by auto
moreover from A1 A2 A3 A4 A5 III have
\( \forall y \in X. \langle y, x \rangle \in s \rightarrow (\exists z \in A. \langle y, z \rangle \in s) \)
using def_of_strict_ver Order_ZF_5_L8 by simp
ultimately show

74
∃x∈X. ( ∀y∈A. ⟨x,y⟩ ∉ s ) ∧ ( ∀y∈X. ⟨y,x⟩ ∈ s → (∃z∈A. ⟨y,z⟩ ∈ s))
by auto
qed

Strict version of a relation on a set is a relation on that set.

lemma strict_ver_rel: assumes A1: r ⊆ A×A
  shows StrictVersion(r) ⊆ A×A
  using assms StrictVersion_def by auto
end

8 Functions - introduction

theory func1 imports ZF.func Fol1 ZF
begin

This theory covers basic properties of function spaces. A set of functions
with domain X and values in the set Y is denoted in Isabelle as X → Y. It
just happens that the colon ":" is a synonym of the set membership symbol
∈ in Isabelle/ZF so we can write f : X → Y instead of f ∈ X → Y. This is
the only case that we use the colon instead of the regular set membership
symbol.

8.1 Properties of functions, function spaces and (inverse) images.

Functions in ZF are sets of pairs. This means that if f : X → Y then
f ⊆ X × Y. This section is mostly about consequences of this understanding
of the notion of function.

We define the notion of function that preserves a collection here. Given two
collection of sets a function preserves the collections if the inverse image
of sets in one collection belongs to the second one. This notion does not
have a name in romantic math. It is used to define continuous functions
in Topology_ZF_2 theory. We define it here so that we can use it for other
purposes, like defining measurable functions. Recall that f-(A) means the
inverse image of the set A.

definition
  PresColl(f,S,T) ≡ ∀ A∈T. f-(A)∈S

A definition that allows to get the first factor of the domain of a binary
function f : X × Y → Z.

definition
  fstdom(f) ≡ domain(domain(f))
If a function maps $A$ into another set, then $A$ is the domain of the function.

**Lemma func1_1_L1:**

- Assumes $f : A \to C$
- Shows $\text{domain}(f) = A$
  - Using assms domain_of_fun by simp

Standard Isabelle defines a function($f$) predicate. The next lemma shows that our functions satisfy that predicate. It is a special version of Isabelle’s `fun_is_function`.

**Lemma fun_is_fun:**

- Assumes $f : X \to Y$
- Shows $\text{function}(f)$
  - Using assms fun_is_function by simp

A lemma explains what $\text{fstdom}$ is for.

**Lemma fstdomdef:**

- Assumes $A1: f : X \times Y \to Z$ and $A2: Y \neq \emptyset$
- Shows $\text{fstdom}(f) = X$
  - Proof -
    - From $A1$ have $\text{domain}(f) = X \times Y$ using `func1_1_L1`
      - By simp
    - With $A2$ show $\text{fstdom}(f) = X$ unfolding `fstdom_def` by auto
  - Qed

A version of the `Pi_type` lemma from the standard Isabelle/ZF library.

**Lemma func1_1_L1A:**

- Assumes $A1: f : X \to Y$ and $A2: \forall x \in X. f(x) \in Z$
- Shows $f : X \to Z$
  - Proof -
    - Fix $x$ assume $x \in X$
      - With $A2$ have $f(x) \in Z$ by simp
    - With $A1$ show $f : X \to Z$ by (rule `Pi_type`)
  - Qed

A variant of `func1_1_L1A`.

**Lemma func1_1_L1B:**

- Assumes $A1: f : X \to Y$ and $A2: Y \subseteq Z$
- Shows $f : X \to Z$
  - Proof -
    - From $A1$ $A2$ have $\forall x \in X. f(x) \in Z$
      - Using `apply_funtype` by auto
    - With $A1$ show $f : X \to Z$ using `func1_1_L1A` by blast
  - Qed

There is a value for each argument.

**Lemma func1_1_L2:**

- Assumes $A1: f : X \to Y$ $x \in X$
- Shows $\exists y \in Y. (x,y) \in f$
  - Proof -
    - From $A1$ have $f(x) \in Y$ using `apply_type` by simp
      - Moreover from $A1$ have $(x,f(x)) \in f$ using `apply_Pair` by simp
    - Ultimately show thesis by auto
  - Qed

The inverse image is the image of converse. True for relations as well.
lemma vimage_converse: shows \( r^{-1}(A) = \text{converse}(r)(A) \)
using vimage_iff image_iff converse_iff by auto

The image is the inverse image of converse.

lemma image_converse: shows \( \text{converse}(r^{-1})(A) = r(A) \)
using vimage_iff image_iff converse_iff by auto

The inverse image by a composition is the composition of inverse images.

lemma vimage_comp: shows \( (r \circ s)^{-1}(A) = s^{-1}(r^{-1})(A) \)
using vimage_converse converse_comp image_comp image_converse by simp

A version of vimage_comp for three functions.

lemma vimage_comp3: shows \( (r \circ s \circ t)^{-1}(A) = t^{-1}(s^{-1}(r^{-1})(A)) \)
using vimage_comp by simp

Inverse image of any set is contained in the domain.

lemma func1_1_L3: assumes \( A1: f:X \rightarrow Y \) shows \( f^{-1}(D) \subseteq X \)
proof -
  have \( \forall x. \ x \in f^{-1}(D) \rightarrow x \in \text{domain}(f) \)
  using vimage_iff domain_iff by auto
  with \( A1 \ough \forall x. \ (x \in f^{-1}(D)) \rightarrow (x \in X) \) using func1_1_L1 by simp
  then show thesis by auto
qed

The inverse image of the range is the domain.

lemma func1_1_L4: assumes \( f:X \rightarrow Y \) shows \( f^{-1}(Y) = X \)
using assms func1_1_L3 func1_1_L2 vimage_iff by blast

The arguments belongs to the domain and values to the range.

lemma func1_1_L5:
assumes \( A1: \langle x,y \rangle \in f \) and \( A2: f:X \rightarrow Y \)
shows \( x \in X \land y \in Y \)
proof
  from \( A1 \) \( A2 \) show \( x \in X \) using apply_iff by simp
  with \( A2 \) have \( f(x) \in Y \) using apply_type by simp
  with \( A1 \) \( A2 \) show \( y \in Y \) using apply_iff by simp
qed

Function is a subset of cartesian product.

lemma fun_subset_prod: assumes \( A1: f:X \rightarrow Y \) shows \( f \subseteq X \times Y \)
proof
  fix \( p \) assume \( p \in f \)
  with \( A1 \) have \( \exists x \in X. \ p = \langle x, f(x) \rangle \)
  using Pi_memberD by simp
  then obtain \( x \) where \( I: p = \langle x, f(x) \rangle \)
  by auto
  with \( A1 \) \( p \in f \) have \( x \in X \land f(x) \in Y \)
  using func1_1_L5 by blast
with I show p ∈ X×Y by auto
qed

The (argument, value) pair belongs to the graph of the function.

**lemma** func1_1_L5A:

assumes A1: f:X→Y  x∈X  y = f(x)
shows ⟨x,y⟩ ∈ f  y ∈ range(f)

**proof**

- from A1 show ⟨x,y⟩ ∈ f using apply_Pair by simp
- then show y ∈ range(f) using rangeI by simp
qed

The next theorem illustrates the meaning of the concept of function in ZF.

**theorem** fun_is_set_of_pairs: assumes A1: f:X→Y
shows f = {⟨x, f(x)⟩. x ∈ X}

**proof**

- from A1 show {⟨x, f(x)⟩. x ∈ X} ⊆ f using func1_1_L5A by auto
- next
- { fix p assume p ∈ f
- with A1 show p ∈ X×Y using fun_subset_prod by auto
- with A1 ⟨p ∈ f⟩ have p ∈ {⟨x, f(x)⟩. x ∈ X}
- using apply_equality by auto
- } thus f ⊆ {⟨x, f(x)⟩. x ∈ X} by auto
qed

The range of function that maps X into Y is contained in Y.

**lemma** func1_1_L5B:

assumes A1: f:X→Y shows range(f) ⊆ Y

**proof**

- fix y assume y ∈ range(f)
- then obtain x where ⟨x, y⟩ ∈ f
- using range_def converse_def domain_def by auto
- with A1 show y ∈ Y using func1_1_L5 by blast
qed

The image of any set is contained in the range.

**lemma** func1_1_L6: assumes A1: f:X→Y
shows f(B) ⊆ range(f) and f(B) ⊆ Y

**proof**

- show f(B) ⊆ range(f) using image_iff rangeI by auto
- with A1 show f(B) ⊆ Y using func1_1_L5B by blast
qed

The inverse image of any set is contained in the domain.

**lemma** func1_1_L6A: assumes A1: f:X→Y shows f−(A) ⊆ X

**proof**
fix x
  assume A2: \( x \in f^{-1}(A) \) then obtain \( y \) where \( \langle x, y \rangle \in f \)
  using vimage_iff by auto
  with A1 show \( x \in X \) using func1_1_L5 by fast
qed

Image of a greater set is greater.

lemma func1_1_L8: assumes A1: \( A \subseteq B \) shows \( f(A) \subseteq f(B) \)
  using assms image_Un by auto

A set is contained in the the inverse image of its image. There is similar theorem in equalities.thy (function_image_vimage) which shows that the image of inverse image of a set is contained in the set.

lemma func1_1_L9: assumes A1: \( f:X \rightarrow Y \) and A2: \( A \subseteq X \)
  shows \( A \subseteq f^{-1}(f(A)) \)
proof -
  from A1 A2 have \( \forall x \in A. \ (x, f(x)) \in f \) using apply_Pair by auto
  then show thesis using image_iff by auto
qed

The inverse image of the image of the domain is the domain.

lemma inv_im_dom: assumes A1: \( f:X \rightarrow Y \) shows \( f^{-1}(f(X)) = X \)
proof
  from A1 show \( f^{-1}(f(X)) \subseteq X \) using func1_1_L3 by simp
  from A1 show \( X \subseteq f^{-1}(f(X)) \) using func1_1_L9 by simp
qed

A technical lemma needed to make the func1_1_L11 proof more clear.

lemma func1_1_L10:
  assumes A1: \( f \subseteq X \times Y \) and A2: \( \exists !y. \ (y \in Y \land \langle x, y \rangle \in f) \)
  shows \( \exists !y. \ (x, y) \in f \)
proof
  from A2 show \( \exists y. \ (x, y) \in f \) by auto
  fix y n assume \( \langle x, y \rangle \in f \) and \( \langle x, n \rangle \in f \)
  with A1 A2 show \( y = n \) by auto
qed

If \( f \subseteq X \times Y \) and for every \( x \in X \) there is exactly one \( y \in Y \) such that \( (x, y) \in f \) then \( f \) maps \( X \) to \( Y \).

lemma func1_1_L11:
  assumes \( f \subseteq X \times Y \) and \( \forall x \in X. \ \exists !y. \ y \in Y \land (x, y) \in f \)
  shows \( f: X \rightarrow Y \) using assms func1_1_L10 Pi_iff_old by simp

A set defined by a lambda-type expression is a function. There is a similar lemma in funct.thy, but I had problems with lambda expressions syntax so I could not apply it. This lemma is a workaround for this. Besides, lambda expressions are not readable.
lemma func1_1_L11A: assumes A1: \( \forall x \in X. \ b(x) \in Y \)
shows \( \{ (x,y) \in X \times Y. \ b(x) = y \} : X \rightarrow Y \)
proof -
  let \( f = \{ (x,y) \in X \times Y. \ b(x) = y \} \)
  have \( f \subseteq X \times Y \) by auto
moreover have \( \forall x \in X. \ \exists !y. y \in Y \land (x,y) \in f \)
proof
  fix \( x \) assume A2: \( x \in X \)
  have \( \exists !y. y \in Y \land (x,y) \in f \) by simp
next
  fix y y1 assume y: \( y \in Y \land (x,y) \in f \)
and y1: \( y1 \in Y \land (x,y1) \in f \)
  then show \( y = y1 \) by simp
qed
ultimately show \( \{(x,y) \in X \times Y. \ b(x) = y \} : X \rightarrow Y \)
using func1_1_L11 by simp
qed

The next lemma will replace func1_1_L11A one day.

lemma ZF_fun_from_total: assumes A1: \( \forall x \in X. \ b(x) \in Y \)
shows \( \{ (x,b(x)). x \in X \} : X \rightarrow Y \)
proof -
  let \( f = \{ (x,b(x)). x \in X \} \)
  { fix x assume A2: \( x \in X \)
    have \( \exists !y. y \in Y \land (x,y) \in f \) by simp
  }
next
  fix y y1 assume y: \( y \in Y \land (x,y) \in f \)
and y1: \( y1 \in Y \land (x,y1) \in f \)
  then show \( y = y1 \) by simp
qed
ultimately show thesis using func1_1_L11 by simp
qed

The value of a function defined by a meta-function is this meta-function
(deprecated, use ZF_fun_from_tot_val(1) instead).

lemma func1_1_L11B:
  assumes A1: \( f : X \rightarrow Y \) \( x \in X \)
and A2: \( f = \{ (x,y) \in X \times Y. \ b(x) = y \} \)

shows \( f(x) = b(x) \)

proof -
  from A1 have \( (x,f(x)) \in f \) using apply_iff by simp 
  with A2 show thesis by simp

qed

The next lemma will replace func1_1_L11B one day.

lemma ZF_fun_from_tot_val:
  assumes \( f:X \rightarrow Y \ \ x \in X \)
  \( f = \{ (x,b(x)). \ x \in X \} \)
  \( \) shows \( f(x) = b(x) \)
  and \( b(x) \in Y \)
  proof - 
    from assms(1,2) have \( (x,f(x)) \in f \) using apply_iff by simp 
    with assms(3) show thesis by simp

from assms(1,2) have \( f(x) \in Y \) by (rule apply_funtype)
  with \( f(x) = b(x) \) show \( b(x) \in Y \) by simp

qed

Identical meaning as ZF_fun_from_tot_val, but phrased a bit differently.

lemma ZF_fun_from_tot_val0:
  assumes \( f:X \rightarrow Y \) and \( f = \{ (x,b(x)). \ x \in X \} \)
  \( \) shows \( \forall x \in X. \ f(x) = b(x) \)
  using assms ZF_fun_from_tot_val by simp

Another way of expressing that lambda expression is a function.

lemma lam_is_fun_range: assumes \( f=\{ (x,g(x)). \ x \in X \} \)
  shows \( f:X \rightarrow \text{range}(f) \)
  proof -
    let \( f = \{ (x,b(x)). \ x \in X \} \)
    have \( \forall x \in X. \ g(x) \in \text{range}(\{ (x,g(x)). \ x \in X \}) \) unfolding range_def 
      by auto
    then have \( \{ (x,g(x)). \ x \in X \} : X \rightarrow \text{range}(\{ (x,g(x)). \ x \in X \}) \) by (rule ZF_fun_from_total)
    with assms show thesis by auto

qed

Yet another way of expressing value of a function.

lemma ZF_fun_from_tot_val1:
  assumes \( x \in X \)
  shows \( \{ (x,b(x)). \ x \in X \}(x) = b(x) \)
  proof -
    let \( f = \{ (x,b(x)). \ x \in X \} \)
    have \( f:X \rightarrow \text{range}(f) \) using lam_is_fun_range by simp 
      with assms show thesis using ZF_fun_from_tot_val0 by simp

qed

An hypotheses-free form of ZF_fun_from_tot_val1: the value of a function \( X \ni x \mapsto p(x) \) is \( p(x) \) for all \( x \in X \).

lemma ZF_fun_from_tot_val2: shows \( \forall x \in X. \ \{ (x,b(x)). \ x \in X \}(x) = b(x) \)
  using ZF_fun_from_tot_val1 by simp

81
The range of a function defined by set comprehension is the set of its values.

```
lemma range_fun: shows range({⟨x,b(x)⟩. x∈X}) = {b(x). x∈X}
  by blast
```

In Isabelle/ZF and Metamath if \( x \) is not in the domain of a function \( f \) then \( f(x) \) is the empty set. This allows us to conclude that if \( y \in f(x) \), then \( x \) must be an element of the domain of \( f \).

```
lemma arg_in_domain: assumes f:X→Y y∈f(x) shows x∈X
proof -
  { assume x∉X
    with assms have False using func1_1_L1 apply_0 by simp
  } thus thesis by auto
qed
```

We can extend a function by specifying its values on a set disjoint with the domain.

```
lemma func1_1_L11C: assumes A1: f:X→Y and A2: ∀x∈A. b(x)∈B
  and A3: X∩A = ∅ and Dg: g = f ∪ {⟨x,b(x)⟩. x∈A}
  shows g : X∪A → Y∪B
  ∀x∈X. g(x) = f(x)
  ∀x∈A. g(x) = b(x)
proof -
  let h = {⟨x,b(x)⟩. x∈A}
  from A1 A2 A3 have
  I: f:X→Y h : A→B X∩A = ∅
    using ZF_fun_from_total by auto
  then have f∪h : X∪A → Y∪B
    by (rule fun_disjoint_Un)
  with Dg show g : X∪A → Y∪B by simp
  { fix x assume A4: x∈A
    with A1 A3 have (f∪h)(x) = h(x)
      using func1_1_L1 fun_disjoint_apply2
      by blast
    moreover from I A4 have h(x) = b(x)
      using ZF_fun_from_tot_val by simp
    ultimately have (f∪h)(x) = b(x)
      by simp
  } with Dg show ∀x∈A. g(x) = b(x) by simp
  { fix x assume A5: x∈X
    with A3 I have x /∈ domain(h)
      using func1_1_L1 by auto
    then have (f∪h)(x) = f(x)
      using fun_disjoint_apply1 by simp
  } with Dg show ∀x∈X. g(x) = f(x) by simp
qed
```

We can extend a function by specifying its value at a point that does not belong to the domain.
lemma func1_1_L11D: assumes A1: \( f : X \rightarrow Y \) and A2: \( a \notin X \)
and Dg: \( g = f \cup \{(a,b)\} \)
shows
\( g : X \cup \{a\} \rightarrow Y \cup \{b\} \)
\( \forall x \in X. \ g(x) = f(x) \)
g(a) = b
proof -
let \( h = \{(a,b)\} \)
from A1 A2 Dg have I:
\( f : X \rightarrow Y \ \forall x \in \{a\}. \ b \in \{b\} \)
\( X \cap \{a\} = \emptyset \) \( g = f \cup \{(x,b). \ x \in \{a\}\} \)
by auto
then show \( g : X \cup \{a\} \rightarrow Y \cup \{b\} \)
by (rule func1_1_L11C)
from I show \( \forall x \in X. \ g(x) = f(x) \)
by (rule func1_1_L11C)
from I have \( \forall x \in \{a\}. \ g(x) = b \)
by (rule func1_1_L11C)
then show \( g(a) = b \) by auto
qed

A technical lemma about extending a function both by defining on a set disjoint with the domain and on a point that does not belong to any of those sets.

lemma func1_1_L11E:
assumes A1: \( f : X \rightarrow Y \) and
A2: \( \forall x \in A. \ b(x) \in B \) and
A3: \( X \cap A = \emptyset \) and A4: \( a \notin X \cup A \)
and Dg: \( g = f \cup \{(x,b(x)). \ x \in A\} \cup \{(a,c)\} \)
shows
\( g : X \cup A \cup \{a\} \rightarrow Y \cup B \cup \{c\} \)
\( \forall x \in X. \ g(x) = f(x) \)
\( \forall x \in A. \ g(x) = b(x) \)
g(a) = c
proof -
let \( h = f \cup \{(x,b(x)). \ x \in A\} \)
from assms show \( g : X \cup A \cup \{a\} \rightarrow Y \cup B \cup \{c\} \)
using func1_1_L11C func1_1_L11D by simp
from A1 A2 A3 have I:
\( f : X \rightarrow Y \ \forall x \in A. \ b(x) \in B \ \ X \cap A = \emptyset \) \( h = f \cup \{(x,b(x)). \ x \in A\} \)
by auto
from assms have
II: \( h : X \cup A \rightarrow Y \cup B \ a \notin X \cup A \ g = h \cup \{(a,c)\} \)
using func1_1_L11C by auto
then have III: \( \forall x \in X. \ g(x) = h(x) \) by (rule func1_1_L11D)
moreover from I have \( \forall x \in X. \ h(x) = f(x) \)
by (rule func1_1_L11C)
ultimately show \( \forall x \in X. \ g(x) = f(x) \) by simp
from I have \( \forall x \in A. \ h(x) = b(x) \) by (rule func1_1_L11C)
with III show \( \forall x \in A. \ g(x) = b(x) \) by simp

83
from II show g(a) = c by (rule func1_1_L11D)
qed

A way of defining a function on a union of two possibly overlapping sets. We decompose the union into two differences and the intersection and define a function separately on each part.

lemma fun_union_overlap: assumes \( \forall x \in A \cap B. \ h(x) \in Y \ \forall x \in A \setminus B. \ f(x) \in Y \ \forall x \in B \setminus A. \ g(x) \in Y \)
shows \( \{<x, \text{if } x \in A \setminus B \text{ then } f(x) \text{ else if } x \in B \setminus A \text{ then } g(x) \text{ else } h(x) >. x \in A \cup B\}: \ A \cup B \rightarrow Y \)
proof -
let \( F = \{<x, \text{if } x \in A \setminus B \text{ then } f(x) \text{ else if } x \in B \setminus A \text{ then } g(x) \text{ else } h(x) >. x \in A \cap B\} \)
from \( \text{assms} \) have \( \forall x \in A \cup B. \ (\text{if } x \in A \setminus B \text{ then } f(x) \text{ else if } x \in B \setminus A \text{ then } g(x) \text{ else } h(x)) \in Y \)
by auto
then show \( \text{thesis by (rule ZF_fun_from_total)} \)
qed

Inverse image of intersection is the intersection of inverse images.

lemma invim_inter_inter_invim: assumes \( f:X \rightarrow Y \)
shows \( f-\{A \cap B\} = f-(A) \cap f-(B) \)
using \( \text{assms} \) \( \text{fun_is_fun} \) \( \text{function_vimage_Int} \) by simp

The inverse image of an intersection of a nonempty collection of sets is the intersection of the inverse images. This generalizes \( \text{invim_inter_inter_invim} \) which is proven for the case of two sets.

lemma func1_1_L12:
assumes \( A1: \ B \subseteq \text{Pow}(Y) \) and \( A2: \ B \neq \emptyset \) and \( A3: \ f:X \rightarrow Y \)
shows \( f-(\bigcap B) = (\bigcap U \in B. \ f-(U)) \)
proof
from \( A2 \) show \( f-(\bigcap B) \subseteq (\bigcap U \in B. \ f-(U)) \) by blast
show \( (\bigcap U \in B. \ f-(U)) \subseteq f-(\bigcap B) \)
proof
fix \( x \) assume \( A4: \ x \in (\bigcap U \in B. \ f-(U)) \)
from \( A3 \) have \( \forall U \in B. \ f-(U) \subseteq X \) using \( \text{func1_1_L6A} \) by simp
with \( A4 \) have \( \forall U \in B. \ x \in X \) by auto
with \( A2 \) have \( x \in X \) by auto
with \( A3 \) have \( \exists ! y. \ (x,y) \in f \) using \( \text{Pi_iff_old} \) by simp
with \( A2 \ A4 \) show \( x \in f-(\bigcap B) \) using \( \text{vimage_iff} \) by blast
qed
qed

The inverse image of a set does not change when we intersect the set with the image of the domain.

lemma inv_im_inter_im: assumes \( f:X \rightarrow Y \)
shows \( f-(A \cap f(X)) = f-(A) \)
using \( \text{assmss} \) \( \text{invim_inter_inter_invim} \) \( \text{inv_im_dom} \) \( \text{func1_1_L6A} \)
If the inverse image of a set is not empty, then the set is not empty. Proof by contradiction.

lemma func1_1_L13: assumes A1: f-(A) ≠ ∅ shows A ≠ ∅ using assms by auto

If the image of a set is not empty, then the set is not empty. Proof by contradiction.

lemma func1_1_L13A: assumes A1: f(A) ≠ ∅ shows A ≠ ∅ using assms by auto

What is the inverse image of a singleton?

lemma func1_1_L14: assumes f: X → Y shows f-{y} = {x ∈ X. f(x) = y} using assms func1_1_L6A vimage_singleton_iff apply_iff by auto

A lemma that can be used instead fun_extension_iff to show that two functions are equal

lemma func_eq: assumes f: X → Y g: X → Z and ∀x ∈ X. f(x) = g(x) shows f = g using assms fun_extension_iff by simp

An alternative syntax for defining a function: instead of writing \{⟨x, p(x)⟩. x ∈ X\} we can write λx ∈ X. p(x).

lemma lambda_fun_alt: shows \{⟨x, p(x)⟩. x ∈ X\} = (λx ∈ X. p(x))
proof -
  let L = \{⟨x, p(x)⟩. x ∈ X\}
  let R = λx ∈ X. p(x)
  have L:X→range(L) and R:X→range(L)
    using lam_is_fun_range range_fun lam_funtype by simp_all
  moreover have ∀x ∈ X. L(x) = R(x) using ZF_fun_from_tot_val1 beta by simp
  ultimately show L = R using func_eq by blast
qed

If a function is equal to an expression b(x) on X, then it has to be of the form \{⟨x, b(x)⟩. x ∈ X\}.

lemma func_eq_set_of_pairs: assumes f: X → Y ∀x ∈ X. f(x) = b(x) shows f = \{⟨x, b(x)⟩. x ∈ X\}
proof -
  from assms(1) have f = \{⟨x, b(x)⟩. x ∈ X\} using fun_is_set_of_pairs by simp
  with assms(2) show thesis by simp
qed

Function defined on a singleton is a single pair.
lemma func_singleton_pair: assumes A1: \( f : \{a\} \rightarrow X \)
shows \( f = \{\langle a, f(a)\rangle\} \)
proof -
  let \( g = \{\langle a, f(a)\rangle\} \)
  note A1
  moreover have \( g : \{a\} \rightarrow \{f(a)\} \) using singleton_fun by simp
  moreover have \( \forall x \in \{a\}. f(x) = g(x) \) using singleton_apply
    by simp
  ultimately show \( f = g \) by (rule func_eq)
qed

A single pair is a function on a singleton. This is similar to \texttt{singleton\_fun}
from standard Isabelle/ZF.

lemma pair_func_singleton: assumes A1: \( y \in Y \)
shows \( \{\langle x, y\rangle\} : \{x\} \rightarrow Y \)
proof -
  have \( \{\langle x, y\rangle\} : \{x\} \rightarrow \{y\} \) using singleton_fun by simp
  moreover from A1 have \( \{y\} \subseteq Y \) by simp
  ultimately show \( \{\langle x, y\rangle\} : \{x\} \rightarrow Y \)
    by (rule func1_1_L1B)
qed

The value of a pair on the first element is the second one.

lemma pair_val: shows \( \{\langle x, y\rangle\}(x) = y \)
using singleton_fun apply_equality by simp

A more familiar definition of inverse image.

lemma func1_1_L15: assumes A1: \( f:X \rightarrow Y \)
shows \( f-(A) = \{x \in X. f(x) \in A\} \)
proof -
  have \( f-(A) = (\bigcup \{ f(y) \mid y \in A \}) \)
    by (rule vimage_eq_UN)
  with A1 show thesis using func1_1_L14 by auto
qed

A more familiar definition of image.

lemma func_imagedef: assumes A1: \( f:X \rightarrow Y \) and A2: \( A \subseteq X \)
shows \( f(A) = \{f(x). x \in A\} \)
proof
  from A1 show \( f(A) \subseteq \{f(x). x \in A\} \)
    using image_iff apply_iff by auto
  show \( \{f(x). x \in A\} \subseteq f(A) \)
  proof
    fix \( y \) assume \( y \in \{f(x). x \in A\} \)
    then obtain \( x \) where \( x \in A \) and \( y = f(x) \)
      by auto
    with A1 A2 have \( \langle x, y\rangle \in f \)
      using apply_iff by force
    with A1 A2 \( \langle x \in A \rangle \) show \( y \in f(A) \)
      using image_iff by auto
  qed

86
If all elements of a nonempty set map to the same element of the codomain, then the image of this set is a singleton.

**Lemma image_constant_singleton:**
- **Assumes:** $f : X \to Y \ A \subseteq X \ A \neq \emptyset \ \forall x \in A. \ f(x) = c$
- **Shows:** $f(A) = \{c\}$
- **Using:** asms func_imagedef by auto

A technical lemma about graphs of functions: if we have two disjoint sets $A$ and $B$ then the cartesian product of the inverse image of $A$ and $B$ is disjoint with (the graph of) $f$.

**Lemma vimage_prod_dis_graph:**
- **Assumes:** $f : X \to Y \ A \cap B = \emptyset$
- **Shows:** $f^{-1}(A) \times B \cap f = \emptyset$
- **Proof:**
  - \{ assume $f^{-1}(A) \times B \cap f \neq \emptyset$
    - then obtain $p$ where $p \in f^{-1}(A) \times B$ and $p \in f$ by blast
      - using fun_is_set_of_pairs by simp
    - then obtain $x$ where $p = \langle x, f(x) \rangle$ by blast
      - with asms $p \in f^{-1}(A) \times B$ have False using func1_1_L15 by auto
  } thus thesis by auto
  - qed

For two functions with the same domain $X$ and the codomain $Y, Z$ resp., we can define a third one that maps $X$ to the cartesian product of $Y$ and $Z$.

**Lemma prod_fun_val:**
- **Assumes:** $(\{\langle x, p(x) \rangle. \ x \in X\} : X \to Y \ \{\langle x, q(x) \rangle. \ x \in X\} : X \to Z$
- **Defines:** $h \equiv \{\langle x, \langle p(x), q(x) \rangle \rangle. \ x \in X\}$
- **Shows:** $h : X \to Y \times Z$ and $\forall x \in X. \ h(x) = \langle p(x), q(x) \rangle$
- **Proof:**
  - from asms(1,2) have $\forall x \in X. \ \langle p(x), q(x) \rangle \in Y \times Z$
    - using ZF_fun_from_tot_val(2) by auto
  - with asms(3) show $h : X \to Y \times Z$ using ZF_fun_from_total by simp
  - with asms(3) show $\forall x \in X. \ h(x) = \langle p(x), q(x) \rangle$ using ZF_fun_from_tot_val0 by simp
  - qed

Suppose we have two functions $f : X \to Y$ and $g : X \to Z$ and the third one is defined as $h : X \to Y \times Z, x \mapsto \langle f(x), g(x) \rangle$. Given two sets $U, V$ we have $h^{-1}(U \times V) = (f^{-1}(U)) \cap (g^{-1}(V))$. We also show that the set where the function $f, g$ are equal is the same as $h^{-1}(\{\langle y, y \rangle : y \in X\})$. It is a bit surprising that we get the last identity without the assumption that $Y = Z$.

**Lemma vimage_prod:**
- **Assumes:** $f : X \to Y \ g : X \to Z$
- **Defines:** $h \equiv \{\langle x, \langle f(x), g(x) \rangle \rangle. \ x \in X\}$

87
shows
h:X→Y×Z
∀x∈X. h(x) = ⟨f(x),g(x)⟩

proof -
from assms show h:X→Y×Z using apply_funtype ZF_fun_from_total
  by simp
with assms(3) show I: ∀x∈X. h(x) = ⟨f(x),g(x)⟩
  using ZF_fun_from_tot_val by simp
with assms(1,2) show h-(U×V) = f-(U) ∩ g-(V)
  using func1_1_L15 by auto
from assms(1) I show {x∈X. f(x) = g(x)} = h-({⟨y,y⟩. y∈Y})
  using apply_funtype func1_1_L15 by auto
qed

The image of a set contained in domain under identity is the same set.

lemma image_id_same: assumes A⊆X shows id(X)(A) = A
  using assms id_type id_conv by auto

The inverse image of a set contained in domain under identity is the same set.

lemma vimage_id_same: assumes A⊆X shows id(X)-(A) = A
  using assms id_type id_conv by auto

What is the image of a singleton?

lemma singleton_image:
  assumes f∈X→Y and x∈X
  shows f{x} = {f(x)}
  using assms func_imagedef by auto

If an element of the domain of a function belongs to a set, then its value belongs to the image of that set.

lemma func1_1_L15D: assumes f:X→Y x∈A A⊆X
  shows f(x) ∈ f(A)
  using assms func_imagedef by auto

Range is the image of the domain. Isabelle/ZF defines range(f) as domain(converse(f)), and that’s why we have something to prove here.

lemma range_image_domain:
  assumes A1: f:X→Y shows f(X) = range(f)
proof
  show f(X) ⊆ range(f) using image_def by auto
  { fix y assume y ∈ range(f)
    then obtain x where (y,x) ∈ converse(f) by auto
    with A1 have x∈X using func1_1_L5 by blast
    with A1 have f(x) ∈ f(X) using func_imagedef
      by auto
  }
with A1 \langle y, x \rangle \in \text{converse}(f) \) have \( y \in f(X) 
using \text{apply\_equality by auto}
\} \ then \ show \ range(f) \subseteq f(X) \ by \ auto
qed

The difference of images is contained in the image of difference.

\begin{itemize}
  \item \textbf{lemma \ diff\_image\_diff:} assumes A1: \( f: X \to Y \) and A2: \( A \subseteq X \)
  \item shows \( f(X) \setminus f(A) \subseteq f(X \setminus A) \)
  \item proof
  \item fix \( y \) assume \( y \in f(X) \setminus f(A) \)
  \item hence \( y \in f(X) \) and I: \( y \notin f(A) \) by auto
  \item with A1 obtain x where x\in X and II: \( y = f(x) \)
  \item using \text{func\_imagedef by auto}
  \item with A1 A2 I have x\notin A
  \item using \text{func1\_1\_L15D by auto}
  \item with \( \langle x \in X \rangle \) have x \in X \setminus A \subseteq X \ by auto
  \item with A1 II show \( y \in f(X \setminus A) \)
  \item using \text{func1\_1\_L15D by simp}
\end{itemize}
qed

The image of an intersection is contained in the intersection of the images.

\begin{itemize}
  \item \textbf{lemma \ image\_of\_Inter:} assumes A1: \( f:X \to Y \) and A2: \( I \neq \emptyset \) and A3: \( \forall i \in I. \ P(i) \subseteq X \)
  \item shows \( f(\bigcap_{i \in I.} P(i)) \subseteq ( \bigcap_{i \in I.} f(P(i)) ) \)
  \item proof
  \item fix \( y \) assume A4: \( y \in f(\bigcap_{i \in I.} P(i)) \)
  \item from A2 A3 have \( f(\bigcap_{i \in I.} P(i)) = \{ f(x). \ x \in ( \bigcap_{i \in I.} P(i) ) \} \)
  \item using \text{ZF1\_1\_L7 func\_imagedef by simp}
  \item with A4 obtain x where x \in ( \bigcap_{i \in I.} P(i) ) \ and y = f(x)
  \item by auto
  \item with A1 A2 A3 show \( y \in ( \bigcap_{i \in I.} f(P(i)) ) \) using \text{func\_imagedef by auto}
\end{itemize}
qed

The image of union is the union of images.

\begin{itemize}
  \item \textbf{lemma \ image\_of\_Union:} assumes A1: \( f:X \to Y \) and A2: \( \forall A \in M. \ A \subseteq X \)
  \item shows \( f(\bigcup M) = \bigcup \{ f(A). \ A \in M \} \)
  \item proof
  \item from A2 have \( \bigcup M \subseteq X \) by auto
  \item \{ fix \( y \) assume \( y \in f(\bigcup M) \)
  \item with A1 \( \langle \bigcup M \subseteq X \rangle \) obtain x where x\in \bigcup M \ and I: \( y = f(x) \)
  \item using \text{func\_imagedef by auto}
  \item then obtain A where A\in M \ and x\in A \ by auto
  \item with assms I have y \in \bigcup \{ f(A). \ A \in M \} \ using \text{func\_imagedef by auto}
\} \ thus \( f(\bigcup M) \subseteq \bigcup \{ f(A). \ A \in M \} \) by auto
  \item \{ fix \( y \) assume \( y \in \bigcup \{ f(A). \ A \in M \} \)
  \item then obtain A where A\in M \ and y \in f(A) \ by auto
  \item with assms \( \langle \bigcup M \subseteq X \rangle \) have \( y \in f(\bigcup M) \) using \text{func\_imagedef by auto}
\} \ thus \( \bigcup \{ f(A). \ A \in M \} \subseteq f(\bigcup M) \) by auto
\end{itemize}
If the domain of a function is nonempty, then the codomain is as well.

**lemma codomain_nonempty:** assumes $f: X \to Y$ $X \neq \emptyset$ shows $Y \neq \emptyset$

using assms apply_funtype by blast

The image of a nonempty subset of domain is nonempty.

**lemma func1_1_L15A:**

assumes $A1: f: X \to Y$ and $A2: A \subseteq X$ and $A3: A \neq \emptyset$

shows $f(A) \neq \emptyset$

proof -
  from $A3$ obtain $x$ where $x \in A$ by auto
  with $A1$ $A2$ have $f(x) \in f(A)$
  using func_imagedef by auto
  then show $f(A) \neq \emptyset$ by auto
qed

The next lemma allows to prove statements about the values in the domain of a function given a statement about values in the range.

**lemma func1_1_L15B:**

assumes $f: X \to Y$ and $A \subseteq X$ and $\forall y \in f(A). P(y)$

shows $\forall x \in A. P(f(x))$

using assms func_imagedef by simp

An image of an image is the image of a composition.

**lemma func1_1_L15C:** assumes $A1: f: X \to Y$ and $A2: g: Y \to Z$

and $A3: A \subseteq X$

shows $g(f(A)) = \{g(f(x)). x \in A\}$

$g(f(A)) = (g \circ f)(A)$

proof -
  from $A1$ $A3$ have $\{f(x). x \in A\} \subseteq Y$
  using apply_funtype by auto
  with $A2$ have $g(f(x)). x \in A\} = \{g(f(x)). x \in A\}$
  using func_imagedef by auto
  with $A1$ $A3$ show $I: g(f(A)) = \{g(f(x)). x \in A\}$
  using func_imagedef by simp
  from $A1$ $A3$ have $\forall x \in A. (g \circ f)(x) = g(f(x))$
  using comp_fun_apply by auto
  with $I$ have $g(f(A)) = \{(g \circ f)(x). x \in A\}$
  by simp
  moreover from $A1$ $A2$ $A3$ have $(g \circ f)(A) = \{(g \circ f)(x). x \in A\}$
  using comp_fun func_imagedef by blast
  ultimately show $g(f(A)) = (g \circ f)(A)$
  by simp
qed

What is the image of a set defined by a meta-fuction?
lemma func1_1_L17: assumes A1: \( f \in X \rightarrow Y \) and A2: \( \forall x \in A. \ b(x) \in X \)
shows \( f(\{b(x). \ x \in A\}) = \{f(b(x)). \ x \in A\} \)
proof -
  from A2 have \( \{b(x). \ x \in A\} \subseteq X \) by auto
  with A1 show thesis using func_imagedef by auto
qed

What are the values of composition of three functions?

lemma func1_1_L18: assumes A1: \( f:A \rightarrow B \) \( g:B \rightarrow C \) \( h:C \rightarrow D \)
and A2: \( x \in A \)
shows \( (h \circ g \circ f)(x) \in D \)
(\( h \circ g \circ f)(x) = h(g(f(x))) \)
proof -
  from A1 A2 have \( (h \circ g \circ f) : A \rightarrow D \)
  using comp_fun by blast
  with A2 show \( (h \circ g \circ f)(x) \in D \) using apply_funtype by simp
  from A1 A2 have \( (h \circ g \circ f)(x) = h(g(f(x))) \)
  using comp_fun comp_fun_apply by blast
  with A1 A2 show \( (h \circ g \circ f)(x) = h(g(f(x))) \)
  using comp_fun_apply by simp
qed

A composition of functions is a function. This is a slight generalization of standard Isabelle's comp_fun.

lemma comp_fun_subset:
  assumes A1: \( g:A \rightarrow B \) and A2: \( f:C \rightarrow D \) and A3: \( B \subseteq C \)
shows \( f \circ g : A \rightarrow D \)
proof -
  from A1 A3 have \( g:A \rightarrow C \) by (rule func1_1_L1B)
  with A2 show \( f \circ g : A \rightarrow D \) using comp_fun by simp
qed

This lemma supersedes the lemma comp_eq_id_iff in Isabelle/ZF. Contributed by Victor Porton.

lemma comp_eq_id_iff1: assumes A1: \( g:B \rightarrow A \) and A2: \( f:A \rightarrow C \)
shows \( (\forall y \in B. \ f(g(y)) = y) \iff f \circ g = id(B) \)
proof -
  from assms have \( f \circ g : B \rightarrow C \) and \( id(B) : B \rightarrow B \)
  using comp_fun id_type by auto
  then have \( (\forall y \in B. \ (f \circ g)y = id(B)(y)) \iff f \circ g = id(B) \)
  by (rule fun_extension_iff)
  moreover from A1 have \( \forall y \in B. \ (f \circ g)y = f(gy) \) and \( \forall y \in B. \ id(B)(y) = y \)
  by auto
  ultimately show \( (\forall y \in B. \ f(gy) = y) \iff f \circ g = id(B) \) by simp
A lemma about a value of a function that is a union of some collection of functions.

**Lemma fun_Union_apply:**

Assumes $A1: \bigcup F : X \rightarrow Y$ and $A2: f \in F$ and $A3: f : A \rightarrow B$ and $A4: x \in A$

Shows $(\bigcup F)(x) = f(x)$

**Proof** -

From $A3$ $A4$ have $(x, f(x)) \in f$ using `apply_Pair` by simp

With $A2$ have $(x, f(x)) \in \bigcup F$ by auto

With $A1$ show $(\bigcup F)(x) = f(x)$ using `apply_equality` by simp

qed

### 8.2 Functions restricted to a set

Standard Isabelle/ZF defines the notion restrict($f, A$) to mean a function (or relation) $f$ restricted to a set. This means that if $f$ is a function defined on $X$ and $A$ is a subset of $X$ then restrict($f, A$) is a function with the same values as $f$, but whose domain is $A$.

What is the inverse image of a set under a restricted function?

**Lemma func1_2_L1:**

Assumes $A1: f : X \rightarrow Y$ and $A2: B \subseteq X$

Shows restrict($f, B$)-(A) = f-(A) $\cap$ B

**Proof** -

Let $g = $ restrict($f, B$)

From $A1$ $A2$ have $g : B \rightarrow Y$

Using `restrict_type2` by simp

With $A2$ $A1$ show $g-(A) = f-(A) \cap B$

Using `func1_1_L15 restrict_if` by auto

qed

A criterion for when one function is a restriction of another. The lemma below provides a result useful in the actual proof of the criterion and applications.

**Lemma func1_2_L2:**

Assumes $A1: f : X \rightarrow Y$ and $A2: g \in A \rightarrow Z$

And $A3: A \subseteq X$ and $A4: f \cap A \times Z = g$

Shows $\forall x \in A . g(x) = f(x)$

**Proof** -

Fix $x$ assume $x \in A$

With $A2$ have $(x, g(x)) \in g$ using `apply_Pair` by simp

With $A4$ $A1$ show $g(x) = f(x)$ using `apply_iff` by auto

qed

Here is the actual criterion.

**Lemma func1_2_L3:**
assumes $A_1: f: X \rightarrow Y$ and $A_2: g: A \rightarrow Z$
and $A_3: A \subseteq X$ and $A_4: f \cap A \times Z = g$
shows $g = \text{restrict}(f, A)$

proof
from $A_4$ show $g \subseteq \text{restrict}(f, A)$ using restrict_iff by auto
show $\text{restrict}(f, A) \subseteq g$
proof
fix $z$
assume $A_5: z \in \text{restrict}(f, A)$$$
then obtain $x, y$ where $D_1: z \in f \land x \in A \land z = (x, y)$$$
using restrict_iff by auto
with $A_1$ have $y = f(x)$ using apply_iff by auto
with $A_1 A_2 A_3 A_4 D_1$ have $y = g(x)$ using func1_2_L2 by simp
with $A_2 D_1$ show $z \in g$ using apply_Pair by simp
qed

Which function space a restricted function belongs to?

lemma func1_2_L4:
assumes $A_1: f: X \rightarrow Y$ and $A_2: A \subseteq X$ and $A_3: \forall x \in A. f(x) \in Z$
shows $\text{restrict}(f, A) : A \rightarrow Z$
proof -
let $g = \text{restrict}(f, A)$
from $A_1 A_2$ have $g : A \rightarrow Y$
using restrict_type2 by simp
moreover {
fix $x$
assume $x \in A$
with $A_1 A_3$ have $g(x) \in Z$ using restrict by simp
ultimately show thesis by (rule Pi_type)
}
qed

A simpler case of func1_2_L4, where the range of the original and restricted function are the same.

corollary restrict_fun: assumes $A_1: f: X \rightarrow Y$ and $A_2: A \subseteq X$
shows $\text{restrict}(f, A) : A \rightarrow Y$
proof -
from assms have $\forall x \in A. f(x) \in Y$ using apply_funtype by auto
with assms show thesis using func1_2_L4 by simp
qed

A function restricted to its domain is itself.

lemma restrict_domain: assumes $f: X \rightarrow Y$
shows $\text{restrict}(f, X) = f$
proof -
have $\forall x \in X. \text{restrict}(f, X)(x) = f(x)$ using restrict by simp
with assms show thesis using func_eq restrict_fun by blast
qed

Suppose a function $f : X \rightarrow Y$ is defined by an expression $q$, i.e. $f =$
\{(x, y) : x \in X\}. Then a function that is defined by the same expression, but on a smaller set is the same as the restriction of \(f\) to that smaller set.

**lemma restrict_def_alt:** assumes \(A \subseteq X\)
shows \(\text{restrict}\{\{(x, q(x)). x \in X\}, A\} = \{(x, q(x)). x \in A\}\)

**proof** -
let \(Y = \{q(x). x \in X\}\)
let \(f = \{(x, q(x)). x \in X\}\)
have \(\forall x \in X. q(x) \in Y\) by blast
with assms have \(f : X \rightarrow Y\) using \(ZF\text{-fun_from_total}\) by simp
moreover from assms have \(\forall x \in A. q(x) \in Y\) by blast
then have \(\{(x, q(x)). x \in A\} : A \rightarrow Y\) using \(ZF\text{-fun_from_total}\) by simp
moreover from assms have \(\forall x \in A. \text{restrict}(f, A)(x) = \{(x, q(x)). x \in A\}(x)\)
using \(\text{restrict} ZF\text{-fun_from_tot_val1}\) by auto
ultimately show thesis by (rule \(\text{func_eq}\))
**qed**

A composition of two functions is the same as composition with a restriction.

**lemma comp_restrict:**
assumes \(A1: f : A \rightarrow B\) and \(A2: g : X \rightarrow C\) and \(A3: B \subseteq X\)
shows \(g \circ f = \text{restrict}(g, B) \circ f\)

**proof** -
from assms have \(g \circ f : A \rightarrow C\) using \(\text{comp_fun_subset}\) by simp
moreover from assms have \(\text{restrict}(g, B) \circ f : A \rightarrow C\)
using \(\text{restrict}_\text{fun comp_fun}\) by simp
moreover from \(A1\) have \(\forall x \in A. (g \circ f)(x) = (\text{restrict}(g, B) \circ f)(x)\)
using \(\text{comp_fun_apply apply_funtype restrict}\) by simp
ultimately show \(g \circ f = \text{restrict}(g, B) \circ f\)
by (rule \(\text{func_eq}\))
**qed**

A way to look at restriction. Contributed by Victor Porton.

**lemma right_comp_id_any:** shows \(r \circ \text{id}(c) = \text{restrict}(r, c)\)
unfolding \(\text{restrict_def}\) by auto

### 8.3 Constant functions

Constant functions are trivial, but still we need to prove some properties to shorten proofs.

We define \(\text{constant}(= c)\) functions on a set \(X\) in a natural way as \(\text{ConstantFunction}(X, c)\).

**definition**
\[
\text{ConstantFunction}(X, c) \equiv X \times \{c\}
\]
Constant function is a function (i.e. belongs to a function space).

**Lemma** func1_3_L1:
- **Assumes** A1: c ∈ Y
- **Shows** ConstantFunction(X, c) : X → Y
**Proof** -
  - From A1 have \( X \times \{c\} = \{(x, y) \in X \times Y. c = y\} \) by auto
  - With A1 show thesis using func1_1_L11A ConstantFunction_def by simp
**Qed**

Constant function is equal to the constant on its domain.

**Lemma** func1_3_L2:
- **Assumes** A1: x ∈ X
- **Shows** ConstantFunction(X, c)(x) = c
**Proof** -
  - Have ConstantFunction(X, c) ∈ X → {c} using func1_3_L1 by simp
  - Moreover from A1 have \( \langle x, c \rangle \in \text{ConstantFunction}(X, c) \) using ConstantFunction_def by simp
  - Ultimately show thesis using apply_iff by simp
**Qed**

Another way of looking at the constant function - it's a set of pairs \( \langle x, c \rangle \) as \( x \) ranges over \( X \).

**Lemma** const_fun_def_alt:
- **Shows** ConstantFunction(X, c) = \{\( \langle x, c \rangle \). x ∈ X\}
- **Unfolds** ConstantFunction_def by auto

If \( c \in A \) then the inverse image of \( A \) by the constant function \( x \mapsto c \) is the whole domain.

**Lemma** const_vimage_domain:
- **Assumes** c ∈ A
- **Shows** ConstantFunction(X, c)-(A) = X
**Proof** -
  - Let \( C = \text{ConstantFunction}(X, c) \)
  - Have \( C-(A) = \{x \in X. C(x) \in A\} \) using func1_3_L1 func1_1_L15 by blast
  - With assms show thesis using func1_3_L2 by simp
**Qed**

If \( c \) is not an element of \( A \) then the inverse image of \( A \) by the constant function \( x \mapsto c \) is empty.

**Lemma** const_vimage_empty:
- **Assumes** c ∉ A
- **Shows** ConstantFunction(X, c)-(A) = ∅
**Proof** -
  - Let \( C = \text{ConstantFunction}(X, c) \)
  - Have \( C-(A) = \{x \in X. C(x) \in A\} \) using func1_3_L1 func1_1_L15 by blast
  - With assms show thesis using func1_3_L2 by simp
**Qed**
8.4 Injections, surjections, bijections etc.

In this section we prove the properties of the spaces of injections, surjections and bijections that we can’t find in the standard Isabelle’s Perm.thy.

For injections the image of intersection is the intersection of images.

**lemma inj_image_inter:**

assumes A1: \( f \in \text{inj}(X,Y) \) and A2: \( A \subseteq X \) \( B \subseteq X \)

shows \( f(A \cap B) = f(A) \cap f(B) \)

**proof**

\[
\text{show } f(A \cap B) \subseteq f(A) \cap f(B) \text{ using image_Int_subset by simp}
\]

\[
\{ \text{ from A1 have } f:X \rightarrow Y \text{ using inj_def by simp } \}
\]

\[
\text{fix } y \text{ assume } y \in f(A) \cap f(B) \text{ then have } y \in f(A) \text{ and } y \in f(B) \text{ by auto}
\]

For injections the image of difference of two sets is the difference of images.

**lemma inj_image_diff:**

assumes A1: \( f \in \text{inj}(A,B) \) and A2: \( C \subseteq A \)

shows \( f(A \setminus C) = f(A) \setminus f(C) \)

**proof**

\[
\text{show } f(A \setminus C) \subseteq f(A) \setminus f(C) \text{ using inj_def by simp}
\]

\[
\text{proof}
\]

\[
\text{fix } y \text{ assume A3: } y \in f(A \setminus C)
\]

\[
\text{from A1 have } f:A \rightarrow B \text{ using inj_def by simp}
\]

\[
\text{moreover have } A \setminus C \subseteq A \text{ by auto}
\]

\[
\text{ultimately have } f(A \setminus C) = \{ f(x). x \in A \setminus C \}
\]

\[
\text{using func_imagedef by simp}
\]

\[
\text{with A3 obtain } x \text{ where I: } f(x) = y \text{ and } x \in A \setminus C
\]

\[
\text{by auto}
\]

\[
\text{hence } x \in A \text{ by auto}
\]

\[
\text{with } \langle f:A \rightarrow B \rangle \text{ I have } y \in f(A)
\]

\[
\text{using func_imagedef by auto}
\]

\[
\text{moreover have } y \notin f(C)
\]

\[
\text{proof -}
\]

\[
\{ \text{ assume } y \in f(C)
\]

\[
\text{with A2 } \langle f:A \rightarrow B \rangle \text{ obtain } x_0
\]

\[
\text{where II: } f(x_0) = y \text{ and } x_0 \in C
\]

\[
\text{using func_imagedef by auto}
\]

\[
\text{with A1 A2 I } \langle x \in A \rangle \text{ have}
\]

\[
\text{f } \in \text{ inj}(A,B) \text{ f(x) = f(x_0) x } \subseteq A \text{ x_0 } \subseteq A
\]

\[
\text{by auto}
\]

\[
\text{then have } x = x_0 \text{ by (rule inj_apply_equality)}
\]

\[
\text{with } \langle x \in A \setminus C \rangle \text{ x_0 } \in C \text{ have False by simp}
\]

\[
\text{ } \}
\]

\[
\text{ thus thesis by auto}
\]

\[
\text{qed}
\]

\[
\text{from A1 A2 show } f(A) \setminus f(C) \subseteq f(A) \setminus f(C)
\]

\[
\text{using inj_def diff_image_diff by auto}
\]

\[
\text{qed}
\]

For injections the image of intersection is the intersection of images.

**lemma inj_image_inter:**

assumes A1: \( f \in \text{inj}(X,Y) \) and A2: \( A \subseteq X \) \( B \subseteq X \)

shows \( f(A \cap B) = f(A) \cap f(B) \)

**proof**

\[
\text{show } f(A \cap B) \subseteq f(A) \cap f(B) \text{ using image_Int_subset by simp}
\]

\[
\{ \text{ from A1 have } f:X \rightarrow Y \text{ using inj_def by simp } \}
\]

\[
\text{fix } y \text{ assume } y \in f(A) \cap f(B) \text{ then have } y \in f(A) \text{ and } y \in f(B) \text{ by auto}
\]
with A2 \langle f:X \rightarrow Y \rangle obtain x_A x_B where 

x_A \in A x_B \in B and I: y = f(x_A) y = f(x_B) 

using func_imagedef by auto

with A2 have x_A \in X x_B \in X and f(x_A) = f(x_B) by auto

with A1 have x_A = x_B using inj_def by auto

moreover from A2 \langle f:X \rightarrow Y \rangle have f(A \cap B) = \{f(x). x \in A \cap B\}

using func_imagedef by blast

ultimately have f(x_A) \in f(A \cap B) by simp

with I have y \in f(A \cap B) by simp 

\} thus \( f(A) \cap f(B) \subseteq f(A \cap B) \) by auto

qed

For surjection from A to B the image of the domain is B.

lemma surj_range_image_domain: assumes A1: f \in \text{surj}(A,B)

shows f(A) = B

proof - 

from A1 have f(A) = \text{range}(f)

using surj_def range_image_domain by auto

with A1 show f(A) = B using surj_range

by simp 

qed

Surjections are functions that map the domain onto the codomain.

lemma surj_def_alt: shows \( \text{surj}(X,Y) = \{f \in X \rightarrow Y. \ f(X) = Y\} \)

proof

show \( \text{surj}(X,Y) \subseteq \{f \in X \rightarrow Y. \ f(X) = Y\} \)

using surj_range_image_domain unfolding surj_def by auto

show \( \{f \in X \rightarrow Y. \ f(X) = Y\} \subseteq \text{surj}(X,Y) \)

using range_image_domain fun_is_surj by auto

qed

Bijections are functions that preserve complements.

lemma bij_def_alt: 

shows \( \text{bij}(X,Y) = \{f \in X \rightarrow Y. \ \forall A \in \text{Pow}(X). \ f(X \setminus A) = Y \setminus f(A)\} \)

proof

let R = \{f \in X \rightarrow Y. \ \forall A \in \text{Pow}(X). \ f(X \setminus A) = Y \setminus f(A)\}

show bij(X,Y) \subseteq R

using inj_image_dif surj_range_image_domain surj_is_fun 

unfolding bij_def by auto

\{ fix f assume f \in R 

hence f:X \rightarrow Y and I: \ \forall A \in \text{Pow}(X). \ f(X \setminus A) = Y \setminus f(A) 

by auto 

\{ fix x_1 x_2 assume x_1 \in X x_2 \in X \ f(x_1) = f(x_2) 

with \langle f:X \rightarrow Y \rangle have 

f(x_1) = \{f(x_1)\} f(x_2) = \{f(x_2)\} f(x_1) = f(x_2) 

using singleton_image by simp_all 

\{ assume x_1 \neq x_2 

from \langle f:X \rightarrow Y \rangle have f(X \setminus \{x_1\}) = \{f(t). \ t \in X \setminus \{x_1\}\} 

\} 

\} 

97
using func_imagedef by blast
with I \( x_2 \in X \) \( x_1 \in X \) \( x_1 \neq x_2 \) \( \{ f(x_1) \} = f(x_2) \)
  have \( f(x_2) \in Y \setminus f(x_2) \) by auto
with \( \{ f(x_2) \} = \{ f(x_2) \} \)
  have False by auto
} hence \( x_1 = x_2 \) by auto
}\[
\text{moreover from } I \text{ have } f(X \setminus \emptyset) = Y \setminus f(\emptyset) \text{ by blast}
\]
with \( f:X \to Y \) have \( f \in \text{inj}(X,Y) \) unfolding inj_def
by auto
ultimately have \( f \in \text{bij}(X,Y) \) unfolding bij_def
by simp
qed

For injections the inverse image of an image is the same set.

lemma inj_vimage_image: assumes \( f \in \text{inj}(X,Y) \) and \( A \subseteq X \)
  shows \( f^{-1}(f(A)) = A \)
proof -
  have \( f^{-1}(f(A)) = (\text{converse}(f) \circ f)(A) \)
    using vimage_converse image_comp by simp
  with assms show thesis using left_comp_inverse image_id_same
    by simp
qed

For surjections the image of an inverse image is the same set.

lemma surj_image_vimage: assumes \( A1: f \in \text{surj}(X,Y) \) and \( A2: A \subseteq Y \)
  shows \( f(f^{-1}(A)) = A \)
proof -
  have \( f(f^{-1}(A)) = (f \circ \text{converse}(f))(A) \)
    using vimage_converse image_comp by simp
  with assms show thesis using right_comp_inverse image_id_same
    by simp
qed

A lemma about how a surjection maps collections of subsets in domain and range.

lemma surj_subsets: assumes \( A1: f \in \text{surj}(X,Y) \) and \( A2: B \subseteq \text{Pow}(Y) \)
  shows \( \{ f(U) \cdot U \in \{ f^{-1}(V) \cdot V \in B \} \} = B \)
proof
\[
\{ \text{fix } W \text{ assume } W \in \{ f(U) \cdot U \in \{ f^{-1}(V) \cdot V \in B \} \} \}
\text{then obtain } U \text{ where } I: U \in \{ f^{-1}(V) \cdot V \in B \} \text{ and } II: W = f(U) \text{ by auto}
\text{then obtain } V \text{ where } V \in B \text{ and } U = f^{-1}(V) \text{ by auto}
\text{with } II \text{ have } W = f(f^{-1}(V)) \text{ by simp}
\text{moreover from } \text{assms } \{ V \in B \} \text{ have } f \in \text{surj}(X,Y) \text{ and } V \subseteq Y \text{ by auto}
\text{ultimately have } W = V \text{ using surj_image_vimage by simp}
\text{with } \{ V \in B \} \text{ have } W \in B \text{ by simp}
\} \text{ thus } \{ f(U) \cdot U \in \{ f^{-1}(V) \cdot V \in B \} \} \subseteq B \text{ by auto}
\{ \text{fix } W \text{ assume } W \in B
\text{let } U = f^{-1}(W) \}
\]
98
from \( \forall \in B \) have \( U \in \{ f^{-1}(V). V \in B \} \) by auto
moreover from \( A1 A2 \) \( \forall \in B \) have \( W = f(U) \) using \( \text{surj_image_vimage} \) by auto
ultimately have \( W \in \{ f(U). U \in \{ f^{-1}(V). V \in B \} \} \) by auto
thus \( B \subseteq \{ f(U). U \in \{ f^{-1}(V). V \in B \} \} \) by auto
qed

Restriction of an bijection to a set without a point is a a bijection.

**lemma** bij_restrict_rem:
assumes \( A1: f \in \text{bij}(A,B) \) and \( A2: a \in A \)
shows \( \text{restrict}(f, A\setminus\{a\}) \in \text{bij}(A\setminus\{a\}, B\setminus\{f(a)\}) \)
**proof** -
let \( C = A\setminus\{a\} \)
from \( A1 \) have \( f \in \text{inj}(A,B) \) \( C \subseteq A \)
using \( \text{bij_def} \) by auto
then have \( \text{restrict}(f,C) \in \text{bij}(C, f(C)) \)
using \( \text{restrict_bij} \) by simp
moreover have \( f(C) = B\setminus\{f(a)\} \)
**proof** -
from \( A2 \) \( f \in \text{inj}(A,B) \) have \( f(C) = f(A)\setminus\{f(a)\} \)
using \( \text{inj_image_dif} \) by simp
moreover from \( A1 \) have \( f(A) = B \)
using \( \text{bij_def} \) \( \text{surj_range_image_domain} \) by auto
moreover from \( A1 A2 \) have \( f\{a\} = \{f(a)\} \)
using \( \text{bij_is_fun} \) \( \text{singleton_image} \) by blast
ultimately show \( f(C) = B\setminus\{f(a)\} \) by simp
qed
ultimately show \( \text{thesis} \) by simp
qed

The domain of a bijection between \( X \) and \( Y \) is \( X \).

**lemma** domain_of_bij:
assumes \( A1: f \in \text{bij}(X,Y) \) shows \( \text{domain}(f) = X \)
**proof** -
from \( A1 \) have \( f:X \rightarrow Y \) using \( \text{bij_is_fun} \) by simp
then show \( \text{domain}(f) = X \) using \( \text{func1_1_L1} \) by simp
qed

The value of the inverse of an injection on a point of the image of a set belongs to that set.

**lemma** inj_inv_back_in_set:
assumes \( A1: f \in \text{inj}(A,B) \) and \( A2: C \subseteq A \) and \( A3: y \in f(C) \)
shows \( \text{converse}(f)(y) \in C \)
\( f(\text{converse}(f)(y)) = y \)
**proof** -
from \( A1 \) have \( I: f:X \rightarrow Y \) using \( \text{inj_is_fun} \) by simp
with \( A2 A3 \) obtain \( x \) where \( II: x \in C \quad y = f(x) \)
using \( \text{func_imagedef} \) by auto

99
with A1 A2 show \( \text{converse}(f)(y) \in C \) using \text{left_inverse} by auto
from A1 A2 I II show \( f(\text{converse}(f)(y)) = y \) using \text{func1_1_L5A right_inverse} by auto
qed

For a bijection between \( Y \) and \( X \) and a set \( A \subseteq X \) an element \( y \in Y \) is in the image \( f(A) \) if and only if \( f^{-1}(y) \) is an element of \( A \). Note this is false with the weakened assumption that \( f \) is an injection, for example consider \( f : \{0,1\} \rightarrow \mathbb{N}, f(n) = n + 1 \) and \( y = 3 \). Then \( f^{-1} : \{1,2\} \rightarrow \{0,1\} \) and (since 3 is not in the domain of the inverse function) \( f^{-1}(3) = \emptyset = 0 \in \{0,1\} \), but 3 is not in the image \( f([0,1]) \).

\begin{lemma}
\textbf{bij_val_image_vimage:} assumes \( f \in \text{bij}(X,Y) \) \( A \subseteq X \) \( y \in Y \) shows \( y \in f(A) \iff \text{converse}(f)(y) \in A \)
proof
assume \( y \in f(A) \)
with assms(1,2) show \( \text{converse}(f)(y) \in A \)
unfolding \text{bij_def} using \text{inj_inv_back_in_set} by blast
next
assume \( \text{converse}(f)(y) \in A \)
with assms(1,3) have \( y \in \{f(x). x \in A\} \) using \text{right_inverse_bij}
by force
with assms(1,2) show \( y \in f(A) \) using \text{bij_is_fun func_imagedef}
by force
qed
\end{lemma}

For injections if a value at a point belongs to the image of a set, then the point belongs to the set.

\begin{lemma}
\textbf{inj_point_of_image:} assumes \( A1: f \in \text{inj}(A,B) \) and \( A2: C \subseteq A \) and \( A3: x \in A \) and \( A4: f(x) \in f(C) \) shows \( x \in C \)
proof
from A1 A2 A4 have \( \text{converse}(f)(f(x)) \in C \)
using \text{inj_inv_back_in_set} by simp
moreover from A1 A3 have \( \text{converse}(f)(f(x)) = x \)
using \text{left_inverse_eq} by simp
ultimately show \( x \in C \) by simp
qed
\end{lemma}

For injections the image of intersection is the intersection of images.

\begin{lemma}
\textbf{inj_image_of_Inter:} assumes \( A1: f \in \text{inj}(A,B) \) and \( A2: I \neq \emptyset \) and \( A3: \forall i \in I. \ P(i) \subseteq A \) shows \( f(\bigcap_{i \in I} P(i)) = ( \bigcap_{i \in I} f(P(i)) ) \)
proof
from A1 A2 A3 show \( f(\bigcap_{i \in I} P(i)) \subseteq ( \bigcap_{i \in I} f(P(i)) ) \)
using \text{inj_is_fun image_of_Inter} by auto
from A1 A2 A3 have \( f: A \rightarrow B \) and \( ( \bigcap_{i \in I} P(i)) \subseteq A \)
using inj_is_fun ZF1_1_L7 by auto
then have I: f(\bigcap_i \in I. P(i)) = \{ f(x). x \in ( \bigcap_i \in I. P(i) ) \}
  using func_imagedef by simp
{ fix y assume A4: y \in ( \bigcap_i \in I. f(P(i)) )
  let x = converse(f)(y)
  from A2 obtain i_0 where i_0 \in I by auto
  with A1 A4 have II: y \in range(f) using inj_is_fun func1_1_L6
        by auto
  with A1 II have III: f(x) = y using right_inverse by simp
  from A1 II have IV: x \in A using inj_converse_fun apply_funtype
        by blast
  { fix i assume i\in I
    with A3 A4 III have P(i) \subseteq A and f(x) \in f(P(i))
        by auto
    with A1 IV have x \in P(i) using inj_point_of_image
        by blast
  } then have \forall i\in I. x \in P(i) by simp
  with A2 I have f(x) \in f( \bigcap_i \in I. P(i) )
    by auto
  with III have y \in f( \bigcap_i \in I. P(i) ) by simp
  } then show ( \bigcap_i \in I. f(P(i)) ) \subseteq f( \bigcap_i \in I. P(i) )
    by auto
qed

An injection is injective onto its range. Suggested by Victor Porton.

lemma inj_inj_range: assumes f \in inj(A,B)
  shows f \in inj(A,range(f))
  using assms inj_def range_of_fun by auto

An injection is a bijection on its range. Suggested by Victor Porton.

lemma inj_bij_range: assumes f \in inj(A,B)
  shows f \in bij(A,range(f))
proof -
  from assms have f \in surj(A,range(f)) using inj_def fun_is_surj
    by auto
  with assms show thesis using inj_inj_range bij_def by simp
qed

A lemma about extending a surjection by one point.

lemma surj_extend_point:
  assumes A1: f \in surj(X,Y) and A2: a\notin X and
  A3: g = f \cup \{(a,b)\}
  shows g \in surj(X\cup\{a\},Y\cup\{b\})
proof -
  from A1 A2 A3 have g : X\cup\{a\} \to Y\cup\{b\}
    using surj_def func1_1_L1D by simp
  moreover have \forall y \in Y\cup\{b\}. \exists x \in X\cup\{a\}. y = g(x)
    proof
      fix y assume y \in Y \cup \{b\}
    qed
  qed

then have $y \in Y$ \lor y = b$ by auto
moreover
\{ assume $y \in Y$
with A1 obtain $x$ where $x \in X$ and $y = f(x)$
\} using surj_def by auto
with A1 A2 A3 have $x \in X \cup \{a\}$ and $y = g(x)$
\} using surj_def func1_1_L11D by auto
moreover
\{ assume $y = b$
with A1 A2 A3 have $y = g(a)$
\} using surj_def func1_1_L11D by auto
ultimately show \exists x \in X \cup \{a\}. y = g(x)
by auto
ultimately show $g \in \text{surj}(X \cup \{a\}, Y \cup \{b\})$
\}
ultimately show $g \in \text{surj}(X \cup \{a\}, Y \cup \{b\})$
\} using surj_def by auto
qed

A lemma about extending an injection by one point. Essentially the same as standard Isabelle’s inj_extend.

\textbf{lemma inj_extend_point:} \textit{assumes} $f \in \text{inj}(X,Y)$ a \notin X b \notin Y
\textit{shows} $(f \cup \{(a,b)\}) \in \text{inj}(X \cup \{a\}, Y \cup \{b\})$
\textit{proof} -
\textit{from} assm have \text{cons}((a,b),f) \in \text{inj}((a,b), (a \times X), (a \times Y) \cup \{a\})
\textit{using} assm inj_extend_\text{by simp}
moreover have \text{cons}((a,b),f) = f \cup \{(a,b)\} and
\text{cons}(a, X) = X \cup \{a\} and \text{cons}(b, Y) = Y \cup \{b\}
\textit{by auto}
ultimately show thesis by simp
\textbf{qed}

A lemma about extending a bijection by one point.

\textbf{lemma bij_extend_point:} \textit{assumes} $f \in \text{bij}(X,Y)$ a \notin X b \notin Y
\textit{shows} $(f \cup \{(a,b)\}) \in \text{bij}(X \cup \{a\}, Y \cup \{b\})$
\textit{using} assm surj_extend_point inj_extend_point bij_def
\textit{by simp}

A quite general form of the $a^{-1}b = 1$ implies $a = b$ law.

\textbf{lemma comp_inv_id_eq:}
\textit{assumes} A1: converse(b) 0 a = id(A) and
A2: a \subseteq A \times B b \in \text{surj}(A,B)
\textit{shows} a = b
\textit{proof} -
\textit{from} A1 have (b 0 converse(b)) 0 a = b 0 id(A)
\textit{using} comp_assoc by simp
\textit{with} A2 have id(B) 0 a = b 0 id(A)
\textit{using} right_comp_inverse by simp

102
moreover
from A2 have a ⊆ A × B and b ⊆ A × B
  using surj_def fun_subset_prod
  by auto
then have id(B) 0 a = a and b 0 id(A) = b
  using left_comp_id right_comp_id by auto
ultimately show a = b by simp
qed

A special case of comp_inv_id_eq - the $a^{-1}b = 1$ implies $a = b$ law for bijections.

lemma comp_inv_id_eq_bij:
  assumes A1: a ∈ bij(A,B) b ∈ bij(A,B) and
  A2: converse(b) 0 a = id(A)
  shows a = b
proof -
  from A1 have a ⊆ A × B and b ∈ surj(A,B)
    using bij_def surj_def fun_subset_prod
    by auto
  with A2 show thesis using comp_inv_id_eq by simp
qed

Converse of a converse of a bijection is the same bijection. This is a special case of converse_converse from standard Isabelle’s equalities theory where it is proved for relations.

lemma bij_converse_converse: assumes a ∈ bij(A,B)
  shows converse(converse(a)) = a
proof -
  from assms have a ⊆ A × B using bij_def surj_def fun_subset_prod by simp
  then show thesis using converse_converse by simp
qed

If a composition of bijections is identity, then one is the inverse of the other.

lemma comp_id_conv: assumes A1: a ∈ bij(A,B) b ∈ bij(B,A) and
  A2: b 0 a = id(A)
  shows a = converse(b) and b = converse(a)
proof -
  from A1 have a ∈ bij(A,B) and converse(b) ∈ bij(A,B) using bij_converse_bij
    by auto
  moreover from assms have converse(converse(b)) 0 a = id(A)
    using bij_converse_converse by simp
  ultimately show a = converse(b) by (rule comp_inv_id_eq_bij)
    with assms show b = converse(a) using bij_converse_converse by simp
qed

A version of comp_id_conv with weaker assumptions.
lemma comp_conv_id: assumes A1: \(a \in \text{bij}(A, B)\) and A2: \(b: B \to A\)
shows \(b \in \text{bij}(B, A)\) and \(a = \text{converse}(b)\) and \(b = \text{converse}(a)\)
proof -
  have \(b \in \text{surj}(B, A)\)
proof -
  have \(\forall x \in A. \exists y \in B. b(y) = x\)
  proof -
  { fix \(x\) assume \(x \in A\)
    let \(y = a(x)\)
    from A1 A3 \(<x \in A> y \in B\ and \ b(y) = x\)
    using bij_def inj_def apply_funtype by auto
    hence \(\exists y \in B. b(y) = x\) by auto
  } thus thesis by simp
qed
with A2 show \(b \in \text{surj}(B, A)\) using surj_def by simp
qed
moreover have \(b \in \text{inj}(B, A)\)
proof -
  have \(\forall w \in B. \forall y \in B. b(w) = b(y) \to w = y\)
  proof -
  { fix \(w\) \(y\) assume \(w \in B\ \ y \in B\ and \ I: b(w) = b(y)\)
    from A1 have \(a \in \text{surj}(A, B)\) unfolding bij_def by simp
    with \(<w \in B>\) obtain \(x_w\) where \(x_w \in A\) and II: \(a(x_w) = w\)
    using surj_def by auto
    with I have \(b(a(x_w)) = b(y)\) by simp
    moreover from \(a \in \text{surj}(A, B)\) \(<y \in B>\) obtain \(x_y\) where
    \(x_y \in A\) and III: \(a(x_y) = y\)
    using surj_def by auto
    moreover from A3 \(<x_w \in A> \ <x_y \in A> have b(a(x_w)) = x_w and b(a(x_y)) = x_y\)
    by auto
    ultimately have \(x_w = x_y\) by simp
    with II III have \(w = y\) by simp
  } thus thesis by auto
qed
with A2 show \(b \in \text{inj}(B, A)\) using inj_def by auto
qed
ultimately show \(b \in \text{bij}(B, A)\) using bij_def by simp
from assms have \(b \circ a = \text{id}(A)\) using bij_def inj_def comp_eq_id_iff1
by auto
with A1 \(<b \in \text{bij}(B, A)> show a = \text{converse}(b)\) and \(b = \text{converse}(a)\)
using comp_id_conv by auto
qed

For a surjection the union if images of singletons is the whole range.

lemma surj_singleton_image: assumes A1: \(f \in \text{surj}(X, Y)\)
shows \((\bigcup_{x \in X. \{f(x)\}}) = Y\)
proof
from A1 show \( (\bigcup x \in X. \{f(x)\}) \subseteq Y \)
using surj_def apply_funtype by auto

next
{ fix y assume y \in Y
  with A1 have y \in (\bigcup x \in X. \{f(x)\})
  using surj_def by auto
} then show Y \subseteq (\bigcup x \in X. \{f(x)\}) by auto
qed

8.5 Functions of two variables

In this section we consider functions whose domain is a cartesian product
of two sets. Such functions are called functions of two variables (although
really in ZF all functions admit only one argument). For every function of
two variables we can define families of functions of one variable by fixing the
other variable. This section establishes basic definitions and results for this
concept.

We can create functions of two variables by combining functions of one
variable.

lemma cart_prod_fun: assumes f_1:X_1 \to Y_1  f_2:X_2 \to Y_2  and
g = \{⟨p,⟨f_1(fst(p)),f_2(snd(p))⟩⟩. p \in X_1 \times X_2\}
shows g: X_1 \times X_2 \to Y_1 \times Y_2 using assms apply_funtype ZF_fun_from_total
by simp

A reformulation of cart_prod_fun above in a slightly different notation.

lemma prod_fun: assumes f:X_1 \to X_3  g:X_2 \to X_4
shows \{⟨⟨x,y⟩,⟨fx,gy⟩⟩. ⟨x,y⟩\in X_1 \times X_3\} \in X_1 \times X_3 \to Y_1 \times Y_2
proof -
  have \{⟨⟨x,y⟩,⟨fx,gy⟩⟩. ⟨x,y⟩\in X_1 \times X_3\} = \{⟨p,⟨f(fst(p)),g(snd(p))⟩⟩. p \in
X_1 \times X_3\}
  by auto
  with assms show thesis using cart_prod_fun by simp
qed

Product of two surjections is a surjection.

theorem prod_functions_surj:
  assumes f:\in surj(A,B)  g:\in surj(C,D)
shows \{(a_1,a_2),(fa_1,ga_2). ⟨a_1,a_2⟩\in A \times C\} \in surj(A \times C,B \times D)
proof -
  let h = \{(⟨x,y⟩,⟨f(x),g(y)⟩). ⟨x,y⟩\in A \times C\}
  from assms have fun: f:A \to Bg:C \to D unfolding surj_def by auto
  then have pfun: h : A \times C \to B \times D using prod_fun by auto
  \{ fix b assume b\in B \times D
  then obtain b_1 b_2 where b=⟨b_1,b_2⟩ b_1\in B b_2\in D by auto
  with assms obtain a_1 a_2 where f(a_1)=b_1 g(a_2)=b_2 a_1\in A a_2\in C
unfolding surj_def by blast
hence \( \langle a_1, a_2 \rangle \langle b_1, b_2 \rangle \in h \) by auto
with pfun have \( h \langle a_1, a_2 \rangle = \langle b_1, b_2 \rangle \) using apply_equality by auto
with \( \langle b = \langle b_1, b_2 \rangle \rangle \langle a \in A \rangle \langle a_2 \in C \rangle \) have \( \exists a \in A \times C \). \( h(a) = b \) by auto
hence \( \forall b \in B \times D \). \( \exists a \in A \times C \). \( h(a) = b \) by auto
with pfun show thesis unfolding surj_def by auto
qed

For a function of two variables created from functions of one variable as in cart_prod_fun above, the inverse image of a cartesian product of sets is the cartesian product of inverse images.

lemma cart_prod_fun_vimage: assumes \( f_1 : X_1 \rightarrow Y_1 \) \( f_2 : X_2 \rightarrow Y_2 \) and \( g = \{ (p, \langle f_1(fst(p)), f_2(snd(p)) \rangle ) \mid p \in X_1 \times X_2 \} \) shows \( g^{-1}(A_1 \times A_2) = f_1^{-1}(A_1) \times f_2^{-1}(A_2) \) proof -
from \( \text{assms} \) have \( g : X_1 \times X_2 \rightarrow Y_1 \times Y_2 \) using cart_prod_fun
by simp
then have \( g^{-1}(A_1 \times A_2) = \{ p \in X_1 \times X_2 \mid g(p) \in A_1 \times A_2 \} \) using func_1_1_L15
by simp
with \( \text{assms} \langle g : X_1 \times X_2 \rightarrow Y_1 \times Y_2 \rangle \) show \( g^{-1}(A_1 \times A_2) = f_1^{-1}(A_1) \times f_2^{-1}(A_2) \)
using ZF_fun_from_tot_val func_1_1_L15 by auto
qed

For a function of two variables defined on \( X \times Y \), if we fix an \( x \in X \) we obtain a function on \( Y \). Note that if \( \text{domain}(f) \) is \( X \times Y \), \( \text{range}(\text{domain}(f)) \) extracts \( Y \) from \( X \times Y \).

definition
\( \text{Fix1stVar}(f, x) \equiv \{ (y, f(x, y)) \mid y \in \text{range}(\text{domain}(f)) \} \)

For every \( y \in Y \) we can fix the second variable in a binary function \( f : X \times Y \rightarrow Z \) to get a function on \( X \).

definition
\( \text{Fix2ndVar}(f, y) \equiv \{ (x, f(x, y)) \mid x \in \text{domain}(\text{domain}(f)) \} \)

We defined \( \text{Fix1stVar} \) and \( \text{Fix2ndVar} \) so that the domain of the function is not listed in the arguments, but is recovered from the function. The next lemma is a technical fact that makes it easier to use this definition.

lemma fix_var_fun_domain: assumes \( A_1 : f : X \times Y \rightarrow Z \)
shows \( x \in X \rightarrow \text{Fix1stVar}(f, x) = \{ (y, f(x, y)) \mid y \in Y \} \)
\( y \in Y \rightarrow \text{Fix2ndVar}(f, y) = \{ (x, f(x, y)) \mid x \in X \} \)
proof -
from \( \text{assms} \) have \( I : \text{domain}(f) = X \times Y \) using func_1_1_L1 by simp
\{ assume \( x \in X \)
with \( I \) have \( \text{range}(\text{domain}(f)) = Y \) by auto

qed

106
then have $\text{Fix1stVar}(f,x) = \{(y,f(x,y)). y \in Y\}$
    using $\text{Fix1stVar_def}$ by simp

  } then show $x \in X \longrightarrow \text{Fix1stVar}(f,x) = \{(y,f(x,y)). y \in Y\}$
        by simp
  { assume $y \in Y$
    with $I$ have $\text{domain}(\text{domain}(f)) = X$ by auto
    then have $\text{Fix2ndVar}(f,y) = \{(x,f(x,y)). x \in X\}$
        using $\text{Fix2ndVar_def}$ by simp
  } then show $y \in Y \longrightarrow \text{Fix2ndVar}(f,y) = \{(x,f(x,y)). x \in X\}$
        by simp
qed

If we fix the first variable, we get a function of the second variable.

lemma $\text{fix\_1st\_var\_fun}$: assumes $A1$: $f : X \times Y \rightarrow Z$ and $A2$: $x \in X$
  shows $\text{Fix1stVar}(f,x) : Y \rightarrow Z$
proof -
  from $A1$ $A2$ have $\forall y \in Y. f(x,y) \in Z$
    using $\text{apply\_funtype}$ by simp
    then have $\{(y,f(x,y)). y \in Y\} : Y \rightarrow Z$ using $\text{ZF\_fun\_from\_total}$ by simp
    with $A1$ $A2$ show $\text{Fix1stVar}(f,x) : Y \rightarrow Z$ using $\text{fix\_var\_fun\_domain}$ by simp
qed

If we fix the second variable, we get a function of the first variable.

lemma $\text{fix\_2nd\_var\_fun}$: assumes $A1$: $f : X \times Y \rightarrow Z$ and $A2$: $y \in Y$
  shows $\text{Fix2ndVar}(f,y) : X \rightarrow Z$
proof -
  from $A1$ $A2$ have $\forall x \in X. f(x,y) \in Z$
    using $\text{apply\_funtype}$ by simp
    then have $\{(x,f(x,y)). x \in X\} : X \rightarrow Z$
        using $\text{ZF\_fun\_from\_total}$ by simp
    with $A1$ $A2$ show $\text{Fix2ndVar}(f,y) : X \rightarrow Z$
        using $\text{fix\_var\_fun\_domain}$ by simp
qed

What is the value of $\text{Fix1stVar}(f,x)$ at $y \in Y$ and the value of $\text{Fix2ndVar}(f,y)$
at $x \in X$?

lemma $\text{fix\_var\_val}$:
  assumes $A1$: $f : X \times Y \rightarrow Z$ and $A2$: $x \in X \quad y \in Y$
  shows $\text{Fix1stVar}(f,x)(y) = f(x,y)$
    $\text{Fix2ndVar}(f,y)(x) = f(x,y)$
proof -
  let $f_1 = \{(y,f(x,y)). y \in Y\}$
    let $f_2 = \{(x,f(x,y)). x \in X\}$
  from $A1$ $A2$ have $I$:
    $\text{Fix1stVar}(f,x) = f_1$
    $\text{Fix2ndVar}(f,y) = f_2$
    using $\text{fix\_var\_fun\_domain}$ by auto
moreover from A1 A2 have
  Fix1stVar(f,x) : Y → Z
  Fix2ndVar(f,y) : X → Z
  using fix_1st_var_fun fix_2nd_var_fun by auto
ultimately have f₁ : Y → Z and f₂ : X → Z
  by auto
with A2 have f₁(y) = f⟨x,y⟩ and f₂(x) = f⟨x,y⟩
  using ZF_fun_from_tot_val by auto
with I show
  Fix1stVar(f,x)(y) = f⟨x,y⟩
  Fix2ndVar(f,y)(x) = f⟨x,y⟩
  by auto
qed

Fixing the second variable commutes with restrictig the domain.

lemma fix_2nd_var_restr_comm:
  shows Fix2ndVar(restrict(f,X₁×Y),y) = restrict(Fix2ndVar(f,y),X₁)
proof -
  let g = Fix2ndVar(restrict(f,X₁×Y),y)
  let h = restrict(Fix2ndVar(f,y),X₁)
  from A3 have I: X₁×Y ⊆ X×Y by auto
  with A1 have II: restrict(f,X₁×Y) : X₁×Y → Z
    using restrict_type2 by simp
  with A2 have g : X₁ → Z
    using fix_2nd_var_fun by simp
  moreover
  from A1 A2 have III: Fix2ndVar(f,y) : X → Z
    using fix_2nd_var_fun by simp
  with A3 have h : X₁ → Z
    using restrict_type2 by simp
  moreover
  { fix z assume A4: z ∈ X₁
    with A2 I II have g(z) = f⟨z,y⟩
      using restrict fix_var_val by simp
    also from A1 A2 A3 A4 have f⟨z,y⟩ = h(z)
      using restrict fix_var_val by auto
    finally have g(z) = h(z) by simp
  } then have ∀z ∈ X₁. g(z) = h(z) by simp
  ultimately show g = h by (rule func_eq)
qed

The next lemma expresses the inverse image of a set by function with fixed
first variable in terms of the original function.

lemma fix_1st_var_vimage:
  assumes A1: f : X×Y → Z and A2: x∈X
  shows Fix1stVar(f,x)-(A) = {y∈Y. ⟨x,y⟩ ∈ f-(A)}
proof -
  from asssms have Fix1stVar(f,x)-(A) = {y∈Y. Fix1stVar(f,x)(y) ∈ A}
The next lemma expresses the inverse image of a set by function with fixed second variable in terms of the original function.

**Lemma fix\_2nd\_var\_vimage:**

**Assumptions:**

1. \( f : X \times Y \rightarrow Z \)
2. \( y \in Y \)

**Show:**

\[ \text{Fix2ndVar}(f,y)-(A) = \{ x \in X. \langle x,y \rangle \in f-(A) \} \]

**Proof:**

1. From assumptions have
2. \( \text{Fix2ndVar}(f,y)-(A) = \{ x \in X. \text{Fix2ndVar}(f,y)(x) \in A \} \)
3. Using \( \text{fix\_2nd\_var\_fun func1\_1\_L15 by blast} \)
4. With assumptions show thesis using \( \text{fix\_var\_val func1\_1\_L15 by auto} \)

**QED**

---

9 Semilattices and Lattices

**Theory Lattice_ZF imports Order_ZF_1a func1**

**Begin**

Lattices can be introduced in algebraic way as commutative idempotent \((x \cdot x = x)\) semigroups or as partial orders with some additional properties. These two approaches are equivalent. In this theory we will use the order-theoretic approach.

### 9.1 Semilattices

We start with a relation \( r \) which is a partial order on a set \( L \). Such situation is defined in Order_ZF as the predicate \( \text{IsPartOrder}(L,r) \).

A partially ordered \((L,r)\) set is a join-semilattice if each two-element subset of \( L \) has a supremum (i.e. the least upper bound).

**Definition**

\[ \text{IsJoinSemilattice}(L,r) \equiv r \subseteq L \times L \land \text{IsPartOrder}(L,r) \land (\forall x \in L. \forall y \in L. \text{HasAsupremum}(r,\{x,y\})) \]

A partially ordered \((L,r)\) set is a meet-semilattice if each two-element subset of \( L \) has an infimum (i.e. the greatest lower bound).

**Definition**

\[ \text{IsMeetSemilattice}(L,r) \equiv r \subseteq L \times L \land \text{IsPartOrder}(L,r) \land (\forall x \in L. \forall y \in L. \text{HasAnInfimum}(r,\{x,y\})) \]

A partially ordered \((L,r)\) set is a lattice if it is both join and meet-semilattice, i.e. if every two element set has a supremum (least upper bound) and infimum (greatest lower bound).
definition
IsAlattice (infixl \{is a lattice on\} 90) where
r \{is a lattice on\} L ≡ IsJoinSemilattice(L,r) ∧ IsMeetSemilattice(L,r)

Join is a binary operation whose value on a pair \langle x, y \rangle is defined as the supremum of the set \{x, y\}.

definition
Join(L,r) ≡ \{\langle p, \operatorname{Supremum}(r,\{\operatorname{fst}(p), \operatorname{snd}(p)\}) \rangle . p ∈ L×L\}

Meet is a binary operation whose value on a pair \langle x, y \rangle is defined as the infimum of the set \{x, y\}.

definition
Meet(L,r) ≡ \{\langle p, \operatorname{Infimum}(r,\{\operatorname{fst}(p), \operatorname{snd}(p)\}) \rangle . p ∈ L×L\}

Linear order is a lattice.

lemma lin_is_latt: assumes r⊆L×L and IsLinOrder(L,r) shows r \{is a lattice on\} L
proof -
  from assm(2) have IsPartOrder(L,r) using Order_ZF_1_L2 by simp
  with assms have IsMeetSemilattice(L,r) unfolding IsLinOrder_def IsMeetSemilattice_def
  using inf_sup_two_el(1) by auto
  moreover from assms \{IsPartOrder(L,r)\} have IsJoinSemilattice(L,r)
  unfolding IsLinOrder_def IsJoinSemilattice_def using inf_sup_two_el(3)
  by auto
  ultimately show thesis unfolding IsAlattice_def by simp
qed

In a join-semilattice join is indeed a binary operation.

lemma join_is_binop: assumes IsJoinSemilattice(L,r) shows Join(L,r) : L×L → L
proof -
  from assms(1) have Join(L,r) : L×L → L using join_is_binop by simp
  with assms(2,3,4) show thesis unfolding Join_def using ZF_fun_from_total by simp
qed

The value of Join(L,r) on a pair \langle x, y \rangle is the supremum of the set \{x, y\}, hence its is greater or equal than both.

lemma join_val:
  assumes IsJoinSemilattice(L,r) x∈L y∈L
  defines j ≡ Join(L,r)\langle x, y \rangle
  shows j∈L j = Supremum(r,\{x, y\}) \langle x, j \rangle ∈ r \langle y, j \rangle ∈ r
proof -
  from assm(1) have Join(L,r) : L×L → L using join_is_binop by simp
  with assms(2,3,4) show thesis unfolding Join_def using ZF_fun_from_total
by auto
from assms(2,3,4) \langle\Join(L,r) : L\times L \rightarrow L\rangle show j \in L using apply_funtype by simp
from assms(1,2,3) have r \subseteq L\times L antisym(r) HasAminimum(r,\bigcap z \in \{x,y\}. r(z))
  unfolding IsJoinSemilattice_def IsPartOrder_def HasAminimum_def by auto
  with \langle j = \text{Supremum}(r,\{x,y\}) \rangle show \langle x, j \rangle \in r and \langle y, j \rangle \in r
  using sup_in_space(2) by auto
qed

In a meet-semilattice meet is indeed a binary operation.

**lemma meet_is_binop:** assumes IsMeetSemilattice(L,r)
  shows Meet(L,r) : L\times L \rightarrow L
proof -
  from assms have \forall p \in L\times L. \text{Infimum}(r,\{\text{fst}(p), \text{snd}(p)\}) \in L
    unfolding IsMeetSemilattice_def IsPartOrder_def HasAnInfimum_def using inf_in_space
    by auto
  then show thesis unfolding Meet_def using ZF_fun_from_total by simp
qed

The value of \text{Meet}(L,r) on a pair \langle x, y \rangle is the infimum of the set \{x, y\}, hence
is less or equal than both.

**lemma meet_val:** assumes IsMeetSemilattice(L,r) x \in L y \in L
defines m \equiv \text{Meet}(L,r) \langle x, y \rangle
shows m \in L m = \text{Infimum}(r,\{x, y\}) \langle m, x \rangle \in r \langle m, y \rangle \in r
proof -
  from assms(1) have \text{Meet}(L,r) : L\times L \rightarrow L using meet_is_binop by simp
  with assms(2,3,4) show m = \text{Infimum}(r,\{x, y\}) unfolding \text{Meet_def} using ZF_fun_from_total_val
    by auto
  from assms(2,3,4) \langle\text{Meet}(L,r) : L\times L \rightarrow L\rangle show m \in L using apply_funtype by simp
  from assms(1,2,3) have r \subseteq L\times L antisym(r) HasAmaximum(r,\bigcap z \in \{x,y\}. r(z))
    unfolding IsMeetSemilattice_def IsPartOrder_def HasAmaximum_def by auto
  with \langle m = \text{Infimum}(r,\{x,y\}) \rangle show \langle m, x \rangle \in r and \langle m, y \rangle \in r
    using inf_in_space(2) by auto
qed

In a (nonempty) meet semi-lattice the relation down-directs the set.

**lemma meet_down_directs:** assumes IsMeetSemilattice(L,r) L \neq 0
shows r \{\text{down-directs}\} L
proof -
  \{ fix x y assume x \in L y \in L
    let m = \text{Meet}(L,r)\langle x, y \rangle

from assms(1) \langle x \in L, y \in L \rangle have \langle m, x \rangle \in r \land \langle m, y \rangle \in r \\
using meet_val by auto 
} 

hence \forall x \in L. \forall y \in L. \exists m \in L. \langle m, x \rangle \in r \land \langle m, y \rangle \in r 

by blast 

with assms(2) show thesis unfolding DownDirects_def by simp 

dqed

In a (nonempty) join semi-lattice the relation up-directs the set.

lemma join_up_directs: assumes IsJoinSemilattice(L,r) L \neq 0 
shows r {up-directs} L 
proof - 
{ 
fix x y assume x \in L y \in L 
let m = Join(L,r) \langle x, y \rangle 
from assms(1) \langle x \in L, y \in L \rangle have \langle m, x \rangle \in r \land \langle m, y \rangle \in r 
using join_val by auto 
} 

hence \forall x \in L. \forall y \in L. \exists m \in L. \langle x, m \rangle \in r \land \langle y, m \rangle \in r 

by blast 

with assms(2) show thesis unfolding UpDirects_def by simp 

dqed

The next locale defines a a notation for join-semilattice. We will use the \⊔ symbol rather than more common \lor to avoid confusion with logical "or".

locale join_semilatt = 
fixes L 
fixes r 
assumes joinLatt: IsJoinSemilattice(L,r) 
fixes join (infixl \⊔ 71) 
defines join_def [simp]: x \⊔ y \equiv Join(L,r) \langle x, y \rangle 
fixes sup (sup _ ) 
defines sup_def [simp]: sup A \equiv Supremum(r,A)

Join of the elements of the lattice is in the lattice.

lemma (in join_semilatt) join_props: assumes x \in L y \in L 
shows x \sqcup y \in L and x \sqcup y = sup \{x, y\} 
proof - 
from joinLatt assms have Join(L,r) \langle x, y \rangle \in L using join_is_binop apply_funtype 

by blast 

thus x \sqcup y \in L by simp 
from joinLatt assms have Join(L,r) \langle x, y \rangle = Supremum(r,\{x, y\}) using join_val(2) 

by simp 

thus x \sqcup y = sup \{x, y\} by simp 

dqed

Join is associative.

lemma (in join_semilatt) join_assoc: assumes x \in L y \in L z \in L 
shows x \sqcup (y \sqcup z) = (x \sqcup y) \sqcup z
proof -
  from joinLatt assms(2,3) have \(x \sqcup (y \sqcup z) = x \sqcup (\sup \{y, z\})\)
  using join_val(2)
  by simp
  also from assms joinLatt have \(\ldots = \sup \{x, \sup \{y, z\}\}\)
    unfolding IsJoinSemilattice_def IsPartOrder_def using join_props sup_inf_singl(2)
    by auto
  also have \(\ldots = \sup \{x, y, z\}\)
    proof -
      let \(T = \{\{x\}, \{y, z\}\}\)
      from joinLatt have \(r \subseteq L \times L\)
        antisym(r) trans(r)
      unfolding IsJoinSemilattice_def IsPartOrder_def by auto
      moreover from joinLatt assms have \(\forall T \in T. \text{HasAsupremum}(r, T)\)
      unfolding IsJoinSemilattice_def IsPartOrder_def HasAsupremum_def
      using sup_in_space(1) sup_inf_singl(2) by auto
      ultimately have \(\text{Supremum}(r, \{\text{Supremum}(r, T). T \in T\}) = \text{Supremum}(r, \bigcup T)\)
      by (rule sup_sup)
      moreover have \(\{\text{Supremum}(r, T). T \in T\} = \{\sup \{x\}, \sup \{y, z\}\}\)
        and \(\bigcup T = \{x, y, z\}\)
      by auto
      ultimately show \((\sup \{x, \sup \{y, z\}\}) = \sup \{x, y, z\}\) by simp
    qed
  also have \(\ldots = \sup \{x, y, z\}\)
    proof -
      let \(T = \{\{x, y\}, \{z\}\}\)
      from joinLatt have \(r \subseteq L \times L\)
        antisym(r) trans(r)
      unfolding IsJoinSemilattice_def IsPartOrder_def by auto
      moreover from joinLatt assms have \(\forall T \in T. \text{HasAsupremum}(r, T)\)
      unfolding IsJoinSemilattice_def IsPartOrder_def HasAsupremum_def
      using sup_in_space(1) sup_inf_singl(2) by auto
      ultimately have \(\text{Supremum}(r, \{\text{Supremum}(r, T). T \in T\}) = \text{Supremum}(r, \bigcup T)\)
      by (rule sup_sup)
      moreover have \(\{\text{Supremum}(r, T). T \in T\} = \{\sup \{x, y\}, \sup \{z\}\}\)
        and \(\bigcup T = \{x, y, z\}\)
      by auto
      ultimately show \((\sup \{x, \sup \{y, z\}\}) = \sup \{x, y, z\}\) by auto
    qed
also from assms joinLatt have ... = (sup \{x,y\}) \sqcup z
  unfolding IsJoinSemilattice_def IsPartOrder_def using join_props by auto
  also from joinLatt assms(1,2) have ... = x\sqcup y \sqcup z using join_val(2) by simp
  finally show x\sqcup(y\sqcup z) = x\sqcup y \sqcup z by simp
qed

Join is idempotent.

lemma (in join_semilatt) join_idempotent: assumes x \in L shows x\sqcup x = x
  using joinLatt assms join_val(2) IsJoinSemilattice_def IsPartOrder_def
  sup_inf_singl(2)
  by auto

The meet_semilatt locale is the dual of the join-semilattice locale defined above. We will use the \sqcap symbol to denote join, giving it a bit higher precedence.

locale meet_semilatt = 
  fixes L
  fixes r
  assumes meetLatt: IsMeetSemilattice(L,r)
  fixes join (infixl \sqcap 72)
  defines join_def [simp]: x \sqcap y \equiv Meet(L,r)\langle x,y \rangle
  fixes sup (inf _ )
  defines sup_def [simp]: inf A \equiv Infimum(r,A)

Meet of the elements of the lattice is in the lattice.

lemma (in meet_semilatt) meet_props: assumes x \in L y \in L
  shows x\sqcap y \in L and x\sqcap y = inf \{x,y\}
proof -
  from meetLatt assms have Meet(L,r)\langle x,y \rangle \in L using meet_is_binop apply_funtype
    by blast
  thus x\sqcap y \in L by simp
  from meetLatt assms have Meet(L,r)\langle x,y \rangle = Infimum(r,\{x,y\}) using meet_val(2)
  by blast
  thus x\sqcap y = inf \{x,y\} by simp
qed

Meet is associative.

lemma (in meet_semilatt) meet_assoc: assumes x \in L y \in L z \in L
  shows x\sqcap(y\sqcap z) = x\sqcap y \sqcap z
proof -
  from meetLatt assms(2,3) have x\sqcap(y\sqcap z) = x\sqcap (inf \{y,z\}) using meet_val
  by simp
  also from assms meetLatt have ... = inf \{inf \{x\}, inf \{y,z\}\}
    unfolding IsMeetSemilattice_def IsPartOrder_def using meet_props sup_inf_singl(4)
by auto
also have \ldots = \inf \{x,y,z\}
proof -
  let \(T = \{\{x\}, \{y,z\}\}\)
  from meetLatt have \(r \subseteq L \times L\) \text{ antisym}(r) trans(r)
    unfolding IsMeetSemilattice_def IsPartOrder_def by auto
moreover from meetLatt assms have \(\forall T \in T\). \text{HasAnInfimum}(r,T)
    unfolding IsMeetSemilattice_def IsPartOrder_def using sup_inf_singl(3)
by blast
moreover from meetLatt assms have \(\text{HasAnInfimum}(r,\{\text{Infimum}(r,T). T \in T\})\)
    unfolding IsMeetSemilattice_def IsPartOrder_def HasAnInfimum_def
using inf_in_space(1) sup_inf_singl(4) by auto
ultimately have \(\text{Infimum}(r,\{\text{Infimum}(r,T). T \in T\}) = \text{Infimum}(r,\bigcup T)\) by (rule inf_inf)
moreover have \(\{\text{Infimum}(r,T). T \in T\} = \{\inf \{x\}, \inf \{y,z\}\}\) and \(\bigcup T = \{x,y,z\}\)
by auto
ultimately show \(\{\inf \{x\}, \inf \{y,z\}\}\) = \(\text{inf} \{x,y,z\}\) by simp
qed
also have \ldots = \inf \{x,y\}, \inf \{z\}\)
proof -
  let \(T = \{\{x,y\}, \{z\}\}\)
  from meetLatt have \(r \subseteq L \times L\) \text{ antisym}(r) trans(r)
    unfolding IsMeetSemilattice_def IsPartOrder_def by auto
moreover from meetLatt assms have \(\forall T \in T\). \text{HasAnInfimum}(r,T)
    unfolding IsMeetSemilattice_def IsPartOrder_def using sup_inf_singl(3)
by blast
moreover from meetLatt assms have \(\text{HasAnInfimum}(r,\{\text{Infimum}(r,T). T \in T\})\)
    unfolding IsMeetSemilattice_def IsPartOrder_def HasAnInfimum_def
using inf_in_space(1) sup_inf_singl(4) by auto
ultimately have \(\text{Infimum}(r,\{\text{Infimum}(r,T). T \in T\}) = \text{Infimum}(r,\bigcup T)\) by (rule inf_inf)
moreover have \(\{\text{Infimum}(r,T). T \in T\} = \{\inf \{x,y\}, \inf \{z\}\}\) and \(\bigcup T = \{x,y,z\}\)
by auto
ultimately show \(\{\inf \{x,y\}, \inf \{z\}\}\) = \(\text{inf} \{x,y,z\}\) by auto
qed
also from assms meetLatt have \ldots = \(\text{inf} \{x,y\}\) \(\cap z\)
unfolding IsMeetSemilattice_def IsPartOrder_def using meet_props sup_inf_singl(4)
by auto
also from assms meetLatt have \ldots = (\text{inf} \{x,y\}) \(\cap z\)
unfolding IsMeetSemilattice_def IsPartOrder_def using meet_props by auto
also from meetLatt assms(1,2) have \ldots = \(x \cap y \cap z\) using meet_val by simp
finally show \(x \cap (y \cap z)\) = \(x \cap y \cap z\) by simp
qed
Meet is idempotent.

lemma (in meet_semilatt) meet_idempotent: assumes x∈L shows x⋒x = x

using meetLatt assms meet_val IsMeetSemilattice_def IsPartOrder_def
sup_inf_singl(4)
by auto

end

10 Order on natural numbers

theory NatOrder_ZF imports Nat_ZF_IML Order_ZF

begin

This theory proves that ≤ is a linear order on \( \mathbb{N} \). ≤ is defined in Isabelle’s Nat theory, and linear order is defined in Order_ZF theory. Contributed by Seo Sanghyeon.

10.1 Order on natural numbers

This is the only section in this theory.

To prove that ≤ is a total order, we use a result on ordinals.

lemma NatOrder_ZF_1_L1:
assumes a∈nat and b∈nat
shows a ≤ b ∨ b ≤ a
proof -
  from assms have I: Ord(a) ∧ Ord(b)
    using nat_into_Ord by auto
  then have a ∈ b ∨ a = b ∨ b ∈ a
    using Ord_linear by simp
  with I have a < b ∨ a = b ∨ b < a
    using ltI by auto
  with I show a ≤ b ∨ b ≤ a
    using le_iff by auto
qed

≤ is antisymmetric, transitive, total, and linear. Proofs by rewrite using definitions.

lemma NatOrder_ZF_1_L2:
  shows antisym(Le)
  trans(Le)
  Le {is total on} nat
  IsLinOrder(nat,Le)
proof -
show antisym(Le)
  using antisym_def Le_def le_anti_sym by auto
moreover show trans(Le)
  using trans_def Le_def le_trans by blast
moreover show Le {is total on} nat
  using IsTotal_def Le_def NatOrder_ZF_1_L1 by simp
ultimately show IsLinOrder(nat,Le)
  using IsLinOrder_def by simp
qed

The order on natural numbers is linear on every natural number. Recall that each natural number is a subset of the set of all natural numbers (as well as a member).

lemma natord_lin_on_each_nat:
  assumes A1: n ∈ nat
  shows IsLinOrder(n,Le)
proof -
  from A1 have n ⊆ nat using nat_subset_nat
    by simp
  then show thesis using NatOrder_ZF_1_L2 ord_linear_subset
    by blast
qed

end

11 Binary operations

theory func_ZF imports func1
begin

In this theory we consider properties of functions that are binary operations, that is they map $X \times X$ into $X$.

11.1 Lifting operations to a function space

It happens quite often that we have a binary operation on some set and we need a similar operation that is defined for functions on that set. For example once we know how to add real numbers we also know how to add real-valued functions: for $f, g : X \rightarrow \mathbb{R}$ we define $(f + g)(x) = f(x) + g(x)$. Note that formally the $+$ means something different on the left hand side of this equality than on the right hand side. This section aims at formalizing this process. We will call it "lifting to a function space", if you have a suggestion for a better name, please let me know.

Since we are writing in generic set notation, the definition below is a bit complicated. Here it what it says: Given a set $X$ and another set $f$ (that represents a binary function on $X$) we are defining $f$ lifted to function space
over $X$ as the binary function (a set of pairs) on the space $F = X \to \text{range}(f)$ such that the value of this function on pair $(a, b)$ of functions on $X$ is another function $c$ on $X$ with values defined by $c(x) = f(a(x), b(x))$.

**definition**
Lift2FcnSpce (infix {lifted to function space over} 65) where
\[ f \text{ {lifted to function space over} } X \equiv \{ (p, \{(x, f(fst(p)(x), snd(p)(x))). x \in X\}). p \in (X \to \text{range}(f)) \times (X \to \text{range}(f))\}

The result of the lift belongs to the function space.

**lemma** func_ZF_1_L1:
assumes
\( A1: f : Y \times Y \to Y \)
and \( A2: p \in (X \to \text{range}(f)) \times (X \to \text{range}(f)) \)
sows
\( \{ (x, f(fst(p)(x), snd(p)(x))). x \in X\} : X \to \text{range}(f) \)
proof -
\begin{itemize}
\item have \( \forall x \in X. f(fst(p)(x), snd(p)(x)) \in \text{range}(f) \)
\end{itemize}
proof
\begin{itemize}
\item fix \( x \) assume \( x \in X \)
\item let \( p = (fst(p)(x), snd(p)(x)) \)
\item from \( A2 \ \langle x \in X \rangle \) have \( \langle x \rangle \in \text{range}(f) \) \( \langle x \rangle \in \text{range}(f) \)
\end{itemize}
using apply_type by auto
\begin{itemize}
\item with \( A1 \) have \( p \in Y \times Y \)
\item using func1_1_L5B by blast
\end{itemize}
\begin{itemize}
\item with \( A1 \) have \( \langle p, f(p) \rangle \in f \)
\end{itemize}
using apply_Pair by simp
\begin{itemize}
\item with \( A1 \) show \( f(p) \in \text{range}(f) \)
\end{itemize}
using rangeI by simp
\begin{itemize}
\item then show thesis using ZF_fun_from_total by simp
\end{itemize}
qed

The values of the lift are defined by the value of the liftee in a natural way.

**lemma** func_ZF_1_L2:
assumes \( A1: f : Y \times Y \to Y \)
and \( A2: p \in (X \to \text{range}(f)) \times (X \to \text{range}(f)) \) and \( A3: x \in X \)
and \( A4: P = \{ (x, f(fst(p)(x), snd(p)(x))). x \in X\} \)
sows \( P(x) = f(fst(p)(x), snd(p)(x)) \)
proof -
\begin{itemize}
\item from \( A1 \) \( A2 \) have \( \{ (x, f(fst(p)(x), snd(p)(x))). x \in X\} : X \to \text{range}(f) \)
\item using func_ZF_1_L1 by simp
\item with \( A4 \) have \( P : X \to \text{range}(f) \) by simp
\item with \( A3 \) \( A4 \) show \( P(x) = f(fst(p)(x), snd(p)(x)) \)
\end{itemize}
using ZF_fun_from_tot_val by simp
\begin{itemize}
\item qed
\end{itemize}

118
Function lifted to a function space results in function space operator.

**theorem** func_ZF_1_L3:
assumes f : Y × Y → Y
and F = f {lifted to function space over} X
shows F : (X → range(f)) × (X → range(f)) → (X → range(f))
using assms Lift2FcnSpce_def func_ZF_1_L1 ZF_fun_from_total by simp

The values of the lift are defined by the values of the liftee in the natural way.

**theorem** func_ZF_1_L4:
assumes A1: f : Y × Y → Y
and A2: F = f {lifted to function space over} X
and A3: s: X → range(f) r: X → range(f)
and A4: x ∈ X
shows (F⟨s, r⟩)(x) = f⟨s(x), r(x)⟩
proof -
let p = ⟨s, r⟩
let P = {(x, f⟨fst(p)(x), snd(p)(x)⟩). x ∈ X}
from A1 A3 A4 have
  f : Y × Y → Y  p ∈ (X → range(f)) × (X → range(f))
x ∈ X  P = {(x, f⟨fst(p)(x), snd(p)(x)⟩). x ∈ X}
by auto
then have P(x) = f⟨fst(p)(x), snd(p)(x)⟩
  by (rule func_ZF_1_L2)
hence P(x) = f⟨s(x), r(x)⟩ by auto
moreover have P = F⟨s, r⟩
proof -
  from A1 A2 have F : (X → range(f)) × (X → range(f)) → (X → range(f))
    using func_ZF_1_L3 by simp
  moreover from A3 have p ∈ (X → range(f)) × (X → range(f))
    by auto
  moreover from A2 have
    F = {{p, ⟨x, f⟨fst(p)(x), snd(p)(x)⟩⟩}. x ∈ X}. p ∈ (X → range(f)) × (X → range(f))
    using Lift2FcnSpce_def by simp
  ultimately show thesis using ZF_fun_from_tot_val by simp
qed
ultimately show (F⟨s, r⟩)(x) = f⟨s(x), r(x)⟩ by auto
qed

### 11.2 Associative and commutative operations

In this section we define associative and commutative operations and prove that they remain such when we lift them to a function space.

Typically we say that a binary operation "·" on a set $G$ is "associative" if $(x \cdot y) \cdot z = x \cdot (y \cdot z)$ for all $x, y, z \in G$. Our actual definition below does
not use the multiplicative notation so that we can apply it equally to
the additive notation + or whatever infix symbol we may want to use. Instead,
we use the generic set theory notation and write \( P(x, y) \) to denote the value
of the operation \( P \) on a pair \( (x, y) \in G \times G \).

**definition**

\[ \text{IsAssociative (infix \{is associative on\} 65) where} \]

\[ P \text{ is associative on } G \equiv P : G \times G \to G \land \]

\[ (\forall x \in G. \forall y \in G. \forall z \in G. \]

\[ ( P(\langle P(x, y), z \rangle) = P(\langle x, P(y, z) \rangle)) ) \]

A binary function \( f : X \times X \to Y \) is commutative if \( f(x, y) = f(y, x) \). Note
that in the definition of associativity above we talk about binary "operation"
and here we say use the term binary "function". This is not set in stone,
but usually the word "operation" is used when the range is a factor of
the domain, while the word "function" allows the range to be a completely
unrelated set.

**definition**

\[ \text{IsCommutative (infix \{is commutative on\} 65) where} \]

\[ f \text{ is commutative on } G \equiv \forall x \in G. \forall y \in G. f(x, y) = f(y, x) \]

The lift of a commutative function is commutative.

**lemma func_ZF_2_L1:**

assumes \( A1: f : G \times G \to G \)

and \( A2: F = f \) \{lifted to function space over\} \( X \)

and \( A3: s : X \to \text{range}(f) \) \( r : X \to \text{range}(f) \)

and \( A4: f \) \{is commutative on\} \( G \)

shows \( F(s, r) = F(r, s) \)

**proof**

from \( A1 \) \( A2 \) have

\( F : (X \to \text{range}(f)) \times (X \to \text{range}(f)) \to (X \to \text{range}(f)) \)

using func_ZF_1_L3 by simp

with \( A3 \) have

\( F(s, r) : X \to \text{range}(f) \) and \( F(r, s) : X \to \text{range}(f) \)

using apply_type by auto

moreover have

\( \forall x \in X. (F(s, r))(x) = (F(r, s))(x) \)

**proof**

fix \( x \) assume \( x \in X \)

from \( A1 \) have range(f) \( \subseteq G \)

using func1_1_L5B by simp

with \( A3 \) \( x \in X \) have \( s(x) \in G \) and \( r(x) \in G \)

using apply_type by auto

with \( A1 \) \( A2 \) \( A3 \) \( A4 \) \( x \in X \) show

\( (F(s, r))(x) = (F(r, s))(x) \)

using func_ZF_1_L4 IsCommutative_def by simp

qed

ultimately show thesis using fun_extension_iff

by simp
The lift of a commutative function is commutative on the function space.

**Lemma** func_ZF_2_L2:

assumes \( f : G \times G \rightarrow G \)
and \( f \) \{is commutative on\} \( G \)
and \( F = f \) \{lifted to function space over\} \( X \)
shows \( F \) \{is commutative on\} \( (X \rightarrow \text{range}(f)) \)
using assms IsCommutative_def func_ZF_2_L1 by simp

The lift of an associative function is associative.

**Lemma** func_ZF_2_L3:

assumes \( A2: F = f \) \{lifted to function space over\} \( X \)
and \( A3: s : X \rightarrow \text{range}(f) \) \( r : X \rightarrow \text{range}(f) \) \( q : X \rightarrow \text{range}(f) \)
and \( A4: f \) \{is associative on\} \( G \)
sows \( F(s,F(r,q)) = F(s,F(r),q) \)
proof -
from \( A4 \) \( A2 \) have \( F : (X \rightarrow \text{range}(f)) \times (X \rightarrow \text{range}(f)) \rightarrow (X \rightarrow \text{range}(f)) \)
using IsAssociative_def func_ZF_1_L3 by auto
with \( A3 \) have I:
\( F(s,r) : X \rightarrow \text{range}(f) \)
\( F(r,q) : X \rightarrow \text{range}(f) \)
\( F(F(s,r),q) : X \rightarrow \text{range}(f) \)
\( F(s,F(r,q)) : X \rightarrow \text{range}(f) \)
using apply_type by auto
moreover have \( \forall x \in X. (F(F(s,r),q))(x) = (F(s,F(r),q))(x) \)
proof
fix \( x \) assume \( x \in X \)
from \( A4 \) \( A2 \) have \( f : G \times G \rightarrow G \)
using IsAssociative_def by simp
then have \( \text{range}(f) \subseteq G \)
using func1_1_L5B by simp
with \( A3 \) \( \forall x \in X \) have \( s(x) \in G \) \( r(x) \in G \) \( q(x) \in G \)
using apply_type by auto
with \( A2 \) \( A3 \) \( A4 \) \( \forall x \in X \) \( \forall f : G \times G \rightarrow G \) show \( (F(F(s,r),q))(x) = (F(s,F(r),q))(x) \)
using func_ZF_1_L4 IsAssociative_def by simp
qed
ultimately show thesis using fun_extension_iff
by simp
qed

The lift of an associative function is associative on the function space.

**Lemma** func_ZF_2_L4:

assumes \( A1: f \) \{is associative on\} \( G \)
and \( A2: F = f \) \{lifted to function space over\} \( X \)
shows $F$ {is associative on} $(X \to \text{range}(f))$

proof -
from $A1$ $A2$ have
$F : (X \to \text{range}(f)) \times (X \to \text{range}(f)) \to (X \to \text{range}(f))$
using IsAssociative_def func_ZF_1_L3 by auto
moreover from $A1$ $A2$
have $\forall s \in X \to \text{range}(f). \ \forall r \in X \to \text{range}(f). \ \forall q \in X \to \text{range}(f).$
$F[F(s,r),q] = F[s,F(r,q)]$
using func_ZF_2_L3 by simp
ultimately show thesis using IsAssociative_def
by simp
qed

11.3 Restricting operations

In this section we consider conditions under which restriction of the operation to a set inherits properties like commutativity and associativity.

The commutativity is inherited when restricting a function to a set.

lemma func_ZF_4_L1:
assumes $A1: f:X \times X \to Y$ and $A2: A \subseteq X$
and $A3: f$ {is commutative on} $X$
shows $\text{restrict}(f,A \times A)$ {is commutative on} $A$
proof -
{ fix $x$ $y$ assume $x \in A$ and $y \in A$
with $A2$ have $x \in X$ and $y \in X$ by auto
with $A3$ $x \in A$ $y \in A$ have
$\text{restrict}(f,A \times A)(x,y) = \text{restrict}(f,A \times A)(y,x)$
using IsCommutative_def restrict_if by simp }
then show thesis using IsCommutative_def by simp
qed

Next we define what it means that a set is closed with respect to an operation.

definition
IsOpClosed {is closed under} 65) where
$A$ {is closed under} $f \equiv \forall x \in A. \ \forall y \in A. \ f(x,y) \in A$

Associative operation restricted to a set that is closed with resp. to this operation is associative.

lemma func_ZF_4_L2: assumes $A1: f$ {is associative on} $X$
and $A2: A \subseteq X$ and $A3: A$ {is closed under} $f$
and $A4: x \in A$ $y \in A$ $z \in A$
and $A5: g = \text{restrict}(f,A \times A)$
shows $g(g(x,y),z) = g(x,g(y,z))$
proof -
from $A4$ $A2$ have $I: x \in X$ $y \in X$ $z \in X$
by auto
from $A_3$ $A_4$ $A_5$ have
$g(g(x,y),z) = f(f(x,y),z)$
$g(x,g(y,z)) = f(x,f(y,z))$
using IsOpClosed_def restrict_if by auto
moreover from $A_1$ I have
$f(f(x,y),z) = f(x,f(y,z))$
using IsAssociative_def by simp
ultimately show thesis by simp

An associative operation restricted to a set that is closed with respect to this
operation is associative on the set.

```
lemma func_ZF_4_L3: assumes A1: f {is associative on} X
and A2: A$\subseteq$X and A3: A {is closed under} f
shows restrict(f,A$\times$A) {is associative on} A
proof -
let g = restrict(f,A$\times$A)
from A1 have f:X$\times$X$\rightarrow$X
using IsAssociative_def by simp
moreover from A2 have A$\times$A$\subseteq$X$\times$X by auto
moreover from A3 have $\forall$ p $\in$ A$\times$A. g(p) $\in$ A
using IsOpClosed_def restrict_if by auto
ultimately have g : A$\times$A$\rightarrow$A
using func1_2_L4 by simp
moreover from A1 A2 A3 have $\forall$ x $\in$ A $\forall$ y $\in$ A $\forall$ z $\in$ A.
g(g(x,y),z) = g(x,g(y,z))
using func_ZF_4_L2 by simp
ultimately show thesis
using IsAssociative_def by simp
qed
```

The essential condition to show that if a set $A$ is closed with respect to an
operation, then it is closed under this operation restricted to any superset
of $A$.

```
lemma func_ZF_4_L4: assumes A: A {is closed under} f
and A$\subseteq$B and x$\in$A y$\in$A and g = restrict(f,B$\times$B)
shows g(x,y) $\in$ A
using assms IsOpClosed_def restrict by auto
```

If a set $A$ is closed under an operation, then it is closed under this operation
restricted to any superset of $A$.

```
lemma func_ZF_4_L5:
  assumes A1: A {is closed under} f
  and A2: A$\subseteq$B
  shows A {is closed under} restrict(f,B$\times$B)
proof -
  let g = restrict(f,B$\times$B)
```

123
from $A_1$ to $A_2$ have $\forall x \in A. \forall y \in A. g(x,y) \in A$

using \textit{func\_ZF\_4\_L4} by simp

then show thesis using IsOpClosed_def by simp

qed

The essential condition to show that intersection of sets that are closed with respect to an operation is closed with respect to the operation.

\textbf{Lemma func\_ZF\_4\_L6:}

\begin{itemize}
  \item assumes $A$ \{is closed under} $f$
  \item and $B$ \{is closed under} $f$
  \item and $x \in A \cap B$ $y \in A \cap B$
  \item shows $f(x,y) \in A \cap B$ using \textit{assms IsOpClosed_def} by auto
\end{itemize}

Intersection of sets that are closed with respect to an operation is closed under the operation.

\textbf{Lemma func\_ZF\_4\_L7:}

\begin{itemize}
  \item assumes $A$ \{is closed under} $f$
  \item and $B$ \{is closed under} $f$
  \item shows $A \cap B$ \{is closed under} $f$
  \item using \textit{assms IsOpClosed_def} by simp
\end{itemize}

\section{11.4 Compositions}

For any set $X$ we can consider a binary operation on the set of functions $f : X \to X$ defined by $C(f,g) = f \circ g$. Composition of functions (or relations) is defined in the standard Isabelle distribution as a higher order function and denoted with the letter $\circ$. In this section we consider the corresponding two-argument ZF-function (binary operation), that is a subset of $((X \to X) \times (X \to X)) \times (X \to X)$.

We define the notion of composition on the set $X$ as the binary operation on the function space $X \to X$ that takes two functions and creates the their composition.

\textbf{Definition}

Composition($X$) $\equiv$

\[ \{ (p, \text{fst}(p) \circ \text{snd}(p)). p \in (X \to X) \times (X \to X) \} \]

Composition operation is a function that maps $X \to X \times X \to X$ into $X \to X$.

\textbf{Lemma func\_ZF\_5\_L1:}

\textit{show} Composition($X$) : $(X \to X) \times (X \to X) \to (X \to X)$

using \textit{comp\_fun Composition\_def} \textit{ZF\_fun\_from\_total} by simp

The value of the composition operation is the composition of arguments.

\textbf{Lemma func\_ZF\_5\_L2:}

\textit{assumes} $f:X \to X$ and $g:X \to X$

\textit{shows} Composition($X$)$\langle f,g \rangle = f \circ g$

\textit{proof -}

\textit{from} \textit{assms} \textit{have}
Composition(X) : (X→X)×(X→X)→(X→X)
⟨f,g⟩ ∈ (X→X)×(X→X)

Composition(X) = \{⟨p,fst(p) O snd(p)⟩. p ∈ (X→X)×(X→X)\}

then show Composition(X)⟨f,g⟩ = f O g

What is the value of a composition on an argument?

The essential condition to show that composition is associative.

Composition is an associative operation on X→X (the space of functions that map X into itself).

Composition is an associative operation on X→X (the space of functions that map X into itself).
11.5 Identity function

In this section we show some additional facts about the identity function defined in the standard Isabelle’s Perm theory. Note there is also image_id_same lemma in func1 theory.

A function that maps every point to itself is the identity on its domain.

**lemma identity_fun:** assumes A1: f:X→Y and A2:∀x∈X. f(x)=x shows f = id(X)

**proof**

- from assms have f:X→Y and id(X):X→X and ∀ x∈X. f(x) = id(X)(x)
  using id_type id_conv by auto
then show thesis by (rule func_eq)
qed

Composing a function with identity does not change the function.

**lemma func_ZF_6_L1A:** assumes A1: f : X→X shows Composition(X)(id(X),f) = f Composition(X)(f,id(X)) = f

**proof**

- have Composition(X) : (X→X)×(X→X)→(X→X)
  using func_ZF_5_L1 by simp
with A1 have Composition(X)(id(X),f) : X→X
  Composition(X)(f,id(X)) : X→X
  using id_type apply_funtype by auto
moreover note A1
moreover from A1 have
  ∀ x∈X. (Composition(X)(id(X),f))(x) = f(x)
  ∀ x∈X. (Composition(X)(f,id(X)))(x) = f(x)
  using id_type func_ZF_5_L3 apply_funtype id_conv by auto
ultimately show Composition(X)(id(X),f) = f Composition(X)(f,id(X)) = f
  using fun_extension_iff by auto
qed

An intuitively clear, but surprisingly nontrivial fact: identity is the only function from a singleton to itself.

**lemma singleton_fun_id:** shows ({x} → {x}) = {id({x})}

**proof**

- show {id({x})} ⊆ ({x} → {x})
  using id_def by simp
  { let g = id({x})
  fix f assume f : {x} → {x}
  then have f : {x} → {x} and g : {x} → {x}

using id_def by auto
moreover from \( f : \{x\} \rightarrow \{x\} \) have \( \forall x \in \{x\}. f(x) = g(x) \)
using apply_funtype id_def by auto
ultimately have \( f = g \) by (rule func_eq)
} then show \( \{x\} \rightarrow \{x\} \subseteq \{\text{id}(\{x\})\} \) by auto
qed

Another trivial fact: identity is the only bijection of a singleton with itself.

**lemma single_bij_id:** shows \( \text{bij}(\{x\},\{x\}) = \{\text{id}(\{x\})\} \)

**proof**
- show \( \{\text{id}(\{x\})\} \subseteq \text{bij}(\{x\},\{x\}) \) using id_bij
  by simp
- \{ fix \( f \) assume \( f \in \text{bij}(\{x\},\{x\}) \)
  then have \( f : \{x\} \rightarrow \{x\} \) using bij_is_fun
  by simp
  then have \( f \in \{\text{id}(\{x\})\} \) using singleton_fun_id
  by simp
} then show \( \text{bij}(\{x\},\{x\}) \subseteq \{\text{id}(\{x\})\} \) by auto
qed

A kind of induction for the identity: if a function \( f \) is the identity on a set with a fixpoint of \( f \) removed, then it is the identity on the whole set.

**lemma id_fixpoint_rem:** assumes \( A1: f:X\rightarrow X \) and
- \( A2: p\in X \) and \( A3: f(p) = p \) and
- \( A4: \text{restrict}(f, X-\{p\}) = \text{id}(X-\{p\}) \)
shows \( f = \text{id}(X) \)

**proof**
- from \( A1 \) have \( f: X\rightarrow X \) and \( \text{id}(X) : X\rightarrow X \)
  using id_def by auto
moreover
- \{ fix \( x \) assume \( x\in X \)
  \{ assume \( x \in X-\{p\} \)
    then have \( f(x) = \text{restrict}(f, X-\{p\})(x) \)
    using restrict by simp
    with \( A4 \) \( \langle x \in X-\{p\} \rangle \) have \( f(x) = x \)
    using id_def by simp \}
  with \( A2 \) \( A3 \) \( \langle x\in X \rangle \) have \( f(x) = x \) by auto
} then have \( \forall x\in X. f(x) = \text{id}(X)(x) \)
  using id_def by simp
ultimately show \( f = \text{id}(X) \) by (rule func_eq)
qed

11.6 Lifting to subsets

Suppose we have a binary operation \( f : X \times X \rightarrow X \) written additively as \( f(x, y) = x + y \). Such operation naturally defines another binary operation on the subsets of \( X \) that satisfies \( A + B = \{x + y : x \in A, y \in B\} \). This new operation which we will call "\( f \) lifted to subsets" inherits many properties of
This notion is useful for considering interval arithmetics.

The next definition describes the notion of a binary operation lifted to subsets. It is written in a way that might be a bit unexpected, but really it is the same as the intuitive definition, but shorter. In the definition we take a pair $p \in Pow(X) \times Pow(X)$, say $p = \langle A, B \rangle$, where $A, B \subseteq X$. Then we assign this pair of sets the set $\{ f(x, y) : x \in A, y \in B \} = \{ f(x') : x' \in A \times B \}$ The set on the right hand side is the same as the image of $A \times B$ under $f$. In the definition we don’t use $A$ and $B$ symbols, but write $fst(p)$ and $snd(p)$, resp. Recall that in Isabelle/ZF $fst(p)$ and $snd(p)$ denote the first and second components of an ordered pair $p$. See the lemma lift_subsets_explained for a more intuitive notation.

**definition**

Lift2Subsets (infix {lifted to subsets of} 65) where

$f$ {lifted to subsets of} $X$ $\equiv$

$\{ \langle p, f(fst(p) \times snd(p)) \rangle : p \in Pow(X) \times Pow(X) \}$

The lift to subsets defines a binary operation on the subsets.

**lemma** lift_subsets_binop: assumes $A1: f : X \times X \rightarrow Y$

shows $(f$ {lifted to subsets of} $X) : Pow(X) \times Pow(X) \rightarrow Pow(Y)$

**proof** -

let $F = \{ \langle p, f(fst(p) \times snd(p)) \rangle : p \in Pow(X) \times Pow(X) \}$

from $A1$ have $\forall p \in Pow(X) \times Pow(X). f(fst(p) \times snd(p)) \in Pow(Y)$

using func1_1_L6 by simp

then have $F : Pow(X) \times Pow(X) \rightarrow Pow(Y)$

by (rule ZF_fun_from_total)

then show thesis unfolding Lift2Subsets_def by simp

qed

The definition of the lift to subsets rewritten in a more intuitive notation. We would like to write the last assertion as $F(A,B) = \{f(x,y). x \in A, y \in B\}$, but Isabelle/ZF does not allow such syntax.

**lemma** lift_subsets_explained: assumes $A1: f : X \times X \rightarrow Y$

and $A2: A \subseteq X \ B \subseteq X$ and $A3: F = f$ {lifted to subsets of} $X$

shows $F(A,B) \subseteq Y$ and

$F(A,B) = f(A \times B)$

$F(A,B) = \{f(p). p \in A \times B\}$

$F(A,B) = \{f(x,y). \langle x,y \rangle \in A \times B\}$

**proof** -

let $p = \langle A,B \rangle$

from asssms have $I: F : Pow(X) \times Pow(X) \rightarrow Pow(Y)$ and $p \in Pow(X) \times Pow(X)$

using lift_subsets_binop by auto

moreover from $A3$ have $F = \{\langle p, f(fst(p) \times snd(p)) \rangle : p \in Pow(X) \times Pow(X) \}$

unfolding Lift2Subsets_def by simp

128
ultimately show $F(A,B) = f(A \times B)$

using $ZF\_fun\_from\_tot\_val$ by auto

also from $A1\ A2$ have $A \times B \subseteq X \times X$ by auto

with $A1$ have $f(A \times B) = \{f(p) . p \in A \times B\}$

by (rule func_imagedef)

finally show $F(A,B) = \{f(p) . p \in A \times B\}$ by simp

also have $\forall x \in A.\ \forall y \in B.\ f(x,y) = f(x,y)$ by simp

then have $\{f(p) . p \in A \times B\} = \{f(x,y).\ (x,y) \in A \times B\}$

by (rule ZF1_1_L4A)

finally show $F(A,B) = \{f(x,y) . (x,y) \in A \times B\}$ by simp

from $A2$ I show $F(A,B) \subseteq Y$ using apply_funtype by blast

qed

A sufficient condition for a point to belong to a result of lifting to subsets.

lemma lift_subset_suff: assumes $A1$: $f : X \times X \rightarrow Y$ and

$A2$: $A \subseteq X$ $B \subseteq X$ and $A3$: $x \in A$ $y \in B$ and

$A4$: $F = f$ \{lifted to subsets of\} $X$

shows $f(x,y) \in F(A,B)$

proof -

from $A3$ have $f(x,y) \in \{f(p) . p \in A \times B\}$ by auto

moreover from $A1\ A2\ A4$ have $\{f(p) . p \in A \times B\} = F(A,B)$

using lift_subsets_explained by simp

ultimately show $f(x,y) \in F(A,B)$ by simp

qed

A kind of converse of lift_subset_apply, providing a necessary condition for a point to be in the result of lifting to subsets.

lemma lift_subset_nec: assumes $A1$: $f : X \times X \rightarrow Y$ and

$A2$: $A \subseteq X$ $B \subseteq X$ and

$A3$: $F = f$ \{lifted to subsets of\} $X$ and

$A4$: $z \in F(A,B)$

shows $\exists x\ y.\ x \in A$ $\land$ $y \in B$ $\land$ $z = f(x,y)$

proof -

from $A1\ A2\ A3$ have $F(A,B) = \{f(p) . p \in A \times B\}$

using lift_subsets_explained by simp

with $A4$ show thesis by auto

qed

Lifting to subsets inherits commutativity.

lemma lift_subset_comm: assumes $A1$: $f : X \times X \rightarrow Y$ and

$A2$: $f$ \{is commutative on\} $X$ and

$A3$: $F = f$ \{lifted to subsets of\} $X$

shows $F$ \{is commutative on\} $Pow(X)$

proof -

have $\forall A \in Pow(X).\ \forall B \in Pow(X).\ F(A,B) = F(B,A)$

proof -
\begin{verbatim}
{ fix A assume A ∈ Pow(X)
  fix B assume B ∈ Pow(X)
  have F⟨A,B⟩ = F⟨B,A⟩
  proof -
  have ∀z ∈ F⟨A,B⟩. z ∈ F⟨B,A⟩
  proof
  fix z assume I: z ∈ F⟨A,B⟩
  with A1 A3 A ∈ Pow(X) B ∈ Pow(X) have
  ∃x y. x∈A ∧ y∈B ∧ z = f(x,y)
  using lift_subset_nec by simp
  then obtain x y where x∈A and y∈B and z = f(x,y)
  by auto
  with A2 A ∈ Pow(X) B ∈ Pow(X) have z = f(y,x)
  using IsCommutative_def by auto
  with A1 A3 I A ∈ Pow(X) B ∈ Pow(X) x∈A y∈B show z ∈ F⟨B,A⟩ using lift_subset_suff by simp
  qed
  moreover have ∀z ∈ F⟨B,A⟩. z ∈ F⟨A,B⟩
  proof
  fix z assume I: z ∈ F⟨B,A⟩
  with A1 A3 A ∈ Pow(X) B ∈ Pow(X) have
  ∃x y. x∈B ∧ y∈A ∧ z = f(x,y)
  using lift_subset_nec by simp
  then obtain x y where x∈B and y∈A and z = f(x,y)
  by auto
  with A2 A ∈ Pow(X) B ∈ Pow(X) have z = f(y,x)
  using IsCommutative_def by auto
  with A1 A3 I A ∈ Pow(X) B ∈ Pow(X) x∈B y∈A show z ∈ F⟨A,B⟩ using lift_subset_suff by simp
  qed
  ultimately show F⟨A,B⟩ = F⟨B,A⟩ by auto
  qed
} thus thesis by auto
qed

Lifting to subsets inherits associativity. To show that \( F(⟨A,B⟩C) = F(⟨A,F(⟨B,C⟩)⟩) \)
we prove two inclusions and the proof of the second inclusion is very similar
to the proof of the first one.

lemma lift_subset_assoc: assumes
  A1: f {is associative on} X and A2: F = f {lifted to subsets of} X
  shows F {is associative on} Pow(X)
proof -
  from A1 have f : X×X → X unfolding IsAssociative_def by simp
  with A2 have F : Pow(X)×Pow(X) → Pow(X)
    using lift_subsets_binop by simp
  moreover have ∀A ∈ Pow(X). ∀B ∈ Pow(X). ∀C ∈ Pow(X).
  \end{verbatim}
\[
F(F(A, B), C) = F(A, F(B, C))
\]

proof -

\{ fix A B C \\
  assume A ∈ Pow(X) B ∈ Pow(X) C ∈ Pow(X) \\
  have F(A, B) ∈ Pow(X) \\
  proof \\
  fix z assume I: z ∈ F(A, B) \\
  from \langle t: X × X → X \rangle A2 \langle a ∈ Pow(X) \rangle \langle b ∈ Pow(X) \rangle \\
  have F(A, B) ∈ Pow(X) \\
  using lift_subsets_binop apply_funtype by blast \\
  with \langle f: X × X → X \rangle A2 \langle c ∈ Pow(X) \rangle I have \\
  \exists x y. x ∈ F(A, B) ∧ y ∈ C ∧ z = f(x, y) \\
  using lift_subset_nec by simp \\
  then obtain x y where \\
  II: x ∈ F(A, B) and y ∈ C and III: z = f(x, y) \\
  by auto \\
  from \langle t: X × X → X \rangle A2 \langle a ∈ Pow(X) \rangle \langle b ∈ Pow(X) \rangle II have \\
  \exists s t. s ∈ A ∧ t ∈ B ∧ x = f(s, t) \\
  using lift_subset_nec by auto \\
  then obtain s t where s ∈ A and t ∈ B and x = f(s, t) \\
  by auto \\
  with A1 \langle a ∈ Pow(X) \rangle \langle b ∈ Pow(X) \rangle \langle c ∈ Pow(X) \rangle III \\
  \langle s ∈ A \rangle \langle t ∈ B \rangle \langle y ∈ C \rangle have IV: z = f(s, f(t, y)) \\
  using IsAssociative_def by blast \\
  from \langle t: X × X → X \rangle A2 \langle a ∈ Pow(X) \rangle \langle b ∈ Pow(X) \rangle \langle t ∈ B \rangle \langle y ∈ C \rangle \\
  have f(t, y) ∈ F(B, C) using lift_subset_suff by simp \\
  moreover from \langle f: X × X → X \rangle A2 \langle b ∈ Pow(X) \rangle \langle c ∈ Pow(X) \rangle \\
  have F(B, C) ⊆ X using lift_subsets_binop apply_funtype \\
  by blast \\
  moreover note \langle f: X × X → X \rangle A2 \langle a ∈ Pow(X) \rangle \langle s ∈ A \rangle IV \\
  ultimately show z ∈ F(A, F(B, C)) \\
  using lift_subset_suff by simp \\
  qed \\
  moreover have F(A, F(B, C)) ⊆ F(A, B, C) \\
  proof \\
  fix z assume I: z ∈ F(A, B, C) \\
  from \langle f: X × X → X \rangle A2 \langle b ∈ Pow(X) \rangle \langle c ∈ Pow(X) \rangle \\
  have F(B, C) ∈ Pow(X) \\
  using lift_subsets_binop apply_funtype by blast \\
  with \langle f: X × X → X \rangle A2 \langle a ∈ Pow(X) \rangle I have \\
  \exists x y. x ∈ A ∧ y ∈ F(B, C) ∧ z = f(x, y) \\
  using lift_subset_nec by simp \\
  then obtain x y where \\
  x ∈ A and II: y ∈ F(B, C) and III: z = f(x, y) \\
  by auto \\
  from \langle f: X × X → X \rangle A2 \langle b ∈ Pow(X) \rangle \langle c ∈ Pow(X) \rangle II have \\
  \exists s t. s ∈ B ∧ t ∈ C ∧ y = f(s, t) \\
  using lift_subset_nec by auto \\
  then obtain s t where s ∈ B and t ∈ C and y = f(s, t) \\
 qed
by auto
with III have z = f⟨x,f⟨s,t⟩⟩ by simp
moreover from A1 ⟨A ∈ Pow(X)⟩ ⟨B ∈ Pow(X)⟩ ⟨C ∈ Pow(X)⟩
  ⟨x∈A⟩ ⟨a∈B⟩ ⟨t∈C⟩ have f⟨f⟨x,s⟩,t⟩ = f⟨x,f⟨s,t⟩⟩
using IsAssociative_def by blast
ultimately have IV: z = f⟨f⟨x,s⟩,t⟩ by simp
from ⟨f:X×X → X⟩ A2 ⟨A ∈ Pow(X)⟩ ⟨B ∈ Pow(X)⟩ ⟨x∈A⟩ ⟨s∈B⟩
have f⟨x,s⟩ ∈ F⟨A,B⟩ using lift_subset_suff by simp
moreover from ⟨f:X×X → X⟩ A2 ⟨A ∈ Pow(X)⟩ ⟨B ∈ Pow(X)⟩
have F⟨A,B⟩ ⊆ X using lift_subsets_binop apply_funtype
  by blast
moreover note ⟨f:X×X → X⟩ A2 ⟨C ∈ Pow(X)⟩ ⟨t∈C⟩ IV
ultimately show z ∈ F⟨F⟨A,B⟩,C⟩ using lift_subset_suff by simp
  qed
  ultimately have F⟨F⟨A,B⟩,C⟩ = F⟨A,F⟨B,C⟩⟩ by auto
  thus thesis by auto
  qed
ultimately show thesis unfolding IsAssociative_def
  by auto
qed

11.7 Distributive operations

In this section we deal with pairs of operations such that one is distributive
with respect to the other, that is a·(b+c) = a·b+a·c and (b+c)·a = b·a+c·a.
We show that this property is preserved under restriction to a set closed
with respect to both operations. In EquivClass1 theory we show that this
property is preserved by projections to the quotient space if both operations
are congruent with respect to the equivalence relation.

We define distributivity as a statement about three sets. The first set is the
set on which the operations act. The second set is the additive operation (a
ZF function) and the third is the multiplicative operation.

definition
  IsDistributive(X,A,M) ≡ (∀a∈X.∀b∈X.∀c∈X.
    M(a,A⟨b,c⟩) = A(M(a,b),M(a,c)) ∧
    M(A⟨b,c⟩,a) = A(M(b,a),M(c,a))))

The essential condition to show that distributivity is preserved by restric-
tions to sets that are closed with respect to both operations.

lemma func_ZF_7_L1:
  assumes A1: IsDistributive(X,A,M)
  and A2: Y⊆X
  and A3: Y {is closed under} A Y {is closed under} M
  and A4: A_r = restrict(A,Y×Y) M_r = restrict(M,Y×Y)
  and A5: a∈Y b∈Y c∈Y
  shows M_r ( a,A_r⟨b,c⟩ ) = A_r ( M_r⟨a,b⟩,M_r⟨a,c⟩ ) ∧
\[ M_r( A_r(b,c), a ) = A_r( M_r(b,a), M_r(c,a) ) \]

**proof**

- from A3 A5 have \( A_r(b,c) \in Y \) \( M_r(a,b) \in Y \) \( M_r(a,c) \in Y \) \( M_r(b,a) \in Y \) \( M_r(c,a) \in Y \) using IsOpClosed_def by auto
- with A5 A4 have \( A_r(b,c) \in Y \) \( M_r(a,b) \in Y \) \( M_r(a,c) \in Y \) \( M_r(b,a) \in Y \) \( M_r(c,a) \in Y \) using restrict by auto
- with A1 A2 A4 A5 show thesis using restrict IsDistributive_def by auto

qed

Distributivity is preserved by restrictions to sets that are closed with respect to both operations.

**lemma** **func_ZF_7_L2:**

assumes IsDistributive(X,A,M) and \( Y \subseteq X \) and \( Y \) {is closed under} A \( Y \) {is closed under} M and \( A_r = \text{restrict}(A,Y \times Y) \) \( M_r = \text{restrict}(M,Y \times Y) \) shows IsDistributive\( (Y,A_r,M_r) \)

**proof**

- from assms have \( \forall a \in Y. \forall b \in Y. \forall c \in Y. \)
  \[ M_r( A_r(b,c), a ) = A_r( M_r(a,b), M_r(a,c) ) \land M_r( A_r(b,c), a ) = A_r( M_r(b,a), M_r(c,a) ) \]
  using func_ZF_7_L1 by simp
- then show thesis using IsDistributive_def by simp

qed

**12 More on functions**

theory **func_ZF_1** imports ZF.Order Order_ZF_1a func_ZF

begin

In this theory we consider some properties of functions related to order relations

**12.1 Functions and order**

This section deals with functions between ordered sets.

If every value of a function on a set is bounded below by a constant, then the image of the set is bounded below.

**lemma** **func_ZF_8_L1:**
assumes \( f: X \rightarrow Y \) and \( A \subseteq X \) and \( \forall x \in A. \langle L, f(x) \rangle \in r \) shows IsBoundedBelow(f(A), r)

proof -
  from assms have \( \forall y \in f(A). \langle L, y \rangle \in r \)
    using func_imagedef by simp
  then show IsBoundedBelow(f(A), r)
    by (rule Order_ZF_3_L9)
qed

If every value of a function on a set is bounded above by a constant, then the image of the set is bounded above.

lemma \( \text{func} \_ZF \_8 \_L2 \):
  assumes \( f: X \rightarrow Y \) and \( A \subseteq X \) and \( \forall x \in A. \langle f(x), U \rangle \in r \) shows IsBoundedAbove(f(A), r)

proof -
  from assms have \( \forall y \in f(A). \langle y, U \rangle \in r \)
    using func_imagedef by simp
  then show IsBoundedAbove(f(A), r)
    by (rule Order_ZF_3_L10)
qed

Identity is an order isomorphism.

lemma \( \text{id} \_ord \_iso \):
  shows id(X) \in ord_iso(X, r, X, r)
  using id_bij id_def ord_iso_def by simp

Identity is the only order automorphism of a singleton.

lemma \( \text{id} \_ord \_auto \_singleton \):
  shows ord_iso({x}, r, {x}, r) = {id({x})}
  using id_ord_iso ord_iso_def single_bij_id by auto

The image of a maximum by an order isomorphism is a maximum. Note that from the fact the \( r \) is antisymmetric and \( f \) is an order isomorphism between \((A, r)\) and \((B, R)\) we can not conclude that \( R \) is antisymmetric (we can only show that \( R \cap (B \times B) \) is).

lemma \( \text{max} \_image \_ord \_iso \):
  assumes A1: antisym(r) and A2: antisym(R) and
  A3: \( f \in \text{ord} \_iso(A, r, B, R) \) and
  A4: HasMaximum(r, A)
  shows HasMaximum(R, B) and Maximum(R, B) = f(Maximum(r, A))

proof -
  let M = Maximum(r, A)
  from A1 A4 have M \in A using Order_ZF_4_L3 by simp
  from A3 have f:A\rightarrow B using ord_iso_def bij_is_fun
    by simp
  with \( \langle M \in A \rangle \) have I: f(M) \in B
    using apply_funtype by simp
  \{ fix y assume y \in B

134
let $x = \text{converse}(f)(y)$
from A3 have $\text{converse}(f) \in \text{ord}_\text{iso}(B,R,A,r)$
  using ord_iso_sym by simp
then have $\text{converse}(f) : B \to A$
  using ord_iso_def bij_is_fun by simp
with $\langle y \in B \rangle$ have $x \in A$
  by simp
with A1 A3 A4 $\langle x \in A \rangle \langle M \in A \rangle$ have $\langle f(x), f(M) \rangle \in R$
  using Order_ZF_4_L3 ord_iso_apply by simp
with A3 $\langle y \in B \rangle$ have $\langle y, f(M) \rangle \in R$
  using right_inverse_bij ord_iso_def by auto
} then have II: $\forall y \in B. \langle y, f(M) \rangle \in R$ by simp
with A2 I show $\text{Maximum}(R,B) = f(M)$
  by (rule Order_ZF_4_L14)
from I II show $\text{HasAmaximum}(R,B)$
  using HasAmaximum_def by auto
qed

Maximum is a fixpoint of order automorphism.

lemma max_auto_fixpoint:
  assumes antisym(r) and $f \in \text{ord}_\text{iso}(A,r,A,r)$
  and $\text{HasAmaximum}(r,A)$
  shows $\text{Maximum}(r,A) = f(\text{Maximum}(r,A))$
  using assms max_image_ord_iso by blast

If two sets are order isomorphic and we remove $x$ and $f(x)$, respectively,
from the sets, then they are still order isomorphic.

lemma ord_iso_rem_point:
  assumes A1: $f \in \text{ord}_\text{iso}(A,r,B,R)$ and A2: $a \in A$
  shows $\text{restrict}(f,A-\{a\}) \in \text{ord}_\text{iso}(A-\{a\},r,B-\{f(a)\},R)$
proof -
let $f_0 = \text{restrict}(f,A-\{a\})$
have $A-\{a\} \subseteq A$ by auto
with A1 have $f_0 \in \text{ord}_\text{iso}(A-\{a\},r,f(A-\{a\}),R)$
  using ord_iso_restrict_image by simp
moreover
from A1 have $f \in \text{inj}(A,B)$
  using ord_iso_def bij_def by simp
with A2 have $f(A-\{a\}) = f(A) - f(a)$
  using inj_image_dif by simp
moreover from A1 have $f(A) = B$
  using ord_iso_def bij_def surj_range_image_domain by auto
moreover
from A1 have $f : A \to B$
  using ord_iso_def bij_is_fun by simp
with A2 have $f(a) = \{f(a)\}$
  using singleton_image by simp
ultimately show thesis by simp
If two sets are order isomorphic and we remove maxima from the sets, then they are still order isomorphic.

**corollary ord_iso_rem_max:**
assumes A1: antisym(r) and f ∈ ord_iso(A,r,B,R) and
A4: HasMaximum(r,A) and A5: M = Maximum(r,A)
shows restrict(f,A-{M}) ∈ ord_iso(A-{M}, r, B-{f(M)},R)
using assms Order_ZF_4_L3 ord_iso_rem_point by simp

Lemma about extending order isomorphisms by adding one point to the domain.

**lemma ord_iso_extend:**  assumes A1: f ∈ ord_iso(A,r,B,R) and
A2: M ∈ A M ∈ B and
A3: ∀ a∈A. ⟨a, M_A⟩ ∈ r ∀ b∈B. ⟨b, M_B⟩ ∈ R and
A4: antisym(r) antisym(R) and
A5: ⟨M_A,M_A⟩ ∈ r ←→ ⟨M_B,M_B⟩ ∈ R
shows f ∪ {⟨M_A,M_B⟩} ∈ ord_iso(A∪{M_A} ,r,B∪{M_B} ,R)
proof -
let g = f ∪ {⟨M_A,M_B⟩} from A1 A2 have
g : A∪{M_A} → B∪{M_B} and
I: ∀ x∈A. g(x) = f(x) and II: g(M_A) = M_B
using ord_iso_def bij_def inj_def func1_1_L11D by auto
from A1 A2 have g ∈ bij(A∪{M_A},B∪{M_B}) using ord_iso_def bij_extend_point by simp
moreover have ∀ x ∈ A∪{M_A}. ∀ y ∈ A∪{M_A}.
⟨x,y⟩ ∈ r ←→ ⟨g(x), g(y)⟩ ∈ R proof -
{ fix x y
 assume x ∈ A∪{M_A} and y ∈ A∪{M_A}
 then have x∈A ∧ y ∈ A ∨ x∈A ∧ y = M_A ∨
x = M_A ∧ y ∈ A ∨ x = M_A ∧ y = M_A
 by auto
 moreover
 { assume x∈A ∧ y ∈ A
 with A1 I have ⟨x,y⟩ ∈ r ←→ ⟨g(x), g(y)⟩ ∈ R
 using ord_iso_def by simp }
 moreover
 { assume x∈A ∧ y = M_A
 with A1 A3 I II have ⟨x,y⟩ ∈ r ←→ ⟨g(x), g(y)⟩ ∈ R
 using ord_iso_def bij_def inj_def apply_funtype
 by auto }
 moreover
 { assume x = M_A ∧ y ∈ A
 with A2 A3 A4 have ⟨x,y⟩ ∉ r
 using antisym_def by auto
 moreover

A kind of converse to \( \text{ordiso}_\text{rem}_\text{max} \): if two linearly ordered sets are order isomorphic after removing the maxima, then they are order isomorphic.

**lemma rem_max_ord_iso:**

- assumes \( A1: \text{IsLinOrder}(X,r) \) \( \text{IsLinOrder}(Y,R) \) and
- \( A2: \text{HasAmaximum}(r,X) \) \( \text{HasAmaximum}(R,Y) \)
- \( \text{ordiso}(X - \{\text{Maximum}(r,X)\},r,Y - \{\text{Maximum}(R,Y)\},R) \neq 0 \)

shows \( \text{ordiso}(X,r,Y,R) \neq 0 \)

**proof -**

- let \( M_A = \text{Maximum}(r,X) \)
- let \( A = X - \{M_A\} \)
- let \( M_B = \text{Maximum}(R,Y) \)
- let \( B = Y - \{M_B\} \)

from \( A2 \) obtain \( f \) where \( f \in \text{ordiso}(A,r,B,R) \)
- by auto

moreover have \( M_A \notin A \) and \( M_B \notin B \)
- by auto

moreover from \( A1 \) \( A2 \) have
- \( \forall a \in A. \; \langle a,M_A \rangle \in r \) and \( \forall b \in B. \; \langle b,M_B \rangle \in R \)
- using \( \text{IsLinOrder}\_\text{def} \) \( \text{Order}\_ZF\_4\_L3 \) by auto

moreover from \( A1 \) have \( \text{antisym}(r) \) and \( \text{antisym}(R) \)
- using \( \text{IsLinOrder}\_\text{def} \) by auto

moreover from \( A1 \) \( A2 \) have \( \langle M_A,M_A \rangle \in r \leftarrow \langle M_B,M_B \rangle \in R \)
- using \( \text{IsLinOrder}\_\text{def} \) \( \text{Order}\_ZF\_4\_L3 \) \( \text{IsLinOrder}\_\text{def} \)
- \( \text{total_is_refl} \) \( \text{refl}\_\text{def} \) by auto
ultimately have
\[ f \cup \{ \langle M_A, M_B \rangle \} \in \text{ord}_\text{iso}(A \cup \{ M_A \}, r, B \cup \{ M_B \}, R) \]
by (rule \text{ord}_\text{iso}_\text{extend})
moreover from A1 A2 have
\[ A \cup \{ M_A \} = X \text{ and } B \cup \{ M_B \} = Y \]
using \text{IsLinOrder}_\text{def} \text{ Order}_\text{ZF.4}_\text{L3} \text{ by auto}
ultimately show \text{ord}_\text{iso}(X, r, Y, R) \neq 0
using \text{ord}_\text{iso}_\text{extend} \text{ by auto}
qed

12.2 Functions in cartesian products

In this section we consider maps arising naturally in cartesian products.

There is a natural bijection between \( X = Y \times \{ y \} \) (a "slice") and \( Y \). We will call this the \text{SliceProjection}(Y \times \{ y \}). This is really the ZF equivalent of the meta-function \text{fst}(x).

**definition**

\text{SliceProjection}(X) \equiv \{ \langle p, \text{fst}(p) \rangle. p \in X \}

A slice projection is a bijection between \( X \times \{ y \} \) and \( X \).

**lemma slice_proj_bij: shows**

\text{SliceProjection}(X \times \{ y \}): X \times \{ y \} \rightarrow X
\text{domain}(\text{SliceProjection}(X \times \{ y \})) = X \times \{ y \}
\forall p \in X \times \{ y \}. \text{SliceProjection}(X \times \{ y \})(p) = \text{fst}(p)
\text{SliceProjection}(X \times \{ y \}) \in \text{bij}(X \times \{ y \}, X)

**proof -**

let \( P = \text{SliceProjection}(X \times \{ y \}) \)
have \( \forall p \in X \times \{ y \}. \text{fst}(p) \in X \) by simp
moreover from this have
\( \{ \langle p, \text{fst}(p) \rangle. p \in X \times \{ y \} \} : X \times \{ y \} \rightarrow X \)
by (rule ZF\_fun\_from\_total)
ultimately show I: \( P: X \times \{ y \} \rightarrow X \) and II: \( \forall p \in X \times \{ y \}. P(p) = \text{fst}(p) \)
using ZF\_fun\_from\_tot\_val \text{SliceProjection}_\text{def} \text{ by auto}
hence
\( \forall a \in X \times \{ y \}. \forall b \in X \times \{ y \}. P(a) = P(b) \rightarrow a=b \)
by auto
with I have P \in \text{inj}(X \times \{ y \}, X) using inj\_def
by simp
moreover from II have \( \forall x \in X. \exists p \in X \times \{ y \}. P(p) = x \)
by simp
with I have P \in \text{surj}(X \times \{ y \}, X) using surj\_def
by simp
ultimately show P \in \text{bij}(X \times \{ y \}, X)
using bij\_def by simp
from I show \text{domain}(\text{SliceProjection}(X \times \{ y \})) = X \times \{ y \}
using func1_1\_L1 by simp
qed
Given 2 functions $f : A \rightarrow B$ and $g : C \rightarrow D$, we can consider a function $h : A \times C \rightarrow B \times D$ such that $h(x, y) = (f(x), g(y))$

**definition**

$ProdFunction$ where

$ProdFunction(f, g) \equiv \{(z, (f(fst(z)), g(snd(z)))) : z \in \text{domain}(f) \times \text{domain}(g)\}$

For given functions $f : A \rightarrow B$ and $g : C \rightarrow D$ the function $ProdFunction(f, g)$ maps $A \times C$ to $B \times D$.

**lemma** $prodFunction$:  
assumes $f : A \rightarrow B$ $g : C \rightarrow D$  
shows $ProdFunction(f, g) : (A \times C) \rightarrow (B \times D)$

**proof** -  
from assms have $\forall z \in \text{domain}(f) \times \text{domain}(g)$. $(f(fst(z)), g(snd(z))) \in B \times D$  
using $\text{func1}_1\_\text{L1}$ apply_type by auto  
with assms show thesis unfolding $ProdFunction$ using $\text{func1}_1\_\text{L1}$  
$ZF\_fun\_from\_tot$  
by simp  
qed

For given functions $f : A \rightarrow B$ and $g : C \rightarrow D$ and points $x \in A$, $y \in C$ the value of the function $ProdFunction(f, g)$ on $\langle x, y \rangle$ is $\langle f(x), g(y) \rangle$.

**lemma** $prodFunctionApp$:  
assumes $f : A \rightarrow B$ $g : C \rightarrow D$ $x \in A$ $y \in C$  
shows $ProdFunction(f, g)(x, y) = (f(x), g(y))$

**proof** -  
let $z = \langle x, y \rangle$  
from assms have $z \in A \times C$ and $ProdFunction(f, g) : (A \times C) \rightarrow (B \times D)$  
using $\text{prodFunction}$ by auto  
moreover from assms(1, 2) have $ProdFunction(f, g) = \{(z, (f(fst(z)), g(snd(z)))) : z \in A \times C\}$  
unfolding $ProdFunction$ using $\text{func1}_1\_\text{L1}$ by blast  
ultimately show thesis using $ZF\_fun\_from\_tot\_val$ by auto  
qed

Somewhat technical lemma about inverse image of a set by a $ProdFunction(f, f)$.

**lemma** $prodFunVimage$:  
assumes $\forall x \in X$. $f:X \rightarrow Y$  
shows $\langle x, t \rangle \in \text{ProdFunction}(f, f) \^{-1}(V) \leftrightarrow t \in X \land \langle fx, ft \rangle \in V$

**proof** -  
from assms(2) have $T:\text{ProdFunction}(f, f) \^{-1}(V) \equiv \{z \in X \times X. \text{ProdFunction}(f, f)(z) \in V\}$  
using $\text{prodFunction}$ $\text{func1}_1\_\text{L15}$ by blast  
with assms show thesis using $\text{prodFunctionApp}$ by auto  
qed
12.3 Induced relations and order isomorphisms

When we have two sets $X,Y$, function $f : X \rightarrow Y$ and a relation $R$ on $Y$ we can define a relation $r$ on $X$ by saying that $x r y$ if and only if $f(x) R f(y)$. This is especially interesting when $f$ is a bijection as all reasonable properties of $R$ are inherited by $r$. This section treats mostly the case when $R$ is an order relation and $f$ is a bijection. The standard Isabelle’s Order theory defines the notion of a space of order isomorphisms between two sets relative to a relation. We expand that material proving that order isomorphisms preserve interesting properties of the relation.

We call the relation created by a relation on $Y$ and a mapping $f : X \rightarrow Y$ the InducedRelation($f,R$).

**definition**

\[
\text{InducedRelation}(f,R) \equiv \\
\{ p \in \text{domain}(f) \times \text{domain}(f). \langle f(\text{fst}(p)), f(\text{snd}(p)) \rangle \in R \}
\]

A reformulation of the definition of the relation induced by a function.

**lemma** def_of_ind_relA:

assumes $(x,y) \in \text{InducedRelation}(f,R)$

shows $(f(x),f(y)) \in R$

using assms InducedRelation_def by simp

A reformulation of the definition of the relation induced by a function, kind of converse of def_of_ind_relA.

**lemma** def_of_ind_relB: assumes $f : A \rightarrow B$ and $x \in A \quad y \in A$ and $(f(x),f(y)) \in R$

shows $(x,y) \in \text{InducedRelation}(f,R)$

using assms func1_1_L1 InducedRelation_def by simp

A property of order isomorphisms that is missing from standard Isabelle’s Order.thy.

**lemma** ord_iso_apply_conv:

assumes $f \in \text{ord_iso}(A,r,B,R)$ and $(f(x),f(y)) \in R \quad \text{and} \quad x \in A \quad y \in A$

shows $(x,y) \in r$

using assms ord_iso_def by simp

The next lemma tells us where the induced relation is defined

**lemma** ind_rel_domain:

assumes $R \subseteq B \times B \quad \text{and} \quad f : A \rightarrow B$

shows InducedRelation($f,R$) $\subseteq A \times A$

using assms func1_1_L1 InducedRelation_def by auto

A bijection is an order homomorphisms between a relation and the induced one.
lemma bij_is_ord_iso: assumes A1: f ∈ bij(A,B) shows f ∈ ord_iso(A,InducedRelation(f,R),B,R)
proof -
let r = InducedRelation(f,R)
{ fix x y assume A2: x∈A y∈A
have ⟨x,y⟩ ∈ r ↔ ⟨f(x),f(y)⟩ ∈ R
proof
assume ⟨x,y⟩ ∈ r then show ⟨f(x),f(y)⟩ ∈ R
using def_of_ind_relA by simp
next assume ⟨f(x),f(y)⟩ ∈ R
with A1 A2 show ⟨x,y⟩ ∈ r
using bij_is_fun def_of_ind_relB by blast
qed }
with A1 show f ∈ ord_iso(A,InducedRelation(f,R),B,R)
using ord_isoI by simp
qed

An order isomorphism preserves antisymmetry.

lemma ord_iso_pres_antsym: assumes A1: f ∈ ord_iso(A,r,B,R) and A2: r ⊆ A×A and A3: antisym(R) shows antisym(r)
proof -
{ fix x y z
assume A4: ⟨x,y⟩ ∈ r ⟨y,z⟩ ∈ r
from A1 have f ∈ inj(A,B)
using ord_iso_is_bij bij_is_inj by simp
moreover from A1 A2 A4 have ⟨f(x), f(y)⟩ ∈ R ∧ ⟨f(y), f(z)⟩ ∈ R
using ord_iso_apply by auto
with A3 have f(x) = f(y) by (rule Fol1_L4)
moreover from A2 A4 have x∈A y∈A by auto
ultimately have x=y by (rule inj_apply_equality)
} then have ∀x y. ⟨x,y⟩ ∈ r ∧ ⟨y,x⟩ ∈ r → x=y by auto
then show antisym(r) using imp_conj antisym_def by simp
qed

Order isomorphisms preserve transitivity.

lemma ord_iso_pres_trans: assumes A1: f ∈ ord_iso(A,r,B,R) and A2: r ⊆ A×A and A3: trans(R) shows trans(r)
proof -
{ fix x y z
assume A4: ⟨x, y⟩ ∈ r ⟨y, z⟩ ∈ r
note A1
moreover from A1 A2 A4 have ⟨f(x), f(y)⟩ ∈ R ∧ ⟨f(y), f(z)⟩ ∈ R

lemmas ord_iso_pres_tot: assumes A1: f ∈ ord_iso(A,r,B,R) and
A2: r ⊆ A×A and A3: R {is total on} B
shows r {is total on} A
proof -
{ fix x y
  assume x∈A y∈A ⟨x,y⟩ ∉ r
  with A1 have ⟨f(x),f(y)⟩ ∉ R using ord_iso_apply_conv
  by simp
  moreover from A1 have f:A → B using ord_iso_is_bij bij_is_fun
  by simp
  with A3 ⟨x∈A⟩ ⟨y∈A⟩ have
  ⟨f(x),f(y)⟩ ∈ R ∨ ⟨f(y),f(x)⟩ ∈ R
  using apply_funtype IsTotal_def by simp
  ultimately have ⟨f(y),f(x)⟩ ∈ R by simp
  with A1 ⟨x∈A⟩ ⟨y∈A⟩ have ⟨y,x⟩ ∈ r
  using ord_iso_apply_conv by simp
} then have ∀x∈A. ∀y∈A. ⟨x,y⟩ ∈ r ∨ ⟨y,x⟩ ∈ r
by blast
then show r {is total on} A using IsTotal_def
by simp
qed

lemmas ord_iso_pres_lin: assumes f ∈ ord_iso(A,r,B,R) and
r ⊆ A×A and IsLinOrder(B,R)
shows IsLinOrder(A,r)
using assms ord_iso_pres_antsym ord_iso_pres_trans ord_iso_pres_tot
IsLinOrder_def by simp

If a relation is a linear order, then the relation induced on another set by a
bijection is also a linear order.

lemmas ind_rel_pres_lin:
  assumes A1: f ∈ bij(A,B) and A2: IsLinOrder(B,R)
  shows IsLinOrder(A,InducedRelation(f,R))
proof -
  let r = InducedRelation(f,R)
  from A1 have f ∈ ord_iso(A,r,B,R) and r ⊆ A×A
The image by an order isomorphism of a bounded above and nonempty set is bounded above.

**Lemma ord_iso_pres_bound_above:**

- Assumes $A1: f \in \text{ord_iso}(A,r,B,R)$ and $A2: r \subseteq A \times A$ and $A3: \text{C} \subseteq A$ and $\text{C} \neq \emptyset$.
- Shows $\text{IsBoundedAbove}(f(C),R)$ if $f(C) \neq \emptyset$.

**Proof:**

1. From $A3$, obtain $u$ where $I: \forall x \in C. \langle x, u \rangle \in r$.
2. Using $\text{IsBoundedAbove}(C,r)$ by auto.
3. From $A1$ have $f: A \rightarrow B$ using $\text{ord_iso_is_bij}(bij_is_fun)$ by simp.
4. From $A2$ and $A3$ have $C \subseteq A$ and $\text{C} \subseteq A$ using $\text{Order_ZF_3_L1A}$ by blast.
5. From $A3$, obtain $x$ where $x \in C$ by auto.
6. With $A2$ and $I$, have $u \in A$ by auto.
7. Fix $y$ assume $y \in f(C)$.
8. With $\langle f: A \rightarrow B \rangle$, $\langle C \subseteq A \rangle$, obtain $x$ where $x \in C$ and $y = f(x)$.
9. Using $\text{func_imagedef}$ by auto.
10. With $A1$ and $\langle C \subseteq A \rangle$, $\langle u \in A \rangle$, have $\langle y, f(u) \rangle \in R$.
11. Using $\text{ord_iso_apply}$ by auto.

Then have $\forall y \in f(C). \langle y, f(u) \rangle \in R$ by simp.

Then show $\text{IsBoundedAbove}(f(C),R)$ by (rule $\text{Order_ZF_3_L10}$).

From $A3$, $\langle f: A \rightarrow B \rangle$, $\langle C \subseteq A \rangle$, show $f(C) \neq \emptyset$ using $\text{func1_1_L15A}$ by simp.

**QED**

Order isomorphisms preserve the property of having a minimum.

**Lemma ord_iso_pres_has_min:**

- Assumes $A1: f \in \text{ord_iso}(A,r,B,R)$ and $A2: r \subseteq A \times A$ and $A3: C \subseteq A$ and $A4: \text{HasAminimum}(R,f(C))$.
- Shows $\text{HasAminimum}(r,C)$.

**Proof:**

1. From $A4$, obtain $m$ where $I: m \in f(C)$ and $II: \forall y \in f(C). \langle m, y \rangle \in R$.
2. Using $\text{HasAminimum}(f(C))$ by auto.
3. Let $k = \text{converse}(f)(m)$.
4. From $A1$ have $f: A \rightarrow B$ using $\text{ord_iso_is_bij}(bij_is_fun)$ by simp.
5. From $A1$ have $f \in \text{inj}(A,B)$ using $\text{ord_iso_is_bij}(bij_is_inj)$ by simp.
6. With $A3$, $I$, $k \in C$ and $III: f(k) = m$.
7. Using $\text{inj_inv_back_in_set}$ by auto.

Moreover,

- Fix $x$ assume $A5: x \in C$.
with A3 II \( f:A \rightarrow B \) \( k \in C \) III have
\[
k \in A \quad x \in A \quad (f(k),f(x)) \in R
\]
using func_imagedef by auto
with A1 have \( \langle k,x \rangle \in r \) using ord_iso_apply_conv
by simp
} then have \( \forall x \in C. \quad (k,x) \in r \) by simp
ultimately show HasAminimum(r,C) using HasAminimum_def by auto
qed

Order isomorphisms preserve the images of relations. In other words taking the image of a point by a relation commutes with the function.

**Lemma ord_iso_pres_rel_image:**
assumes A1: \( f \in \text{ord}_iso(A,r,B,R) \) and
A2: \( r \subseteq A \times A \quad R \subseteq B \times B \) and
A3: \( a \in A \)
shows \( f(r\{a\}) = R\{f(a)\} \)
proof
from A1 have \( f:A \rightarrow B \) using ord_iso_is_bij bij_is_fun
by simp
moreover from A2 A3 have I: \( r\{a\} \subseteq A \) by auto
ultimately have I: \( f(r\{a\}) = \{f(x). \quad x \in r\{a\} \} \)
using func_imagedef by simp
{ fix y assume A4: \( y \in f(r\{a\}) \)
with I obtain \( x \) where
\( x \in r\{a\} \) and II: \( y = f(x) \)
by auto
with A1 A2 have \( \langle f(a),f(x) \rangle \in R \) using ord_iso_apply
by auto
with II have \( y \in R\{f(a)\} \) by auto
} then show \( f(r\{a\}) \subseteq R\{f(a)\} \) by auto
{ fix \( y \) assume A5: \( y \in R\{f(a)\} \)
let \( x = \text{converse}(f)(y) \)
from A2 A5 have \( \langle f(a),y \rangle \in R \quad f(a) \in B \) and IV: \( y \in B \)
by auto
with A1 have III: \( \langle \text{converse}(f)(f(a)),x \rangle \in r \)
using ord_iso_converse by simp
moreover from A1 A3 have \( \text{converse}(f)(f(a)) = a \)
using ord_iso_is_bij left_inverse_bij by blast
ultimately have \( f(x) \in \{f(x). \quad x \in r\{a\} \} \)
by auto
moreover from A1 IV have \( f(x) = y \)
using ord_iso_is_bij right_inverse_bij by blast
moreover from A1 I have \( f(r\{a\}) = \{f(x). \quad x \in r\{a\} \} \)
using ord_iso_is_bij bij_is_fun func_imagedef by blast
ultimately have \( y \in f(r\{a\}) \) by simp
} then show \( R\{f(a)\} \subseteq f(r\{a\}) \) by auto
qed
Order isomorphisms preserve collections of upper bounds.

**Lemma ord_iso_pres_up_bounds:**

- Assumes $A1: f \in \text{ord_iso}(A,r,B,R)$ and $A2: r \subseteq A \times A \quad R \subseteq B \times B$ and $A3: C \subseteq A$
- Shows $\{f(r\{a\}). \quad a \in C\} = \{R\{b\}. \quad b \in f(C)\}$

**Proof**

- From $A1$ have $f:A \rightarrow B$
  - Using $\text{ord_iso_is_bij}$ bij_is_fun by simp
- From $A3$ $\langle a \in C \rangle$ have $a \in A$ by auto
  - With $A1$ $A2$ have $f(r\{a\}) = R\{f(a)\}$
  - Moreover from $A3$ $\langle f:A \rightarrow B \rangle \langle a \in C \rangle$ have $f(a) \in f(C)$
  - Using $\text{func_imagedef}$ by auto
  - Ultimately have $f(r\{a\}) \in \{R\{b\}. \quad b \in f(C)\}$
  - By auto
  - With $I$ have $Y \in \{R\{b\}. \quad b \in f(C)\}$ by simp
- Fix $Y$ assume $Y \in \{R\{b\}. \quad b \in f(C)\}$
  - Then obtain $b$ where $b \in f(C)$ and $Y = R\{b\}$
  - By auto
  - With $A3$ $\langle f:A \rightarrow B \rangle$ obtain $a$ where $a \in C$ and $b = f(a)$
  - Using $\text{func_imagedef}$ by auto
  - With $A3$ $\langle f:A \rightarrow B \rangle$ have $a \in A$ and $Y = R(f(a))$ by auto
  - With $A1$ $A2$ have $Y = f(r\{a\})$
  - Using $\text{ord_iso_pres_rel_image}$ by simp
  - With $\langle a \in C \rangle$ have $Y \in \{f(r\{a\}). \quad a \in C\}$ by auto
- Then show $\{R\{b\}. \quad b \in f(C)\} \subseteq \{f(r\{a\}). \quad a \in C\}$
  - By auto

**QED**

The image of the set of upper bounds is the set of upper bounds of the image.

**Lemma ord_iso_pres_min_up_bounds:**

- Assumes $A1: f \in \text{ord_iso}(A,r,B,R)$ and $A2: r \subseteq A \times A \quad R \subseteq B \times B$ and $A3: C \subseteq A$ and $A4: C \neq 0$
- Shows $f(\bigcap a \in C. \quad r\{a\}) = (\bigcap b \in f(C). \quad R\{b\})$

**Proof**

- From $A1$ have $f \in \text{inj}(A,B)$
  - Using $\text{ord_iso_is_bij}$ bij_is_inj by simp
  - Moreover note $A4$
  - Moreover from $A2$ $A3$ have $\forall a \in C. \quad r\{a\} \subseteq A$ by auto
  - Ultimately have $f(\bigcap a \in C. \quad r\{a\}) = (\bigcap a \in C. \quad f(r\{a\}))$
  - Using $\text{inj_image_of_Inter}$ by simp

145
also from A1 A2 A3 have
( \bigcap_{a \in C} f(r(a)) ) = ( \bigcap_{b \in f(C)} R(b) )
using ord_iso_pres_up_bounds by simp
finally show f(\bigcap_{a \in C} r(a)) = (\bigcap_{b \in f(C)} R(b))
by simp
qed

Order isomorphisms preserve completeness.

lemma ord_iso_pres_compl:
  assumes A1: f \in ord_iso(A,r,B,R) and
  A2: r \subseteq A \times A R \subseteq B \times B and A3: R \{is complete\}
  shows r \{is complete\}
proof -
  { fix C
    assume A4: IsBoundedAbove(C,r) C \neq 0
    with A1 A2 A3 have
      HasAminimum(R,\bigcap_{b \in f(C)} R(b))
      using ord_iso_pres_bound_above IsComplete_def
      by simp
    moreover
    from A2 \{IsBoundedAbove(C,r)\} have I: C \subseteq A using Order_ZF_3_L1A
      by blast
    with A1 A2 \{C \neq 0\} have f(\bigcap_{a \in C} r(a)) = (\bigcap_{b \in f(C)} R(b))
      using ord_iso_pres_min_up_bounds by simp
    ultimately have HasAminimum(R,f(\bigcap_{a \in C} r(a)))
      by simp
    moreover
    from A2 have \forall a \in C. r(a) \subseteq A
      by auto
    with \{C \neq 0\} have ( \bigcap_{a \in C} r(a) ) \subseteq A using ZF1_1_L7
      by simp
    moreover note A1 A2
    ultimately have HasAminimum(r, \bigcap_{a \in C} r(a) )
      using ord_iso_pres_has_min by simp
  } then show r \{is complete\} using IsComplete_def
    by simp
qed

If the original relation is complete, then the induced one is complete.

lemma ind_rel_pres_compl: assumes A1: f \in bij(A,B)
  and A2: R \subseteq B \times B and A3: R \{is complete\}
  shows InducedRelation(f,R) \{is complete\}
proof -
  let r = InducedRelation(f,R)
  from A1 have f \in ord_iso(A,r,B,R)
    using bij_is_ord_iso by simp
  moreover from A1 A2 have r \subseteq A \times A
    using bij_is_fun ind_rel_domain by simp
  moreover note A2 A3

ultimately show r {is complete}
using ord_iso_pres_compl by simp
qed
end

13 Finite sets - introduction

theory Finite_ZF imports ZF1 Nat_ZF_IML ZF.Cardinal
begin

Standard Isabelle Finite.thy contains a very useful notion of finite powerset: the set of finite subsets of a given set. The definition, however, is specific to Isabelle and based on the notion of "datatype", obviously not something that belongs to ZF set theory. This theory file develops the notion of finite powerset similarly as in Finite.thy, but based on standard library’s Cardinal.thy. This theory file is intended to replace IsarMathLib’s Finite1 and Finite_ZF_1 theories that are currently derived from the ”datatype” approach.

13.1 Definition and basic properties of finite powerset

The goal of this section is to prove an induction theorem about finite powersets: if the empty set has some property and this property is preserved by adding a single element of a set, then this property is true for all finite subsets of this set.

We defined the finite powerset FinPow(X) as those elements of the powerset that are finite.

definition
FinPow(X) ≡ \{A ∈ Pow(X). Finite(A)\}

The cardinality of an element of finite powerset is a natural number.

lemma card_fin_is_nat: assumes A ∈ FinPow(X)
shows |A| ∈ nat and A ≈ |A|
using assms FinPow_def Finite_def cardinal_cong nat_into_Card
Card_cardinal_eq by auto

A reformulation of card_fin_is_nat: for a finit set A there is a bijection between |A| and A.

lemma fin_bij_card: assumes A1: A ∈ FinPow(X)
shows ∃b. b ∈ bij(|A|, A)
proof -
from A1 have |A| ≈ A using card_fin_is_nat eqpoll_sym
by blast
then show thesis using eqpoll_def by auto
qed

If a set has the same number of elements as \( n \in \mathbb{N} \), then its cardinality is \( n \).
Recall that in set theory a natural number \( n \) is a set with \( n \) elements.

**Lemma card_card:**

**Assumes** \( A \approx n \) and \( n \in \text{nat} \)

**Shows** \(|A| = n\)

**Using** asms cardinal_cong nat_into_Card Card_cardinal_eq

by auto

If we add a point to a finite set, the cardinality increases by one. To understand the second assertion \(|A \cup \{a\}| = |A| \cup \{|A|\}\) recall that the cardinality \(|A|\) of \( A \) is a natural number and for natural numbers we have \( n+1 = n \cup \{n\}\).

**Lemma card_fin_add_one:**

**Assumes** \( A1: A \in \text{FinPow}(X) \) and \( A2: a \in X-A \)

**Shows**

\(|A \cup \{a\}| = \text{succ}(|A|)\)

\(|A \cup \{a\}| = |A| \cup \{|A|\}\)

**Proof** -

from \( A1 \) \( A2 \) have \( \text{cons}(a,A) \approx \text{cons}(|A|,|A|) \)

using card_fin_is_nat mem_not_refl cons_eqpoll_cong

by auto

moreover have \( \text{cons}(a,A) = A \cup \{a\} \) by (rule consdef)

moreover have \( \text{cons}(|A|,|A|) = |A| \cup \{|A|\} \)

by (rule consdef)

ultimately have \( A \cup \{a\} \approx \text{succ}(|A|) \) using succ_explained

by simp

with \( A1 \) show

\(|A \cup \{a\}| = \text{succ}(|A|)\) and \(|A \cup \{a\}| = |A| \cup \{|A|\}\)

using card_fin_is_nat card_card by auto

qed

We can decompose the finite powerset into collection of sets of the same natural cardinalities.

**Lemma finpow_decomp:**

**Shows** \( \text{FinPow}(X) = (\bigcup n \in \text{nat}. \{A \in \text{Pow}(X). A \approx n\}) \)

**Using** Finite_def FinPow_def by auto

Finite powerset is the union of sets of cardinality bounded by natural numbers.

**Lemma finpow_union_card_nat:**

**Shows** \( \text{FinPow}(X) = (\bigcup n \in \text{nat}. \{A \in \text{Pow}(X). A \lesssim n\}) \)

**Proof** -

have \( \text{FinPow}(X) \subseteq (\bigcup n \in \text{nat}. \{A \in \text{Pow}(X). A \lesssim n\}) \)

using finpow_decomp FinPow_def eqpoll_imp_lepoll

by auto

moreover have

\( (\bigcup n \in \text{nat}. \{A \in \text{Pow}(X). A \lesssim n\}) \subseteq \text{FinPow}(X) \)
A different form of finpow_union_card_nat (see above) - a subset that has not more elements than a given natural number is in the finite powerset.

lemma lepoll_nat_in_finpow:
    assumes n ∈ nat  A ⊆ X  A ⊲ n
    shows A ∈ FinPow(X)
    using assms finpow_union_card_nat by auto

Natural numbers are finite subsets of the set of natural numbers.

lemma nat_finpow_nat: assumes n ∈ nat shows n ∈ FinPow(nat)
    using assms nat_into_Finite nat_subset_nat FinPow_def by simp

A finite subset is a finite subset of itself.

lemma fin_finpow_self: assumes A ∈ FinPow(X) shows A ∈ FinPow(A)
    using assms FinPow_def by auto

If we remove an element and put it back we get the set back.

lemma rem_add_eq: assumes a ∈ A shows (A-{a}) ∪ {a} = A
    using assms by auto

Induction for finite powerset. This is similar to the standard Isabelle’s Fin_induct.

theorem FinPow_induct: assumes A1: P(0) and
    A2: ∀ A ∈ FinPow(X). P(A) −→ (∀ a ∈ X. P(A ∪ {a})) and
    A3: B ∈ FinPow(X)
    shows P(B)
proof -
{ fix n assume n ∈ nat
  moreover from A1 have I: ∀ B ∈ Pow(X). B ⊲ 0 −→ P(B)
    using lepoll_0_is_0 by auto
  moreover have ∀ k ∈ nat.
    (∀ B ∈ Pow(X). (B ⊲ k −→ P(B))) −→
    (∀ B ∈ Pow(X). (B ⊲ succ(k) −→ P(B)))
  proof -
    { fix k assume A4: k ∈ nat
      assume A5: ∀ B ∈ Pow(X). (B ⊲ k −→ P(B))
      fix B assume A6: B ∈ Pow(X)  B ⊲ succ(k)
      have P(B)
      proof -
        have B = 0 −→ P(B)
        proof -
          { assume B = 0
            then have B ⊲ 0 using lepoll_0_iff
            by simp
          }{...}
with I A6 have P(B) by simp
} thus B = 0 → P(B) by simp
qed
moreover have B≠0 → P(B)
proof -
{ assume B ≠ 0
  then obtain a where II: a∈B by auto
  let A = B - {a}
  from A6 II have A ⊆ X and A ≲ k
  using Diff_sing_lepoll by auto
  with A4 A5 have A ∈ FinPow(X) and P(A)
  using lepoll_nat_in_finpow finpow_decomp
  by auto
  with A2 A6 II have P(A∪{a})
  by auto
  moreover from II have A∪{a} = B
  by auto
  ultimately have P(B) by simp
} thus B≠0 → P(B) by simp
qed
ultimately show P(B) by auto
qed

A subset of a finite subset is a finite subset.

lemma subset_finpow: assumes A ∈ FinPow(X) and B ⊆ A
  shows B ∈ FinPow(X)
  using assms FinPow_def subset_Finite by auto

If we subtract anything from a finite set, the resulting set is finite.

lemma diff_finpow:
  assumes A ∈ FinPow(X) shows A-B ∈ FinPow(X)
  using assms subset_finpow by blast

If we remove a point from a finite subset, we get a finite subset.

corollary fin_rem_point_fin: assumes A ∈ FinPow(X)
  shows A - {a} ∈ FinPow(X)
  using assms diff_finpow by simp

Cardinality of a nonempty finite set is a successor of some natural number.

lemma card_non_empty_succ:
assumes A1: A ∈ FinPow(X) and A2: A ≠ 0
shows ⋁ n ∈ nat. |A| = succ(n)
proof -
  from A2 obtain a where a ∈ A by auto
  let B = A - {a}
  from A1 ‹a ∈ A› have
  B ∈ FinPow(X) and a ∈ X - B
  using FinPow_def fin_rem_point_fin by auto
  then have |B ∪ {a}| = succ( |B| )
  using card_fin_add_one by auto
  moreover from ‹a ∈ A› ‹B ∈ FinPow(X)› have
  A = B ∪ {a} and |B| ∈ nat
  using card_fin_is_nat by auto
  ultimately show ⋁ n ∈ nat. |A| = succ(n) by auto
qed

Nonempty set has non-zero cardinality. This is probably true without the
assumption that the set is finite, but I couldn’t derive it from standard
Isabelle theorems.

lemma card_non_empty_non_zero:
  assumes A ∈ FinPow(X) and A ≠ 0
  shows |A| ≠ 0
proof -
  from assms obtain n where |A| = succ(n)
  using card_non_empty_succ by auto
  then show |A| ≠ 0 using succ_not_0 by simp
qed

Another variation on the induction theme: If we can show something holds
for the empty set and if it holds for all finite sets with at most k elements
then it holds for all finite sets with at most k + 1 elements, the it holds for
all finite sets.

theorem FinPow_card_ind: assumes A1: P(0) and
  A2: ∀ k ∈ nat. (∀ A ∈ FinPow(X). A ⊊ k −→ P(A)) −→
  (∀ A ∈ FinPow(X). A ⊊ succ(k) −→ P(A))
  and A3: A ∈ FinPow(X) shows P(A)
proof -
  from A3 have |A| ∈ nat and A ∈ FinPow(X) and A ⊊ |A|
  using card_fin_is_nat eqpoll_imp_lepoll by auto
  moreover have ∀ n ∈ nat. (∀ A ∈ FinPow(X). A ⊊ n −→ P(A))
  A ⊊ n −→ P(A)
  proof
  ‹fix n assume n ∈ nat
  moreover from A1 have ∀ A ∈ FinPow(X). A ⊊ 0 −→ P(A)
  using lepoll_0_is_0 by auto
  moreover note A2

151
ultimately show
\[ \forall A \in \text{FinPow}(X). ~ A \subseteq n \rightarrow P(A) \]
by (rule ind_on_nat)
qed
ultimately show \( P(A) \) by simp
qed

Another type of induction (or, maybe recursion). In the induction step we try to find a point in the set that if we remove it, the fact that the property holds for the smaller set implies that the property holds for the whole set.

lemma FinPow_ind_rem_one: assumes A1: \( P(0) \) and
A2: \( \forall A \in \text{FinPow}(X). ~ A \neq 0 \rightarrow (\exists a\in A. ~ P(A\{-a\}) \rightarrow P(A)) \)
and A3: \( B \in \text{FinPow}(X) \)
shows \( P(B) \)
proof -
note A1
moreover have \( \forall k \in \text{nat}. \)
(\( \forall B \in \text{FinPow}(X). ~ B \subseteq k \rightarrow P(B) \)) \rightarrow
(\( \forall C \in \text{FinPow}(X). ~ C \subseteq \text{succ}(k) \rightarrow P(C) \))
proof -
{ fix \( k \) assume \( k \in \text{nat} \)
assume A4: \( \forall B \in \text{FinPow}(X). ~ B \subseteq k \rightarrow P(B) \)
have \( \forall C \in \text{FinPow}(X). ~ C \subseteq \text{succ}(k) \rightarrow P(C) \)
proof -
{ fix \( C \) assume \( C \in \text{FinPow}(X) \)
assume \( C \subseteq \text{succ}(k) \)
note A1
moreover
{ assume \( C \neq 0 \)
with A2 \( \langle C \in \text{FinPow}(X) \rangle \) obtain \( a \) where
\( a \in C \) and \( P(C\{-a\}) \rightarrow P(C) \)
by auto
with A4 \( \langle C \in \text{FinPow}(X) \rangle \langle C \subseteq \text{succ}(k) \rangle \)
have \( P(C) \) using Diff_sing_lepoll fin_rem_point_fin
by simp }
ultimately have \( P(C) \) by auto
} thus thesis by simp
qed
} thus thesis by blast
qed
moreover note A3
ultimately show \( P(B) \) by (rule FinPow_card_ind)
qed

Yet another induction theorem. This is similar, but slightly more complicated than FinPow_ind_rem_one. The difference is in the treatment of the empty set to allow to show properties that are not true for empty set.

lemma FinPow_rem_ind: assumes A1: \( \forall A \in \text{FinPow}(X). ~ A = 0 \lor (\exists a\in A. ~ A = \{a\} \lor P(A\{-a\}) \rightarrow P(A)) \)

152
and A2: A ∈ FinPow(X) and A3: A ≠ 0
shows P(A)
proof -
  have 0 = 0 ∨ P(0) by simp
moreover have
  ∀k∈nat.
  (∀B ∈ FinPow(X). B ⊆ k → (B=0 ∨ P(B))) →
  (∀A ∈ FinPow(X). A ⊆ succ(k) → (A=0 ∨ P(A)))
proof -
  { fix k assume k ∈ nat
    assume A4: ∀B ∈ FinPow(X). B ⊆ k → (B=0 ∨ P(B))
    have ∀A ∈ FinPow(X). A ⊆ succ(k) → (A=0 ∨ P(A))
    proof -
    { fix A assume A ∈ FinPow(X)
      assume A ⊆ succ(k) A ≠ 0
      from A1 ‹A ∈ FinPow(X)› ‹A ≠ 0› obtain a
        where a∈A and A = {a} ∨ P(A-{a}) → P(A)
        by auto
      let B = A-{a}
      from A4 ‹A ∈ FinPow(X)› ‹A ⊆ succ(k)› ‹a∈A›
      have B = 0 ∨ P(B)
        using Diff_sing_lepoll fin_rem_point_fin
        by simp
      with ‹a∈A› ‹A = {a} ∨ P(A-{a}) → P(A)›
      have P(A) by auto
    } thus thesis by auto
    qed
    } thus thesis by blast
    qed
moreover note A2
ultimately have A=0 ∨ P(A) by (rule FinPow_card_ind)
with A3 show P(A) by simp
qed

If a family of sets is closed with respect to taking intersections of two sets
then it is closed with respect to taking intersections of any nonempty finite
collection.

lemma inter_two_inter_fin:
  assumes A1: ∀V∈T. ∀W∈T. V ∩ W ∈ T and
  A2: N ≠ 0 and A3: N ∈ FinPow(T)
  shows (⋂N ∈ T)
proof -
  have 0 = 0 ∨ (∅0 ∈ T) by simp
moreover have ∀M ∈ FinPow(T). (M = 0 ∨ ⋂M ∈ T) →
  (∀W ∈ T. M ∪ {W} = 0 ∨ ⋂(M ∪ {W}) ∈ T)
proof -
  { fix M assume M ∈ FinPow(T)
    assume A4: M = 0 ∨ ⋂M ∈ T
    { assume M = 0
hence \( \forall W \in T. M \cup \{W\} = 0 \lor \bigcap (M \cup \{W\}) \in T \)
by auto 
}
moreover 
{ assume \( M \neq 0 \)
with A4 have \( \bigcap M \in T \) by simp 
{ fix \( W \)
assume \( W \in T \)
from \( \langle M \neq 0 \rangle \) have \( \bigcap (M \cup \{W\}) = (\bigcap M) \cap W \)
by auto 
with A1 \( \langle \bigcap M \in T \rangle \) \( \langle W \in T \rangle \) have \( \bigcap (M \cup \{W\}) \in T \)
by simp 
} hence \( \forall W \in T. M \cup \{W\} = 0 \lor \bigcap (M \cup \{W\}) \in T \)
by simp 
ultimately have \( \forall W \in T. M \cup \{W\} = 0 \lor \bigcap (M \cup \{W\}) \in T \)
by blast 
} thus thesis by simp 
qed 
moreover note \( \langle N \in \text{FinPow}(T) \rangle \)
ultimately have \( N = 0 \lor (\bigcap N \in T) \)
by (rule FinPow_induct) 
with A2 show \( (\bigcap N \in T) \) by simp 
qed 

If a family of sets contains the empty set and is closed with respect to taking
unions of two sets then it is closed with respect to taking unions of any finite
collection.

lemma union_two_union_fin: 
asumes A1: \( 0 \in C \) and A2: \( \forall A \in C. \forall B \in C. A \cup B \in C \) and 
A3: \( N \in \text{FinPow}(C) \)
shows \( \bigcup N \in C \)
proof - 
from \( \langle 0 \in C \rangle \) have \( \bigcup 0 \in C \) by simp 
moreover have \( \forall M \in \text{FinPow}(C). \bigcup M \in C \longrightarrow (\forall A \in C. \bigcup (M \cup \{A\}) \in C) \)
proof - 
{ fix \( M \)
assume \( M \in \text{FinPow}(C) \)
assume \( \bigcup M \in C \)
fix \( A \)
assume \( A \in C \)
have \( \bigcup (M \cup \{A\}) = (\bigcup M) \cup A \) by auto 
with A2 \( \langle \bigcup M \in C \rangle \) \( \langle A \in C \rangle \) have \( \bigcup (M \cup \{A\}) \in C \)
by simp 
} thus thesis by simp 
qed 
moreover note \( \langle N \in \text{FinPow}(C) \rangle \)
ultimately show \( \bigcup N \in C \) by (rule FinPow_induct) 
qed 

Empty set is in finite power set.

lemma empty_in_finpow: shows \( 0 \in \text{FinPow}(X) \)
using FinPow_def by simp
Singleton is in the finite powerset.

**lemma singleton_in_finpow:** assumes \( x \in X \)
\( \\text{shows} \ \{x\} \in \text{FinPow}(X) \) using assms FinPow_def by simp

Union of two finite subsets is a finite subset.

**lemma union_finpow:** assumes \( A \in \text{FinPow}(X) \) and \( B \in \text{FinPow}(X) \)
\( \\text{shows} \ A \cup B \in \text{FinPow}(X) \) using assms FinPow_def by auto

Union of finite number of finite sets is finite.

**lemma fin_union_finpow:** assumes \( M \in \text{FinPow}(\text{FinPow}(X)) \)
\( \\text{shows} \ \bigcup M \in \text{FinPow}(X) \) using assms empty_in_finpow union_finpow union_two_union_fin by simp

If a set is finite after removing one element, then it is finite.

**lemma rem_point_fin_fin:** assumes A1: \( x \in X \) and A2: \( A - \{x\} \in \text{FinPow}(X) \)
\( \\text{shows} \ A \in \text{FinPow}(X) \)
proof -
\( \text{from} \text{assms} \ \text{have} \ (A - \{x\}) \cup \{x\} \in \text{FinPow}(X) \)
using singleton_in_finpow union_finpow by simp
moreover have \( A \subseteq (A - \{x\}) \cup \{x\} \) by auto
ultimately show \( A \in \text{FinPow}(X) \)
using FinPow_def subset_Finite by auto
qed

An image of a finite set is finite.

**lemma fin_image_fin:** assumes \( \forall V \in B. \ K(V) \in C \) and \( N \in \text{FinPow}(B) \)
\( \\text{shows} \ \{K(V). \ V \in N\} \in \text{FinPow}(C) \)
proof -
\( \text{have} \ \{K(V). \ V \in 0\} \in \text{FinPow}(C) \) using FinPow_def
by auto
moreover have \( \forall A \in \text{FinPow}(B). \ \{K(V). \ V \in A\} \in \text{FinPow}(C) \) using assms by simp
proof -
\{ fix A assume A \in FinPow(B)
assume \( \{K(V). \ V \in A\} \in \text{FinPow}(C) \)
fix a assume a\inB
have \( \{K(V). \ V \in (A \cup \{a\})\} \in \text{FinPow}(C) \)
proof -
have \( \{K(V). \ V \in (A \cup \{a\})\} = \{K(V). \ V\in A\} \cup \{K(a)\} \)
by auto
moreover note \( \{K(V). \ V\in A\} \in \text{FinPow}(C) \)
moreover from \( \forall V \in B. \ K(V) \in C \) \( \forall a \in B \) have \( \{K(a)\} \in \text{FinPow}(C) \)
using singleton_in_finpow by simp
ultimately show thesis using union_finpow by simp
qed
Union of a finite indexed family of finite sets is finite.

lemma union_fin_list_fin:
  assumes A1: n ∈ nat and A2: ∀k ∈ n. N(k) ∈ FinPow(X)
  shows
  {N(k). k ∈ n} ∈ FinPow(FinPow(X)) and (∪k ∈ n. N(k)) ∈ FinPow(X)
proof -
  from A1 have n ∈ FinPow(n)
    using nat_finpow_nat fin_finpow_self by auto
  with A2 show {N(k). k ∈ n} ∈ FinPow(FinPow(X))
    by (rule fin_image_fin)
  then show (∪k ∈ n. N(k)) ∈ FinPow(X)
    using fin_union_finpow by simp
qed
end

14 Finite sets

theory Finitel imports ZF.EquivClass ZF.Finite func1 ZF1
begin

This theory extends Isabelle standard Finite theory. It is obsolete and should not be used for new development. Use the Finite_ZF instead.

14.1 Finite powerset

In this section we consider various properties of Fin datatype (even though there are no datatypes in ZF set theory).

In Topology_ZF theory we consider induced topology that is obtained by taking a subset of a topological space. To show that a topology restricted to a subset is also a topology on that subset we may need a fact that if $T$ is a collection of sets and $A$ is a set then every finite collection $\{V_i\}$ is of the form $V_i = U_i \cap A$, where $\{U_i\}$ is a finite subcollection of $T$. This is one of those trivial facts that require surprisingly long formal proof. Actually, the need for this fact is avoided by requiring intersection two open sets to be open (rather than intersection of a finite number of open sets). Still, the fact is left here as an example of a proof by induction. We will use Fin_induct.
lemma from Finite.thy. First we define a property of finite sets that we want
to show.

**definition**

\[ \text{Prfin}(T,A,M) \equiv ( (M = 0) \mid (\exists N \in \text{Fin}(T). \ \forall V \in M. \ \exists U \in N. \ (V = U \cap A)) ) \]

Now we show the main induction step in a separate lemma. This will make
the proof of the theorem \text{FinRestr} below look short and nice. The premises
of the \text{ind_step} lemma are those needed by the main induction step in lemma
\text{Fin_induct} (see standard Isabelle's Finite.thy).

**lemma ind_step:** assumes

\[ A : \ \forall V \in T. A. \ \exists U \in T. \ V = U \cap A \]

and

\[ A1: W \in T. A \]

and

\[ A2: M \in \text{Fin}(T) \]

and

\[ A3: W \not\in M \]

and

\[ A4: \text{Prfin}(T,A,M) \]

shows \[ \text{Prfin}(T,A,\text{cons}(W,M)) \]

**proof** -

{ assume \[ A7: M = 0 \]

have \[ \text{Prfin}(T,A,\text{cons}(W,M)) \]

proof-

from \[ A1 \]

obtain \[ U \] where

\[ A5: U \in T \]

and

\[ A6: W = U \cap A \]

by fast

let \[ N = \{U\} \]

from \[ A5 \]

have \[ T1: N \in \text{Fin}(T) \]

by simp

from \[ A7 \]

\[ A6 \]

have \[ T2: \ \forall V \in \text{cons}(W,M). \ \exists U \in N. \ V = U \cap A \]

by simp

from \[ A7 \]

\[ T1 \]

\[ T2 \]

show \[ \text{Prfin}(T,A,\text{cons}(W,M)) \]

using \[ \text{Prfin_def} \]

by auto

qed }

moreover

{ assume \[ A8: M \neq 0 \]

have \[ \text{Prfin}(T,A,\text{cons}(W,M)) \]

proof-

from \[ A1 \]

\[ A \]

obtain \[ U \]

where

\[ A5: U \in T \]

and

\[ A6: W = U \cap A \]

by fast

from \[ A8 \]

\[ A4 \]

obtain \[ N0 \]

where

\[ A9: N0 \in \text{Fin}(T) \]

and

\[ A10: \ \forall V \in M. \ \exists U0 \in N0. \ (V = U0 \cap A) \]

using \[ \text{Prfin_def} \]

by auto

let \[ N = \text{cons}(U,N0) \]

moreover from \[ A10 \]

\[ A6 \]

have \[ \forall V \in \text{cons}(W,M). \ \exists U \in N. \ V = U \cap A \]

by simp

ultimately have \[ \exists N \in \text{Fin}(T). \ \forall V \in \text{cons}(W,M). \ \exists U \in N. \ V = U \cap A \]

by auto

with \[ A8 \]

show \[ \text{Prfin}(T,A,\text{cons}(W,M)) \]

using \[ \text{Prfin_def} \]

by simp

ultimately show thesis by auto

qed }

ultimately show thesis by auto

qed

Now we are ready to prove the statement we need.

**theorem \text{FinRestr0}:** assumes

\[ A: \ \forall V \in T. A. \ \exists U \in T. \ V = U \cap A \]

shows \[ \forall M \in \text{Fin}(T). \ \text{Prfin}(T,A,M) \]

**proof** -

{ fix \[ M \]

\[ \text{assume} \ M \in \text{Fin}(T) \]

moreover have \[ \text{Prfin}(T,A,0) \]

using \[ \text{Prfin_def} \]

moreover

...
{ fix \( W \) assume \( W \in T \subseteq M \subseteq \text{Fin}(T) \) with \( A \) have \( \text{Prfin}(T,A,W) \) by (rule ind_step) } 
ultimately have \( \text{Prfin}(T,A,M) \) by (rule Fin_induct) 
thus thesis by simp 
qed

This is a different form of the above theorem:

**TheoremZF1FinRestr:**
assumes \( A_1 : M \in \text{Fin}(T) \) and \( A_2 : M \neq 0 \) and \( A_3 : \forall V \in T. \exists U \subseteq V \) shows \( \exists N \subseteq \text{Fin}(T). (\forall V \in M. \exists U \subseteq N. (V = U \cap A)) \land N \neq 0 \)

**Proof:**
- from \( A_3 \) \( A_1 \) have \( \text{Prfin}(T,A,M) \) using FinRestr0 by blast
then have \( \exists N \subseteq \text{Fin}(T). (\forall V \in M. \exists U \subseteq N. (V = U \cap A)) \) by simp
- with \( A_2 \) have \( N \neq 0 \) by auto
- with \( D_1 \) show thesis by auto 
qed

Purely technical lemma used in Topology_ZF_1 to show that if a topology is \( T_2 \), then it is \( T_1 \).

**LemmaFinite1_L2:**
assumes \( A : \exists U V. (U \in T \land V \in T \land x \in U \land y \in V \land U \cup V = 0) \) shows \( \exists U \in T. (x \in U \land y \notin U) \)

**Proof:**
- from \( A \) obtain \( U V \) where \( D_1 : U \in T \land V \in T \land x \in U \land y \in V \land U \cup V = 0 \) by auto
- with \( D_1 \) show thesis by auto
qed

A collection closed with respect to taking a union of two sets is closed under taking finite unions. Proof by induction with the induction step formulated in a separate lemma.

**LemmaFinite1_L3_IndStep:**
assumes \( A_1 : \forall A B. ((A \in C \land B \in C) \rightarrow A \cup B \in C) \) and \( A_2 : A \in C \) and \( A_3 : N \in \text{Fin}(C) \) and \( A_4 : A \notin N \) and \( A_5 : \bigcup N \subseteq C \) shows \( \bigcup \text{cons}(A,N) \subseteq C \)

**Proof:**
- have \( \bigcup \text{cons}(A,N) = A \cup \bigcup N \) by blast
- with \( A_1 A_2 A_5 \) show thesis by simp
qed

The lemma: a collection closed with respect to taking a union of two sets is closed under taking finite unions.

**LemmaFinite1_L3:**
assumes \( A_1 : 0 \in C \) and \( A_2 : \forall A B. ((A \in C \land B \in C) \rightarrow A \cup B \in C) \) and \( A_3 : N \in \text{Fin}(C) \)
A collection closed with respect to taking an intersection of two sets is closed under taking finite intersections. Proof by induction with the induction step formulated in a separate lemma. This is slightly more involved than the union case in Finite1_L3, because the intersection of empty collection is undefined (or should be treated as such). To simplify notation we define the property to be proven for finite sets as a separate notion.

definition
  \text{IntPr}(T,N) \equiv (N = 0 \mid \bigcap N \in T)

The induction step.

lemma Finite1_L4_IndStep:
  assumes A1: \forall A B. ((A \in T \land B \in T) \longrightarrow A \cap B \in T)
  and A2: A \in T
  and A3: \neg \in \text{Fin}(T)
  and A4: A \notin N
  and A5: \text{IntPr}(T,N)
  shows \text{IntPr}(T,\text{cons}(A,N))

proof -
  { assume A6: N=0
    with A2 have \text{IntPr}(T,\text{cons}(A,N))
    using \text{IntPr_def} by simp }
  moreover
  { assume A7: N\notin0 have \text{IntPr}(T, \text{cons}(A, N))
    proof -
      from A7 A5 A2 A1 have \bigcap N \cap A \in T
      using \text{IntPr_def} by simp
      moreover from A7 have \bigcap \text{cons}(A, N) = \bigcap N \cap A
      by auto
      ultimately show \text{IntPr}(T, \text{cons}(A, N))
      using \text{IntPr_def} by simp
    qed }
  ultimately show thesis by auto
qed

The lemma.

lemma Finite1_L4:
  assumes A1: \forall A B. A \in T \land B \in T \longrightarrow A \cap B \in T
  and A2: N \in \text{Fin}(T)
  shows \text{IntPr}(T,N)

proof -
  note A2
  moreover have \text{IntPr}(T,0)
  using \text{IntPr_def} by simp
  moreover
A N
assume A ∈ T
N ∈ Fin(T) A /∈ N
IntPr(T,N)
with A1 have IntPr(T,cons(A,N)) by (rule Finite1_L4_IndStep) }
ultimately show IntPr(T,N) by (rule Fin_induct)
qed

Next is a restatement of the above lemma that does not depend on the IntPr
meta-function.

lemma Finite1_L5:
assumes A1: ∀ A B. ((A ∈ T ∧ B ∈ T) → A ∩ B ∈ T)
and A2: N ≠ 0 and A3: N ∈ Fin(T)
shows \( \bigcap N ∈ T \)
proof -
from A1 A3 have IntPr(T,N) using Finite1_L4 by simp
with A2 show thesis using IntPr_def by simp
qed

The images of finite subsets by a meta-function are finite. For example in
topology if we have a finite collection of sets, then closing each of them
results in a finite collection of closed sets. This is a very useful lemma with
many unexpected applications. The proof is by induction. The next lemma
is the induction step.

lemma fin_image_fin_IndStep:
assumes ∀ V ∈ B. K(V) ∈ C
and U ∈ B and N ∈ Fin(B) and U /∈ N
{K(V). V ∈ N} ∈ Fin(C)
shows {K(V). V ∈ cons(U,N)} ∈ Fin(C)
using assms by simp

The lemma:

lemma fin_image_fin:
assumes A1: ∀ V ∈ B. K(V) ∈ C and A2: N ∈ Fin(B)
shows {K(V). V ∈ N} ∈ Fin(C)
proof -
note A2
moreover have {K(V). V ∈ 0} ∈ Fin(C) by simp
moreover
{ fix U N
assume U ∈ B N ∈ Fin(B) U /∈ N {K(V). V ∈ N} ∈ Fin(C)
with A1 have {K(V). V ∈ cons(U,N)} ∈ Fin(C)
by (rule fin_image_fin_IndStep) }
ultimately show thesis by (rule Fin_induct)
qed

The image of a finite set is finite.

lemma Finite1_L6A: assumes A1: f:X → Y and A2: N ∈ Fin(X)
shows f(N) ∈ Fin(Y)
proof -
from A1 have ∀ x ∈ X. f(x) ∈ Y

160
using apply_type by simp

moreover note A2
ultimately have \{f(x). x \in N\} \in \text{Fin}(Y)
  by (rule fin_image_fin)
with A1 A2 show thesis
  using FinD func_imagedef by simp
qed

If the set defined by a meta-function is finite, then every set defined by a
composition of this meta function with another one is finite.

lemma Finite1_L6B:
  assumes A1: \(\forall x \in X. a(x) \in Y\) and A2: \{b(y). y \in Y\} \in \text{Fin}(Z)
  shows \{b(a(x)). x \in X\} \in \text{Fin}(Z)
proof -
  from A1 have \{b(a(x)). x \in X\} \subseteq \{b(y). y \in Y\} by auto
  with A2 show thesis using Fin_subset_lemma by blast
qed

If the set defined by a meta-function is finite, then every set defined by a
composition of this meta function with another one is finite.

lemma Finite1_L6C:
  assumes A1: \(\forall y \in Y. b(y) \in Z\) and A2: \{a(x). x \in X\} \in \text{Fin}(Y)
  shows \{b(a(x)). x \in X\} \in \text{Fin}(Z)
proof -
  let N = \{a(x). x \in X\}
  from A1 A2 have \{b(a(x)). x \in X\} \subseteq \{b(y). y \in N\} by auto
  moreove have \{b(a(x)). x \in X\} = \{b(y). y \in N\}
  by auto
  ultimately show thesis by simp
qed

Cartesian product of finite sets is finite.

lemma Finite1_L12: assumes A1: A \in \text{Fin}(A) and A2: B \in \text{Fin}(B)
  shows A \times B \in \text{Fin}(A \times B)
proof -
  have T1: \(\forall a \in A. \forall b \in B. \{(a,b)\} \in \text{Fin}(A \times B)\) by simp
  have \(\forall a \in A. \{\{a,b\}. b \in B\} \in \text{Fin}(\text{Fin}(A \times B))\)
  proof
    fix a assume A3: a \in A
    with T1 have \(\forall b \in B. \{(a,b)\} \in \text{Fin}(A \times B)\)
      by simp
    moreover note A2
    ultimately show \(\{\{a,b\}. b \in B\} \in \text{Fin}(\text{Fin}(A \times B))\)
      by (rule fin_image_fin)
  qed
  then have \(\forall a \in A. \bigcup \{\{a,b\}. b \in B\} \in \text{Fin}(A \times B)\)
    using Fin_UnionI by simp
moreover have
∀a∈A. \{\langle a,b \rangle. b ∈ B\} = \{a\}× B by blast
ultimately have ∀a∈A. \{a\}× B ∈ Fin(A×B) by simp
moreover note A1
ultimately have \{\{a\}× B. a∈A\} ∈ Fin(Fin(A×B))
  by (rule fin_image_fin)
then have \bigcup\{\{a\}× B. a∈A\} ∈ Fin(A×B)
  using Fin_UnionI by simp
moreover have \bigcup\{\{a\}× B. a∈A\} = A×B by blast
ultimately show thesis by simp
qed

We define the characteristic meta-function that is the identity on a set and assigns a default value everywhere else.

definition
Characteristic(A,default,x) ≡ (if x∈A then x else default)

A finite subset is a finite subset of itself.

lemma Finite1_L13: assumes A1: A ∈ Fin(X) shows A ∈ Fin(A)
proof -
  { assume A=0 hence A ∈ Fin(A) by simp }
moreover
  { assume A2: A≠0 then obtain c where D1:c∈A
    by auto
    then have ∀x∈X. Characteristic(A,c,x) ∈ A
      using Characteristic_def by simp
    moreover note A1
    ultimately have \{Characteristic(A,c,x). x∈A\} ∈ Fin(A) by (rule fin_image_fin)
    moreover from D1 have
      \{Characteristic(A,c,x). x∈A\} = A using Characteristic_def by simp
    ultimately have A ∈ Fin(A) by simp }
ultimately show thesis by blast
qed

Cartesian product of finite subsets is a finite subset of cartesian product.

lemma Finite1_L14: assumes A1: A ∈ Fin(X) B ∈ Fin(Y)
  shows A×B ∈ Fin(X×Y)
proof -
  from A1 have A×B ⊆ X×Y using FinD by auto
  then have Fin(A×B) ⊆ Fin(X×Y) using Fin_mono by simp
  moreover from A1 have A×B ∈ Fin(A×B)
    using Finite1_L13 Finite1_L12 by simp
  ultimately show thesis by auto
qed

The next lemma is needed in the Group_ZF_3 theory in a couple of places.

lemma Finite1_L15:
assumes A1: \{b(x). x \in A\} \in \text{Fin}(B) \quad \text{and} \quad \{c(x). x \in A\} \in \text{Fin}(C)

and A2: f : B \times C \rightarrow E

shows \{f(b(x),c(x)). x \in A\} \in \text{Fin}(E)

proof -
  from A1 have \{b(x). x \in A\} \times \{c(x). x \in A\} \in \text{Fin}(B \times C)
    using Finite1_L14 by simp
  moreover have \{ (b(x),c(x)). x \in A\} \subseteq \{b(x). x \in A\} \times \{c(x). x \in A\}
    by blast
  ultimately have T0: \{ (b(x),c(x)). x \in A\} \in \text{Fin}(B \times C)
    by (rule Fin_subset_lemma)
  with A2 have T1: \{f(b(x),c(x)). x \in A\} \in \text{Fin}(E)
    using Finite1_L6A by auto
  from T0 have \forall x \in A. \{ (b(x),c(x)) \} \in B \times C
    using FinD by auto
  with A2 have \{f(b(x),c(x)). x \in A\} = \{f(b(x),c(x)). x \in A\}
    using func1_1_L17 by simp
  with T1 show thesis by simp
qed

Singletons are in the finite powerset.

lemma Finite1_L16: assumes x \in X shows \{x\} \in \text{Fin}(X)
  using assms emptyI consI by simp

A special case of Finite1_L15 where the second set is a singleton. In Group_ZF_3 theory this corresponds to the situation where we multiply by a constant.

lemma Finite1_L16AA: assumes \{b(x). x \in A\} \in \text{Fin}(B) \quad \text{and} \quad c \in C \quad \text{and} \quad f : B \times C \rightarrow E

shows \{f(b(x),c). x \in A\} \in \text{Fin}(E)

proof -
  from assms have \forall y \in B. f(y,c) \in E
    \{b(x). x \in A\} \in \text{Fin}(B)
    using apply_funtype by auto
  then show thesis by (rule Finite1_L6C)
qed

First order version of the induction for the finite powerset.

lemma Finite1_L16B: assumes A1: P(0) and A2: B \in \text{Fin}(X)
  and A3: \forall A \in \text{Fin}(X). \forall x \in X. x \notin A \land P(A) \rightarrow P(A \cup \{x\})

shows P(B)

proof -
  note <B \in \text{Fin}(X)> and <P(0)>
  moreover
  \{ fix A x
    assume x \in X \quad A \in \text{Fin}(X) \quad x \notin A \quad P(A)

dr245742
moreover have \( \text{cons}(x,A) = A \cup \{x\} \) by auto
moreover note \( A^3 \)
ultimately have \( P(\text{cons}(x,A)) \) by simp }
ultimately show \( P(B) \) by (rule Fin_induct)
qed

14.2 Finite range functions

In this section we define functions \( f : X \rightarrow Y \), with the property that \( f(X) \) is a finite subset of \( Y \). Such functions play an important role in the construction of real numbers in the Real_ZF series.

Definition of finite range functions.

**definition**

\[ \text{FinRangeFunctions}(X,Y) \equiv \{f:X \rightarrow Y. f(X) \in \text{Fin}(Y)\} \]

Constant functions have finite range.

**lemma** Finite1_L17: assumes \( c \in Y \) and \( X \neq 0 \)
shows \( \text{ConstantFunction}(X,c) \in \text{FinRangeFunctions}(X,Y) \)
using assms func1_3_L1 func_imagedef func1_3_L2 Finite1_L16
\[ \text{FinRangeFunctions_def by simp} \]

Finite range functions have finite range.

**lemma** Finite1_L18: assumes \( f \in \text{FinRangeFunctions}(X,Y) \)
shows \( \{f(x). x \in X\} \in \text{Fin}(Y) \)
using assms FinRangeFunctions_def func_imagedef by simp

An alternative form of the definition of finite range functions.

**lemma** Finite1_L19: assumes \( f:X \rightarrow Y \)
and \( \{f(x). x \in X\} \in \text{Fin}(Y) \)
shows \( f \in \text{FinRangeFunctions}(X,Y) \)
using assms func_imagedef FinRangeFunctions_def by simp

A composition of a finite range function with another function is a finite range function.

**lemma** Finite1_L20: assumes A1: \( f \in \text{FinRangeFunctions}(X,Y) \)
and A2: \( g : Y \rightarrow Z \)
says \( g \circ f \in \text{FinRangeFunctions}(X,Z) \)
proof -
from A1 A2 have \( g\{f(x). x \in X\} \in \text{Fin}(Z) \)
using Finite1_L18 Finite1_L6A
by simp
with A1 A2 have \( \{(g \circ f)(x). x \in X\} \in \text{Fin}(Z) \)
using FinRangeFunctions_def apply_funtype
func1_1_L17 comp_fun_apply by auto
with A1 A2 show thesis using
\[ \text{FinRangeFunctions_def comp_fun Finite1_L19} \]
lemma Finite1_L21:
  assumes f ∈ FinRangeFunctions(X,Y) and A ⊆ X
  shows f(A) ∈ Fin(Y)
proof -
  from assms have f(X) ∈ Fin(Y) f(A) ⊆ f(X)
    using FinRangeFunctions_def func1_1_L8
    by auto
  then show f(A) ∈ Fin(Y) using Fin_subset_lemma
    by blast
qed

15 Finite sets 1

theory Finite_ZF_1 imports Finite1 Order_ZF_1a

begin

This theory is based on Finite1 theory and is obsolete. It contains properties of finite sets related to order relations. See the FinOrd theory for a better approach.

15.1 Finite vs. bounded sets

The goal of this section is to show that finite sets are bounded and have maxima and minima.

Finite set has a maximum - induction step.

lemma Finite_ZF_1_1_L1:
  assumes A1: r {is total on} X and A2: trans(r)
  and A3: A ∈ Fin(X) and A4: x ∈ X and A5: A=0 ∨ HasAmaximum(r,A)
  shows A ∪ {x} = 0 ∨ HasAmaximum(r,A ∪ {x})
proof -
  { assume A=0 then have T1: A ∪ {x} = {x} by simp
    from A1 have refl(X,r) using total_is_refl by simp
    with T1 A4 have A ∪ {x} = 0 ∨ HasAmaximum(r,A ∪ {x})
      using Order_ZF_4_L8 by simp }
  moreover
  { assume A≠0
    with A1 A2 A3 A4 A5 have A ∪ {x} = 0 ∨ HasAmaximum(r,A ∪ {x})
      using FinD Order_ZF_4_L9 by simp }
  ultimately show thesis by blast

165
qed

For total and transitive relations finite set has a maximum.

**Theorem Finite_ZF_1_1_T1A:**

_**Assumptions:**_
- \( A1: r \text{ is total on } X \)
- \( A2: \text{trans}(r) \)
- \( A3: B \in \text{Fin}(X) \)

_**Shows:**_
- \( B = 0 \lor \text{HasAmaximum}(r, B) \)

_**Proof:**_

- \( 0 = 0 \lor \text{HasAmaximum}(r, 0) \) by simp
- moreover note \( A3 \)
- moreover from \( A1 \) \( A2 \) have \( \forall A \in \text{Fin}(X). \forall x \in X. \)
  \( x \notin A \land (A = 0 \lor \text{HasAmaximum}(r, A)) \rightarrow (A \cup \{x\} = 0 \lor \text{HasAmaximum}(r, A \cup \{x\})) \)
  using Finite_ZF_1_1_L1 by simp
- ultimately show \( B = 0 \lor \text{HasAmaximum}(r, B) \) by (rule Finite1_L16B)

qed

Finite set has a minimum - induction step.

**Lemma Finite_ZF_1_1_L2:**

_**Assumptions:**_
- \( A1: r \text{ is total on } X \)
- \( A2: \text{trans}(r) \)
- \( A3: A \in \text{Fin}(X) \)
- \( A4: x \in X \)
- \( A5: A = 0 \lor \text{HasAminimum}(r, A) \)

_**Shows:**_
- \( A \cup \{x\} = 0 \lor \text{HasAminimum}(r, A \cup \{x\}) \)

_**Proof:**_

- \( \{ \) assume \( A = 0 \) then have \( T1: A \cup \{x\} = \{x\} \) by simp
  from \( A1 \) have \( \text{refl}(X, r) \) using total_is_refl by simp
  with \( T1 \) \( A4 \) have \( A \cup \{x\} = 0 \lor \text{HasAminimum}(r, A \cup \{x\}) \)
    using Order_ZF_4_L8 by simp \}
- moreover
- \( \{ \) assume \( A \neq 0 \)
  with \( A1 \) \( A2 \) \( A3 \) \( A4 \) \( A5 \) have \( A \cup \{x\} = 0 \lor \text{HasAminimum}(r, A \cup \{x\}) \)
    using FinD Order_ZF_4_L10 by simp \}
- ultimately show thesis by blast

qed

For total and transitive relations finite set has a minimum.

**Theorem Finite_ZF_1_1_T1B:**

_**Assumptions:**_
- \( A1: r \text{ is total on } X \)
- \( A2: \text{trans}(r) \)
- \( A3: B \in \text{Fin}(X) \)

_**Shows:**_
- \( B = 0 \lor \text{HasAminimum}(r, B) \)

_**Proof:**_

- \( 0 = 0 \lor \text{HasAminimum}(r, 0) \) by simp
- moreover note \( A3 \)
- moreover from \( A1 \) \( A2 \) have \( \forall A \in \text{Fin}(X). \forall x \in X. \)
  \( x \notin A \land (A = 0 \lor \text{HasAminimum}(r, A)) \rightarrow (A \cup \{x\} = 0 \lor \text{HasAminimum}(r, A \cup \{x\})) \)
  using Finite_ZF_1_1_L1 by simp
- ultimately show \( B = 0 \lor \text{HasAminimum}(r, B) \) by (rule Finite1_L16B)

qed

For transitive and total relations finite sets are bounded.

**Theorem Finite_ZF_1_T1:**
assumes A1: \( r \) \{is total on\} \( X \) and A2: trans\((r)\)
and A3: \( B \in \text{Fin}(X) \)
shows IsBounded\((B,r)\)

proof -
from A1 A2 A3 have B=0 \( \lor \) HasAminimum\((r,B)\) B=0 \( \lor \) HasAmaximum\((r,B)\)
using Finite_ZF_1_1_T1A Finite_ZF_1_1_T1B by auto
then have
\( B = 0 \lor \) IsBoundedBelow\((B,r)\) B=0 \( \lor \) IsBoundedAbove\((B,r)\)
using Order_ZF_4_L7 Order_ZF_4_L8A by auto
then show IsBounded\((B,r)\) using
IsBounded_def IsBoundedBelow_def IsBoundedAbove_def by simp
qed

For linearly ordered finite sets maximum and minimum have desired properties. The reason we need linear order is that we need the order to be total and transitive for the finite sets to have a maximum and minimum and then we also need antisymmetry for the maximum and minimum to be unique.

theorem Finite_ZF_1_T2:
assumes A1: IsLinOrder\((X,r)\) and A2: \( A \in \text{Fin}(X) \) and A3: \( A \neq 0 \)
shows 
Maximum\((r,A)\) \( \in \) \( A \)
Minimum\((r,A)\) \( \in \) \( A \)
\( \forall x \in A \). \( \langle x,\text{Maximum}(r,A) \rangle \) \( \in \) \( r \)
\( \forall x \in A \). \( \langle \text{Minimum}(r,A),x \rangle \) \( \in \) \( r \)

proof -
from A1 have T1: \( r \) \{is total on\} \( X \) trans\((r)\) antisym\((r)\)
using IsLinOrder_def by auto
moreover from T1 A2 A3 have HasAmaximum\((r,A)\)
using Finite_ZF_1_1_T1A by auto
moreover from T1 A2 A3 have HasAminimum\((r,A)\)
using Finite_ZF_1_1_T1B by auto
ultimately show
Maximum\((r,A)\) \( \in \) \( A \)
Minimum\((r,A)\) \( \in \) \( A \)
\( \forall x \in A \). \( \langle x,\text{Maximum}(r,A) \rangle \) \( \in \) \( r \) \( \forall x \in A \). \( \langle \text{Minimum}(r,A),x \rangle \) \( \in \) \( r \)
using Order_ZF_4_L3 Order_ZF_4_L4 by auto
qed

A special case of Finite_ZF_1_T2 when the set has three elements.

corollary Finite_ZF_1_L2A:
assumes A1: IsLinOrder\((X,r)\) and A2: \( a \in X \) \( b \in X \) \( c \in X \)
shows 
Maximum\((r,\{a,b,c\})\) \( \in \) \( \{a,b,c\} \)
Minimum\((r,\{a,b,c\})\) \( \in \) \( \{a,b,c\} \)
Maximum\((r,\{a,b,c\})\) \( \in \) \( X \)
Minimum\((r,\{a,b,c\})\) \( \in \) \( X \)
\( \langle a,\text{Maximum}(r,\{a,b,c\}) \rangle \) \( \in \) \( r \)
\( \langle b,\text{Maximum}(r,\{a,b,c\}) \rangle \) \( \in \) \( r \)
\[ \langle c, \text{Maximum}(r, \{a, b, c\}) \rangle \in r \]

**proof** -

from A2 have I: \( \{a, b, c\} \in \text{Fin}(X) \) \( \{a, b, c\} \neq 0 \)
by auto
with A1 show II: \( \text{Maximum}(r, \{a, b, c\}) \in \{a, b, c\} \)
by (rule Finite_ZF_1_T2)
moreover from A1 I show III: \( \text{Minimum}(r, \{a, b, c\}) \in \{a, b, c\} \)
by (rule Finite_ZF_1_T2)
moreover from A2 have \( \{a, b, c\} \subseteq X \)
by auto
ultimately show \( \text{Maximum}(r, \{a, b, c\}) \in X \)
Minimum(r,\{a,b,c\})\in X
by auto
from A1 I have \( \forall x \in \{a, b, c\}. \langle x, \text{Maximum}(r, \{a, b, c\}) \rangle \in r \)
by (rule Finite_ZF_1_T2)
then show
\[ \langle a, \text{Maximum}(r, \{a, b, c\}) \rangle \in r \]
\[ \langle b, \text{Maximum}(r, \{a, b, c\}) \rangle \in r \]
\[ \langle c, \text{Maximum}(r, \{a, b, c\}) \rangle \in r \]
by auto

qed

If for every element of \( X \) we can find one in \( A \) that is greater, then the \( A \) cannot be finite. Works for relations that are total, transitive and antisymmetric.

**lemma** Finite_ZF_1_1_L3:

assumes A1: \( r \text{ is total on } X \)
and A2: \( \text{trans}(r) \) and A3: \( \text{antisym}(r) \)
and A4: \( r \subseteq X \times X \) and A5: \( X \neq 0 \)
and A6: \( \forall x \in X. \exists a \in A. x \neq a \land \langle x, a \rangle \in r \)
shows \( A \notin \text{Fin}(X) \)

**proof** -

from assms have \( \neg \text{IsBounded}(A, r) \)
using Order_ZF_3_L14 IsBounded_def
by simp
with A1 A2 show \( A \notin \text{Fin}(X) \)
by (rule Finite_ZF_1_T1) by auto

qed

**end**

**16 Finite sets and order relations**

theory FinOrd_ZF imports Finite_ZF func_ZF_1 NatOrder_ZF

begin

This theory file contains properties of finite sets related to order relations.
Part of this is similar to what is done in Finite_ZF_1 except that the development is based on the notion of finite powerset defined in Finite_ZF rather than the one defined in standard Isabelle finite theory.

### 16.1 Finite vs. bounded sets

The goal of this section is to show that finite sets are bounded and have maxima and minima.

For total and transitive relations nonempty finite set has a maximum.

**Theorem fin_has_max:**

assumes \( A1: \ r \text{ (is total on)} \ X \text{ and } A2: \text{trans}(r) \)
and \( A3: \ B \in \text{FinPow}(X) \) and \( A4: \ B \neq 0 \)
shows HasAmaximum\( (r,B) \)

**Proof -**

have \( 0 = 0 \lor \text{HasAmaximum}(r,0) \) by simp
moreover have
\( \forall A \in \text{FinPow}(X). \ A = 0 \lor \text{HasAmaximum}(r,A) \longrightarrow \)
\( (\forall x \in X. \ (A \cup \{x\}) = 0 \lor \text{HasAmaximum}(r,A \cup \{x\})) \)

**Proof -**

{ fix \( A \)
  assume \( A \in \text{FinPow}(X) \) \( A = 0 \lor \text{HasAmaximum}(r,A) \)
  have \( \forall x \in X. \ (A \cup \{x\}) = 0 \lor \text{HasAmaximum}(r,A \cup \{x\}) \)
  proof -
    { fix \( x \)
      assume \( x \in X \)
      note \( \langle A = 0 \lor \text{HasAmaximum}(r,A) \rangle \)
      moreover
      { assume \( A = 0 \)
        then have \( A \cup \{x\} = \{x\} \) by simp
        from \( A1 \) have \( \text{refl}(X,r) \) using total_is_refl
        by simp
        with \( \langle x \in X \rangle \langle A \cup \{x\} = \{x\} \rangle \) have \( \text{HasAmaximum}(r,A \cup \{x\}) \)
        using \( \text{OrderZF}_4\_L8 \) by simp }
      moreover
      { assume \( \text{HasAmaximum}(r,A) \)
        with \( A1 \ A2 \langle A \in \text{FinPow}(X) \rangle \langle x \in X \rangle \)
        have \( \text{HasAmaximum}(r,A \cup \{x\}) \)
        using \( \text{FinPow}\_\text{def} \text{OrderZF}_4\_L9 \) by simp }
      ultimately have \( A \cup \{x\} = 0 \lor \text{HasAmaximum}(r,A \cup \{x\}) \)
      by auto
    } thus \( \forall x \in X. \ (A \cup \{x\}) = 0 \lor \text{HasAmaximum}(r,A \cup \{x\}) \)
    by simp
    qed
  } thus thesis by simp
  qed
moreover note \( A3 \)
ultimately have \( B = 0 \lor \text{HasAmaximum}(r,B) \)
by (rule \( \text{FinPow}\_\text{induct} \)
with A4 show HasAmaximum(r,B) by simp 
qed

For linearly ordered nonempty finite sets the maximum is in the set and indeed it is the greatest element of the set.

lemma linord_max_props: assumes A1: IsLinOrder(X,r) and 
A2: A ∈ FinPow(X) A ≠ 0 
shows 
Maximum(r,A) ∈ A 
Maximum(r,A) ∈ X 
∀a∈A. ⟨a,Maximum(r,A)⟩ ∈ r
proof - 
from A1 A2 show 
Maximum(r,A) ∈ A and ∀a∈A. ⟨a,Maximum(r,A)⟩ ∈ r 
using IsLinOrder_def fin_has_max Order_ZF_4_L3 
by auto 
with A2 show Maximum(r,A) ∈ X using FinPow_def 
by auto 
qed

Every nonempty subset of a natural number has a maximum with expected properties.

lemma nat_max_props: assumes n∈nat A⊆n A≠0 
shows 
Maximum(Le,A) ∈ A 
Maximum(Le,A) ∈ nat 
∀k∈A. k ≤ Maximum(Le,A)
proof - 
from assms(1,2) have A ∈ FinPow(nat) 
using nat_finpow_nat subset_finpow by blast 
with assms(3) show 
Maximum(Le,A) ∈ A 
Maximum(Le,A) ∈ nat 
using NatOrder_ZF_1_L2(4) linord_max_props(1,2) by simp_all 
from assms(3) ⟨A ∈ FinPow(nat)⟩ have ∀k∈A. ⟨k,Maximum(Le,A)⟩ ∈ Le 
using linord_max_props NatOrder_ZF_1_L2(4) by blast 
then show ∀k∈A. k ≤ Maximum(Le,A) by simp 
qed

Yet another version of induction where the induction step is valid only up to n ∈ N rather than for all natural numbers. This lemma is redundant as it is easier to prove this assertion using lemma fin_nat_ind from Nat_ZF_IML which was done in lemma fin_nat_ind1 there. It is left here for now as an alternative proof based on properties of the maximum of a finite set.

lemma ind_on_nat2: 
assumes n∈nat and P(0) and ∀j∈n. P(j)→P(j #+ 1) 
shows ∀j∈n #+ 1. P(j) and P(n)
proof -
let \( A = \{ k \in \text{succ}(n). \ \forall j \in \text{succ}(k). \ P(j) \} \)
let \( M = \text{Maximum}(\text{Le}, A) \)

from \( \text{assms}(1,2) \) have \( I: \text{succ}(n) \in \text{nat} A \subseteq \text{succ}(n) A \neq 0 \)
using \( \text{empty_in_every_succ} \) by \( \text{auto} \)
then have \( n = M \) by \( \text{rule nat_max_props} \)

have \( n = M \)
proof -
  from \( \langle M \in A \rangle \) have \( M \in \text{succ}(n) \) by \( \text{blast} \)
with \( \text{assms}(1) \) have \( M \in n \lor M = n \) by \( \text{auto} \)
moreover
  \{ assume \( M \in n \)
   from \( I \) have \( M \in \text{nat} \) by \( \text{rule nat_max_props} \)
   from \( \text{assms}(3) \) \( \langle M \in A \rangle \langle M \in n \rangle \) have \( P(M + 1) \) by \( \text{blast} \)
   with \( \langle M \in \text{nat} \rangle \) have \( P(\text{succ}(M)) \) using \( \text{succ_add_one}(1) \) by \( \text{simp} \)
   with \( \langle M \in A \rangle \) have \( \forall j \in \text{succ}(\text{succ}(M)). \ P(j) \) by \( \text{blast} \)
   moreover from \( \text{assms}(1) \) \( \langle M \in n \rangle \) have \( \text{succ}(M) \in \text{succ}(n) \)
   using \( \text{succ_ineq1} \) by \( \text{simp} \)
   moreover from \( I \) have \( \forall k \in A. \ k \leq M \)
   by \( \text{rule nat_max_props} \)
   ultimately have \( \text{False} \) by \( \text{blast} \)
  \}
ultimately show \( n = M \) by \( \text{auto} \)
qed

with \( \langle M \in A \rangle \) have \( n \in A \) by \( \text{rule eq_mem} \)
with \( \text{assms}(1) \) show \( \forall j \in n \# 1 . P(j) \) and \( P(n) \)
using \( \text{succ_add_one}(1) \) by \( \text{simp_all} \)

16.2 Order isomorphisms of finite sets

In this section we establish that if two linearly ordered finite sets have the same number of elements, then they are order-isomorphic and the isomorphism is unique. This allows us to talk about "enumeration" of a linearly ordered finite set. We define the enumeration as the order isomorphism between the number of elements of the set (which is a natural number \( n = \{0, 1, \ldots, n-1\} \)) and the set.

A really weird corner case - empty set is order isomorphic with itself.

lemma \( \text{empty_ord_iso} \): shows \( \text{ord_iso}(0, r, 0, R) \neq 0 \)
proof -
  have \( 0 \approx 0 \) using \( \text{eqpoll_refl} \) by \( \text{simp} \)
  then obtain \( f \) where \( f \in \text{bij}(0, 0) \)
  using \( \text{eqpoll_def} \) by \( \text{blast} \)
  then show thesis using \( \text{ord_iso_def} \) by \( \text{auto} \)
qed

Even weirder than \( \text{empty_ord_iso} \) The order automorphism of the empty set is unique.
lemma empty_ord_iso_uniq:
  assumes f ∈ ord_iso(0,r,0,R) g ∈ ord_iso(0,r,0,R)
  shows f = g
proof -
  from assms have f : 0 → 0 and g: 0 → 0
    using ord_iso_def bij_def surj_def by auto
  moreover have ∀x∈0. f(x) = g(x) by simp
  ultimately show f = g by (rule func_eq)
qed

The empty set is the only order automorphism of itself.

lemma empty_ord_iso_empty: shows ord_iso(0,r,0,R) = {0}
proof -
  have 0 ∈ ord_iso(0,r,0,R)
  proof -
    have ord_iso(0,r,0,R) ≠ 0 by (rule empty_ord_iso)
    then obtain f where f ∈ ord_iso(0,r,0,R) by auto
    then show 0 ∈ ord_iso(0,r,0,R)
      using ord_iso_def bij_def surj_def fun_subset_prod
        by auto
  qed
  then show ord_iso(0,r,0,R) = {0} using empty_ord_iso_uniq
    by blast
qed

An induction (or maybe recursion?) scheme for linearly ordered sets. The
induction step is that we show that if the property holds when the set is
a singleton or for a set with the maximum removed, then it holds for the
set. The idea is that since we can build any finite set by adding elements on
the right, then if the property holds for the empty set and is invariant with
respect to this operation, then it must hold for all finite sets.

lemma fin_ord_induction:
  assumes A1: IsLinOrder(X,r) and A2: P(0) and
  A3: ∀A ∈ FinPow(X). A ≠ 0 −→ (P(A - {Maximum(r,A)}) −→ P(A))
  and A4: B ∈ FinPow(X) shows P(B)
proof -
  note A2
  moreover have ∀ A ∈ FinPow(X). A ≠ 0 −→ (∃a∈A. P(A-{a}) −→ P(A))
  proof -
    { fix A assume A ∈ FinPow(X) and A ≠ 0
      with A1 A3 have ∃a∈A. P(A-{a}) −→ P(A)
        using IsLinOrder_def fin_has_max
          IsLinOrder_def Order_ZF_4_L3
          by blast
      } thus thesis by simp
  qed
  moreover note A4
  ultimately show P(B) by (rule FinPow_ind_rem_one)

172
A slightly more complicated version of fin_ord_induction that allows to prove properties that are not true for the empty set.

**lemma fin_ord_ind:**

assumes A1: IsLinOrder(X,r) and A2: ∀A ∈ FinPow(X).
A = 0 ∨ (A = {Maximum(r,A)}) ∨ P(A - {Maximum(r,A)}) → P(A)
and A3: B ∈ FinPow(X) and A4: B ≠ 0
shows P(B)

**proof -**

1. fix A assume A ∈ FinPow(X) and A ≠ 0
   2. with A1 A2 have
      ∃a∈A. A = {a} ∨ P(A-{a}) → P(A)
      using IsLinOrder_def fin_has_max
      IsLinOrder_def Order_ZF_4_L3
      by blast
   3. then have ∀A ∈ FinPow(X).
      A = 0 ∨ (∃a∈A. A = {a} ∨ P(A-{a}) → P(A))
      by auto
   4. with A3 A4 show P(B) using FinPow_rem_ind
      by simp

qed

Yet another induction scheme. We build a linearly ordered set by adding elements that are greater than all elements in the set.

**lemma fin_ind_add_max:**

assumes A1: IsLinOrder(X,r) and A2: P(0) and A3: ∀ A ∈ FinPow(X).
( ∀ x ∈ X-A. P(A) ∧ (∀a∈A. ⟨a,x⟩ ∈ r ) → P(A ∪ {x}))
and A4: B ∈ FinPow(X)
shows P(B)

**proof -**

1. note A1 A2
   2. moreover have
      ∀C ∈ FinPow(X). C ≠ 0 → (P(C - {Maximum(r,C)}) → P(C))
      proof -
      1. fix C assume C ∈ FinPow(X) and C ≠ 0
      2. let x = Maximum(r,C)
      3. let A = C - {x}
      4. assume P(A)
      5. moreover from <C ∈ FinPow(X)> have A ∈ FinPow(X)
         using fin_rem_point_fin by simp
      6. moreover from A1 <C ∈ FinPow(X)> <C ≠ 0> have
         x ∈ C and x ∈ X - A and ∀a∈A. ⟨a,x⟩ ∈ r
         using linord_max_props by auto
      7. moreover note A3
      8. ultimately have P(A ∪ {x}) by auto
      9. moreover from <x ∈ C> have A ∪ {x} = C
         by auto

173
ultimately have \( P(C) \) by simp

thus thesis by simp qed

moreover note A4
ultimately show \( P(B) \) by (rule fin_ord_induction) qed

The only order automorphism of a linearly ordered finite set is the identity.

**Theorem fin_ord_auto_id:** assumes A1: IsLinOrder(X,r) and A2: B ∈ FinPow(X) and A3: B ≠ 0 shows ord_iso(B,r,B,r) = {id(B)}

**Proof** -

note A1
moreover
{ fix A assume A ∈ FinPow(X) A ≠ 0
  let M = Maximum(r,A)
  let A₀ = A - {M}
  assume A = {M} ∨ ord_iso(A₀,r,A₀,r) = {id(A₀)}
  moreover
  { assume A = {M}
    have ord_iso({M},r,{M},r) = {id({M})}
    using id_ord_auto_singleton by simp
    with A1 {A ∈ FinPow(X)} {A ≠ 0}
    have ord_iso(A,r,A,r) = {id(A)}
    proof
      show {id(A)} ⊆ ord_iso(A,r,A,r)
        using id_ord_iso by simp
    { fix f assume f ∈ ord_iso(A,r,A,r)
      with A1 {A ∈ FinPow(X)} {A ≠ 0} have restrict(f,A₀) ∈ ord_iso(A₀,r,A-{f(M)},r)
        using IsLinOrder_def fin_has_max ord_iso_rem_max by auto
      with A1 {A ∈ FinPow(X)} {A ≠ 0} {f ∈ ord_iso(A,r,A,r)}
      {ord_iso(A₀,r,A₀,r) = {id(A₀)}
        have restrict(f,A₀) = id(A₀)
        using IsLinOrder_def fin_has_max max_auto_fixpoint by auto
        moreover from A1 {f ∈ ord_iso(A,r,A,r)}
        {A ∈ FinPow(X)} {A ≠ 0} have
          f : A → A and M ∈ A and f(M) = M
          using ord_iso_def bij_is_fun IsLinOrder_def
          fin_has_max Order_ZF_4_L3 max_auto_fixpoint by auto
        ultimately have f = id(A) using id_fixpoint_rem by simp
      } then show ord_iso(A,r,A,r) ⊆ {id(A)}
    }
Every two finite linearly ordered sets are order isomorphic. The statement is formulated to make the proof by induction on the size of the set easier, see fin_ord_iso_ex for an alternative formulation.
Every two finite linearly ordered sets are order isomorphic.

lemma fin_ord_iso_ex:
assumes A1: IsLinOrder(X,r) IsLinOrder(Y,R) and
A2: A ∈ FinPow(X) B ∈ FinPow(Y) and A3: B ≈ A
shows ord_iso(A,r,B,R) ≠ 0
proof
  from A2 obtain n where n ∈ nat and A ≈ n
    using finpow_decomp by auto
  from A3 ⟨A ≈ n⟩ have B ≈ n by (rule eqpoll_trans)
  with A1 A2 ⟨A ≈ n⟩ ⟹ n ∈ nat show ord_iso(A,r,B,R) ≠ 0
    using fin_order_iso by simp
qed

Existence and uniqueness of order isomorphism for two linearly ordered sets with the same number of elements.

theorem fin_ord_iso_ex_uniq:
assumes A1: IsLinOrder(X,r) IsLinOrder(Y,R) and  
A2: A ∈ FinPow(X) B ∈ FinPow(Y) and A3: B ≈ A  
shows ∃!f. f ∈ ord_iso(A,r,B,R)  

proof  
from assms show ∃f. f ∈ ord_iso(A,r,B,R)  
  using fin_ord_iso_ex by blast  
fix f g  
assume A4: f ∈ ord_iso(A,r,B,R) g ∈ ord_iso(A,r,B,R)  
then have converse(g) ∈ ord_iso(B,R,A,r)  
  using ord_iso_sym by simp  
with ⟨f ∈ ord_iso(A,r,B,R)⟩ have  
  I: converse(g) O f ∈ ord_iso(A,r,A,r)  
  by (rule ord_iso_trans)  
{ assume A ≠ 0  
  with A1 A2 I have converse(g) O f = id(A)  
    using fin_ord_auto_id by auto  
  with A4 have f = g  
    using ord_iso_def comp_inv_id_eq_bij by auto }  
moreover  
{ assume A = 0  
  then have A ≈ 0 using eqpoll_0_iff  
    by simp  
  with A3 have B ≈ 0 by (rule eqpoll_trans)  
  with A4 ⟨A = 0⟩ have  
    f ∈ ord_iso(0,r,0,R) and g ∈ ord_iso(0,r,0,R)  
    using eqpoll_0_iff by auto  
  then have f = g by (rule empty_ord_iso_uniq) }  
ultimately show f = g  
  using ord_iso_def comp_inv_id_eq_bij by auto  
qed  

end  

17 Equivalence relations  

theory EquivClass1 imports ZF.EquivClass func_ZF ZF1  
begin  
In this theory file we extend the work on equivalence relations done in the  
standard Isabelle's EquivClass theory. That development is very good and  
all, but we really would prefer an approach contained within the a standard  
ZF set theory, without extensions specific to Isabelle. That is why this  
theory is written.
17.1 Congruent functions and projections on the quotient

Suppose we have a set $X$ with a relation $r \subseteq X \times X$ and a function $f : X \to X$. The function $f$ can be compatible (congruent) with $r$ in the sense that if two elements $x$, $y$ are related then the values $f(x)$, $f(y)$ are also related. This is especially useful if $r$ is an equivalence relation as it allows to "project" the function to the quotient space $X/r$ (the set of equivalence classes of $r$) and create a new function $F$ that satisfies the formula $F([x]_r) = [f(x)]_r$. When $f$ is congruent with respect to $r$ such definition of the value of $F$ on the equivalence class $[x]_r$ does not depend on which $x$ we choose to represent the class. In this section we also consider binary operations that are congruent with respect to a relation. These are important in algebra - the congruency condition allows to project the operation to obtain the operation on the quotient space.

First we define the notion of function that maps equivalent elements to equivalent values. We use similar names as in the Isabelle’s standard EquivClass theory to indicate the conceptual correspondence of the notions.

definition Congruent $(r,f)$ ≡ $(\forall x y. \langle x,y \rangle \in r \implies \langle f(x),f(y) \rangle \in r)$

Now we will define the projection of a function onto the quotient space. In standard math the equivalence class of $x$ with respect to relation $r$ is usually denoted $[x]_r$. Here we reuse notation $r\{x\}$ instead. This means the image of the set $\{x\}$ with respect to the relation, which, for equivalence relations is exactly its equivalence class if you think about it.

definition ProjFun $(A,r,f)$ ≡ $\{ \langle c, \bigcup x \in c. r\{f(x)\} \rangle. c \in (A//r) \}$

Elements of equivalence classes belong to the set.

lemma EquivClass_1_L1:
assumes A1: equiv $(A,r)$ and A2: $C \in A//r$ and A3: $x \in C$
shows $x \in A$
proof -
  from A2 have $C \subseteq \bigcup (A//r)$ by auto
  with A1 A3 show $x \in A$
    using Union_quotient by auto
qed

The image of a subset of $X$ under projection is a subset of $A/r$.

lemma EquivClass_1_L1A:
assumes $A \subseteq X$ shows $\{r(x). x \in A\} \subseteq X/r$
using assms quotientI by auto
If an element belongs to an equivalence class, then its image under relation is this equivalence class.

**lemma** EquivClass_1_L2:

- assumes A1: equiv(A,r)  C ∈ A//r and A2: x∈C
- shows r{x} = C

**proof** -

- from A1 A2 have x ∈ r{x} using EquivClass_1_L1 equiv_class_self by simp
- with A2 have I: r{x}∩C ≠ 0 by auto
- from A1 A2 have r{x} ∈ A//r using EquivClass_1_L1 quotientI by simp
- with A1 I show thesis using quotient_disj by blast

**qed**

Elements that belong to the same equivalence class are equivalent.

**lemma** EquivClass_1_L2A:

- assumes equiv(A,r) C ∈ A//r  x∈C  y∈C
- shows ⟨x,y⟩ ∈ r using assms EquivClass_1_L2 EquivClass_1_L1 equiv_class_eq_iff by simp

Elements that have the same image under an equivalence relation are equivalent. This is the same as eq_equiv_class from standard Isabelle/ZF’s EqvClass theory, just copied here to be easier to find.

**lemma** same_image-equiv:

- assumes equiv(A,r) y∈A r{x} = r{y}
- shows ⟨x,y⟩ ∈ r using assms eq_equiv_class by simp

Every x is in the class of y, then they are equivalent.

**lemma** EquivClass_1_L2B:

- assumes A1: equiv(A,r) and A2: y∈A and A3: x ∈ r{y}
- shows ⟨x,y⟩ ∈ r

**proof** -

- from A2 have r{y} ∈ A//r using quotientI by simp
- with A1 A3 show thesis using EquivClass_1_L1 equiv_class_self equiv_class_nondisjoint by blast

**qed**

If a function is congruent then the equivalence classes of the values that come from the arguments from the same class are the same.

**lemma** EquivClass_1_L3:

- assumes A1: equiv(A,r) and A2: Congruent(r,f) and A3: C ∈ A//r  x∈C  y∈C
- shows r{f(x)} = r{f(y)}

**proof** -
The values of congruent functions are in the space.

**Lemma EquivClass_1_L4:**
- Assumes $A1: \text{equiv}(A, r)$ and $A2: C \in A/r \ x \in C$
- Assumes $A3: \text{Congruent}(r, f)$
- Shows $f(x) \in A$

**Proof** -
- From $A1$ $A2$ have $x \in A$
- Using EquivClass_1_L1 by simp
- With $A1$ have $\langle x, x \rangle \in r$
- Using equiv_def refl_def by simp
- With $A3$ have $\langle f(x), f(x) \rangle \in r$
- Using Congruent_def by simp
- With $A1$ show thesis using equiv_type by auto

**Qed**

Equivalence classes are not empty.

**Lemma EquivClass_1_L5:**
- Assumes $A1: \text{refl}(A, r)$ and $A2: C \in A//r$
- Shows $C \neq 0$

**Proof** -
- From $A2$ obtain $x$ where $I: C = r\{x\}$ and $x \in A$
- Using quotient_def by auto
- From $A1$ $\langle x \in A \rangle$ have $x \in r(x)$ using refl_def by auto
- With $I$ show thesis by auto

**Qed**

To avoid using an axiom of choice, we define the projection using the expression $\bigcup_{x \in C} r(\{f(x)\})$. The next lemma shows that for congruent function this is in the quotient space $A/r$.

**Lemma EquivClass_1_L6:**
- Assumes $A1: \text{equiv}(A, r)$ and $A2: \text{Congruent}(r, f)$
- Assumes $A3: C \in A//r$
- Shows $(\bigcup_{x \in C} r\{f(x)\}) \in A//r$

**Proof** -
- From $A1$ have refl(A, r) unfolding equiv_def by simp
- With $A3$ have $C \neq 0$ using EquivClass_1_L5 by simp
- Moreover from $A2$ $A3$ $A1$ have $\forall x \in C. \ r\{f(x)\} \in A//r$
- Using EquivClass_1_L4 quotientI by auto
- Moreover from $A1$ $A2$ $A3$ have $\forall x \ y. \ x \in C \land y \in C \longrightarrow r\{f(x)\} = r\{f(y)\}$
- Using EquivClass_1_L3 by blast

180
ultimately show thesis by (rule ZF1_1_L2)

qed

Congruent functions can be projected.

lemma EquivClass_1_T0:
  assumes equiv(A,r) Congruent(r,f)
  shows ProjFun(A,r,f) : A//r → A//r
  using assms EquivClass_1_L6 ProjFun_def ZF_fun_from_total
  by simp

We now define congruent functions of two variables (binary funtions). The
predicate Congruent2 corresponds to congruent2 in Isabelle’s standard EquivClass
theory, but uses ZF-functions rather than meta-functions.

definition Congruent2(r,f) ≡
  (∀x₁ x₂ y₁ y₂. ⟨x₁,x₂⟩ ∈ r ∧ ⟨y₁,y₂⟩ ∈ r → ⟨f⟨x₁,y₁⟩, f⟨x₂,y₂⟩⟩ ∈ r)

Next we define the notion of projecting a binary operation to the quotient
space. This is a very important concept that allows to define quotient
groups, among other things.

definition ProjFun2(A,r,f) ≡
  {⟨p,⋃z ∈ fst(p)×snd(p). r{f(z)}⟩. p ∈ (A//r)×(A//r) }

The following lemma is a two-variables equivalent of EquivClass_1_L3.

lemma EquivClass_1_L7:
  assumes A1: equiv(A,r) and A2: Congruent2(r,f)
  and A3: C₁ ∈ A//r C₂ ∈ A//r
  and A4: z₁ ∈ C₁×C₂ z₂ ∈ C₁×C₂
  shows r{f(z₁)} = r{f(z₂)}
  proof -
  from A4 obtain x₁ y₁ x₂ y₂ where
  x₁∈C₁ and y₁∈C₂ and z₁ = ⟨x₁,y₁⟩ and
  x₂∈C₁ and y₂∈C₂ and z₂ = ⟨x₂,y₂⟩
  by auto
  with A1 A3 have ⟨x₁,x₂⟩ ∈ r and ⟨y₁,y₂⟩ ∈ r
  using EquivClass_1_L2A by auto
  with A2 have ⟨f⟨x₁,y₁⟩,f⟨x₂,y₂⟩⟩ ∈ r
  using Congruent2_def by simp
  with A1 ⟨z₁ = ⟨x₁,y₁⟩⟩ ⟨z₂ = ⟨x₂,y₂⟩⟩ show thesis
  using equiv_class_eq by simp
  qed

The values of congruent functions of two variables are in the space.

lemma EquivClass_1_L8:
  assumes A1: equiv(A,r) and A2: C₁ ∈ A//r and A3: C₂ ∈ A//r
  and A4: z ∈ C₁×C₂ and A5: Congruent2(r,f)
shows $f(z) \in A$

proof -
  from A4 obtain $x \ y$ where $x \in C_1$ and $y \in C_2$ and $z = \langle x,y \rangle$
  by auto
  with A1 A2 A3 have $x \in A$ and $y \in A$
    using EquivClass_1_L1 by auto
  with A1 A4 have $(x,x) \in r$ and $(y,y) \in r$
    using equiv_def refl_def by auto
  with A5 have $(f(x,y), f(x,y)) \in r$
    using Congruent2_def by simp
  with A1 $\langle z = \langle x,y \rangle \rangle$ show thesis using equiv_type by auto
qed

The values of congruent functions are in the space. Note that although this lemma is intended to be used with functions, we don’t need to assume that $f$ is a function.

lemma EquivClass_1_L8A:
  assumes A1: equiv(A,r) and A2: $x \in A$ $y \in A$
  and A3: Congruent2(r,f)
  shows $f\langle x,y \rangle \in A$

proof -
  from A3 have $r\{x\} \in A//r$ $r\{y\} \in A//r$
    $(x,y) \in r\{x\} \times r\{y\}$
    using equiv_class_self quotientI by auto
  with A1 A2 show thesis using EquivClass_1_L8 by simp
qed

The following lemma is a two-variables equivalent of EquivClass_1_L6.

lemma EquivClass_1_L9:
  assumes A1: equiv(A,r) and A2: Congruent2(r,f)
  and A3: $p \in (A//r) \times (A//r)$
  shows $(\bigcup z \in fst(p) \times snd(p). r\{f(z)\}) \in A//r$

proof -
  from A3 have $fst(p) \in A//r$ and $snd(p) \in A//r$
    by auto
  with A1 A2 have $I: \forall z \in fst(p) \times snd(p). f(z) \in A$
    using EquivClass_1_L8 by simp
  from A3 A1 have $fst(p) \times snd(p) \neq 0$
    using equiv_def EquivClass_1_L5 Sigma_empty_iff by auto
  moreover from A1 I have
    $\forall z \in fst(p) \times snd(p). r\{f(z)\} \in A//r$
    using quotientI by simp
  moreover from A1 A2 $\langle fst(p) \in A//r \rangle$ $\langle snd(p) \in A//r \rangle$ have
    $\forall z_1 \ z_2. \ z_1 \in fst(p) \times snd(p) \land z_2 \in fst(p) \times snd(p) \rightarrow$
    $r\{f(z_1)\} = r\{f(z_2)\}$
    using EquivClass_1_L7 by blast
  ultimately show thesis by (rule ZF1_1_L2)
Congruent functions of two variables can be projected.

**Theorem** EquivClass_1_T1:
- assumes $\text{equiv}(A,r)$, $\text{Congruent2}(r,f)$
- shows $\text{ProjFun2}(A,r,f) : (A//r) \times (A//r) \rightarrow A//r$
- using assms EquivClass_1_L9 ProjFun2_def ZF_fun_from_total
  by simp

The projection diagram commutes. I wish I knew how to draw this diagram in LaTeX.

**Lemma** EquivClass_1_L10:
- assumes $A1: \text{equiv}(A,r)$ and $A2: \text{Congruent2}(r,f)$
- and $A3: x \in A$ $y \in A$
- shows $\text{ProjFun2}(A,r,f)\langle x, y \rangle = r\{f(x,y)\}$
- proof -
  - from $A3$ $A1$ have $r\{x\} \times r\{y\} \neq 0$
    - using quotientI equiv_def EquivClass_1_L5 Sigma_empty_iff
    - by auto
  - moreover have $\forall z \in r\{x\} \times r\{y\}$. $r\{f(z)\} = r\{f(x,y)\}$
    - proof
      - fix $z$ assume $A4: z \in r\{x\} \times r\{y\}$
        - from $A1$ $A3$ have
          $r\{x\} \in A//r$ $r\{y\} \in A//r$
          $\langle x, y \rangle \in r\{x\} \times r\{y\}$
          - using quotientI equiv_class_self by auto
        - with $A1$ $A2$ $A4$ show $r\{f(z)\} = r\{f(x,y)\}$
          - using EquivClass_1_L7 by blast
      qed
    - ultimately have $(\bigcup z \in r\{x\} \times r\{y\}. r\{f(z)\}) = r\{f(x,y)\}$
      - by (rule ZF1_1_L1)
  - moreover have $\text{ProjFun2}(A,r,f)\langle x, y \rangle = (\bigcup z \in r\{x\} \times r\{y\}. r\{f(z)\})$
    - proof -
      - from assms have $\text{ProjFun2}(A,r,f) : (A//r) \times (A//r) \rightarrow A//r$
        $\langle r\{x\}, r\{y\} \rangle \in (A//r) \times (A//r)$
        - using EquivClass_1_T1 quotientI by auto
      - then show thesis using ProjFun2_def ZF_fun_from_tot_val
        by auto
    qed
  - ultimately show thesis by simp
qed
17.2 Projecting commutative, associative and distributive operations.

In this section we show that if the operations are congruent with respect to an equivalence relation then the projection to the quotient space preserves commutativity, associativity and distributivity.

The projection of commutative operation is commutative.

**Lemma** `EquivClass_2_L1` assumes
- `equiv(A, r)` and `Congruent2(r, f)`
- `f` is commutative on `A`
- `c1 ∈ A//r` `c2 ∈ A//r`
shows `ProjFun2(A, r, f)⟨c1, c2⟩ = ProjFun2(A, r, f)⟨c2, c1⟩`

**Proof**
- from `A4` obtain `x` `y` where `D1`:
  - `c1 = r{x}` `c2 = r{y}`
  - `x ∈ A` `y ∈ A`
  - using `quotient_def` by `auto`
- with `A1` `A2` have `ProjFun2(A, r, f)⟨c1, c2⟩ = r{f⟨x, y⟩}`
  - using `EquivClass_1_L10` by `simp`
- also from `A3` `D1` have
  - `r{f⟨x, y⟩} = r{f⟨y, x⟩}`
  - using `IsCommutative_def` by `simp`
- also from `A1` `A2` `D1` have
  - `r{f⟨y, x⟩} = ProjFun2(A, r, f)⟨c2, c1⟩`
  - using `EquivClass_1_L10` by `simp`
- finally show thesis by `simp`

**Qed**

The projection of commutative operation is commutative.

**Theorem** `EquivClass_2_T1`:
- assumes `equiv(A, r)` and `Congruent2(r, f)`
- `f` is commutative on `A`
- shows `ProjFun2(A, r, f)` is commutative on `A//r`
- using `assms` `IsCommutative_def` `EquivClass_2_L1` by `simp`

The projection of an associative operation is associative.

**Lemma** `EquivClass_2_L2`:
- assumes `equiv(A, r)` and `Congruent2(r, f)`
- `f` is associative on `A`
- `c1 ∈ A//r` `c2 ∈ A//r` `c3 ∈ A//r`
- `g = ProjFun2(A, r, f)`
- shows `g⟨g⟨c1, c2⟩, c3⟩ = g⟨c1, g⟨c2, c3⟩⟩`

**Proof**
- from `A4` obtain `x` `y` `z` where `D1`:
  - `c1 = r{x}` `c2 = r{y}` `c3 = r{z}`
  - `x ∈ A` `y ∈ A` `z ∈ A`
  - using `quotient_def` by `auto`
The projection of an associative operation is associative on the quotient.

**Theorem EquivClass_2_T2:**

assumes A1: equiv(A,r) and A2: Congruent2(r,f)
and A3: f (is associative on) A
shows ProjFun2(A,r,f) (is associative on) A//r

**Proof -**

let g = ProjFun2(A,r,f)
from A1 A2 have
g ∈ (A//r) × (A//r) → A//r
using EquivClass_1_T1 by simp
moreover from A1 A2 A3 have
∀ c1 ∈ A//r. ∀ c2 ∈ A//r. ∀ c3 ∈ A//r.
g(g(c1,c2),c3) = g(c1,g(c2,c3))
using EquivClass_2_L2 by simp
ultimately show thesis
using IsAssociative_def by simp
qed

The essential condition to show that distributivity is preserved by projections to quotient spaces, provided both operations are congruent with respect to the equivalence relation.

**Lemma EquivClass_2_L3:**

assumes A1: IsDistributive(X,A,M)
and A2: equiv(X,r)
and A3: Congruent2(r,A) Congruent2(r,M)
and A4: a ∈ X//r b ∈ X//r c ∈ X//r
and A5: A_p = ProjFun2(X,r,A) M_p = ProjFun2(X,r,M)
suches M_p(a,A_p(b,c)) = A_p(M_p(a,b),M_p(a,c)) ∧
M_p(A_p(b,c),a) = A_p(M_p(b,a),M_p(c,a))

**Proof -**

from A4 obtain x y z where x∈X y∈X z∈X
a = r(x) b = r(y) c = r(z)
using quotient_def by auto
with A1 A2 A3 A5 show
M_p(a,A_p(b,c)) = A_p(M_p(a,b),M_p(a,c)) and
\[ M_p \langle A_p(b,c), a \rangle = A_p(M_p(b,a), M_p(c,a)) \]

using EquivClass_1_L8A EquivClass_1_L10 IsDistributive_def
by auto

qed

Distributivity is preserved by projections to quotient spaces, provided both operations are congruent with respect to the equivalence relation.

lemma EquivClass_2_L4: assumes A1: IsDistributive(X,A,M)
and A2: equiv(X,r)
and A3: Congruent2(r,A) Congruent2(r,M)
shows IsDistributive(X//r,ProjFun2(X,r,A),ProjFun2(X,r,M))
proof-
let A_p = ProjFun2(X,r,A)
let M_p = ProjFun2(X,r,M)
from A1 A2 A3 have
\[ \forall a \in X//r. \forall b \in X//r. \forall c \in X//r. \]
\[ M_p(a, A_p(b,c)) = A_p(M_p(a,b), M_p(c,a)) \]
\[ M_p(A_p(b,c), a) = A_p(M_p(b,a), M_p(c,a)) \]
using EquivClass_2_L3 by simp
then show thesis using IsDistributive_def by simp
qed

17.3 Saturated sets

In this section we consider sets that are saturated with respect to an equivalence relation. A set \( A \) is saturated with respect to a relation \( r \) if \( A = r^{-1}(r(A)) \). For equivalence relations saturated sets are unions of equivalence classes. This makes them useful as a tool to define subsets of the quotient space using properties of representants. Namely, we often define a set \( B \subseteq X/r \) by saying that \([x]_r \in B \) iff \( x \in A \). If \( A \) is a saturated set, this definition is consistent in the sense that it does not depend on the choice of \( x \) to represent \([x]_r \).

The following defines the notion of a saturated set. Recall that in Isabelle \( r^{-1}(A) \) is the inverse image of \( A \) with respect to relation \( r \). This definition is not specific to equivalence relations.

definition IsSaturated(r,A) ≡ A = r^{-1}(r(A))

For equivalence relations a set is saturated iff it is an image of itself.

lemma EquivClass_3_L1: assumes A1: equiv(X,r)
shows IsSaturated(r,A) ←→ A = r(A)
proof
assume IsSaturated(r,A)
then have A = (converse(r) 0 r)(A)
  using IsSaturated_def vimage_def image_comp
  by simp
qed
also from $A_1$ have ... = $r(A)$
  using equiv_comp_eq by simp
finally show $A = r(A)$ by simp

next assume $A = r(A)$
  with $A_1$ have $A = (\text{converse}(r) \circ r)(A)$
  using equiv_comp_eq by simp
also have ... = $r^{-1}(r(A))$
  using vimage_def image_comp by simp
finally have $A = r^{-1}(r(A))$ by simp
then show IsSaturated($r, A$) using IsSaturated_def
  by simp
qed

For equivalence relations sets are contained in their images.

lemma EquivClass_3_L2: assumes $A_1$: equiv($X, r$) and $A_2$: $A \subseteq X$
  shows $A \subseteq r(A)$
proof
  fix $a$ assume $a \in A$
  with $A_1$ $A_2$ have $a \in r\{a\}$
    using equiv_class_self by auto
  with $\langle a \in A \rangle$ show $a \in r(A)$ by auto
qed

The next lemma shows that if "$\sim$" is an equivalence relation and a set $A$ is such that $a \in A$ and $a \sim b$ implies $b \in A$, then $A$ is saturated with respect to the relation.

lemma EquivClass_3_L3: assumes $A_1$: equiv($X, r$) and $A_2$: $r \subseteq X \times X$ and $A_3$: $A \subseteq X$
and $A_4$: $\forall x \in A. \forall y \in X. \langle x, y \rangle \in r \rightarrow y \in A$
  shows IsSaturated($r, A$)
proof
  from $A_2$ $A_4$ have $r(A) \subseteq A$
    using image_iff by blast
moreover from $A_1$ $A_3$ have $A \subseteq r(A)$
  using EquivClass_3_L2 by simp
ultimately have $A = r(A)$ by auto
with $A_1$ show IsSaturated($r, A$) using EquivClass_3_L1
  by simp
qed

If $A \subseteq X$ and $A$ is saturated and $x \sim y$, then $x \in A$ iff $y \in A$. Here we show only one direction.

lemma EquivClass_3_L4: assumes $A_1$: equiv($X, r$) and $A_2$: IsSaturated($r, A$) and $A_3$: $A \subseteq X$
and $A_4$: $\langle x, y \rangle \in r$
and $A_5$: $x \in X \quad y \in A$
  shows $x \in A$
proof

from A1 A5 have x ∈ r{x}
  using equiv_class_self by simp
with A1 A3 A4 A5 have x ∈ r(A)
  using equiv_class_eq equiv_class_self
by auto
with A1 A2 show x ∈ A
  using EquivClass_3_L1 by simp
qed

If \( A \subseteq X \) and \( A \) is saturated and \( x \sim y \), then \( x \in A \) iff \( y \in A \).

lemma EquivClass_3_L5: assumes A1: equiv(X,r)
  and A2: IsSaturated(r,A) and A3: A ⊆ X
  and A4: x ∈ X \( y \in X \)
  and A5: \( \langle x,y \rangle \in r \)
shows x ∈ A ←→ y ∈ A
proof
  assume y ∈ A
  with assms show x ∈ A using EquivClass_3_L4
  by simp
next assume x ∈ A
  from A1 A5 have \( \langle y,x \rangle \in r \)
    using equiv_is_sym by blast
  with A1 A2 A3 A4 \( \langle x, y \rangle \in r \)
    shows x ∈ A ←→ y ∈ A
qed

If \( A \) is saturated then \( x \in A \) iff its class is in the projection of \( A \).

lemma EquivClass_3_L6: assumes A1: equiv(X,r)
  and A2: IsSaturated(r,A) and A3: A ⊆ X
  and A4: x ∈ X
  and A5: B = {r{x}. x ∈ A}
shows x ∈ A ←→ r{x} ∈ B
proof
  assume x ∈ A
  with A5 show r{x} ∈ B by auto
next assume r{x} ∈ B
  with A5 obtain y where y ∈ A and r{x} = r{y}
    by auto
  with A1 A3 have \( \langle x,y \rangle \in r \)
    using eq_equiv_class by auto
  with A1 A2 A3 A4 \( \langle y \in A \rangle \) show x ∈ A
    using EquivClass_3_L4 by simp
qed

A technical lemma involving a projection of a saturated set and a logical expression with exclusive or. Note that we don’t really care what \( \text{Xor} \) is here, this is true for any predicate.

lemma EquivClass_3_L7: assumes equiv(X,r)
  and IsSaturated(r,A) and A ⊆ X
and \( x \in X \) \( y \in X \)
and \( B = \{r(x). x \in A\} \)
and \((x \in A) \text{ Xor } (y \in A)\)
shows \((r(x) \in B) \text{ Xor } (r(y) \in B)\)
using assms EquivClass_3_L6 by simp

end

18 Finite sequences

theory FiniteSeq_ZF imports Nat_ZF_IML func1

begin

This theory treats finite sequences (i.e. maps \( n \to X \), where \( n = \{0, 1, \ldots, n-1\} \) is a natural number) as lists. It defines and proves the properties of basic operations on lists: concatenation, appending and element etc.

18.1 Lists as finite sequences

A natural way of representing (finite) lists in set theory is through (finite) sequences. In such view a list of elements of a set \( X \) is a function that maps the set \( \{0, 1, \ldots, n-1\} \) into \( X \). Since natural numbers in set theory are defined so that \( n = \{0, 1, \ldots, n-1\} \), a list of length \( n \) can be understood as an element of the function space \( n \to X \).

We define the set of lists with values in set \( X \) as \( \text{Lists}(X) \).

**definition**

\[ \text{Lists}(X) \equiv \bigcup n \in \text{nat.} \,(n \to X) \]

The set of nonempty \( X \)-value listst will be called \( \text{NELists}(X) \).

**definition**

\[ \text{NELists}(X) \equiv \bigcup n \in \text{nat.} \,(\text{succ}(n) \to X) \]

We first define the shift that moves the second sequence to the domain \( \{n, \ldots, n + k - 1\} \), where \( n, k \) are the lengths of the first and the second sequence, resp. To understand the notation in the definitions below recall that in Isabelle/ZF \( \text{pred}(n) \) is the previous natural number and denotes the difference between natural numbers \( n \) and \( k \).

**definition**

\[ \text{ShiftedSeq}(b,n) \equiv \{(j, b(j \#- n)). \ j \in \text{NatInterval}(n,\text{domain}(b))\} \]

We define concatenation of two sequences as the union of the first sequence with the shifted second sequence. The result of concatenating lists \( a \) and \( b \) is called \( \text{Concat}(a,b) \).

**definition**
\text{Concat}(a, b) \equiv a \cup \text{ShiftedSeq}(b, \text{domain}(a))

For a finite sequence we define the sequence of all elements except the first one. This corresponds to the "tail" function in Haskell. We call it \text{Tail} here as well.

\textbf{definition}
\text{Tail}(a) \equiv \{ \langle k, a(\text{succ}(k)) \rangle. k \in \text{pred}(\text{domain}(a)) \} 

A dual notion to \text{Tail} is the list of all elements of a list except the last one. Borrowing the terminology from Haskell again, we will call this \text{Init}.

\textbf{definition}
\text{Init}(a) \equiv \text{restrict}(a, \text{pred}(\text{domain}(a))) 

Another obvious operation we can talk about is appending an element at the end of a sequence. This is called \text{Append}.

\textbf{definition}
\text{Append}(a, x) \equiv a \cup \{ (\text{domain}(a), x) \} 

If lists are modeled as finite sequences (i.e. functions on natural intervals \{0, 1, ..., n − 1\} = n) it is easy to get the first element of a list as the value of the sequence at 0. The last element is the value at \(n - 1\). To hide this behind a familiar name we define the \text{Last} element of a list.

\textbf{definition}
\text{Last}(a) \equiv a(\text{pred}(\text{domain}(a)))

A formula for tail of a finite list.

\textbf{lemma tail_as_set: assumes} n \in \text{nat} \text{ and } a: n \#+ 1 \rightarrow X \text{ shows } \text{Tail}(a) = \{ \langle k, a(k \#+ 1) \rangle. k \in n \} \text{ using assms func1_1_L1 elem_nat_is_nat(2) succ_add_one(1) unfolding Tail_def by simp}

Formula for the tail of a list defined by an expression:

\textbf{lemma tail_formula: assumes} n \in \text{nat} \text{ and } \forall k \in n \#+ 1. q(k) \in X \text{ shows } \text{Tail}((\{ \langle k, q(k) \rangle. k \in n \#+ 1 \})) = \{ \langle k, q(k \#+ 1) \rangle. k \in n \} 

\textbf{proof -}
\{ \text{ fix } k \text{ assume } k \in n \\
\text{ with assms(1) have } k \#+ 1 \in n \#+ 1 \\
\text{ using succ_ineq1 elem_nat_is_nat(2) succ_add_one(1) by simp} \\\n\text{ then have } a(k \#+ 1) = q(k \#+ 1) \}

190
Codomain of a nonempty list is nonempty.

**Lemma** \texttt{nelist_vals_nonempty}: assumes \texttt{a:succ(n)→Y} shows \texttt{Y≠0} using \texttt{assms codomain_nonempty} by simp

Shifted sequence is a function on a the interval of natural numbers.

**Lemma** \texttt{shifted_seq_props}:
assumes \texttt{A1: n ∈ nat k ∈ nat and A2: b:k→X} shows \texttt{ShiftedSeq(b,n): NatInterval(n,k)→X}
\texttt{∀ i ∈ NatInterval(n,k). ShiftedSeq(b,n)(i) = b(i #- n)}
\texttt{∀ j∈k. ShiftedSeq(b,n)(n #+ j) = b(j)}

**Proof**
- let \texttt{I = NatInterval(n,domain(b))}
  from \texttt{A2} have \texttt{Fact: I = NatInterval(n,k)} using \texttt{func1_1_L1} by simp
  with \texttt{A1 A2} have \texttt{∀ j∈I. b(j #- n) ∈ X}
    using \texttt{inter_diff_in_len apply_funtype} by simp
  then have \texttt{⟨⟨j, b(j #- n)⟩. j ∈ I⟩} : \texttt{I} → \texttt{X} by \texttt{(rule ZF_fun_from_total)}
  with \texttt{Fact} show \texttt{thesis_1: ShiftedSeq(b,n): NatInterval(n,k)→X}
    using \texttt{ShiftedSeq_def} by simp

  \{ fix \texttt{i}
    from \texttt{Fact} \texttt{thesis_1} have \texttt{ShiftedSeq(b,n): I → X} by simp
    moreover assume \texttt{i ∈ NatInterval(n,k)}
    with \texttt{Fact} have \texttt{i ∈ I} by simp
    moreover from \texttt{Fact} have
      \texttt{ShiftedSeq(b,n) = ⟨⟨i, b(i #- n)⟩. i ∈ I⟩}
      using \texttt{ShiftedSeq_def} by simp
    ultimately have \texttt{ShiftedSeq(b,n)(i) = b(i #- n)}
      by \texttt{(rule ZF_fun_from_tot_val)}
  \}
  then show \texttt{thesis1:}
    \texttt{∀ i ∈ NatInterval(n,k). ShiftedSeq(b,n)(i) = b(i #- n)}
    by simp

  \{ fix \texttt{j}
    let \texttt{i = n #+ j}
    assume \texttt{A3: j∈k}
    with \texttt{A1} have \texttt{j ∈ nat} using \texttt{elem_nat_is_nat} by blast
    then have \texttt{i #- n = j} using \texttt{diff_add_inverse} by simp
    with \texttt{A3} \texttt{thesis1} have \texttt{ShiftedSeq(b,n)(i) = b(j)}
      using \texttt{NatInterval_def} by auto
  \}
  then show \texttt{∀ j∈k. ShiftedSeq(b,n)(n #+ j) = b(j)}
    by simp

qed
Basis properties of the concatenation of two finite sequences.

**Theorem** \( \text{concat\_props} \):
- **Assumes** \( A1: n \in \text{nat} \) \( k \in \text{nat} \) and \( A2: a:n \rightarrow X \) \( b:k \rightarrow X \)
- **Shows**
  \[
  \forall i \in n. \text{Concat}(a,b)(i) = a(i) \\
  \forall i \in \text{NatInterval}(n,k). \text{Concat}(a,b)(i) = b(i - n) \\
  \forall j \in k. \text{Concat}(a,b)(n + j) = b(j)
  \]
- **Proof**

Properties of concatenating three lists.

**Lemma** \( \text{concat\_concat\_list} \):
- **Assumes** \( A1: n \in \text{nat} \) \( k \in \text{nat} \) \( m \in \text{nat} \) and
  \( A2: a:n \rightarrow X \) \( b:k \rightarrow X \) \( c:m \rightarrow X \) and
  \( A3: d = \text{Concat}(\text{Concat}(a,b),c) \)
- **Shows**
\[ d : n \#+ k \#+ m \rightarrow X \]
\[ \forall j \in n. d(j) = a(j) \]
\[ \forall j \in k. d(n \#+ j) = b(j) \]
\[ \forall j \in m. d(n \#+ k \#+ j) = c(j) \]

**proof** -

from A1 A2 have I:
\[ n \#+ k \in \text{nat} \quad m \in \text{nat} \]
Concat(a,b): n \#+ k \rightarrow X \quad c:m \rightarrow X
using concat_props by auto

with A3 have d: n \#+ k \#+ m \rightarrow X
using concat_props by simp

from I have II: \[ \forall i \in n \#+ k. \]
Concat(Concat(a,b),c)(i) = Concat(a,b)(i)
by (rule concat_props)

{ fix \( j \) assume A4: \( j \in n \)
  moreover from A1 have \( n \subseteq n \#+ k \) using add_nat_le by simp
  ultimately have \( j \in n \#+ k \) by auto
  with A3 II have d(j) = Concat(a,b)(j) by simp
  with A1 A2 A4 have d(j) = a(j)
    using concat_props by simp
  }
thus \[ \forall j \in n. d(j) = a(j) \] by simp

{ fix \( j \) assume A5: \( j \in k \)
  with A1 A3 II have d(n \#+ j) = Concat(a,b)(n \#+ j)
    using add_lt_mono by simp
  also from A1 A2 A5 have \ldots = b(j)
    using concat_props by simp
  finally have d(n \#+ j) = b(j) by simp
  }
thus \[ \forall j \in k. d(n \#+ j) = b(j) \] by simp

from I have \( \forall j \in m. \]
Concat(Concat(a,b),c)(n \#+ k \#+ j) = c(j)
by (rule concat_props)

with A3 show \[ \forall j \in m. d(n \#+ k \#+ j) = c(j) \]
  by simp

qed

Properties of concatenating a list with a concatenation of two other lists.

**lemma** concat_list_concat:
assumes A1: \( n \in \text{nat} \quad k \in \text{nat} \quad m \in \text{nat} \) and
A2: \( a:n \rightarrow X \quad b:k \rightarrow X \quad c:m \rightarrow X \) and
A3: \( e = \text{Concat}(a, \text{Concat}(b,c)) \)
shows
\[ e : n \#+ k \#+ m \rightarrow X \]
\[ \forall j \in n. e(j) = a(j) \]
\[ \forall j \in k. e(n \#+ j) = b(j) \]
\[ \forall j \in m. e(n \#+ k \#+ j) = c(j) \]

**proof** -

from A1 A2 have I:
\[ n \in \text{nat} \quad k \#+ m \in \text{nat} \]
a:n \rightarrow X \quad \text{Concat}(b,c): k \#+ m \rightarrow X
using concat_props by auto
with A3 show \( e : n #+ k #+ m \to X \)
   using concat_props add_assoc by simp
from 1 have \( \forall j \in n. \text{Concat}(a, \text{Concat}(b,c))(j) = a(j) \)
   by (rule concat_props)
with A3 show \( \forall j \in n. e(j) = a(j) \) by simp
from 1 have II:
  \( \forall j \in k #+ m. \text{Concat}(a, \text{Concat}(b,c))(n #+ j) = \text{Concat}(b,c)(j) \)
  by (rule concat_props)
{ fix \( j \) assume A4: \( j \in k \)
  moreover from A1 have \( k \subseteq k #+ m \) using add_nat_le by simp
  ultimately have \( j \in k #+ m \) by auto
  with A3 II have \( e(n #+ j) = \text{Concat}(b,c)(j) \) by simp
  also from A1 A2 A4 have \( \ldots = b(j) \)
    using concat_props by simp
  finally have \( e(n #+ j) = b(j) \) by simp
} thus \( \forall j \in k. e(n #+ j) = b(j) \) by simp
{ fix \( j \) assume A5: \( j \in m \)
  with A1 II A3 have \( e(n #+ k #+ j) = \text{Concat}(b,c)(k #+ j) \)
    using add_lt_mono add_assoc by simp
  also from A1 A2 A5 have \( \ldots = c(j) \)
    using concat_props by simp
  finally have \( e(n #+ k #+ j) = c(j) \) by simp
} then show \( \forall j \in m. e(n #+ k #+ j) = c(j) \)
  by simp
qed

Concatenation is associative.

theorem concat_assoc:
  assumes A1: \( n \in \text{nat} \ k \in \text{nat} \ m \in \text{nat} \) and
  A2: \( a : n \to X \ b : k \to X \ c : m \to X \)
  shows \( \text{Concat}(\text{Concat}(a,b),c) = \text{Concat}(a, \text{Concat}(b,c)) \)
proof -
  let d = \( \text{Concat}(\text{Concat}(a,b),c) \)
  let e = \( \text{Concat}(a, \text{Concat}(b,c)) \)
  from A1 A2 have
    d : n #+k #+ m \to X and e : n #+k #+ m \to X
    using concat_concat_list concat_list_concat by auto
  moreover have \( \forall i \in n #+k #+ m. d(i) = e(i) \)
  proof -
    { fix \( i \) assume i \in n #+k #+ m
      moreover from A1 have
        n #+k #+ m = n \cup \text{NatInterval}(n,k) \cup \text{NatInterval}(n #+ k,m)
        using adjacent_intervals3 by simp
      ultimately have
        i \in n \lor i \in \text{NatInterval}(n,k) \lor i \in \text{NatInterval}(n #+ k,m)
        by simp
      moreover
        { assume i \in n
          with A1 A2 have d(i) = e(i)
        }
using `concat_concat_list concat_list_concat` by simp } 

moreover 
{ assume i ∈ NatInterval(n,k) 
then obtain j where j∈k and i = n #+ j 
using NatInterval_def by auto 
with A1 A2 have d(i) = e(i) 
using `concat_concat_list concat_list_concat` by simp } 

moreover 
{ assume i ∈ NatInterval(n #+ k,m) 
then obtain j where j∈m and i = n #+ k #+ j 
using NatInterval_def by auto 
with A1 A2 have d(i) = e(i) 
using `concat_concat_list concat_list_concat` by simp } 

ultimately have d(i) = e(i) by auto 

thus thesis by simp 
qed 

ultimately show d = e by (rule func_eq) 
qed 

Properties of Tail. 

theorem tail_props: 
assumes A1: n ∈ nat and A2: a: succ(n) → X 
shows 
Tail(a) : n → X 
∀k ∈ n. Tail(a)(k) = a(succ(k)) 

proof - 
from A1 A2 have ∀k ∈ n. a(succ(k)) ∈ X 
using succ_ineq apply_funtype by simp 
then have `{(k, a(succ(k))). k ∈ n} : n → X 
by (rule ZF_fun_from_total) 
with A2 show I: Tail(a) : n → X 
using func1_1_L1 pred_succ_eq Tail_def by simp 
moreover from A2 have Tail(a) = `{(k, a(succ(k))). k ∈ n} 
using func1_1_L1 pred_succ_eq Tail_def by simp 
ultimately show ∀k ∈ n. Tail(a)(k) = a(succ(k)) 
by (rule ZF_fun_from_tot_val0) 
qed 

Essentially the second assertion of tail_props but formulated using notation 
\( n + 1 \) instead of \( \text{succ}(n) \): 

lemma tail_props2: assumes n ∈ nat a: n #+ 1 → X k∈n 
shows Tail(a)(k) = a(k #+ 1) 
using assms succ_add_one(1) tail_props(2) elem_nat_is_nat(2) 
by simp 

A nonempty list can be decomposed into concatenation of its first element 
and the tail. 

lemma first_concat_tail: assumes n∈nat a:succ(n)→X 

195
shows \( a = \text{Concat} (\{ \langle 0, a(0) \rangle \}, \text{Tail}(a)) \)

proof -

let \( b = \text{Concat} (\{ \langle 0, a(0) \rangle \}, \text{Tail}(a)) \)

have \( b : \text{succ}(n) \rightarrow X \) and \( \forall k \in \text{succ}(n). \ a(k) = b(k) \)

proof -

from assms(1) have \( 0 \in \text{succ}(n) \) using empty_in_every_succ by simp

with assms(2) have \( a(0) \in X \) using apply_funtype by simp

then have I: \( \{ \langle 0, a(0) \rangle \} : \{ 0 \} \rightarrow X \) using pair_func_singleton by simp

from assms have \( \text{Tail}(a) : n \rightarrow X \) using tail_props(1) by simp

with assms(1) \( \{ 0 \} \in \text{nat} \) I have \( b : \{ 0 \} \#+ n \rightarrow X \)

using concat_props(1) by simp

with assms(1) show \( b : \text{succ}(n) \rightarrow X \) using succ_add_one(3) by simp

\{ fix \( k \) assume \( k \in \text{succ}(n) \)

\{ assume \( k \neq 0 \)

from assms(1) \( k \in \text{succ}(n) \) have \( k \in \text{nat} \)

using elem_nat_is_nat(2) by blast

with \( k \neq 0 \) obtain \( m \) where \( m \in \text{nat} \) and \( k = \text{succ}(m) \)

using Nat_ZF_1_L3 by blast

with assms(1) \( k \in \text{succ}(n) \) have \( m \in n \) using succ_mem by simp

with \( \{ 0 \} \in \text{nat} \) assms(1) I have \( b : \{ 0 \} \#+ m \rightarrow X \)

using concat_props(4) by simp

with assms \( m \in \text{nat} \) \( k \in \text{succ}(n) \) \( k = \text{succ}(m) \) \( m \in n \)

have \( a(k) = b(k) \)

using succ_add_one(3) tail_props(2) by simp

\}

ultimately have \( a(k) = b(k) \) by blast

\} thus \( \forall k \in \text{succ}(n). \ a(k) = b(k) \) by simp

qed

with assms(2) show thesis by (rule func_eq)

qed

Properties of Append. It is a bit surprising that the we don’t need to assume that \( n \) is a natural number.

theorem append_props:

assumes A1: \( a : n \rightarrow X \) and A2: \( x \in X \) and A3: \( b = \text{Append}(a, x) \)

shows \( b : \text{succ}(n) \rightarrow X \)

\( \forall k \in n. \ b(k) = a(k) \)

\( b(n) = x \)

proof -
A special case of append_props: appending to a nonempty list does not change the head (first element) of the list.

corollary head_of_append:
assumes n ∈ nat and a: succ(n) → X and x ∈ X
shows Append(a,x)(0) = a(0)
using assms append_props empty_in_every_succ by auto

Tail commutes with Append.

theorem tail_append_commute:
assumes A1: n ∈ nat and A2: a: succ(n) → X and A3: x ∈ X
shows Append(Tail(a),x) = Tail(Append(a,x))
proof -
let b = Append(Tail(a),x)
let c = Tail(Append(a,x))
from A1 A2 have I: Tail(a) : n → X using tail_props
by simp
from A1 A2 A3 have
  succ(n) ∈ nat and Append(a,x) : succ(succ(n)) → X
  using append_props by auto
then have II: ∀ k ∈ succ(n). c(k) = Append(a,x)(succ(k))
  by (rule tail_props)
from assms have
  b : succ(n) → X and c : succ(n) → X
  using tail_props append_props by auto
moreover have ∀ k ∈ succ(n). b(k) = c(k)
proof -
  { fix k assume k ∈ succ(n)
    hence k ∈ n ∨ k = n by auto
    moreover
    { assume A4: k ∈ n
      with assms II have c(k) = a(succ(k))
        using succ_ineq append_props by simp
      moreover
      from A3 I have ∀ k ∈ n. b(k) = Tail(a)(k)
        using append_props by simp
      with A1 A2 A4 have b(k) = a(succ(k))
        using tail_props by simp
  }
ultimately have \( b(k) = c(k) \) by simp 

moreover
{ assume A5: \( k = n \)
with A2 A3 I II have \( b(k) = c(k) \)
using append_props by auto }
ultimately have \( b(k) = c(k) \) by auto

thus thesis by simp

qed
ultimately show \( b = c \) by (rule func_eq)

qed

NELists are non-empty lists

lemma non_zero_List_func_is_NEList:
shows NELists(X) = \{a ∈ Lists(X). a ≠ 0\}

proof
{ fix a assume as: a ∈ \{a ∈ Lists(X). a ≠ 0\}
from as obtain n where a: n ∈ nat a: n → X unfolding Lists_def
by auto
{ assume n=0
with a(2) have a=0 unfolding Pi_def by auto
with as have False by auto
}

hence n ≠ 0 by auto

with a(1) obtain k where k ∈ nat n = succ(k) using Nat_ZF_1_L3
by auto

with a(2) have a ∈ NELists(X) unfolding NELists_def by auto
}

moreover
{ fix a assume as: a ∈ NELists(X)
then obtain k where k: a: succ(k) → X k ∈ nat
unfolding NELists_def by auto
{ assume a=0
hence domain(a) = 0 by auto

with k(1) have succ(k) = 0 using domain_of_fun by auto

hence False by auto
}

moreover
{ from k(2) have succ(k) ∈ nat using nat_succI by auto

with k(1) have a ∈ Lists(X) unfolding Lists_def by auto
}

ultimately
have a ∈ \{a ∈ Lists(X). a ≠ 0\} by auto

}

ultimately show thesis by auto

qed

Properties of Init.

theorem init_props:
assumes A1: \( n ∈ \text{nat} \) and A2: \( a: \text{succ}(n) \to X \)

shows
Init(a) : n → X
∀ k ∈ n. Init(a)(k) = a(k)
\[ a = \text{Append}(\text{Init}(a), a(n)) \]

proof -
  
  have \( n \subseteq \text{succ}(n) \) by auto
  
  with A2 have \( \text{restrict}(a,n) : n \to X \)
  
    using \text{restrict_type2} by simp

moreover from A1 A2 have I: \( \text{restrict}(a,n) = \text{Init}(a) \)
  
    using func1_1_L1 pred_succ_eq Init_def by simp

ultimately show thesis1: \( \text{Init}(a) : n \to X \) by simp

{ fix \( k \) assume \( k \in n \)
  
    then have \( \text{restrict}(a,n)(k) = a(k) \)
  
    using \text{restrict} by simp

  with I have \( \text{Init}(a)(k) = a(k) \) by simp
}

then show thesis2: \( \forall k \in n. \text{Init}(a)(k) = a(k) \) by simp

let \( b = \text{Append}(\text{Init}(a), a(n)) \)

from A2 thesis1 have II:
  
  \( \text{Init}(a) : n \to X \ a(n) \in X \)
  
  \( b = \text{Append}(\text{Init}(a), a(n)) \)

using \text{apply_funtype} by auto

note A2

moreover from II have \( b : \text{succ}(n) \to X \)
  
    by (rule append_props)

moreover have \( \forall k \in \text{succ}(n). a(k) = b(k) \)

proof -

{ fix \( k \) assume A3: \( k \in n \)
  
    from II have \( \forall j \in n. b(j) = \text{Init}(a)(j) \)

  by (rule append_props)

  with \( A3 \) have \( a(k) = b(k) \) by simp } 

moreover

from II have \( b(n) = a(n) \)
  
    by (rule append_props)

hence \( a(n) = b(n) \) by simp

ultimately show \( \forall k \in \text{succ}(n). a(k) = b(k) \)
  
    by simp

qed

ultimately show \( a = b \) by (rule func_eq)

qed

The initial part of a non-empty list is a list, and the domain of the original list is the successor of its initial part.

theorem init_NElist:
  
  assumes \( a \in \text{NELists}(X) \)
  
  shows \( \text{Init}(a) \in \text{Lists}(X) \) and \( \text{succ}(\text{domain}(\text{Init}(a))) = \text{domain}(a) \)

proof -

from assms obtain \( n \) where \( n \in \text{nat} a : \text{succ}(n) \to X \)
  
    unfolding \text{NELists_def} by auto

then have tailF: \( \text{Init}(a) : n \to X \) using init_props(1) by auto

with \( n(1) \) show \( \text{Init}(a) \in \text{Lists}(X) \)
  
    unfolding Lists_def by auto

from tailF have domain(\text{Init}(a)) = n using domain_of_fun by auto

moreover from \( n(2) \) have \( \text{domain}(a) = \text{succ}(n) \) using domain_of_fun

199
by auto
ultimately show succ(domain(Init(a))) = domain(a) by auto
qed

If we take init of the result of append, we get back the same list.

shows Init(Append(a,x)) = a
proof -
  from A2 A3 have Append(a,x): succ(n)→X using append_props by simp
  with A1 have Init(Append(a,x)):n→X and ∀k∈n. Init(Append(a,x))(k)
  = Append(a,x)(k)
    using init_props by auto
  with A2 A3 have ∀k∈n. Init(Append(a,x))(k) = a(k) using append_props
    by simp
  with ‹Init(Append(a,x)):n→X› A2 show thesis by (rule func_eq)
qed

A reformulation of definition of Init.

lemma init_def: assumes n ∈ nat and a:succ(n)→X
  shows Init(a) = restrict(a,n)
  using assms func1_1_L1 Init_def by simp

Another reformulation of the definition of Init, starting with the expression defining the list.

lemma init_def_alt: assumes n∈nat and ∀k∈n #+ 1. q(k) ∈ X
  shows Init({⟨k,q(k)⟩. k∈n #+ 1}) = {⟨k,q(k)⟩. k∈n}
proof -
  let a = {⟨k,q(k)⟩. k∈n #+ 1}
  from assms(2) have a:n #+ 1→X by (rule ZF_fun_from_total)
  moreover from assms(1) have n #+ 1 = succ(n) using succ_add_one(1)
    by simp
  ultimately have a:succ(n)→X by simp
  with assms(1) have Init(a) = restrict(a,n) using init_def by simp
  moreover
  from assms(1) have n ⊆ n #+ 1 by auto
  then have restrict(a,n) = {⟨k,q(k)⟩. k∈n}
    by (rule restrict_def_alt)
  ultimately show thesis by simp
qed

A lemma about extending a finite sequence by one more value. This is just a more explicit version of append_props.

lemma finseq_extend: assumes a:n→X y∈X b = a ∪ {(n,y)}
sshows b: succ(n) → X
∀k∈n. b(k) = a(k)
b(n) = y
The next lemma is a bit displaced as it is mainly about finite sets. It is proven here because it uses the notion of \texttt{Append}. Suppose we have a list of element of \(A\) is a bijection. Then for every element that does not belong to \(A\) we can we can construct a bijection for the set \(A \cup \{x\}\) by appending \(x\). This is just a specialised version of lemma \texttt{bij_extend_point} from \texttt{func1.thy}.

\begin{verbatim}
lemma bij_append_point:
  assumes A1: n \in\ nat and A2: b \in bij(n,X) and A3: x /\in\ X
  shows Append(b,x) \in bij(succ(n), X \cup \{x\})
proof -
  from A2 A3 have b \cup \{(n,x)\} \in bij(n \cup \{n\},X \cup \{x\})
    using mem_not_refl bij_extend_point by simp
  moreover have Append(b,x) = b \cup \{(n,x)\}
    proof -
      from A2 have b : n \rightarrow X
        using bij_def surj_def by simp
      then have b : n \rightarrow X \cup \{x\}
        using func1_1_L1B by blast
      then show Append(b,x) = b \cup \{(n,x)\}
        using Append_def func1_1_L1 by simp
    qed
  ultimately show thesis
    using succ_explained by auto
qed
\end{verbatim}

The next lemma rephrases the definition of \texttt{Last}. Recall that in ZF we have \(\{0,1,2,\ldots,n\} = n + 1 = \text{succ}(n)\).

\begin{verbatim}
lemma last_seq_elem: assumes a: succ(n) \rightarrow X
  shows Last(a) = a(n)
using assms func1_1_L1 pred_succ_eq Last_def by simp
\end{verbatim}

The last element of a non-empty list valued in \(X\) is in \(X\).

\begin{verbatim}
lemma last_type: assumes a \in NELists(X)
  shows Last(a) \in X
using assms last_seq_elem apply_funtype unfolding NELists_def by auto
\end{verbatim}

The last element of a list of length at least 2 is the same as the last element of the tail of that list.

\begin{verbatim}
lemma last_tail_last: assumes n\in nat a: succ(succ(n)) \rightarrow X
  shows Last(Tail(a)) = Last(a)
proof -
  from assms have Last(Tail(a)) = Tail(a)(n)
    using tail_props(1) last_seq_elem by blast
  also from assms have Tail(a)(n) = a(succ(n)) using tail_props(2)
    by blast
  also from assms(2) have a(succ(n)) = Last(a) using last_seq_elem
    by simp
  finally show thesis by simp
qed
\end{verbatim}
If two finite sequences are the same when restricted to domain one shorter than the original and have the same value on the last element, then they are equal.

lemma finseq_restr_eq: assumes A1: n ∈ nat and A2: a: succ(n) → X b: succ(n) → X and A3: restrict(a,n) = restrict(b,n) and A4: a(n) = b(n)
shows a = b
proof -
  { fix k assume k ∈ succ(n)
      then have k ∈ n ∨ k = n by auto
      moreover
      { assume k ∈ n
        then have restrict(a,n)(k) = a(k) and restrict(b,n)(k) = b(k)
        using restrict by auto
          with A3 have a(k) = b(k) by simp }
      moreover
      { assume k = n
        with A4 have a(k) = b(k) by simp }
    } then have ∀ k ∈ succ(n). a(k) = b(k) by simp
  with A2 show a = b by (rule func_eq)
qed

Concatenating a list of length 1 is the same as appending its first (and only) element. Recall that in ZF set theory 1 = {0}.

shows Concat(a,b) = Append(a,b(0))
proof -
  let C = Concat(a,b)
  let A = Append(a,b(0))
  from A1 A2 A3 have I:
    n ∈ nat  1 ∈ nat
    a:n→X  b:1→X by auto
  have C : succ(n) → X
  proof -
    from I have C : n #+ 1 → X
      by (rule concat_props)
    with A1 show C : succ(n) → X by simp
  qed
  moreover from A2 A3 have A : succ(n) → X
    using apply_funtype append_props by simp
  moreover have ∀k ∈ succ(n). C(k) = A(k)
  proof
    fix k assume k ∈ succ(n)
    moreover
    { assume k ∈ n
      then have C(k) = A(k) by simp
      with A1 have C(k) = Append(a,b(0))(k)
        by (rule indep)
      with A2 have C(k) = Append(a,b(0))(k) = Append(a,b(k-1))(k+1)
        by simp
      with A3 have C(k) = Append(a,b(k-1))(k+1) = A(k+1)
        by simp
      with A1 have C(k) = A(k+1) by simp
    }
  qed
  qed
moreover from I have \( \forall i \in n. \ C(i) = a(i) \)
by (rule concat_props)
moreover from A2 A3 have \( \forall i \in n. \ A(i) = a(i) \)
using apply_funtype append_props by simp
ultimately have \( C(n) = A(n) \)
proof -
  from I have \( \forall j \in 1. \ C(n \# j) = b(j) \)
by (rule concat_props)
with A1 A2 A3 show \( C(n) = A(n) \)
using apply_funtype append_props by simp
qed
ultimately show \( C = A \) by (rule func_eq)
qed

If \( x \in X \) then the singleton set with the pair \( \langle 0, x \rangle \) as the only element is a list of length 1 and hence a nonempty list.

**Lemma list_len1_singleton:** assumes \( x \in X \) shows \( \{ \langle 0, x \rangle \} : 1 \to X \) and \( \{ \langle 0, x \rangle \} \in \text{NELists}(X) \)
proof -
  from assms have \( \{ \langle 0, x \rangle \} : \{0\} \to X \) using pair_func_singleton
  by simp
moreover have \( \{0\} = 1 \) by auto
ultimately show \( \{ \langle 0, x \rangle \} : 1 \to X \) and \( \{ \langle 0, x \rangle \} \in \text{NELists}(X) \)
  unfolding NELists_def by auto
qed

A singleton list is in fact a singleton set with a pair as the only element.

**Lemma list_singleton_pair:** assumes A1: \( x:1 \to X \) shows \( x = \{ \langle 0, x(0) \rangle \} \)
proof -
  from A1 have \( x = \{ \langle t, x(t) \rangle. t \in \{0\} \} \) by (rule fun_is_set_of_pairs)
hence \( x = \{ \langle t, x(t) \rangle. t \in \{0\} \} \) by simp
thus thesis by simp
qed

When we append an element to the empty list we get a list with length 1.

**Lemma empty_append1:** assumes A1: \( x \in X \) shows \( \text{Append}(0, x) : 1 \to X \) and \( \text{Append}(0, x)(0) = x \)
proof -
  let \( a = \text{Append}(0, x) \)
  have \( a = \{ \langle 0, x \rangle \} \) using Append_def by auto
  with A1 show \( a : 1 \to X \) and \( a(0) = x \)
    using list_len1_singleton pair_func_singleton
    by auto
qed

Appending an element is the same as concatenating with certain pair.
lemma append_concat_pair:
  assumes n : \nat and a : n \to X and x \in X
  shows Append(a,x) = Concat(a,\{\langle 0,x \rangle \})
  using assms list_len1_singleton append_1elem pair_val
  by simp

An associativity property involving concatenation and appending. For proof
we just convert appending to concatenation and use concat_assoc.

lemma concat_append_assoc: assumes A1: n \in \nat k \in \nat and
  A2: a:n \to X b:k \to X and A3: x \in X
  shows Append(Concat(a,b),x) = Concat(a, Append(b,x))
proof -
  from A1 A2 A3 have
    n #+ k \in \nat Concat(a,b) : n #+ k \to X x \in X
    using concat_props by auto
  then have
    Append(Concat(a,b),x) = Concat(Concat(a,b),\{\langle 0,x \rangle \})
    by (rule append_concat_pair)
moreover
from A1 A2 A3 have
  n \in \nat k \in \nat 1 \in \nat
  a:n \to X b:k \to X \{\langle 0,x \rangle \} : 1 \to X
  using list_len1_singleton by auto
then have
  Concat(Concat(a,b),\{\langle 0,x \rangle \}) = Concat(a, Concat(b,\{\langle 0,x \rangle \}))
  by (rule concat_assoc)
moreover from A1 A2 A3 have Concat(b,\{\langle 0,x \rangle \}) = Append(b,x)
  using list_len1_singleton append_1elem pair_val by simp
ultimately show Append(Concat(a,b),x) = Concat(a, Append(b,x))
  by simp
qed

An identity involving concatenating with init and appending the last ele-

lemma concat_init_last_elem:
  assumes n \in \nat k \in \nat and
  a: n \to X and b : succ(k) \to X
  shows Append(Concat(a,init(b)),b(k)) = Concat(a,b)
  using assms init_props apply_funtype concat_append_assoc
  by simp

A lemma about creating lists by composition and how Append behaves in
such case.

lemma list_compose_append:
  assumes A1: n \in \nat and A2: a : n \to X and
  A3: x \in X and A4: c : X \to Y
  shows
  c 0 Append(a,x) : succ(n) \to Y
\[ c \circ \text{Append}(a, x) = \text{Append}(c \circ a, c(x)) \]

**proof**
- Let \( b = \text{Append}(a, x) \)
- Let \( d = \text{Append}(c \circ a, c(x)) \)
- From A2 A4 have \( c \circ a : n \rightarrow Y \)
  - Using \text{comp_fun} by simp
- From A2 A3 have \( b : \text{succ}(n) \rightarrow X \)
  - Using \text{append_props} by simp
- With A4 show \( c \circ b : \text{succ}(n) \rightarrow Y \)
  - Using \text{comp_fun} by simp
- Moreover from A3 A4 \( c \circ a : n \rightarrow Y \)
  - Using \text{append_props} by simp
- Moreover have \( \forall k \in \text{succ}(n). (c \circ b)(k) = d(k) \)
  - Proof -
    - \{ fix \( k \) assume \( k \in \text{succ}(n) \)
      - With \( b : \text{succ}(n) \rightarrow X \)
        - \( (c \circ b)(k) = c(b(k)) \)
    - Using \text{comp_fun_apply} by simp
      - With A2 A3 A4 \( c \circ a : n \rightarrow Y \) \( c \circ a : n \rightarrow Y \) \( k \in \text{succ}(n) \)
        - Have \( (c \circ b)(k) = d(k) \)
      - Using \text{append_props} \text{comp_fun_apply} apply_functype
        - By auto
    - \} thus thesis by simp
  -Qed
- Ultimately show \( c \circ b = d \) by (rule \text{func_eq})
  -Qed

A lemma about appending an element to a list defined by set comprehension.

**lemma set_list_append:** assumes
\[ A1: \forall i \in \text{succ}(k). b(i) \in X \] and
\[ A2: a = \{ \langle i, b(i) \rangle. i \in \text{succ}(k) \} \]
shows
\[ a: \text{succ}(k) \rightarrow X \]
\[ \{ \langle i, b(i) \rangle. i \in k \}: k \rightarrow X \]
\[ a = \text{Append}(\{ \langle i, b(i) \rangle. i \in k \}, b(k)) \]

**proof**
- From A1 have \( \{ \langle i, b(i) \rangle. i \in \text{succ}(k) \} : \text{succ}(k) \rightarrow X \)
  - By (rule \text{ZF_fun_from_total})
- With A2 show \( a: \text{succ}(k) \rightarrow X \) by simp
- From A1 have \( \forall i \in k. b(i) \in X \)
  - By simp
- Then show \( \{ \langle i, b(i) \rangle. i \in k \}: k \rightarrow X \)
  - By (rule \text{ZF_fun_from_total})
- With A2 show \( a = \text{Append}(\{ \langle i, b(i) \rangle. i \in k \}, b(k)) \)
  - Using \text{func1_1_L1 Append_def} by auto
  -Qed

A version of \text{set_list_append} using \( n + 1 \) instead of \text{succ}(n).
lemma set_list_append1:
assumes n∈nat and ∀k∈n #+ 1. q(k) ∈ X
defines a≡{(k,q(k)). k∈n #+ 1}
shows a: n #+ 1 → X
{⟨k,q(k)⟩. k ∈ n}: n → X
Init(a) = {(k,q(k)). k ∈ n}
a = Append({⟨k,q(k)⟩. k ∈ n},q(n))
a = Append(Init(a), q(n))
a = Append(Init(a), a(n))
proof -
from assms(1) have I: n #+ 1 = succ(n) using succ_add_one(1)
by simp

with assms show a: n #+ 1 → X and {⟨k,q(k)⟩. k ∈ n}: n → X
and II: Init(a) = {(k,q(k)). k ∈ n}
using set_list_append(1,2) init_def_alt by simp_all

from assms(2,3) I have ∀k∈succ(n). q(k) ∈ X and a = {(k,q(k)). k ∈ succ(n)}
by simp_all

then show a = Append({⟨k,q(k)⟩. k ∈ n},q(n))
using set_list_append(3) by simp

with II show a = Append(Init(a), q(n)) by simp

from I have n ∈ n #+ 1 by simp
then have {(k,q(k)). k∈n #+ 1}(n) = q(n)
by (rule ZF_fun_from_tot_val1)

with assms(3) a = Append(Init(a), q(n)) show a = Append(Init(a), a(n))
by simp

qed

An induction theorem for lists.

lemma list_induct: assumes A1: ∀b∈1→X. P(b) and
A2: ∀b∈NELists(X). P(b) −→ (∀x∈X. P(Append(b,x))) and
A3: d ∈ NELists(X)
serves P(d)
proof -
{ fix n
  assume n∈nat
  moreover from A1 have ∀b∈succ(0)→X. P(b) by simp
  moreover have ∀k∈nat. ((∀b∈succ(k)→X. P(b)) −→ (∀c∈succ(succ(k))→X. P(c)))
  proof
    { fix k assume k ∈ nat assume ∀b∈succ(k)→X. P(b)
      have ∀c∈succ(succ(k))→X. P(c)
      proof
        fix c assume c: succ(succ(k))→X
        let b = Init(c)
        let x = c(succ(k))
        qed
      qed
    qed
  qed
  qed

206
from \( k \in \text{nat} \) \( \langle c: \text{succ}(\text{succ}(k)) \rightarrow X \rangle \) have \( b: \text{succ}(k) \rightarrow X \) using init_props by simp

with \( A2 \) \( \langle k \in \text{nat} \rangle \langle \forall b \in \text{succ}(k) \rightarrow X. P(b) \rangle \) have \( \forall x \in X. P(\text{Append}(b, x)) \) using NELists_def by auto

by simp

with \( \langle k \in \text{nat} \rangle \langle c: \text{succ}(\text{succ}(k)) \rightarrow X \rangle \) show \( P(\text{Append}(b, x)) \) using apply_funtype by simp

with \( \langle k \in \text{nat} \rangle \langle c: \text{succ}(\text{succ}(k)) \rightarrow X \rangle \) have \( P(\text{Append}(b, x)) \) using init_props by simp

qed

ultimately have \( \forall b \in \text{succ}(n) \rightarrow X. P(b) \) by (rule ind_on_nat)

with \( A3 \) show thesis using NELists_def by auto

qed

A dual notion to \( \text{Append} \) is \( \text{Prepend} \) where we add an element to the list at the beginning of the list. We define the value of the list \( a \) prepended by an element \( x \) as \( x \) if index is 0 and \( a(k - 1) \) otherwise.

**definition**

\[
\text{Prepend}(a, x) \equiv \{ (k, \text{if } k = 0 \text{ then } x \text{ else } a(k - 1)) \}. k \in \text{domain}(a) + 1
\]

If \( a : n \rightarrow X \) is a list, then \( a \) with prepended \( x \in X \) is a list as well and its first element is \( x \).

**lemma prepend_props:**

assumes \( n \in \text{nat} \) \( a : n \rightarrow X \) \( x \in X \)

shows \( \text{Prepend}(a, x) : (n + 1) \rightarrow X \) and \( \text{Prepend}(a, x)(0) = x \)

proof -

let \( b = \{ (k, \text{if } k = 0 \text{ then } x \text{ else } a(k - 1)) \}. k \in n + 1 \}

have \( \forall k \in n + 1. (\text{if } k = 0 \text{ then } x \text{ else } a(k - 1)) \in X \)

proof -

\{
fix \( k \) assume \( k \in n + 1 \)
let \( v = \text{if } k = 0 \text{ then } x \text{ else } a(k - 1) \)
\{ assume \( k \neq 0 \)
with \( \langle k \in n + 1 \rangle \) have \( n \neq 0 \) by auto
from assms(1) \( \langle k \in n + 1 \rangle \) have \( k \in \text{nat} \)
using elem_nat_is_nat(2) by blast
from assms(1) have \( \text{succ}(n) = n + 1 \)
using succ_add_one(1) by simp
with \( \langle k \in n + 1 \rangle \) have \( k \in \text{succ}(n) \) by simp
with assms(1) \( \langle n \neq 0 \rangle \) have \( \text{pred}(k) \in n \)
using pred_succ_mem by simp
with assms(2) \( \langle k \in n \rangle \langle k \neq 0 \rangle \) have \( v \in X \)
using pred_minus_one apply_funtype by simp
\}

with assms(3) have \( v \in X \) by simp
\}

thus thesis by simp
qed

then have \( b : (n + 1) \rightarrow X \) by (rule ZF_fun_from_total)

with assms(2) show \( \text{Prepend}(a, x) : (n + 1) \rightarrow X \)
using func1_1_L1 unfolding Prepend_def by simp
from assms(1) have 0 ∈ n #+ 1
using succ_add_one(1) empty_in_every_succ by simp
then have b(0) = (if 0 = 0 then x else a(0 #- 1))
  by (rule ZF_fun_from_tot_val1)
with assms(2) show Prepend(a,x)(0) = x
  using func1_1_L1 unfolding Prepend_def by simp
qed

When prepending an element to a list the values at positive indices do not change.

lemma prepend_val: assumes n ∈ nat a: n → X x ∈ X k ∈ n
shows Prepend(a,x)(k #+ 1) = a(k)
proof -
  let b = {⟨k,if k = 0 then x else a(k #- 1)⟩. k ∈ n #+ 1}
from assms(1,4) have k ∈ nat
    using elem_nat_is_nat(2) by simp
with assms(1) have succ(n) = n #+ 1 and succ(k) = k #+ 1
    using succ_add_one(1) by auto
with assms(1,4) have k #+ 1 ∈ n #+ 1
    using succ_ineq by simp
from ‹k #+ 1 ∈ n #+ 1› have b(k #+ 1) = (if k #+ 1 = 0 then x else a((k #+ 1) #- 1))
    by (rule ZF_fun_from_tot_val1)
with assms(2) ‹k ∈ nat› show thesis
  using func1_1_L1 unfolding Prepend_def by simp
qed

18.2 Lists and cartesian products

Lists of length \( n \) of elements of some set \( X \) can be thought of as a model of the cartesian product \( X^n \) which is more convenient in many applications.

There is a natural bijection between the space \((n+1) \to X\) of lists of length \( n+1 \) of elements of \( X \) and the cartesian product \((n \to X) \times X\).

lemma lists_cart_prod: assumes n ∈ nat
  shows \( \{⟨x,⟨\text{Init}(x),x(n)⟩⟩. x ∈ \text{succ}(n)\to X\} \in \text{bij}(\text{succ}(n)\to X, (n\to X)\times X)\)
proof -
  let f = \( \{⟨x,⟨\text{Init}(x),x(n)⟩⟩. x ∈ \text{succ}(n)\to X\}\)
from assms have \( \forall x ∈ \text{succ}(n)\to X. (\text{Init}(x),x(n)) ∈ (n\to X)\times X\)
    using init_props succ_iff apply_functype by simp
then have I: f: (succ(n)→X)→((n→X)×X) by (rule ZF_fun_from_total)
moreover from assms I have \( \forall x∈\text{succ}(n)\to X. \forall y∈\text{succ}(n)\to X. f(x)=f(y) \)
→ x=y
  using ZF_fun_from_tot_val init_def finseq_restr_eq by auto
moreover have \( \forall p∈(n→X)×X. \exists x∈\text{succ}(n)→ X. f(x) = p \)
proof
  fix p assume p ∈ (n→X)×X

208
let x = Append(fst(p),snd(p))
from assms ⟨p ∈ (n→X)×X⟩ have x: succ(n)→ X using append_props by simp
with I have f(x) = ⟨Init(x), x(n)⟩ using succ_iff ZF_fun_from_tot_val by simp
moreover from assms ⟨p ∈ (n→X)×X⟩ have Init(x) = fst(p) and x(n) = snd(p)
using init_append append_props by auto
ultimately have f(x) = ⟨fst(p), snd(p)⟩ by auto
with ‹p ∈ (n→X)×X› ‹x:succ(n)→ X› show ∃x∈ succ(n)→ X. f(x) = p by auto
qed
ultimately show thesis using inj_def surj_def bij_def by auto
qed

We can identify a set X with lists of length one of elements of X.

lemma singleton_list_bij: shows \{⟨x,x(0)⟩. x∈ 1→X\} ∈ bij(1→X,X)
proof -
let f = \{⟨x,x(0)⟩. x∈ 1→X\}
have ∀x∈ 1→X. x(0) ∈ X using apply_funtype by simp
then have I: f:(1→X)→ X by (rule ZF_fun_from_total)
moreover have ∀x∈ 1→X. ∀y∈ 1→X. f(x) = f(y) ⟷ x=y
proof -
{ fix x y
  assume x:1→X y:1→X and f(x) = f(y)
  with I have x(0) = y(0) using ZF_fun_from_tot_val by auto
  moreover from \{x∈ 1→X. x(0) = y(0)\} have x = \{0,x(0)\} and y = \{0,y(0)\}

  using list_singleton_pair by auto
  ultimately have x=y by simp
} thus thesis by auto
qed
moreover have ∀y∈X. ∃x∈ 1→X. f(x)=y
proof
fix y assume y∈X
let x = \{0,y\}
from I ⟨y∈X⟩ have x:1→X and f(x) = y
  using list_len1_singleton ZF_fun_from_tot_val pair_val by auto
thus ∃x∈ 1→X. f(x)=y by auto
qed
ultimately show thesis using inj_def surj_def bij_def by simp
qed

We can identify a set of X-valued lists of length with X.

lemma list_singleton_bij: shows \{⟨x,{⟨0,x⟩}⟩. x∈ X\} ∈ bij(X,1→X) and
\{⟨y,y(0)⟩. y∈ 1→X\} = converse(\{⟨x,{⟨0,x⟩}⟩. x∈ X\}) and
\{⟨x,{⟨0,x⟩}⟩. x∈ X\} = converse(\{⟨y,y(0)⟩. y∈ 1→X\})
proof -
let f = \{(y,y(0)). y \in 1 \to X\}
let g = \{(x,\{(0,x)\}).x \in X\}

have 1 = \{0\} by auto
then have f \in bij(1 \to X,X) and g:X \to (1 \to X)
   using singleton_list_bij pair_func_singleton ZF_fun_from_total
   by auto
moreover have \forall y \in 1 \to X.g(f(y)) = y
proof
  fix y assume y:1 \to X
  have f:(1 \to X) \to X using singleton_list_bij bij_def inj_def by simp
  with 1 = \{0\} and y:1 \to X g:X \to (1 \to X)
  show g(f(y)) = y
  using ZF_fun_from_tot_val apply_funtype func_singleton_pair
  by simp
qed
ultimately show g \in bij(X,1 \to X) and f = converse(g) and g = converse(f)
  using comp_conv_id by auto
qed

What is the inverse image of a set by the natural bijection between X-valued
singleton lists and X?

lemma singleton_vimage: assumes U \subseteq X shows \{x \in 1 \to X. x(0) \in U\} = \{ \{(0,y)\}. y \in U\}
proof
  have 1 = \{0\} by auto
  { fix x assume x \in \{x \in 1 \to X. x(0) \in U\}
    with 1 = \{0\} have x = \{(0, x(0))\} using func_singleton_pair by auto
    } thus \{x \in 1 \to X. x(0) \in U\} \subseteq \{ \{(0,y)\}. y \in U\} by auto
  { fix x assume x \in \{ \{(0,y)\}. y \in U\}
    then obtain y where x = \{(0,y)\} and y \in U by auto
    with 1 = \{0\} obtain x:1 \to X using pair_func_singleton by auto
    } thus \{ \{(0,y)\}. y \in U\} \subseteq \{x \in 1 \to X. x(0) \in U\} by auto
qed

A technical lemma about extending a list by values from a set.

lemma list_append_from: assumes A1: n \in nat and A2: U \subseteq n \to X and A3: V \subseteq X
    shows \{x \in succ(n) \to X. Init(x) \in U \land x(n) \in V\} = (\bigcup y \in V.\{Append(x,y).x \in U\})
proof -
  { fix x assume x \in \{x \in succ(n) \to X. Init(x) \in U \land x(n) \in V\}
    then have x \in succ(n) \to X and Init(x) \in U and I: x(n) \in V
      by auto
    let y = x(n)
    from A1 and \langle x \in succ(n) \to X\rangle have x = Append(Init(x),y)
      using init_props by simp
    with I and \langle Init(x) \in U\rangle have x \in (\bigcup y \in V.\{Append(a,y).a \in U\}) by auto
  } moreover
19 Formal languages

theory Finite_State_Machines_ZF imports FiniteSeq_ZF Finite1 ZF.CardinalArith
begin

19.1 Introduction

This file deals with finite state machines. The goal is to define regular languages and show that they are closed by finite union, finite intersection, complements and concatenation.

We show that the languages defined by deterministic, non-deterministic and non-deterministic with \(\varepsilon\) moves are equivalent.

First, a transitive closure variation on \(r^* = id(field(r)) \cup (r \circ r^*)\).
ultimately have \( t \in \text{converse}(\text{id}(\text{field}(r)) \cup (r^* 0 r)) \) by auto
}

ultimately have \( t \in \text{converse}(\text{id}(\text{field}(r)) \cup (r^* 0 r)) \) by auto

moreover
{
fix \( t \) assume \( t : t \in \text{converse}(\text{id}(\text{field}(r)) \cup (r^* 0 r)) \)
then obtain \( t_1 \ t_2 \) where
\[
\langle t_1, t_2 \rangle \langle t_2, t_1 \rangle \in \text{id}(\text{field}(r)) \cup (r^* O r)
\]
by auto
{
assume \( \langle t_2, t_1 \rangle \in \text{id}(\text{field}(r)) \)
with \( t_{12}(1) \) have \( t \in \text{id}(\text{field}(r)) \) by auto
then have \( t \in \text{id}(\text{field}(r)) \cup \text{converse}(r^* 0 r) \) by auto
} moreover
{
assume \( \langle t_2, t_1 \rangle \notin \text{id}(\text{field}(r)) \)
with \( t_{12}(2) \) have \( t \in (r^* 0 r) \) by auto
with \( t_{12}(1) \) have \( t \in \text{converse}(r^* 0 r) \) by auto
then have \( t \in \text{id}(\text{field}(r)) \cup \text{converse}(r^* 0 r) \) by auto
}
ultimately have \( t \in \text{converse}(\text{id}(\text{field}(r)) \cup (r^* 0 r)) \) by auto

ultimately have converse(\text{id}(\text{field}(r)) \cup (r^* 0 r)) = converse(r^*) by auto
then have converse(converse(\text{id}(\text{field}(r)) \cup (r^* 0 r))) = r^* using converse_converse[OF rtrancl_type]
by auto moreover
{
fix \( t \) assume \( t : t \in (\text{id}(\text{field}(r)) \cup (r^* 0 r)) \)
{
assume \( t \in \text{id}(\text{field}(r)) \)
then have \( t : \text{field}(r)^* \text{field}(r) \) by auto
} moreover
{
assume \( t \notin \text{id}(\text{field}(r)) \)
with \( t \in r^* 0 r \) by auto
then have \( t : \text{field}(r)^* \text{field}(r) \) using rtrancl_type unfolding comp_def
by auto
}
ultimately
have \( t \in \text{field}(r)^* \text{field}(r) \) by auto
}
ultimately show thesis using converse_converse[of \text{id}(\text{field}(r))\cup(r^* 0 r) \text{field}(r) \lambda_. \text{field}(r)] by auto
qed

A language is a subset of words.

definition
IsALanguage (_{is a language with alphabet}_) where
Finite(Σ) ⇒ L {is a language with alphabet} Σ ⊆ L ⊆ Lists(Σ)

The set of all words, and the set of no words are languages.

**lemma** full_empty_language:
assumes Finite(Σ)
shows Lists(Σ) {is a language with alphabet} Σ
and 0 {is a language with alphabet} Σ
unfolding IsALanguage_def[OF assms] by auto

## 19.2 Deterministic Finite Automata

A deterministic finite state automaton is defined as a finite set of states, an initial state, a transition function from state to state based on the word and a set of final states.

**definition**
DFSA ('(_,_,_,_') {is an DFSA for alphabet}_) where
Finite(Σ) =⇒ (S,s₀,t,F) {is an DFSA for alphabet} Σ = Finite(S) ∧ s₀ ∈ S ∧ F ⊆ S ∧ t : S × Σ → S

A finite automaton defines transitions on pairs of words and states. Two pairs are transition related if the second word is equal to the first except it is missing the last symbol, and the second state is generated by this symbol and the first state by way of the transition function.

**definition**
DFSAExecutionRelation (reduce D-relation)'(_,_,_') {in alphabet}_) where
Finite(Σ) =⇒ (S,s₀,t,F) {is an DFSA for alphabet} Σ =⇒ {reduce D-relation}(S,s₀,t) {in alphabet} Σ =⇒ {⟨⟨w,s⟩,⟨Init(w),t{s,Last(w)}⟩⟩. ⟨w,s⟩ ∈ NELists(Σ) × S}

We define a word to be fully reducible by a finite state automaton if in the transitive closure of the previous relation it is related to the pair of the empty word and a final state.

Since the empty word with the initial state need not be in field({reduce D-relation}(S,s₀,t) {in alphabet} Σ), we add the extra condition that ⟨⟨∅, s₀⟩, ∅, s₀⟩ is also a valid transition.

**definition**
DFSA_Satisfy (_ <-D '(_,_,_,_') {in alphabet}_) where
Finite(Σ) =⇒ (S,s₀,t,F) {is an DFSA for alphabet} Σ =⇒ i ∈ Lists(Σ) =⇒ i <-D (S,s₀,t,F) {in alphabet} Σ =⇒ (∃q ∈ F. ⟨⟨i,s₀⟩,⟨0,q⟩⟩ ∈ {reduce D-relation}(S,s₀,t) {in alphabet} Σ)∗ ∨ (i = 0 ∧ s₀ ∈ F)

We define a locale for better notation

**locale** DetFinStateAuto =
fixes S and s₀ and t and F and Σ
assumes finite_alphabet: Finite(\Sigma)

assumes DFSA: (S,s_0,t,F)\{is an DFSA for alphabet\}\Sigma

We abbreviate the reduce relation to a single symbol within this locale.

abbreviation (in DetFinStateAuto) r_D where
r_D \equiv \{\text{reduce D-relation}\}(S,s_0,t)\{\text{in alphabet}\}\Sigma

We abbreviate the full reduction condition to a single symbol within this locale.

abbreviation (in DetFinStateAuto) reduce (_{reduces}) where
i_{reduces} \equiv i \prec_D (S,s_0,t,F)\{\text{in alphabet}\}\Sigma

Destruction lemma about deterministic finite state automata.

lemma (in DetFinStateAuto) DFSA_dest:
shows s_0 \in S \subseteq S \times \Sigma \rightarrow S \text{ Finite}(S)\text{ using DFSA unfolding DFSA_def [OF finite_alphabet] by auto}

The set of words that reduce to final states forms a language. This is by definition.

lemma (in DetFinStateAuto) DFSA_language:
shows \{i \in \text{Lists}(\Sigma). i \prec_D (S,s_0,t,F)\{\text{in alphabet}\}\Sigma\} \{\text{is a language with alphabet}\}\Sigma
\text{ unfolding IsALanguage_def [OF finite_alphabet] by auto}

Define this language as an abbreviation to reduce terms

abbreviation (in DetFinStateAuto) LanguageDFSA
where LanguageDFSA \equiv \{i \in \text{Lists}(\Sigma). i \prec_D (S,s_0,t,F)\{\text{in alphabet}\}\Sigma\}

The relation is an actual relation, but even more it is a function (hence the adjective deterministic).

lemma (in DetFinStateAuto) reduce_is_relation_function:
shows relation(r_D) function(r_D) unfolding DFSAExecutionRelation_def [OF finite_alphabet DFSA]
relation_def function_def by auto

The relation, that is actually a function has the following domain and range:

lemma (in DetFinStateAuto) reduce_function:
shows r_D: NELists(\Sigma) \times S \rightarrow Lists(\Sigma) \times S
proof-
from DFSA have T:t:S \times \Sigma \rightarrow S unfolding DFSA_def [OF finite_alphabet]
by auto
{ fx x assume x \in r_D
then obtain l s where x:l \in NELists(\Sigma) s \in S x = \langle l, s, (\text{Init}(l), t(s, \text{Last}(l)))\rangle
unfolding
DFSAExecutionRelation_def [OF finite_alphabet DFSA] by auto

214
from x(1) have Init(l) ∈ Lists(Σ) using init_NElist(1) by auto moreover
from x(1) have Last(l) ∈ Σ using last_type by auto
with x(2) have t⟨s,Last(l)⟩ ∈ S using apply_type[OF T] by auto
moreover note x
ultimately have x∈(NELists(Σ)×S)×(Lists(Σ)×S) by auto
\}
then have \( r:D ∈ \text{Pow}((NELists(Σ)×S)×(Lists(Σ)×S)) \) by auto moreover
\{
fix x assume x∈NELists(Σ)×S
then obtain 1 s where x:1∈NELists(Σ) s∈S x=⟨1,s⟩ by auto
then have \( ⟨⟨1,s⟩,⟨\text{Init}(1),t⟨s,\text{Last}(1)⟩⟩⟩∈r \)
\( \text{unfolding} \) DFSAExecutionRelation_def[OF finite_alphabet DFSA] by auto
with x(3) have x∈domain(r) \( \text{unfolding} \) domain_def by auto
\}
then have NELists(Σ)×S ⊆ domain(r) by auto moreover
note reduce_is_relation_function(2)
ultimately show thesis unfolding Pi_def by auto
qed

The field of the relation contains all pairs with non-empty words, but we cannot assume that it contains all pairs.

corollary (in DetFinStateAuto) reduce_field:
shows field(r) ⊆ Lists(Σ)×S NELists(Σ)×S ⊆ field(r)
proof-
from DFSA have T:t:S×Σ → S unfolding DFSA_def[OF finite_alphabet] by auto
\{
fix x assume x∈field(r)
then have E:\( \exists y. \ (x,y)∈r ∨ ⟨y,x⟩∈r \) unfolding domain_def range_def field_def by auto
\{
assume \( \exists y. \ (x,y)∈r \)
then have \( x∈NELists(Σ)×S \) unfolding DFSAExecutionRelation_def[OF finite_alphabet DFSA] by auto
then have \( x∈Lists(Σ)×S \) unfolding Lists_def NELists_def by auto
\}
moreover
\{
assume \( \neg(\exists y. \ (x,y)∈r) \)
with \( E \) have \( \exists y. \ (y,x)∈r \) by auto
then obtain u v where y:u∈NELists(Σ) v∈S x=⟨\text{Init}(u),t⟨v,\text{Last}(u)⟩⟩
\( \text{unfolding} \) DFSAExecutionRelation_def[OF finite_alphabet DFSA] by auto
from y(1) have Init(u) ∈ Lists(Σ) using init_NElist(1) by auto
moreover
from y(1) have Last(u) ∈ Σ using last_type by auto
with y(2) have t⟨v,Last(u)⟩ ∈ S using apply_type[OF T] by auto
moreover note y(3) ultimately have x∈Lists(Σ)×S by auto
\}

215
ultimately have \( x \in \text{Lists}(\Sigma) \times S \) by auto 

then show field\((r_D) \subseteq \text{Lists}(\Sigma) \times S \) by auto 

show NELists\((\Sigma) \times S \subseteq \text{field}(r_D)\)

using domain_of_fun[OF reduce_function] unfolding field_def by auto

qed

If a word is a reduced version of another, then it can be encoded as a restriction.

lemma (in DetFinStateAuto) seq_is_restriction:

fixes w s u v

assumes \( \langle \langle w, s \rangle, \langle u, v \rangle \rangle \in r_D^* \)

shows \( \text{restrict}(w, \text{domain}(u)) = u \)

proof-

from assms have \( \langle w, s \rangle \in \text{field}(r_D) \) using rtrancl_field[of r_D] relation_field_times_field[of relation_rtrancl[of r_D]] by auto

then have w \( \in \text{Lists}(\Sigma) \) using reduce_field(1) by auto

then obtain n where \( w : \Sigma \to \Sigma \) unfolding Lists_def by auto

by auto

then have base: \( \text{restrict}(w, \text{domain}(w)) = w \) using domain_of_fun by auto

\{

fix y z

assume as: \( \langle \langle w, s \rangle, y \rangle \in r_D^* \) \( \langle y, z \rangle \in r_D \) restrict\((w, \text{domain}(u)) = u \)

= \( \text{fst}(y) \)

from as(1) have \( y : \text{field}(r_D) \) using rtrancl_field[of r_D] relation_field_times_field[of relation_rtrancl[of r_D]] by auto

then obtain y1 y2 where \( y : y = \langle y1, y2 \rangle \) \( y1 \in \text{Lists}(\Sigma) \) \( y2 \in S \) using reduce_field(1) by auto

with as(2) have \( z : z = \langle \text{Init}(y1), \text{t}(y2, \text{Last}(y1)) \rangle \) \( y1 \in \text{NELists}(\Sigma) \) unfolding DFSAExecutionRelation_def[of finite_alphabet DFSA] by auto

then have \( \text{fst}(z) = \text{Init}(y1) \) by auto

with z(2) have \( S : \text{succ}(\text{domain}(\text{fst}(z))) = \text{domain}(y1) \) using init_NEList(2) by auto

from as(3) \( y(1) \) have \( \text{restrict}(w, \text{domain}(y1)) = y1 \) by auto

then have \( \text{restrict}(\text{restrict}(w, \text{domain}(y1)), \text{pred}(\text{domain}(y1))) = \text{Init}(y1) \) unfolding Init_def by auto

then have \( w : \text{restrict}(w, \text{domain}(y1) \cap \text{pred}(\text{domain}(y1))) = \text{Init}(y1) \) unfolding restrict_restrict by auto

from z(2) obtain q where \( q : \text{domain}(y1) = \text{succ}(q) \) \( q \in \text{nat} \) using domain_of_fun unfolding NELists_def by auto

then have \( \text{pred}(\text{domain}(y1)) \subseteq \text{domain}(y1) \) using pred_succ_eq by auto

then have \( \text{domain}(y1) \cap \text{pred}(\text{domain}(y1)) = \text{pred}(\text{domain}(y1)) \) by auto

with \( w \) have \( \text{restrict}(w, \text{pred}(\text{domain}(y1))) = \text{Init}(y1) \) by auto moreover

from \( q \) \( z(2) \) init_props(1)[of _ y1 \( \Sigma \)] have \( \text{domain}(\text{Init}(y1)) = \text{pred}(\text{domain}(y1)) \)

216
using domain_of_fun[of y1 _ λ_... Σ] domain_of_fun[of Init(y1) _ λ_...

Σ]

unfolding NELists_def by auto ultimately
have restrict(w,domain(Init(y1))) = Init(y1) by auto
with z(1) have restrict(w,domain(fst(z))) = fst(z) by auto
}

then have reg:∀ y z. (⟨⟨ w, s ⟩, y ⟩ ∈ rD^* ⟷
(⟨ y, z ⟩ ∈ rD ⟹ restrict(w, domain(fst(y))) = fst(y) ⟹
restrict(w, domain(fst(z))) = fst(z))) by auto
have restrict(w, domain(fst((u, v)))) = fst((u, v))
proof(rule rtrancl_induct[of ⟨ w,s ⟩ ⟨ u,v ⟩ rD λ q. restrict(w,domain(fst(q)))
= fst(q))]
  from base show restrict(w,domain(fst(⟨ w,s ⟩))) = fst((w,s)) by auto
  from asms show ⟨⟨ w, s ⟩, u, v ⟩ ∈ rD^* by auto
  { fix y z
    assume as:⟨⟨ w, s ⟩, y ⟩ ∈ rD^* ⟨ y, z ⟩ ∈ rD restrict(w, domain(fst(y)))
    = fst(y)
    from as(1) have y:field(rD) using rtrancl_field[of rD] relation_field_times_field[of relation_rtrancl[of rD]] by auto
    then obtain y1 y2 where y:y=⟨ y1,y2 ⟩ y1 ∈ NELists(Σ) unfolding DFSAExecutionRelation_def[of finite_alphabet DFSA] by auto
    with as(2) have z:z=⟨ Init(y1),t⟨ y2,Last(y1) ⟩⟩ y1 ∈ NELists(Σ) unfolding DFSAExecutionRelation_def[of finite_alphabet DFSA] by auto
    then have fst(z) = Init(y1) by auto
    with z(2) have S:succ(domain(fst(z))) = domain(y1) using init_NElist(2) by auto
    from as(3) y(1) have restrict(w,domain(y1)) = y1 by auto
    then have restrict(w,domain(y1),pred(domain(y1))) = Init(y1)
    unfolding Init_def by auto
    then have w:restrict(w,domain(y1)∩pred(domain(y1))) = Init(y1)
    using restrict_restrict by auto
    from z(2) obtain q where q:domain(y1) = succ(q) q∈nat using domain_of_fun
    unfolding NELists_def by auto
    then have pred(domain(y1)) ⊆ domain(y1) using pred_succeq by auto
    then have domain(y1) ∩ pred(domain(y1)) = pred(domain(y1)) by auto
    with w have restrict(w,pred(domain(y1))) = Init(y1) by auto
    moreover
    from q z(2) init_props(1)[of _ y1 Σ] have domain(Init(y1)) = pred(domain(y1))
    using domain_of_fun[of y1 _ λ_... Σ] domain_of_fun[of Init(y1) _
λ_... Σ]
    unfolding NELists_def by auto ultimately
    have restrict(w,domain(Init(y1))) = Init(y1) by auto
    with z(1) show restrict(w,domain(fst(z))) = fst(z) by auto
  }
qed
then show thesis by auto

qed

lemma (in DetFinStateAuto) relation_detemnistic:
  assumes \( \langle \langle w, s \rangle, \langle u, v \rangle \rangle \in r_D^{-*} \cap \langle \langle w, s \rangle, \langle u, m \rangle \rangle \in r_D^{-*} \)
  shows \( v = m \)
proof-
  let \( P = \lambda y. \forall q1 q2. \langle \langle w, s \rangle, \langle q1, q2 \rangle \rangle \in r_D^{-*} \rightarrow \text{fst}(y) = q1 \rightarrow \text{snd}(y) = q2 \)
  { fix q1 q2 assume \( \langle \langle w, s \rangle, \langle q1, q2 \rangle \rangle \in r_D^{-*} \rightarrow \text{fst}(w, ss) = q1 \)
    then have \( \langle \langle w, s \rangle, \langle w, q2 \rangle \rangle \in r_D^{-*} \) by auto
    then have \( \langle \langle w, s \rangle, \langle w, q2 \rangle \rangle \in \text{id}(\text{field}(r_D)) \cup (r_D^{-*} \cap r_D) \) using rtrancl_rev by auto
    then have \( A : s_{q2} \lor \langle \langle w, s \rangle, \langle w, q2 \rangle \rangle \in r_D^{-*} \cap r_D \) by auto
    { assume \( s \neq q2 \)
      with \( A \) have \( \langle \langle w, s \rangle, \langle w, q2 \rangle \rangle \in r_D^{-*} \cap r_D \) by auto
      then obtain \( b \) where \( b :\langle \langle w, s \rangle, \langle b, w, q2 \rangle \rangle : r_D^{-*} \) unfolding compE by auto
      from \( b(1) \) have \( b = \langle \text{Init}(w), t \langle s, \text{Last}(w) \rangle \rangle \) and \( w : w \in \text{NELists}(\Sigma) \)
      unfolding DFSAExecutionRelation_def[OF finite_alphabet DFSA] by auto
      with \( b(2) \) have \( \text{restrict}(\text{Init}(w), \text{domain}(w)) = w \) using seq_is_restriction by auto
      then have \( \text{domain}(\text{Init}(w)) \cap \text{domain}(w) = \text{domain}(w) \) using domain_restrict[of \( \text{Init}(w) \) domain(w)] by auto
      with \( w \) have \( e : \text{domain}(\text{Init}(w)) \cap \text{domain}(w) = \text{succ}(\text{domain}(\text{Init}(w))) \)
      using init_NEList(2)[of \( w \)] by auto
      { fix \( tt \) assume \( t : \text{tt} : \text{succ}(\text{domain}(\text{Init}(w))) \)
        with \( e \) have \( \text{tt} : \text{domain}(\text{Init}(w)) \cap \text{domain}(w) \) by auto
        then have \( \text{tt} : \text{domain}(\text{Init}(w)) \) by auto
      }
      then have \( \text{succ}(\text{domain}(\text{Init}(w))) \subseteq \text{domain}(\text{Init}(w)) \) by auto
      then have \( \text{domain}(\text{Init}(w)) \subseteq \text{domain}(\text{Init}(w)) \) by auto
      then have \( \text{False} \) using mem_irrefl[of \( \text{domain}(\text{Init}(w)) \)] by auto
    }
    then have \( s = q2 \) by auto
    then have \( \text{snd}(\langle \langle w, s \rangle \rangle) = q2 \) by auto
  }
  then have \( P \in P(\langle \langle w, s \rangle \rangle) \) by auto
  { fix \( y z \) assume \( \langle \langle w, s \rangle, \langle y, z \rangle \rangle : r_D^{-*} \cap \langle \langle y, z \rangle \rangle : r_D \)
    { fix \( q1 q2 \) assume \( z : \langle \langle w, s \rangle, \langle q1, q2 \rangle \rangle \in r_D^{-*} \rightarrow \text{fst}(z) = q1 \)
      from this(1) have \( \langle \langle w, s \rangle, \langle q1, q2 \rangle \rangle \in \text{id}(\text{field}(r_D)) \cup (r_D^{-*} \cap r_D) \) using rtrancl_unfold by auto
      then have \( A : s_{q2} \lor w = q1 \lor \langle \langle w, s \rangle, \langle q1, q2 \rangle \rangle \in r_D^{-*} \cap r_D \) by auto
      from \( y(2) \) obtain \( y1 \) \( y2 \) where \( z : \langle \langle y1, y2 \rangle \rangle \) \( y1 : \text{NELists}(\Sigma) \) \( y2 : \text{S} \) \( z : \text{Init}(y1), t : y2, \text{Last}(y1) \)
      unfolding DFSAExecutionRelation_def[OF finite_alphabet DFSA] by auto
    }
  }

218
assume w:w=q1
with z(2) zz(4) have w=Init(y1) by auto
with y(1) zz(1) have restrict(Init(y1),domain(y1)) = y1 using seq_is_restriction
  by auto
then have domain(Init(y1))∩domain(y1) = domain(y1) using domain_restrict[of Init(y1) domain(y1)] by auto
with zz(2) have e:domain(Init(y1))∩domain(y1) = succ(domain(Init(y1)))
using init_NElist(2)[of y1] by auto

fix tt assume t:tt:succ(domain(Init(y1)))
with e have tt:domain(Init(y1))∩domain(y1) by auto
then have tt:domain(Init(y1)) by auto

then have succ(domain(Init(y1)))⊆ domain(Init(y1)) by auto
then have domain(Init(y1)) ∈ domain(Init(y1)) by auto
then have False using mem_irrefl[of domain(Init(y1))] by auto

with A have ⟨⟨w, s⟩, q1, q2⟩:(r D O r D^*) by auto
then obtain pp where pp:⟨⟨w, s⟩, pp⟩:r D^* ⟨pp, q1, q2⟩:r D unfolding compE by auto
from this(2) obtain ppL ppS where ppl:ppl:NELists(Σ) ppS∈S pp=⟨ppL,ppS⟩
q1=Init(ppL) q2=t⟨ppS,Last(ppL)⟩ unfolding DFSAExecutionRelation_def[OF finite_alphabet DFSA] by auto
from this(3) pp(1) have rr:restrict(w,domain(ppL)) = ppL using seq_is_restriction by auto
then have r:restrict(w,domain(ppL))pred(domain(ppL)) = Last(ppL)
unfolding Last_def by auto
from ppl(1) obtain q where q:ppl:succ(q) → Σ q∈nat unfolding NELists_def by blast
from q(1) have D:domain(ppL) = succ(q) using func1_1_L1[of ppL] by auto
moreover
from D have pred(domain(ppL)) = q using pred_succ_eq by auto
then have pred(domain(ppL)) ∈ succ(q) by auto
ultimately have pred(domain(ppL)) ∈ domain(ppL) by auto
with restrict r have W:wpred(domain(ppL)) = Last(ppL) by auto
from y(1) zz(1) have restrict(w,domain(y1)) = y1 using seq_is_restriction
by auto
moreover note rr moreover
from q have Init(ppL):q → Σ using init_props(1) by auto
then have DInit:domain(Init(ppL)) = q using func1_1_L1 by auto
from zz(4) z(2) have q1=Init(y1) by auto
with ppl(4) have Init(ppL) = Init(y1) by auto
with DInit have Dy1:domain(Init(y1)) = q by auto
from zz(2) obtain o where o:o∈nat y1:succ(o) → Σ unfolding NELists_def by auto
then have Init(y1):o→Σ using init_props(1) by auto
with Dy1 have q=0 using func1_1_L1 by auto
with o(2) have y1:succ(q)→Σ by auto moreover
note q(1) ultimately have y1=ppL using func1_1_L1 by auto moreover
then have fst(y) = ppL using zz(1) by auto
moreover
then have P(z) by auto
then have R:⋀y z. ⟨⟨w,s⟩,y⟩∈rD^* =⇒ ⟨y,z⟩∈rD =⇒ P(y) =⇒ P(z) by auto
then have P(⟨u,v⟩) using rtrancl_induct[of ⟨w,s⟩ ⟨u,v⟩ rD]\ λy. ∀q1 q2. ⟨⟨w,s⟩,(q1,q2)⟩:rD^* −→ fst(y) = q1 −→ snd(y) = q2]
P0 assms(1) by auto
then show thesis using assms(2) by auto
qed

Any non-empty word can be reduced to the empty string, but it does not always end in a final state.

lemma (in DetFinStateAuto) endpoint_exists:
assumes w ∈ NELists(Σ)
shows ∃q ∈ S. ⟨⟨w,s_0⟩,⟨0,q⟩⟩ ∈ rD^-

proof-
let P=λk. ∀y∈Lists(Σ). domain(y) = k −→ y=0 ∨ (∀ss∈S. (∃q∈S. ⟨⟨y,ss⟩,⟨0,q⟩⟩∈rD^*))

{ fix y assume y∈Lists(Σ) domain(y) = 0
  with assms have y=0 unfolding Lists_def using domain_of_fun by auto
  then have y=0 ∨ (∀ss∈S. (∃q∈S. ⟨⟨y,ss⟩,⟨0,q⟩⟩∈rD^*)) by auto
}
then have base:P(0) by auto
{ fix k assume hyp:P(k) k∈nat
  { fix y assume as:y∈Lists(Σ) domain(y) = succ(k)
    from as have y:succ(k)→Σ unfolding Lists_def using domain_of_fun
    by auto
    with hyp(2) have y:y:NELists(Σ) unfolding NELists_def by auto
    then have Init(y):Lists(Σ) succ(domain(Init(y))) = domain(y) using init_NEList by auto
    with as(2) have D:Init(y):Lists(Σ) domain(Init(y)) = k using pred_succ_eq
    by auto
    with hyp(1) have d:Init(y) = 0 ∨ (∀ss∈S. (∃q∈S. ⟨⟨Init(y),ss⟩,⟨0,q⟩⟩∈rD^*))
    by auto
    { assume iy0:Init(y) = 0
      220
    }
  }
}


{ fix ss assume ss∈S 
with i0 y have ⟨⟨y,ss⟩⟩,0,t(ss,Last(y))⟩∈rD unfolding DFSAExecutionRelation_def[OF finite_alphabet DFSA] 
  by auto 
then have ⟨⟨y,ss⟩⟩,0,t(ss,Last(y))⟩∈rD−* using r_into_rtrancl 
by auto 
moreover from `ss∈S` have t(ss,Last(y)) ∈S using apply_type[OF DFSA_dest(3)] 
DFSA_dest(3)) 
  last_type[OF y] by auto 
ultimately have ∃q∈S. ⟨⟨y,ss⟩⟩,0,q⟩:rD−* by auto 
} 
then have ∀ss∈S. ∃q∈S. ⟨⟨y,ss⟩⟩,0,q⟩:rD−* by auto 
then have y = 0 ∨ (∀ss∈S. ∃q∈S. ⟨⟨y,ss⟩⟩,0,q⟩:rD−*) by auto 
moreover 
{ assume qS:∀ss∈S. ∃q∈S. ⟨⟨Init(y),ss⟩⟩,⟨0,q⟩⟩∈rD−* 
  fix ss assume ss∈S 
with y have ⟨⟨y,ss⟩⟩,Init(y),t(ss,Last(y))⟩∈rD unfolding DFSAExecutionRelation_def[ 
finite_alphabet DFSA] 
  by auto 
moreover from `ss∈S` y have t(ss,Last(y)) ∈S using apply_type[OF DFSA_dest(3)] 
DFSA_dest(3)) 
  last_type[OF y] by auto 
ultimately have ∃q∈S. ⟨⟨y,ss⟩⟩,0,q⟩:rD−* by auto 
} 
then obtain q where q∈S ⟨⟨y,ss⟩⟩,⟨0,q⟩⟩∈rD−* by auto 
then have P(succ(k)) by auto 
} then have ind:∀k. k∈nat ⇒ P(k) ⇒ P(succ(k)) by blast 
have dom:domain(w) ∈ nat using assms unfolding NELists_def using domain_of_fun 
by auto 
from ind have P(domain(w)) using nat_induct[of _ P, OF dom base] by auto 
with assms have (∀ss∈S. ∃q∈S. ⟨⟨w, ss⟩⟩,0,q⟩ ∈ rD−*) 
  using non_zero_List_func_is_NEList by auto 
then show thesis using DFSA_dest(1) by auto 
qed 

Example of Finite Automaton of binary lists starting with 0 and ending with
locale ListFrom0To1
begin

Empty state
definition empty where
  empty ≡ 2

The string starts with 0 state
definition ends0 where
  ends0 ≡ succ(2)

The string ends with 1 state
definition starts1 where
  starts1 ≡ 1

The string ends with 0 state
definition starts0 where
  starts0 ≡ 0

The states are the previous 4 states. They are encoded as natural numbers to make it easier to reason about them, and as human readable variable names to make it easier to understand.
definition states where
  states ≡ {empty, starts0, starts1, ends0}

The final state is starts0
definition finalStates where
  finalStates ≡ {starts0}

The transition function is defined as follows:
From the empty state, we transition to state starts1 in case there is a 1 and to state ends0 in case there is a 0.
From the state ends0 we stay in it.
From the states starts1 and starts0 we transition to starts0 in case there is a 0, and to starts1 in case there is a 1.
definition transFun where
  transFun ≡ {⟨⟨empty,1⟩⟩,⟨⟨empty,0⟩⟩,⟨⟨ends0,x⟩⟩. x∈2}∪
  {⟨⟨starts1,0⟩⟩,⟨⟨starts1,1⟩⟩,⟨⟨starts0,0⟩⟩,⟨⟨starts0,1⟩⟩,⟨⟨starts0,0⟩⟩,⟨⟨starts0,1⟩⟩,⟨⟨starts0,0⟩⟩,⟨⟨starts0,1⟩⟩}

Add lemmas to simplify
lemma from0To1[simp] = states_def empty_def transFun_def finalStates_def
ends0_def starts1_def starts0_def
Interpret the example as a deterministic finite state automaton

interpretation dfsaFrom0To1: DetFinStateAuto states empty transFun finalStates
2 unfolding DetFinStateAuto_def apply safe
using Finite_0 apply simp
proof-
  have finA:Finite(2) using nat_into_Finite[of 2] by auto
  have finS:Finite(states) using nat_into_Finite by auto
  moreover have funT:transFun:states×2 → states unfolding Pi_def function_def
  by auto
  moreover have finalStates ⊆ states by auto
  moreover have empty ∈ states by auto
  ultimately show (states,empty,transFun,finalStates){is an DFSA for alphabet}2
  unfolding DFSA_def[OF finA] by auto
qed

Abbreviate the relation to something readable.

abbreviation r0to1 (r{0.*1}) where
r{0.*1} ≡ dfsaFrom0To1.rD

If a word reaches the state starts0, it does not move from it.

lemma invariant_state_3:
  fixes w u y
  assumes ⟨⟨w,ends0⟩,⟨u,y⟩⟩∈ r{0.*1}^*
  shows y = ends0
proof-
  have finA:Finite(2) by auto
  have funT:transFun:states×2 → states
    using dfsaFrom0To1.DFSA_dest(3).
  have snd(⟨u,y⟩) = ends0
  proof(rule rtrancl_induct[of ⟨w,ends0⟩
    (⟨u,y⟩ r0,0  λt. snd(t) = ends0))
    show snd(⟨w, ends0⟩) = ends0 by auto
    from asms show ⟨⟨w, ends0⟩, u, y⟩ ∈ r{0.*1}^*.
    { fix y z assume as:⟨⟨w, ends0⟩, y⟩ ∈ r{0.*1}^* (y, z) ∈ r{0.*1} snd(y)
      = ends0
      from as(3,2) obtain y1 where yy:y=y1,ends0 y1∈NELLists(2)
        z=⟨Init(y1),transFun{ends0,Last(y1)}⟩
        unfolding DFSAExecutionRelation_def[OF finA dfsaFrom0To1.DFSA]
        by auto
      from yy(2) have Last(y1)∈2 using last_type by auto
      then have ‹(ends0,Last(y1)),ends0⟩∈transFun by auto
      then have transFun{ends0,Last(y1)} = ends0
        using apply_equality[OF _ funT, of _ ends0] by auto
      with yy(3) have z=⟨Init(y1),ends0⟩ by auto
      then show snd(z) = ends0 by auto
    }
qed

223
then show thesis by auto 
Qed 

If the string starts in 0 and has reached states starts0 or starts1; then it reduces to starts0. 

lemma invariant_state_0_1: 
fixes w 
assumes w ∈ NELists(2) w0 = 0 
shows ⟨⟨w,starts0⟩,⟨0,starts0⟩⟩∈r{0.*1}^* ⟨⟨w,starts1⟩,⟨0,starts0⟩⟩∈r{0.*1}^*  
proof- 
from assms(1) obtain n where w:n:nat w:succ(n) unfolding NELists_def by auto 
then have dom:domain(w) ∈ nat using domain_of_fun by auto 
{ 
fix q assume q∈NELists(2) 
then obtain m where m∈nat q:succ(m)→2 unfolding NELists_def by auto 
then have domain(q) ≠ 0 using domain_of_fun by auto 
} 
then have base:∀ w. w ∈ NELists(2) ∧ w0 = 0 ∧ domain(w) = 0 → ⟨⟨w,starts0⟩,0,starts0⟩ ∈ r{0.*1}^* ∧ ⟨⟨w,starts1⟩,0,starts0⟩ ∈ r{0.*1}^* 
by auto 
from w(2) have domNO:domain(w) ≠ 0 using domain_of_fun by auto 
have funT:transFun:states×2→states unfolding Pi_def function_def by auto 
have t00:transFun⟨starts0,0⟩ = starts0 using funT apply_equality[of starts0,0] startsFun states×2 λ_. states by auto 
have t01:transFun⟨starts1,0⟩ = starts0 using funT apply_equality[of starts1,0] starts0 transFun states×2 λ_. states by auto 
have t10:transFun⟨starts0,1⟩ = starts1 using funT apply_equality[of starts0,1] starts0 transFun states×2 λ_. states by auto 
have t11:transFun⟨starts1,1⟩ = starts1 using funT apply_equality[of starts1,1] starts1 transFun states×2 λ_. states by auto 
{ 
fix ka assume kaNat:ka:nat 
assume k:∀ w. w ∈ NELists(2) ∧ w0 = 0 ∧ domain(w) = ka → (⟨⟨w,starts0⟩,⟨0,starts0⟩⟩∈r{0.*1}^* ∧ ⟨⟨w,starts1⟩,⟨0,starts0⟩⟩∈r{0.*1}^*) 
{ 
fix y assume y:y∈NELists(2) y0 = 0 domain(y) = succ(ka) 
from y(1) obtain s where s:y:succ(s)→2 s:nat unfolding NELists_def by auto 
then have L:Last(y) = ys using last_seq_elem by auto 
then have last_2:Last(y) ∈ 2 using apply_type[of s(i), of s] by auto 
} 
assume ka:ka =0 
with y(3) have pred(domain(y)) = 0 using pred_succ_eq by auto 
then have y00:y0 = 0 using y(2) unfolding Last_def by auto 
then have ⟨⟨y,starts0⟩,⟨Init(y),transFun⟨starts0,Last(y)⟩⟩⟩∈r{0.*1}^* 
}
unfolding
  DFSAExecutionRelation_def[OF finA dfsaFrom0To1.DFSA]
  using y(1) by auto
with last_2 t00 t10 have ⟨⟨y,starts0⟩, ⟨Init(y), Last(y)⟩⟩ ∈ r{0.*1} by auto
by auto moreover
from ka y(3) s(1) have Init(y):0→2 using init_props(1)[OF s(2,1)]
domain_of_fun[OF s(1)] by auto
then have y0:Init(y) = 0 unfolding Pi_def by auto
with last_2 t00 t10 have ⟨⟨y,starts0⟩, ⟨Init(y), Last(y)⟩⟩ ∈ r{0.*1} by auto
by auto
moreover from ka y(3) s(1) have Init(y):0→2 using init_props(1)[OF s(2,1)]
domain_of_fun[OF s(1)] by auto
then have A: ⟨⟨y,starts0⟩, ⟨Init(y), Last(y)⟩⟩ ∈ r{0.*1} by auto
ultimately have ⟨⟨y,starts0⟩, ⟨Init(y), Last(y)⟩⟩ ∈ r{0.*1}^* using r_into_rtrancl
by auto

note t01 moreover have ⟨⟨y,starts1⟩, ⟨Init(y), transFun⟨starts1,Last(y)⟩⟩⟩ ∈ r{0.*1}
using y(1,2) by auto
moreover note t11 last_2 y0 LL y(2) ultimately have ⟨⟨y,starts1⟩, ⟨Init(y), Last(y)⟩⟩ ∈ r{0.*1}^* using r_into_rtrancl
by auto
with A have ⟨⟨y,starts1⟩, ⟨Init(y), Last(y)⟩⟩ ∈ r{0.*1}^* ∧ ⟨⟨y,starts1⟩, ⟨Init(y), Last(y)⟩⟩ ∈ r{0.*1}^* by auto
by auto
}

moreover
{
assume ka: ka≠0
with kaNat obtain u where u: ka = succ(u) u ∈ nat using Nat_ZF_1_L3
by auto
from y(3) s(1) kaNat s(2) have s = ka using domain_of_fun succ_inject
by auto
with u(1) have s = succ(u) by auto moreover
have Init(y):s→2 using init_props(1)[OF s(2,1)].
ultimately have yu:Init(y):succ(u)→2 by auto
with u have Init(y) ∈ NELists(2) domain(Init(y)) = ka
using domain_of_fun unfolding NELists_def by auto
moreover from yu have Init(y)0 = 0 using init_props(2)[OF nat_succI[OF u(2)], of y 2] s(1) `s=succ(u)` empty_in_every_succ[OF u(2)] y(2) by auto
moreover note k ultimately have ⟨⟨Init(y),starts0⟩, ⟨Init(y),Last(y)⟩⟩ ∈ r{0.*1}^* by auto
then have A: ∀ x ∈ {starts0, starts1}. ⟨⟨Init(y),x⟩, ⟨Init(y), Last(y)⟩⟩ ∈ r{0.*1}^* by auto
by auto
have Q: ⟨⟨y,starts0⟩, ⟨Init(y), transFun⟨starts0,Last(y)⟩⟩⟩ ∈ r{0.*1} by auto
using y(2)
unfolding DFSAExecutionRelation_def[OF finA dfsaFrom0To1.DFSA]
using y(1) by auto
{
assume as: Last(y) = 0
}
with \( Q \) t00 have \( \langle\langle y, \text{starts}_0\rangle, \langle\langle y, \text{starts}_1\rangle, \rangle \rangle \in r\{0.*1\} \) by auto

moreover
\{
  \text{assume as:} \text{Last}(y) = 1
  \text{with} \ Q \ t10 \text{ have} \langle\langle y, \text{starts}_0\rangle, \langle\langle y, \text{starts}_1\rangle, \rangle \rangle \in r\{0.*1\} \) by auto
\}

moreover note \text{last}_2 ultimately have
\((\langle\langle y, \text{starts}_0\rangle, \langle\langle y, \text{starts}_1\rangle, \rangle \rangle \in r\{0.*1\} \) \lor \( \langle\langle y, \text{starts}_1\rangle, \langle\langle y, \text{starts}_1\rangle, \rangle \rangle \in r\{0.*1\}\) by auto

with A have B: \( \langle\langle y, \text{starts}_1\rangle, \langle\langle y, \text{starts}_0\rangle, \rangle \rangle \in r\{0.*1\} \) using \text{rtrancl}\_\text{into}\_\text{trancl2}
\text{trancl}\_\text{into}\_\text{rtrancl} by auto

have \( Q : \langle\langle y, \text{starts}_1\rangle, \langle\langle y, \text{transFun}\text{starts}_1, \text{Last}(y)\rangle \rangle \rangle \in r\{0.*1\} \) using \text{y}(2)

unfolding \text{DFSA}\_\text{ExecutionRelation}\_\text{def}[OF \text{finA dfsaFrom0To1.DFSA}]

using \text{y}(1) by auto
\{
  \text{assume as:} \text{Last}(y) = 0
  \text{with} \ Q \ t01 \text{ have} \langle\langle y, \text{starts}_1\rangle, \langle\langle y, \text{starts}_0\rangle, \rangle \rangle \in r\{0.*1\} \) by auto
\}

moreover
\{
  \text{assume as:} \text{Last}(y) = 1
  \text{with} \ Q \ t11 \text{ have} \langle\langle y, \text{starts}_1\rangle, \langle\langle y, \text{starts}_1\rangle, \rangle \rangle \in r\{0.*1\}\) by auto
\}

moreover note \text{last}_2 ultimately have
\((\langle\langle y, \text{starts}_1\rangle, \langle\langle y, \text{starts}_0\rangle, \rangle \rangle \in r\{0.*1\} \) \lor \( \langle\langle y, \text{starts}_1\rangle, \langle\langle y, \text{starts}_1\rangle, \rangle \rangle \in r\{0.*1\}\) by auto

with A have \( \langle\langle y, \text{starts}_1\rangle, \langle\langle y, \text{starts}_0\rangle, \rangle \rangle \in r\{0.*1\} \) using \text{rtrancl}\_\text{into}\_\text{trancl2}
\text{trancl}\_\text{into}\_\text{rtrancl} by auto

with B have \( \text{rr} : \langle\langle y, \text{starts}_0\rangle, \langle\langle y, \text{starts}_0\rangle, \rangle \rangle \in r\{0.*1\} \) \land \( \langle\langle y, \text{starts}_1\rangle, \langle\langle y, \text{starts}_0\rangle, \rangle \rangle \in r\{0.*1\} \)

by auto
\}

ultimately have \( \langle\langle y, \text{starts}_0\rangle, \langle\langle y, \text{starts}_0\rangle, \rangle \rangle \in r\{0.*1\} \) \land \( \langle\langle y, \text{starts}_1\rangle, \langle\langle y, \text{starts}_0\rangle, \rangle \rangle \in r\{0.*1\} \)

by auto

then have \( \forall w. w \in \text{NELists}(2) \land w_0 = 0 \land \text{domain}(w) = \text{succ}(k) \longrightarrow \langle\langle w, \text{starts}_0\rangle, \langle\langle w, \text{starts}_0\rangle, \rangle \rangle \in r\{0.*1\} \) \land \( \langle\langle w, \text{starts}_1\rangle, \langle\langle w, \text{starts}_0\rangle, \rangle \rangle \in r\{0.*1\} \)

by auto

then have rule: \( \forall k \in \text{nat}. \)
(\( \forall w. w \in \text{NELists}(2) \land w_0 = 0 \land \text{domain}(w) = k \longrightarrow \langle\langle w, \text{starts}_0\rangle, \langle\langle w, \text{starts}_0\rangle, \rangle \rangle \in r\{0.*1\} \) \land \( \langle\langle w, \text{starts}_1\rangle, \langle\langle w, \text{starts}_0\rangle, \rangle \rangle \in r\{0.*1\} \)

\( \forall w. w \in \text{NELists}(2) \land w_0 = 0 \land \text{domain}(w) = \text{succ}(k) \longrightarrow \langle\langle w, \text{starts}_0\rangle, \langle\langle w, \text{starts}_0\rangle, \rangle \rangle \in r\{0.*1\} \) \land \( \langle\langle w, \text{starts}_1\rangle, \langle\langle w, \text{starts}_0\rangle, \rangle \rangle \in r\{0.*1\} \)

by auto

from \text{ind_on_nat}[OF domain(w) \lambda t . \forall w. w\in\text{NELists}(2) \land w_0 =0 \land \text{domain}(w) = t \longrightarrow \langle\langle w, \text{starts}_0\rangle, \langle\langle w, \text{starts}_0\rangle, \rangle \rangle \in r\{0.*1\} \) \land \( \langle\langle w, \text{starts}_1\rangle, \langle\langle w, \text{starts}_0\rangle, \rangle \rangle \in r\{0.*1\} \), \text{OF dom base rule}]

show \( R : \langle\langle w, \text{starts}_0\rangle, \langle\langle w, \text{starts}_0\rangle, \rangle \rangle \in r\{0.*1\} \) \land \( \langle\langle w, \text{starts}_1\rangle, \langle\langle w, \text{starts}_0\rangle, \rangle \rangle \in r\{0.*1\} \)
using assms(2,1) by auto

qed

A more readable reduction statement

abbreviation red (_{reduces in 0.*1}) where
  i{reduces in 0.*1} ≡ dfsaFrom0To1.reduce(i)

Any list starting with 0 and ending in 1 reduces.

theorem starts1ends0_DFSA_reduce:
  fixes i
  assumes i ∈ Lists(2) and i0=0 and Last(i) = 1
  shows i{reduces in 0.*1}
proof
  from assms(1) obtain tt where t:tt ∈ nat i:tt → 2 unfolding Lists_def
  by auto
  then have domNat:domain(i) = tt using domain_of_fun by auto
  { assume domain(i) = 0 moreover
    from assms(3) have ∪(i(Arith.pred(domain(i))))=1 unfolding Last_def
    apply_def by auto
    then have i≠0 by auto
    ultimately have False using t domain_of_fun by auto
  }
  with domNat t(1) obtain y where y:domain(i) = succ(y) y ∈ nat using Nat_ZF_1_L3
  by auto
  with domNat t(2) have iList:i ∈ NELists(2) unfolding NELists_def by auto
  have funT:transFun:states × 2 → states unfolding Pi_def function_def by auto
  have ⟨⟨i,empty⟩,⟨Init(i),transFun⟨empty,Last(i)⟩⟩⟩:r{0.*1}
  unfolding DFSAExecutionRelation_def[OF finA dfsaFrom0To1.DFSA] by auto
  moreover have transFun⟨empty,Last(i)⟩ = 1 using apply_equality[OF _
  funT] assms(3)
  by auto
  ultimately have U:⟨⟨i,empty⟩,⟨Init(i),starts1⟩⟩:r{0.*1} by auto
  { assume y = 0
    with y(1) t(2) have iFun:i:1→2 using domain_of_fun by auto
    then have i = {⟨0,i0⟩} using fun_is_set_of_pairs[of i 1 2] by auto
    with assms(2) have ii:i={⟨0,0⟩} by auto
    then have ∀y. ∃x∈Arith.pred(domain(i)). x, y ∈ i → y=0
    assume y = 0
    with y(1) t(2) have iFun:i:1→2 using domain_of_fun by auto
    then have i = {⟨0,i0⟩} using fun_is_set_of_pairs[of i 1 2] by auto
    with assms(2) have ii:i={⟨0,0⟩} by auto
    then have ∀y. ∃x∈Arith.pred(domain(i)). x, y ∈ i → y=0
    by auto
  }
  moreover from assms(3) have eq:1 = ipred(domain(i)) unfolding Last_def
  by auto
  from iFun have domain(i) = i using domain_of_fun by auto
  then have pred(domain(i)) = 0 using pred_succ_eq by auto
  then have ipred(domain(i)) = i0 by auto
  with eq have 1 = i0 by auto

227
then have False using assms(2) by auto
}
then have y≠0 by auto
then obtain k where yy:y= succ(k) k∈nat using Nat_ZF_1_L3 y(2) by auto
with iList y(1) have iss:i:succ(succ(k)) → 2 unfolding NELists_def using domain_of_fun by auto
then have Init(i):succ(k) → 2 using init_props(1) nat_succI[OF yy(2)] by auto
then have Init(i)∈NELists(2) unfolding NELists_def using yy(2) by auto
moreover have 0∈succ(k) using empty_in_every_succ yy(2) by auto
with iss assms(2) have Init(i)0 = 0 using init_props(2)[of succ(k)]
i 2] nat_succI[OF yy(2)] by auto
ultimately have ⟨⟨Init(i),starts1⟩,⟨0,starts0⟩⟩∈ r{0.*1}^* using invariant_state_0_1
by auto
with U have ⟨⟨i,empty⟩,⟨0,starts0⟩⟩∈ r{0.*1}^* using rtrancl_into_trancl2[THEN trancl_into_rtrancl] by auto
then have 3∈finalStates. ⟨⟨i,empty⟩,⟨0,q⟩⟩ ∈ r{0.*1}^* by auto
then show thesis using DFSASatisfy_def[OF finA dfsaFrom0To1.DFSA assms(1)]
by auto
qed

Any list that reduces starts with 0 and ends in 1

theorem starts1ends0_DFSA_reduce_rev:
defines i as i∈Lists(2)
assumes i∈Lists(2) and i {reduces in 0.*1}
shows i0=0 and Last(i) = 1

proof-
have finA:Finite(2) using nat_into_Finite[of 2] by auto
have funT:transFun:states×2→states unfolding Pi_def function_def by auto
from assms(2) have ⟨⟨i,empty⟩,⟨0,starts0⟩⟩∈ r{0.*1}^*
unfolding DFSASatisfy_def[OF finA dfsaFrom0To1.DFSA assms(1)]
by auto
then have ⟨⟨i,empty⟩,⟨0,starts0⟩⟩∈ id(field(r{0.*1})) \ r{0.*1}^* using rtrancl_unfold[of r{0.*1}] by auto
then have k:⟨⟨i,empty⟩,⟨0,starts0⟩⟩∈ id(field(r{0.*1})) ∨ ⟨⟨i,empty⟩,⟨0,starts0⟩⟩ ∈ (r{0.*1} O r{0.*1}^*)
by auto
then have d:domain(r{0.*1}) = NELists(2)×states
using domainI[of _ _ r{0.*1}]
unfolding DFSASATuration_def[OF finA dfsaFrom0To1.DFSA]
by auto
from k have ⟨⟨i,empty⟩,⟨0,starts0⟩⟩ ∈ (r{0.*1} O r{0.*1}^*) using id_iff by auto
then have ⟨⟨i,empty⟩,⟨0,starts0⟩⟩ ∈ (r{0.*1} O r{0.*1}^*) using id_iff by auto
ultimately have ⟨⟨i,empty⟩,⟨0,starts0⟩⟩∈ r{0.*1} O r{0.*1}^* by auto

228
then obtain $q$ where $q : (q, \langle 0, \text{starts0} \rangle) \in r^{\bullet}$ unfolding $\text{comp_def}$ by auto
from $q(1)$ have $q \in \text{domain}(r^{\bullet})$ unfolding $\text{domain_def}$ by auto
with $d$ have $q \in \text{NELists}(2) \times \text{states}$ by auto
then obtain $q_1 q_2$ where $q : (q_1, q_2) \in \text{NELists}(2)$ $q_2 \in \text{states}$ by auto
from $q(1)$ $q(q(1))$ have $A : 0 = \text{Init}(q_1)$ $0 = \text{transFun}(q_2, \text{Last}(q_1))$
unfolding $\text{DFSAExecutionRelation_def}[\text{OF finA dfsaFrom0To1.DFSA}]$ by auto
from $q(q(1))$ $q(2)$ have $\text{restrict}(i, \text{domain}(q_1)) = q_1$
using $\text{dfsaFrom0To1.seq_is_restriction}$ by auto
from $q(q(1))$ $q(2)$ have $\text{restrict}(i, \text{domain}(q_1)) = q_1$ $0 = \text{transFun}(q_2, \text{Last}(q_1))$ unfolding $\text{domain_of_fun}$ by auto
then have $\text{iRes} : \text{restrict}(i, \text{domain}(q_1)) = q_1$ $0$ by auto
from $q(q(1))$ $q(2)$ obtain $k$ where $k : q_1 : \text{succ}(k) \to 2$ $k \in \text{nat}$ unfolding $\text{NELists_def}$ by auto
then have $\text{Init}(q_1) : k \to 2$ $\text{Last}(q_1) = 0$ using $\text{init_props}(1)$ by auto
with $A$ have $k = 0$ unfolding $\text{Pi_def}$ by auto
with $k(1)$ have $qfun : q_1 : 1 \to 2$ using $\text{apply_type}[\text{OF q_1 1 \_ \_ 2}]$ $\text{empty_in_every_succ}[\text{OF nat_0I}]$ by auto
with $qfun$ have $q_{10} : \text{Last}(q_1) = q_{10}$ unfolding $\text{Last_def}$ using $\text{domain_of_fun}$ by auto
from $k$ have $0 \in \text{domain}(q_1)$ using $\text{domain_of_fun}$ $\text{empty_in_every_succ}$ by auto
with $\text{iRes} : \text{restrict}(i, \text{domain}(q_1)) = q_1$ $0 = \text{transFun}(q_2, \text{Last}(q_1))$ unfolding $\text{domain_of_fun}$ $\text{empty_in_every_succ}$ by auto
then have $\text{iRes} : \text{restrict}(i, \text{domain}(q_1)) = q_1$ $0 = \text{transFun}(q_2, \text{Last}(q_1))$ unfolding $\text{domain_of_fun}$ by auto
from $k$ have $0 \in \text{domain}(q_1)$ using $\text{domain_of_fun}$ $\text{empty_in_every_succ}$ by auto
with $\text{iRes} : \text{restrict}(i, \text{domain}(q_1)) = q_1$ $0 = \text{transFun}(q_2, \text{Last}(q_1))$ unfolding $\text{domain_of_fun}$ $\text{empty_in_every_succ}$ by auto
then have $\text{iRes} : \text{restrict}(i, \text{domain}(q_1)) = q_1$ $0 = \text{transFun}(q_2, \text{Last}(q_1))$ unfolding $\text{domain_of_fun}$ by auto
moreover
\[
\begin{align*}
\text{assume } & \langle \langle i, 2 \rangle, \langle q_1, q_2 \rangle \rangle \in \text{id} (\text{field}(r^{\bullet})) \\
& \text{then have } i = q_1 \land q_2 = 2 \text{ by auto} \\
& \text{with } \langle 2 \rangle \text{ have False by auto}
\end{align*}
\]
ultimately
have $\langle \langle i, 2 \rangle, \langle q_1, q_2 \rangle \rangle \in (r^{\bullet})$ $0 \in (r^{\bullet})$ by auto
then obtain $z$ where $z : (\langle i, 2 \rangle, z) : r^{\bullet}$ unfolding $\text{comp_def}$ by auto
from $z(2)$ have $z \in \text{field}(r^{\bullet})$ using $\text{rtrancl_type}[\text{OF r^{\bullet}}]$ by auto
then obtain $z_1 z_2$ where $z : (z_1, z_2) z_1 \in \text{List}(2)$ $z_2 \in \text{states}$ using $\text{dfsaFrom0To1.reduce_field}$ by blast
from $z(1)$ $z(1)$ have $\text{zzz} : z_1 = \text{Init}(i)$ $z_2 = \text{transFun}(2, \text{Last}(i))$
unfolding $\text{DFSAExecutionRelation_def}[\text{OF finA dfsaFrom0To1.DFSA}]$ by auto
\[
\begin{align*}
\text{assume } & \text{Last}(i) = 0 \\
& \text{with } \text{zzz}(2) \text{ have } z_2 = \text{succ}(2) \text{ using } \text{apply_equality}[\text{OF _ funT}] \text{ by auto}
\end{align*}
\]
auto
  with zz(1) z(2) have q2 = succ(2) using invariant_state_3 by auto
  with `q2 ∈ 2` have False by auto
}
with z(1) show Last(i) = 1 using last_type[of i 2]
  unfolding DFSAEvaluationRelation_def[OF finA dfsaFrom0To1.DFSA]
by auto
qed

We conclude that this example constitutes the language of binary strings
starting in 0 and ending in 1

theorem determine_strings:
  shows dfsaFrom0To1.LanguageDFSA = \{i ∈ Lists(2). i0 = 0 ∧ Last(i) = 1\}
  using starts1ends0_DFSA_reduce_rev
  starts1ends0_DFSA_reduce by blast
end

We define the languages determined by a deterministic finite state automa-
ton as regular.

definition
  IsRegularLanguage (_, {is a regular language on}_) where
  Finite(\Sigma) ⇒ L{is a regular language on}\Sigma ≡ ∃ S s t F. ((S,s,t,F){is
an DFSA for alphabet}\Sigma) ∧ L=DetFinStateAuto.LanguageDFSA(S,s,t,F,\Sigma)

By definition, the language in the locale is regular.

corollary (in DetFinStateAuto) regular_intersect:
  shows LanguageDFSA{is a regular language on}\Sigma
  using IsRegularLanguage_def finite_alphabet DFSA
  by auto

A regular language is a language.

lemma regular_is_language:
  assumes Finite(\Sigma)
  and L{is a regular language on}\Sigma
  shows L{is a language with alphabet}\Sigma unfolding IsALanguage_def[OF
assms(1)]
proof
  from assms(2) obtain S s t F where (S,s,t,F){is an DFSA for alphabet}\Sigma
L=DetFinStateAuto.LanguageDFSA(S,s,t,F,\Sigma)
    unfolding IsRegularLanguage_def[OF assms(1)] by auto
  then show L ⊆ Lists(\Sigma)
    unfolding DetFinStateAuto_def using assms(1) by auto
qed

19.3 Operations on regular languages

The intersection of two regular languages is a regular language.

theorem regular_intersect:
assumes Finite(\(\Sigma\))
and \(L_1\) is a regular language on \(\Sigma\)
and \(L_2\) is a regular language on \(\Sigma\)
shows \((L_1 \cap L_2)\) is a regular language on \(\Sigma\)

proof-

from assms(1,2) obtain \(S_1\) s1 t1 F1 where \(L_1 = \text{DetFinStateAuto.LanguageDFSA}(S_1, s1, t1, F1, \Sigma)\)
using IsRegularLanguage_def by auto
then have \(L_1 = \text{DetFinStateAuto.LanguageDFSA}(S_1, s1, t1, F1, \Sigma)\)
using DetFinStateAuto_def assms(1) l1(1)
by auto

from assms(1,3) obtain \(S_2\) s2 t2 F2 where \(L_2 = \text{DetFinStateAuto.LanguageDFSA}(S_2, s2, t2, F2, \Sigma)\)
using IsRegularLanguage_def by auto
then have \(L_2 = \text{DetFinStateAuto.LanguageDFSA}(S_2, s2, t2, F2, \Sigma)\)
using DetFinStateAuto_def assms(1) l2(1)
by auto

let \(S = S_1 \times S_2\)
let \(s = (s_1, s_2)\)
let \(t = \{((x_1, x_2), y), (t_1(x_1, y), t_2(x_2, y))\}. (x_1, x_2), y \in S \times \Sigma\)
let \(F = F_1 \times F_2\)
let \(r = \text{DetFinStateAuto.r}_D(S, s, t, \Sigma)\)
let \(r_1 = \text{DetFinStateAuto.r}_D(S_1, s_1, t_1, \Sigma)\)
let \(r_2 = \text{DetFinStateAuto.r}_D(S_2, s_2, t_2, \Sigma)\)

have \(D : (S, s, t, F)\) is an DFSA for alphabet \(\Sigma\)
proof-

have A:s\(\in\)S using 11(1) 12(1) unfolding DFSA_def[of assms(1)] by auto
have B:F \(\subseteq\) S using 11(1) 12(1) unfolding DFSA_def[of assms(1)] by auto

have function(t) unfolding function_def by auto
moreover have \(S \times \Sigma \subseteq\) domain(t) by auto
moreover

\{ fix x1 x2 y assume as:((x1, x2), y) \(\in\) S \times \Sigma
then have t1(x1, y)\(\in\)S1 t2(x2, y)\(\in\)S2 using apply_type
11(1) 12(1) unfolding DFSA_def[of assms(1)] by auto
then have (t1(x1, y), t2(x2, y))\(\in\)S by auto
\}
then have \(\{((x_1, x_2), y), (t_1(x_1, y), t_2(x_2, y))\}. (x_1, x_2), y \in (S_1 \times S_2) \times \Sigma\) \(\in\) Pow\((S\times\Sigma)\times S\) by auto

ultimately have C:t:S\(\times\)\(\Sigma\) \(\rightarrow\) S unfolding Pi_def by auto
have Finite(S) using Finite1_L12[of S1 S2] Finite_into_Finite Finite_into_Fin
then have 11(1) 12(1) unfolding DFSA_def[of assms(1)] by auto

with A B C show thesis unfolding DFSA_def[of assms(1)] by auto

qed

then have DFS\(\alpha\):\(\text{DetFinStateAuto}(S, s, t, F, \Sigma)\) unfolding DetFinStateAuto_def using assms(1) by auto

231
have \( R \subseteq \{ m \in \text{NELLists}(\Sigma) \mid \langle \langle m, s_1, s_2 \rangle, 0, y, z \rangle \rangle : r^{-*} \Rightarrow \langle \langle m, s_1 \rangle, 0, y \rangle \rangle : r_1^{-*} \) ∧ \( \langle \langle m, s_2 \rangle, 0, z \rangle \rangle : r_2^{-*} \)

proof

fix \( m \), \( y \), \( z \).

assume \( \langle \langle m, s_1, s_2 \rangle, 0, y, z \rangle \rangle : r^{-*} \), \( m \in \text{NELLists}(\Sigma) \).

note \( \langle \rangle : r^{-*} \Rightarrow \langle \langle m, s_1 \rangle, \langle 0, y \rangle \rangle : r_1^{-*} \), \( \langle \langle m, s_2 \rangle, \langle 0, z \rangle \rangle : r_2^{-*} \).

proof

fix \( m \), \( y \), \( z \).

assume \( \langle \langle m, s_1, s_2 \rangle, 0, y, z \rangle \rangle : r^{-*} \), \( m \in \text{NELLists}(\Sigma) \).

note \( \langle \rangle : r^{-*} \Rightarrow \langle \langle m, s_1 \rangle, \langle 0, y \rangle \rangle : r_1^{-*} \).

ultimately have \( \langle \langle m, s_1 \rangle, \langle 0, y \rangle \rangle \rangle : r_1^{-*} \) and \( \langle \langle m, s_2 \rangle, \langle 0, z \rangle \rangle \rangle : r_2^{-*} \) using \( r_1^{-*} \).

moreover have \( \langle \langle m, s_1 \rangle, \langle 0, y \rangle \rangle \rangle : r_1^{-*} \) and \( \langle \langle m, s_2 \rangle, \langle 0, z \rangle \rangle \rangle : r_2^{-*} \) using \( r_1^{-*} \).

ultimately show \( \langle \langle m, s_1 \rangle, 0, y \rangle \rangle : r_1^{-*} \) and \( \langle \langle m, s_2 \rangle, 0, z \rangle \rangle : r_2^{-*} \) by auto.

qed
then have \( M : \text{c} \subseteq \text{Lists}(\Sigma) \) \( m \leftarrow D (S,s,t,F) \{ \text{in alphabet} \} \Sigma \) by auto
{
  assume \( m_0 : m=0 \)
  from \( m_0 \) \( M \) have \( 0 \leftarrow D (S,s,t,F) \{ \text{in alphabet} \} \Sigma \) by auto
  then obtain \( yy \), \( zz \) where \( \langle \langle 0, s_1, s_2 \rangle, \langle 0, yy, zz \rangle \rangle : r^* \vee s : F \langle yy, zz \rangle : F \)
  using \( m_0 \)
  DFSASatisfy_def[\{ OF assms(1) D M(1) \}] by auto
moreover
{
  fix \( y \) \( z \)
  assume \( \langle \langle 0, s_1, s_2 \rangle, y \rangle \in r^* \langle y, z \rangle \in r \) \( y = \langle 0, s_1, s_2 \rangle \)
  from this(2,3) have \( 0 \in \text{NELists}(\Sigma) \) unfolding DFSExecutionRelation_def[\{ OF assms(1) D \}] by auto
  ultimately have \( s : F \)
    using \( \lambda q. q = \langle 0, s \rangle \) by auto
  then have \( m \leftarrow D (S_1,s_1,t_1,F_1) \{ \text{in alphabet} \} \Sigma \) \( m \leftarrow D (S_2,s_2,t_2,F_2) \{ \text{in alphabet} \} \Sigma \)
    using DFSASatisfy_def[\{ OF assms(1) l1(1) M(1) \}] DFSASatisfy_def[\{ OF assms(1) l2(1) M(1) \}]
    using \( m_0 \) by auto
}
ultimately have \( m \leftarrow D (S_1,s_1,t_1,F_1) \{ \text{in alphabet} \} \Sigma \) \( m \leftarrow D (S_2,s_2,t_2,F_2) \{ \text{in alphabet} \} \Sigma \)
  using DFSASatisfy_def[\{ OF assms(1) D M(1) \}]
  by auto

assume \( m_0 : m \neq 0 \)
with \( M(2) \) obtain \( yy \), \( zz \) where \( F : \langle \langle m, s_1, s_2 \rangle, \langle 0, yy, zz \rangle \rangle : r^* \langle yy, zz \rangle : F \)

using \( \text{DFSASatisfy_def}\{ OF assms(1) D M(1) \} \) by auto
from \( m_0 \) \( M(1) \) have \( m \in \text{NELists}(\Sigma) \) using non_zero_List_func_is_NEList
by auto
  then have \( RR : \forall yy \), \( zz \). \( \langle \langle m, s_1, s_2 \rangle, \langle 0, yy, zz \rangle \rangle : r^* \Longrightarrow \langle \langle m, s_1 \rangle, \langle 0, yy \rangle \rangle : r_1^* \)
  \( \land \langle \langle m, s_2 \rangle, \langle 0, zz \rangle \rangle : r_2^* \)
  using \( RR \) by auto
  with \( F \) have \( \exists q \in F_1. \langle \langle m, s_1 \rangle, 0, q \rangle \in r_1^* \) \( \exists q \in F_2. \langle \langle m, s_2 \rangle, 0, q \rangle \in r_2^* \)
  by auto
  then have \( m \leftarrow D (S_1,s_1,t_1,F_1) \{ \text{in alphabet} \} \Sigma \) \( m \leftarrow D (S_2,s_2,t_2,F_2) \{ \text{in alphabet} \} \Sigma \)
    using DFSASatisfy_def[\{ OF assms(1) l1(1) M(1) \}] DFSASatisfy_def[\{ OF assms(1) l2(1) M(1) \}]
    using \( m_0 \) by auto
} ultimately have \( m \leftarrow D (S_1,s_1,t_1,F_1) \{ \text{in alphabet} \} \Sigma \) \( m \leftarrow D (S_2,s_2,t_2,F_2) \{ \text{in alphabet} \} \Sigma \)
  by auto
then have \( S_1 : \{ i \in \text{Lists}(\Sigma). i \leftarrow D (S,s,t,F) \{ \text{in alphabet} \} \Sigma \} \subseteq L_1 \cap L_2 \)
using \( l_1(2) \) \( l_2(2) \) by auto
{
  fix \( m \) assume \( m \in L_1 \cap L_2 \)
}

233
with \( l_1(2) \) \( l_2(2) \) have \( M : m \in Lists(\Sigma) \) \( m \leftarrow \mathcal{D} \{ (S_1, s_1, t_1, F_1) \} \{ \text{in alphabet} \} \Sigma \)

by auto

then obtain \( f_1 \) \( f_2 \) where \( \forall f_1 \in F_1 \) \( f_2 \in F_2 \) \( \langle \langle m, s_1 \rangle, 0, f_1 \rangle \in r_1^* \land (m = 0 \land s_1 \in F_1) \langle \langle m, s_2 \rangle, 0, f_2 \rangle \in r_2^* \land (m = 0 \land s_2 \in F_2) \)

unfolding DFSASatisfy_def[OF assms(1) l1(1) M(1)] DFSASatisfy_def[OF assms(1) l2(1) M(1)]

by auto

\{$\text{fix } y \ z$

\begin{align*}
\text{assume } & \langle \langle 0, s_1 \rangle, y \rangle \in r_1^* \langle y, z \rangle \in r_1 \ y = \langle 0, s_1 \rangle \\
\text{from this(2,3) have } & 0 \in \text{NELists}(\Sigma) \text{ unfolding DFSExecutionRelation_def[OF assms(1) l1(1)]} \\
\text{by auto}
\end{align*}

\}

with \( ff(1, 3) \) have \( m = 0 \rightarrow s_1 \in F_1 \)

r_1 \ \lambda q. q = \langle 0, s_1 \rangle 

by auto

moreover

\{$\text{fix } y \ z$

\begin{align*}
\text{assume } & \langle \langle 0, s_2 \rangle, y \rangle \in r_2^* \langle y, z \rangle \in r_2 \ y = \langle 0, s_2 \rangle \\
\text{from this(2,3) have } & 0 \in \text{NELists}(\Sigma) \text{ unfolding DFSExecutionRelation_def[OF assms(1) l2(1)]} \\
\text{by auto}
\end{align*}

\}

with \( ff(2, 4) \) have \( m = 0 \rightarrow s_2 \in F_2 \)

r_2 \ \lambda q. q = \langle 0, s_2 \rangle 

by auto

moreover

\{$\text{assume } m \neq 0$

\begin{align*}
\text{with } & \langle \langle m, s_1 \rangle, 0, f_1 \rangle : \langle \langle m, s_2 \rangle, 0, f_2 \rangle \in r_1^* \land r_2^* \text{ by auto} \\
\text{from this(1) have } & \langle m, s_1 \rangle : \text{field}(r_1) \text{ using rtrancl_type by auto} \\
\text{then have } & (m, s_1) : \text{Lists}(\Sigma) \times S_1 \text{ using DetFinStateAuto.reduce_field(1)[OF S1 s1 t1 F1 \Sigma] 11(1) unfolding DetFinStateAuto_def} \\
\text{using assms(1) by auto}
\end{align*}

\}

with \( m \in \text{NELists}(\Sigma) \) \( s_1 \in S_1 \) \( s_2 \in S_2 \) using 11(1) 12(1) non_zero_List_func_is_NEList unfolding DFSA_def[OF assms(1)] by auto

then have \( (m, s_1, s_2) : \text{NELists}(\Sigma) \times S \) by auto

then have \( (m, s_1, s_2) : \text{field}(r) \text{ using DetFinStateAuto.reduce_field(2)[OF DFSAO] by auto} \\
\text{then have } & K : \langle \langle m, s_1, s_2 \rangle, \langle m, s_1, s_2 \rangle \rangle : r^* \text{ using rtrancl_refl by auto} \\
\text{with } & s_2 \in S_2 \text{ have } \exists f_2 \in S_2. \langle \langle m, s_1, s_2 \rangle, \langle m, s_1, f_2 \rangle \rangle : r^* \text{ by auto moreover}
\}

\{$\text{fix } y \ z$

\begin{align*}
\text{assume as: } & \langle \langle m, s_1 \rangle, y \rangle : r_1^* \langle y, z \rangle : r_1 \ \exists f_2 \in S_2. \langle \langle m, s_1, s_2 \rangle, \langle \text{fst}(y), \text{snd}(y), f_2 \rangle \rangle : r^*
\end{align*}

\}
from as(2) obtain \( yL \ y1 \) where \( y:y=(yL,y1) \) \( z=(\text{Init}(yL),t1(y1,\text{Last}(yL))) \)
\[ yL\in \text{NELists}(\Sigma) \] unfolding \text{DFSAExecutionRelation_def[OF }\text{assms(1)} \text{l1(1)}\] by auto
with as(3) obtain \( ff2 \) where \( sf:ff2\in S2 \) \( \langle \langle m,s1,s2\rangle,\langle yL,y1,ff2\rangle \rangle : r^* \)
by auto
from this(1) \( y(3,4) \) have \( \langle \langle yL,y1,ff2\rangle,\langle \text{Init}(yL),t(\langle y1,ff2\rangle,\text{Last}(yL))\rangle \rangle \)
\( \in r \)
unfolding \text{DFSAExecutionRelation_def[OF }\text{assms(1)} \text{D}\] by auto moreover
have \( \text{fun}\ t:t:S\times\Sigma \rightarrow S \) using \( D \) unfolding \text{DFSA_def[OF }\text{assms(1)}\]
by auto
with \( y(3,4) \) \( sf(1) \) have \( t(\langle y1,ff2\rangle,\text{Last}(yL)) = (t1(y1,\text{Last}(yL)),t2(ff2,\text{Last}(yL))) \)
using \text{apply\_equality[of }\langle \langle y1,ff2\rangle,\text{Last}(yL)\rangle \_ t S\times\Sigma \lambda_. \_ S\text{]} last\_type
by auto
ultimately have \( \langle \langle yL,y1,ff2\rangle,\langle \text{Init}(yL),t1(y1,\text{Last}(yL)),t2(ff2,\text{Last}(yL))\rangle \rangle \)
\( \in r \) by auto
with \( y(2) \) have \( \langle \langle yL,y1,ff2\rangle,\langle \text{fst}(z),(\text{snd}(z),t2(ff2,\text{Last}(yL)))\rangle \rangle \)
\( \in r \) by auto
with \( sf(2) \) have \( \langle \langle m,s1,s2\rangle,\langle \text{fst}(z),(\text{snd}(z),t2(ff2,\text{Last}(yL)))\rangle \rangle : r^* \)
using \text{rtrancl\_into\_rtrancl by auto}
moreover from \( sf(1) \) have \( t2(ff2,\text{Last}(yL)) \in S2 \) using \( l2(1) \)
apply\_type[of \( t2 S2\times\Sigma \lambda_. \_ S2\text{]} last\_type[OF }y(3)\] unfolding \text{DFSA\_def[OF }\text{assms(1)}\]
by auto
ultimately have \( \exists f2\in S2. \langle \langle m,s1,s2\rangle,\langle \text{fst}(z),(\text{snd}(z),f2)\rangle \rangle : r^* \)
using \( sf(1) \) by auto
} moreover note \( A(1) \)
ultimately have \( \exists f2\in S2. \langle \langle m,s1,s2\rangle,\langle 0,(f1,f2)\rangle \rangle : r^* \)
using \text{rtrancl\_induct[of }\langle \langle m,s1\rangle,\langle 0,f1\rangle \rangle \_ r1 \_ \lambda q. \exists f2\in S2. \langle \langle m,s1,s2\rangle,\langle \text{fst}(q),(\text{snd}(q),f2)\rangle \rangle : r^* \]
by auto
then obtain \( uu \) where \( uu:uu\in S2 \) \( \langle \langle m,s1,s2\rangle,\langle 0,(f1,uu)\rangle \rangle : r^* \) by auto
from \( K \ ` s1\in S1` \ have \( \exists f1\in S1. \langle \langle m,s1,s2\rangle,\langle f1,f1,s2\rangle \rangle : r^* \) by auto moreover
\{ \fix \ y \ z 
assume as: \( \langle \langle m,s2\rangle,\langle y,z\rangle \rangle : r^* \) \( y,z: r2 \ \exists f1\in S1. \langle \langle m,s1,s2\rangle,\langle \text{fst}(y),(f1,\text{snd}(y))\rangle \rangle : r^* \)
from as(2) obtain \( yL \ y1 \) where \( y:y=(yL,y1) \) \( z=(\text{Init}(yL),t2(y1,\text{Last}(yL))) \)
\( yL\in \text{NELists}(\Sigma) \] unfolding \text{DFSAExecutionRelation_def[OF }\text{assms(1)} \text{l2(1)}\] by auto
with as(3) obtain \( ff2 \) where \( sf:ff2\in S1 \) \( \langle \langle m,s1,s2\rangle,\langle yL,ff2,y1\rangle \rangle : r^* \)
by auto
from this(1) \( y(3,4) \) have \( \langle \langle yL,ff2,y1\rangle,\langle \text{Init}(yL),t(\langle ff2,y1\rangle,\text{Last}(yL))\rangle \rangle \)
\( \in r \)
unfolding \text{DFSAExecutionRelation_def[OF }\text{assms(1)} \text{D}\] by auto moreover
have \( \text{fun}\ t:t:S\times\Sigma \rightarrow S \) using \( D \) unfolding \text{DFSA\_def[OF }\text{assms(1)}\]
by auto
with \( y(3,4) \) \( sf(1) \) have \( t(\langle ff2,y1\rangle,\text{Last}(yL)) = (t1(ff2,\text{Last}(yL)),t2(y1,\text{Last}(yL))) \)

235
ultimately have \( \langle\langle \text{Init}(yL), \langle t1(\text{ff2}, \text{Last}(yL)), t2(y1, \text{Last}(yL)) \rangle \rangle \rangle \in r \) by auto

moreover from sf(1) have \( t1(\text{ff2}, \text{Last}(yL)) \in S1 \) using l1(1)

apply_type[of t1 S1 \times \Sigma \lambda_. S1] last_type[of y(3)] unfolding DFSA_def[of assms(1)] by auto

ultimately have \( \exists f2 \in S1. \langle\langle m, s1, s2, \langle \text{fst}(z), f2, \text{snd}(z) \rangle \rangle \rangle : r^* \) using sf(1) by auto

moreover note A(2)

ultimately have \( \exists f1 \in S1. \langle\langle m, s1, s2, \langle 0, \langle f1, f2 \rangle \rangle \rangle : r^* \) using rtrancl_induct[of \( m, s2 \) \langle 0, f2 \rangle r2 \lambda q. \exists f1 \in S1. \langle\langle m, s1, s2, \langle \text{fst}(q), f1, \text{snd}(q) \rangle \rangle : r^* \rangle by auto

then obtain vv where vv:vv \in S1 by auto

from uu(2) vv(2) have f1=vv uu=f2 using DetFinStateAuto.relation_deteministic[of] by auto

from this(1) vv(2) ff(1,2) have m <-D (S,s,t,F){in alphabet}\Sigma unfolding DFSASatisfy_def[of assms(1) D M(1)] by auto

ultimately have m <-D (S,s,t,F){in alphabet}\Sigma unfolding DFSASatisfy_def[of assms(1) D M(1)] by auto

with l1(2) l2(2) have \( L1 \cap L2 \subseteq \{i \in \text{Lists}(\Sigma). i <-D (S,s,t,F)\{\text{in alphabet}\}\Sigma \} \) by auto

ultimately have m <-D (S,s,t,F){in alphabet}\Sigma unfolding DFSASatisfy_def[of assms(1) D M(1)] by auto

with l1(2) l2(2) have \( L1 \cap L2 \subseteq \{i \in \text{Lists}(\Sigma). i <-D (S,s,t,F)\{\text{in alphabet}\}\Sigma \} \) by auto

with S1 have \( \{i \in \text{Lists}(\Sigma). i <-D (S,s,t,F)\{\text{in alphabet}\}\Sigma \} = L1 \cap L2 \) by auto

then have L1\cap L2 = DetFinStateAuto.LanguageDFSA(S,s,t,F,\Sigma) by auto

with D have \( \exists S \ s \ t \ F. \langle (S,s,t,F)\{\text{is an DFSA for alphabet}\}\Sigma \rangle \land L1 \cap L2 = \text{DetFinStateAuto.LanguageDFSA}(S,s,t,h,\Sigma) \) unfolding exI[of \( \lambda h. \langle (S,s,m,h)\{\text{is an DFSA for alphabet}\}\Sigma \rangle \land L1 \cap L2 = \text{DetFinStateAuto.LanguageDFSA}(S,s,m,h,\Sigma) t \)]

using exI[of \( \lambda h. \langle (S,s,m,h)\{\text{is an DFSA for alphabet}\}\Sigma \rangle \land L1 \cap L2 = \text{DetFinStateAuto.LanguageDFSA}(S,s,m,h,\Sigma) s \)]

using exI[of \( \lambda p. \exists n \ m. \langle (p,n,m,h)\{\text{is an DFSA for alphabet}\}\Sigma \rangle \land L1 \cap L2 = \text{DetFinStateAuto.LanguageDFSA}(p,n,m,h,\Sigma) S \)]

by auto

with assms(2,3) show thesis unfolding IsRegularLanguage_def[of assms(1)] IsALanguage_def[of assms(1)] by auto
done

The complement of a regular language is a regular language.
theorem regular_opp:
  assumes Finite(Σ)
  and L{is a regular language on}Σ
  shows (Lists(Σ)-L) {is a regular language on}Σ
proof-
  from assms(1,2) obtain S s t F where
    l:(S,s,t,F){is an DFSA for alphabet}Σ
  L=DetFinStateAuto.LanguageDFSA(S,s,t,F,Σ) unfolding IsRegularLanguage_def[OF assms(1)]
  by auto
  then have l:(S,s,t,F){is an DFSA for alphabet}Σ
  L=DetFinStateAuto.LanguageDFSA(S,s,t,F,Σ) unfolding IsRegularLanguage_def[OF assms(1)]
  by auto
  let F = S-F
  let r = DetFinStateAuto.rD(S,s,t,Σ)
  from l(1) have D:(S,s,t,F){is an DFSA for alphabet}Σ unfolding DFSA_def[OF assms(1)]
  by auto
  with assms(1) have D0:DetFinStateAuto(S,s,t,F,Σ)
  unfolding DetFinStateAuto_def
  by auto
  { fix m assume m∈{i∈Lists(Σ). i <-D (S,s,t,F){in alphabet}Σ}
    then have M:m ∈ Lists(Σ) m <-D (S,s,t,F){in alphabet}Σ by auto
    assume m∈L
    with l(2) have MM:m <-D (S,s,t,F){in alphabet}Σ by auto
    assume as:m=0
    from MM(1) as(1) obtain yy where
      ⟦⟨0,s⟩,⟨0,yy⟩⟧:r^* ∨ s:F yy∈F
    unfolding DFSASatisfy_def[OF assms(1) l(1) M(1)] by auto
    moreover
    { fix y z
      assume ⟨⟨0,s⟩,y⟩ ∈ r^* ⟨y,z⟩ ∈ r y = ⟨0,s⟩
      from this(2,3) have 0∈NELists(Σ) unfolding DFSAExecutionRelation_def[OF assms(1) l(1)]
      by auto
      then have False unfolding NELists_def Pi_def by auto
    }
    ultimately have sf:s:F using rtrancl_induct[of ⟨0,s⟩ ⟨0,yy⟩ r λq. q=⟨0,s⟩] by auto
    from M(2) as(1) obtain yy where
      ⟨⟨0,s⟩,⟨0,yy⟩⟩:r^* ∨ s:S-F yy∈S-F
    unfolding DFSASatisfy_def[OF assms(1) D M(1)] by auto
    moreover
    { fix y z
      assumption
    }
  }

237
\textbf{assume \((0,s),y\) ∈ \(r^* y \in r y = \langle 0,s \rangle\)}

\textbf{from this(2,3) have 0∈\textsc{NELists}(\(\Sigma\)) unfolding \textsc{DFSAExecutionRelation_def[OF \textbf{assms(1) \(D\)]}}

\textbf{by auto}

\textbf{then have \(\textbf{False}\) unfolding \textsc{NELists_def \(\Pi\) def} by auto}

\textbf{then have \(z=\langle 0,s \rangle\) by auto}

\textbf{ultimately have \(s:S-F\) using \textsc{rtrancl_induct[of \(0,s\) \(0,yy\) \(r \lambda q.\)}}}

\textbf{q=\(0,s\)\] by auto}

\textbf{with \(sf\) have \(\textbf{False}\) by auto}

\textbf{then have \(m0:m\neq0\) by auto}

\textbf{with \(MM\) obtain \(q1\) where \(q1:q1\in F \langle\langle m, s\rangle, 0, q1\rangle \in r^*\)}

\textbf{unfolding \textsc{DFSAccSim} def[OF \textbf{assms(1) \(1(1) \(M(1)\)]}} by auto

\textbf{from \(m0\) \(M(2)\) obtain \(q2\) where \(q2:q2\in S-F \langle\langle m, s\rangle, 0, q2\rangle \in r^*\)}

\textbf{unfolding \textsc{DFSAccSim} def[OF \textbf{assms(1) \(D \(M(1)\)]}} by auto

\textbf{from \(q1(2)\) \(q2(2)\) have \(q1=q2\) using \textsc{DetFinStateAuto.relation_deterministic[OF \textbf{D0}, \(of \(m\) \(s\) \(0\)] by auto}

\textbf{with \(q1(1)\) \(q2(1)\) have \(\textbf{False}\) by auto}

\textbf{then have \(m\in\textsc{Lists}(\(\Sigma\)) - L\) using \(M(1)\) by auto}

\textbf{then have \(S:\{i \in \textsc{Lists}(\(\Sigma\)) . i \lessdot D \(S,s,t,S-F\){in alphabet}\(\) \(\Sigma = \Rightarrow \textbf{False}\) using \(\lambda(2)\) by auto}

\textbf{\(\{\textbf{fix} m \textbf{assume} m\in\textsc{Lists}(\(\Sigma\))-L\)

\textbf{then have \(m:m\in\textsc{Lists}(\(\Sigma\)) \lessdot D \(S,s,t,F\){in alphabet} \(\Sigma = \Rightarrow \textbf{False}\) using \(\lambda(2)\) by auto}

\textbf{\(\} \) moreover

\textbf{\(\{\textbf{assume} as:m=0 s\in F\)

\textbf{with \(m(1)\) have \(m \lessdot D \(S,s,t,F\){in alphabet} \(\Sigma = \textsc{unfolding\textsc{DFSAccSim} def[OF \textbf{assms(1) \(1(1) \(m(1)\)]}} by auto

\textbf{with \(m(2)\) have \(\textbf{False}\) by auto}

\textbf{then have \(m\in\{i \in \textsc{Lists}(\(\Sigma\)) . i \lessdot D \(S,s,t,S-F\){in alphabet}\(\) \(\Sigma\}\} by auto

\textbf{\(}\) by auto

\textbf{\(\}\) by auto

\textbf{\(238\)
The union of two regular languages is a regular language.

Theorem regular_union:
assumes Finite($\Sigma$)
and L1{is a regular language on}$\Sigma$
and L2{is a regular language on}$\Sigma$
shows (L1 $\cup$ L2) {is a regular language on}$\Sigma$
proof-
have L1$\cup$L2 = Lists($\Sigma$) - ((Lists($\Sigma$) - L1) $\cap$ (Lists($\Sigma$) - L2)) using regular_is_language[of assms(1)]
  assms(2,3) unfolding IsALanguage_def[of assms(1)] by auto
moreover
  have A:(Lists($\Sigma$) - L1) {is a regular language on}$\Sigma$ using regular_opp[of assms(1,2)].
  have B:(Lists($\Sigma$) - L2) {is a regular language on}$\Sigma$ using regular_opp[of assms(1,3)].
  from A B have ((Lists($\Sigma$) - L1) $\cap$ (Lists($\Sigma$) - L2)) {is a regular language on}$\Sigma$ using regular_intersect[of assms(1)] by auto
  then have (Lists($\Sigma$) - (((Lists($\Sigma$) - L1) $\cap$ (Lists($\Sigma$) - L2)))) {is a regular language on}$\Sigma$ using regular_opp[of assms(1)] by auto
qed
ultimately show thesis by auto
qed

Another natural operation on words is concatenation, hence we can defined the concatenated language as the set of concatenations of words of one language with words of another.

definition concat where
L1 {is a language with alphabet}$\Sigma$ $\Rightarrow$ L2 {is a language with alphabet}$\Sigma$
$\Rightarrow$ concat(L1,L2) = {Concat(w1,w2). (w1,w2)$\in$L1$\times$L2}

The result of concatenating two languages is a language.

lemma concat_language:
assumes Finite($\Sigma$)
and L1 {is a language with alphabet}$\Sigma$
and L2 {is a language with alphabet}$\Sigma$
shows concat(L1,L2) {is a language with alphabet}$\Sigma$
proof-
{ fix w assume w$\in$concat(L1,L2)
then obtain w1 w2 where w:w=Concat(w1,w2) w1$\in$L1 w2$\in$L2 unfolding concat_def[OF assms(2,3)]
by auto
from this(2,3) assms(2,3) obtain n1 n2 where n1$\in$nat n2$\in$nat w1:n1$\rightarrow$$\Sigma$
w2:n2$\rightarrow$$\Sigma$
unfolding IsALanguage_def[OF assms(1)] Lists_def by blast
then have Concat(w1,w2):n1#+n2 $\rightarrow$$\Sigma$ n1#*n2 $\in$nat using concat_props(1)
by auto
with w(1) have w$\in$Lists($\Sigma$) unfolding Lists_def by auto
} then show thesis unfolding IsALanguage_def[OF assms(1)] by auto
qed

19.4 Non-deterministic finite state automata

We have reached a point where it is not easy to realize a concatenated language of two regular languages as a regular language. Nevertheless, if we extend our instruments to allow non-determinism it is much easier.

The cost, a priori, is that our class of languages would be larger since our automata are more generic.

The non-determinism is introduced by allowing the transition function to return not just a state, but more than one or even none.

definition NFSA ('(_,_,_,_') {is an NFSA for alphabet} _) where
Finite($\Sigma$) $\Rightarrow$ (S,s0,t,F){is an NFSA for alphabet}$\Sigma$ $\equiv$ Finite(S) $\land$ s0$\in$S $\land$ F$\subseteq$S $\land$ t:S$\times$$\Sigma$$\rightarrow$Pow(S)
The transition relation is then realized by considering all possible steps the transition function returns.

**definition**

\[ \text{NFSAExecutionRelation} \left( \text{reduce N-relation} \ '(_\ldots_\ldots\)' \{in alphabet\}_ \right) \]

**where**

\[ \text{Finite}(\Sigma) \implies (S, s_0, t, F)\{\text{is an NFSA for alphabet}\}_ \implies \{\text{reduce N-relation}\}(S, s_0, t)\{\text{in alphabet}\}_ \equiv \{(w, Q), \{\text{Init}(w)\}, \{t(s, Last(w)). s \in Q\}\}. (w, Q) \in \text{NELists}(\Sigma) \times \text{Pow}(S) \]

The full reduction is conceived as one of those possible paths reaching a final state.

**definition**

\[ \text{NFSASatisfy} \left( _\ldots_\ldots_\ldots, \text{\{is an NFSA for alphabet\}_} \right) \]

**where**

\[ \text{Finite}(\Sigma) \implies (S, s_0, t, F)\{\text{is an NFSA for alphabet}\}_ \implies i \in \text{Lists}(\Sigma) \implies i \text{\{reduce N-relation\}}(S, s_0, t)\{\text{in alphabet}\}_ \equiv (\exists q \in \text{Pow}(S). (q \cap F \neq \emptyset \land (i, (s_0), (0, q)) \in (\{\text{reduce N-relation}\}(S, s_0, t)\{\text{in alphabet}\}_)^*) \lor (i = 0 \land s_0 \in F) \]

An extra generalization can be consider if we allow the transition relation to go forward without consuming elements from the word. This is implemented as allowing \( \Sigma \) to symbolize an step without the word being touched. We might call it a \( \Sigma \) transition or a \( \varepsilon \)-transition.

**definition**

\[ \text{FullNFSA} \left( _\ldots_\ldots_\ldots, \text{\{is an \varepsilon-NFSA for alphabet\}_} \right) \]

**where**

\[ \text{Finite}(\Sigma) \implies (S, s_0, t, F)\{\text{is an \varepsilon-NFSA for alphabet}\}_ \equiv \text{Finite}(S) \land s_0 \in S \land F \subseteq S \land t : S \times \text{succ}(\Sigma) \rightarrow \text{Pow}(S) \]

The closure of a set of states can then be viewed as all the states reachable from that set with a transition of type \( \Sigma \).

**definition**

\[ \text{EpsilonClosure} \left( \varepsilon\text{-cl} \right) \]

**where**

\[ \text{Finite}(\Sigma) \implies (S, s_0, t, F)\{\text{is an \varepsilon-NFSA for alphabet}\}_ \implies E \subseteq S \implies \varepsilon\text{-cl}(S, t, \Sigma, E) \equiv \bigcup \{P \in \text{Pow}(S). \langle E, P \rangle \in (\{Q, \exists q \in Q. t(q, \Sigma) = s\}). Q \in \text{Pow}(S)\}^* \}

The reduction relation is then extended by considering any such transitions.

**definition**

\[ \text{FullNFSAExecutionRelation} \left( \{\text{reduce \varepsilon-N-relation} \ '(_\ldots_\ldots\)' \{in alphabet\}_ \right) \]

**where**

\[ \text{Finite}(\Sigma) \implies (S, s_0, t, F)\{\text{is an \varepsilon-NFSA for alphabet}\}_ \implies \{\text{reduce \varepsilon-N-relation}\}(S, s_0, t)\{\text{in alphabet}\}_ \equiv \{(w, Q), \{\text{Init}(w), \varepsilon\text{-cl}(S, t, \Sigma, \bigcup \{t(s, Last(w)). s \in Q\})\}. (w, Q) \in \text{NELists}(\Sigma) \times \text{Pow}(S) \}

The full reduction of a word is similar to that of the automata without \( \varepsilon \)-transitions.

**definition**

\[ \text{FullNFSASatisfy} \left( _\ldots_\ldots_\ldots, \text{\{is an \varepsilon-NFSA for alphabet\}_} \right) \]

**where**

\[ \text{Finite}(\Sigma) \implies (S, s_0, t, F)\{\text{is an \varepsilon-NFSA for alphabet}\}_ \implies i \in \text{Lists}(\Sigma) \implies \]

\[ i \neq 0 \land s_0 \in F \]
Finite(\Sigma) \implies (S,s_0,t,F) \{\text{is an } \varepsilon\text{-NFSA for alphabet}\} \Sigma \implies i \in \text{Lists}(\Sigma)
\implies
i \leftarrow \varepsilon\text{-N} (S,s_0,t,F) \{\text{in alphabet}\} \Sigma \equiv (\exists q \in \text{Pow}(S). (q \cap F \neq 0 \land \langle \langle i,\{s_0\}\rangle,\{0,q\}\rangle \in (\{\text{reduce } \varepsilon\text{-N-relation}\}(S,s_0,t)\{\text{in alphabet}\} \Sigma)^*) \lor (i = 0 \land s_0 \in F)

We define a locale to create some notation:

locale NonDetFinStateAuto =
fixes S and s_0 and t and F and \Sigma
assumes finite_alphabet: Finite(\Sigma)
assumes NFSA: (S,s_0,t,F) \{\text{is an NFSA for alphabet}\} \Sigma

Notation for the transition relation:
abbreviation (in NonDetFinStateAuto) nd_rel (r_N) where
r_N \equiv \{\text{reduce N-relation}\}(S,s_0,t)\{\text{in alphabet}\} \Sigma

Notation for the language generated by the non-deterministic automaton:
abbreviation (in NonDetFinStateAuto) LanguageNFSA where
LanguageNFSA \equiv \{i \in \text{Lists}(\Sigma). i \leftarrow \varepsilon\text{-N} (S,s_0,t,F)\{\text{in alphabet}\} \Sigma\}

19.5 Equivalence of Non-deterministic and Deterministic Finite State Automata

We will show that the non-deterministic automata generate languages that are regular in the sense that there is a deterministic automaton that generates the same language.

The transition function of the deterministic automata we will construct:
definition (in NonDetFinStateAuto) tPow where
tPow \equiv \{(U,u), (\bigcup v \in U. t \langle v, u \rangle). \langle U,u \rangle \in \text{Pow}(S) \times \Sigma\}

The transition relation of the deterministic automata we will construct:
definition (in NonDetFinStateAuto) rPow where
rPow \equiv \text{DetFinStateAuto}.r_D(\text{Pow}(S),\{s_0\},tPow,\Sigma)

We show that we do have a deterministic automaton:
sublocale NonDetFinStateAuto < dfsa:DetFinStateAuto Pow(S) \{s_0\} tPow \{Q \in \text{Pow}(S). Q \cap F \neq 0\} \Sigma
 unfolding DetFinStateAuto_def DFSA_def[OF finite_alphabet] unfolding tPow_def
 apply safe using finite_alphabet apply simp
 using NFSA unfolding NFSA_def[OF finite_alphabet]
 apply simp using NFSA unfolding NFSA_def[OF finite_alphabet]
 apply simp unfolding Pi_def function_def apply auto
proof-
fix b y x v assume as:y \in \Sigma \ b \subseteq S \ v \in b \ x \in t \ \langle v, y \rangle
from as(2,3) have v:v \in S by auto
have \( t \in S \times \Sigma \to \text{Pow}(S) \) using NFSA

unfolding NFSA_def[OF finite_alphabet] by auto

with as(1,4) v show \( x \in S \) using apply_type[of \( t \times \Sigma \lambda_. \text{Pow}(S) \langle \cdot, y \rangle \)] by auto

qed

The two automata have the same relations associated with them.

First, we show that if the non-deterministic automaton produces a reduction step to a word, then the deterministic one we constructed does the same reduction step.

lemma (in NonDetFinStateAuto) nd_impl_det:
assumes \( \langle \langle w, Q \rangle, \langle u, G \rangle \rangle \in r_N \)
shows \( \langle \langle w, Q \rangle, \langle u, G \rangle \rangle \in r_{\text{Pow}} \)
proof-
from assms have \( w : w \in \text{NELists}(\Sigma) \ u = \text{Init}(w) \ Q \in \text{Pow}(S) \ G = (\bigcup s \in Q. \ t\langle s, \text{Last}(w) \rangle) \)
unfolding NFSAExecutionRelation_def[OF finite_alphabet NFSA] by auto
then have \( t_{\text{Pow}}(Q, \text{Last}(w)) = (\bigcup s \in Q. \ t\langle s, \text{Last}(w) \rangle) \Longrightarrow \text{thesis} \)
unfolding DFSAExecutionRelation_def[OF finite_alphabet dfsa.DFSA] rPow_def
by auto
moreover have \( \langle \langle Q, \text{Last}(w) \rangle, \bigcup s \in Q. \ t\langle s, \text{Last}(w) \rangle \rangle : t_{\text{Pow}} \) using last_type[of \( w(1) \) \( w(3) \)] unfolding tPow_def
by auto
ultimately show \text{thesis} using apply_equality[of _ dfsa.DFSA_dest(3),
of \( \langle Q, \text{Last}(w) \rangle \bigcup s \in Q. \ t\langle s, \text{Last}(w) \rangle \)] by blast
qed

Next, we show that if the deterministic automaton produces a reduction step to a word, then the non-deterministic one we constructed does the same reduction step.

lemma (in NonDetFinStateAuto) det_impl_nd:
assumes \( \langle \langle w, Q \rangle, \langle u, G \rangle \rangle \in r_{\text{Pow}} \)
shows \( \langle \langle w, Q \rangle, \langle u, G \rangle \rangle \in r_N \)
proof-
from assms have \( w : w \in \text{NELists}(\Sigma) \ u = \text{Init}(w) \ Q \in \text{Pow}(S) \ G = t_{\text{Pow}} \langle Q, \text{Last}(w) \rangle \)
unfolding DFSAExecutionRelation_def[OF finite_alphabet dfsa.DFSA] rPow_def by auto
then have \( t_{\text{Pow}}(Q, \text{Last}(w)) = (\bigcup s \in Q. \ t\langle s, \text{Last}(w) \rangle) \Longrightarrow \text{thesis} \)
unfolding NFSAExecutionRelation_def[OF finite_alphabet NFSA] by auto
moreover have \( \langle \langle Q, \text{Last}(w) \rangle, \bigcup s \in Q. \ t\langle s, \text{Last}(w) \rangle \rangle : t_{\text{Pow}} \) unfolding tPow_def
using last_type[of \( w(1) \) \( w(3) \)] by auto
ultimately show \text{thesis} using apply_equality[of _ dfsa.DFSA_dest(3),
of \( \langle Q, \text{Last}(w) \rangle \bigcup s \in Q. \ t\langle s, \text{Last}(w) \rangle \)] by blast
qed

Since both are relations, they are equal

243
corollary (in NonDetFinStateAuto) relation_NFSA_to_DFSA:
  shows r_N = rPow using nd_impl_det det_impl_nd
unfolding DFSAExecutionRelation_def[OF finite_alphabet dfsa.DFSA]
  NFSAExecutionRelation_def[OF finite_alphabet NFSA] rPow_def
by auto

As a consequence, by the definition of a language generated by an automa-
ton, both languages are equal.

theorem (in NonDetFinStateAuto) language_nfsa:
  shows dfsa.LanguageDFSA = LanguageNFSA
proof
  let S = Pow(S)
  let s = {s_0}
  let f = {{(U, u), | v ∈ U. t. (v, u) . (U, u) ∈ Pow(S) × Σ}}
  let F = {Q ∈ Pow(S) . Q ∩ F ≠ 0}
  { fix i assume i:i∈Lists(Σ) i <-D (S, s, f, F){in alphabet}Σ
    { assume i=0 s∈F
      then have i=0 s_0∈F by auto
      then have i <-N (S, s_0, t, F){in alphabet}Σ
        unfolding NFSASatisfy_def[OF finite_alphabet NFSA i(1)] by auto
    } moreover
    { assume ¬(i=0 ∧ s∈F)
      with i(2) obtain q where q:q∈F ⟨⟨i, s, (0, q)⟩⟩∈rPow^*
        using DFSASatisfy_def[OF finite_alphabet dfsa.DFSA i(1)]
        unfolding rPow_def tPow_def by auto
      then have ⟨⟨i, s, (0, q)⟩⟩∈r_N^* using relation_NFSA_to_DFSA
        by auto
      with q(1) have i <-N (S, s_0, t, F){in alphabet}Σ
        unfolding NFSASatisfy_def[OF finite_alphabet NFSA i(1)] by auto
    } ultimately
    have i <-N (S, s_0, t, F){in alphabet}Σ by auto
  } then have A:{i ∈ Lists(Σ) . dfsa.reduce(i)} ⊆ {i ∈ Lists(Σ) . i <-N
    (S, s_0, t, F){in alphabet}Σ}
    unfolding rPow_def tPow_def by auto
  { fix i assume i:i∈Lists(Σ) i <-N (S, s_0, t, F){in alphabet}Σ
    { assume i=0 s_0∈F
      then have i=0 s∈F using NFSA
        unfolding NFSA_def[OF finite_alphabet] by auto
      then have i <-D (S, s_0, t, F){in alphabet}Σ
        using DFSASatisfy_def[OF finite_alphabet dfsa.DFSA i(1)]
        unfolding tPow_def rPow_def by auto
    } moreover
    {
assume ¬(i=0 ∧ s_0 ∈ F)
with i(2) obtain q where q:q ∈ Pow(S) q ∩ F ≠ 0 ⟨⟨i,s⟩, ⟨0,q⟩⟩ ∈ rN^∗
  unfolding NFSASatisfy_def[OF finite_alphabet NFSA i(1)] by auto
then have ⟨⟨i,s⟩, ⟨0,q⟩⟩ ∈ rPow^∗ using relation_NFSA_to_DFSA
  by auto
with q(1,2) have i <-D (S,s,f,F){in alphabet}Σ
  using DFSASatisfy_def[OF finite_alphabet dfsa.DFSA i(1)]
  unfolding tPow_def rPow_def by auto
ultimately
have i <-D (S,s,f,F){in alphabet}Σ by auto
}
then have B:{i ∈ Lists(Σ) . i <-N (S,s_0,t,F){in alphabet}Σ} ⊆
  {i ∈ Lists(Σ) . dfsa.reduce(i)} unfolding tPow_def rPow_def by auto
with A show dfsa.LanguageDFSA = LanguageNFSA by auto
qed

The language of a non-deterministic finite state automaton is regular.
corollary (in NonDetFinStateAuto) lang_is_regular:
  shows LanguageNFSA{is a regular language on}Σ
  unfolding IsRegularLanguage_def[OF finite_alphabet]
  apply (rule exI[of _ Pow(S)],
       rule exI[of _ s_0],
       rule exI[of _ tPow],
       rule exI[of _ {Q ∈ Pow(S). Q ∩ F ≠ 0}])
  using language_nfsa dfsa.DFSA by auto

end

20 Inductive sequences

theory InductiveSeq_ZF imports Nat_ZF_IML FiniteSeq_ZF FinOrd_ZF
begin

In this theory we discuss sequences defined by conditions of the form a_0 = x, a_{n+1} = f(a_n) and similar.

20.1 Sequences defined by induction

One way of defining a sequence (that is a function a : N → X) is to provide
the first element of the sequence and a function to find the next value when
we have the current one. This is usually called ”defining a sequence by
induction”. In this section we set up the notion of a sequence defined by
induction and prove the theorems needed to use it.

245
First we define a helper notion of the sequence defined inductively up to a given natural number \( n \).

**definition**  
\[
\text{InductiveSequenceN}(x,f,n) \equiv \{ a. a: \text{succ}(n) \rightarrow \text{domain}(f) \land a(0) = x \land (\forall k \in n. a(\text{succ}(k)) = f(a(k))) \}
\]

From that we define the inductive sequence on the whole set of natural numbers. Recall that in Isabelle/ZF the set of natural numbers is denoted nat.

**definition**  
\[
\text{InductiveSequence}(x,f) \equiv \bigcup n \in \text{nat}. \text{InductiveSequenceN}(x,f,n)
\]

First we will consider the question of existence and uniqueness of finite inductive sequences. The proof is by induction and the next lemma is the \( P(0) \) step. To understand the notation recall that for natural numbers in set theory we have \( n = \{0,1,\ldots,n-1\} \) and \( \text{succ}(n) = \{0,1,\ldots,n\} \).

**lemma** \( \text{indseq_exun0} \): assumes \( A1: f: X \rightarrow X \) and \( A2: x \in X \)  
shows \( \exists ! a. a: \text{succ}(0) \rightarrow X \land a(0) = x \land (\forall k \in 0. a(\text{succ}(k)) = f(a(k))) \)

**proof**  
fix \( a, b \)
assume \( A3: a: \text{succ}(0) \rightarrow X \land a(0) = x \land (\forall k \in 0. a(\text{succ}(k)) = f(a(k))) \)
\( b: \text{succ}(0) \rightarrow X \land b(0) = x \land (\forall k \in 0. b(\text{succ}(k)) = f(b(k))) \)
moreover have \( \text{succ}(0) = \{0\} \) by auto
ultimately have \( a: \{0\} \rightarrow X \) b: \( \{0\} \rightarrow X \) by auto
then have \( a = \{(0, a(0))\} \) b = \( \{(0, b(0))\} \) using func_singleton_pair
by auto
with \( A3 \) show \( a=b \) by simp
next
let \( a = \{(0, x)\} \)
have \( a: \{0\} \rightarrow \{x\} \) using singleton_fun by simp
moreover from \( A1 A2 \) have \( \{x\} \subseteq X \) by simp
ultimately have \( a: \{0\} \rightarrow X \)
using func1_1_L1B by blast
moreover have \( \{0\} = \text{succ}(0) \) by auto
ultimately have \( a: \text{succ}(0) \rightarrow X \) by simp
with \( A1 \) show \( \exists a. a: \text{succ}(0) \rightarrow X \land a(0) = x \land (\forall k \in 0. a(\text{succ}(k)) = f(a(k))) \)
using singleton_apply by auto
qed

A lemma about restricting finite sequences needed for the proof of the inductive step of the existence and uniqueness of finite inductive sequences.

**lemma** \( \text{indseq_restrict} \):  
assumes \( A1: f: X \rightarrow X \) and \( A2: x \in X \) and \( A3: n \in \text{nat} \) and  
\( A4: a: \text{succ}(\text{succ}(n)) \rightarrow X \land a(0) = x \land (\forall k \in \text{succ}(n). a(\text{succ}(k)) = f(a(k))) \)

\[\text{246}\]
and A5: \( a_r = \text{restrict}(a, \text{succ}(n)) \)

shows
\[ a_r: \text{succ}(n) \rightarrow X \land a_r(0) = x \land ( \forall k \in n. a_r(\text{succ}(k)) = f(a_r(k)) ) \]

proof -
from A3 have succ\( (n) \subseteq \text{succ}(\text{succ}(n)) \) by auto
with A4 A5 have \( a_r: \text{succ}(n) \rightarrow X \) using restrict_type2 by auto
moreover
from A3 have 0 \( \in \text{succ}(n) \) using empty_in_every_succ by simp
with A4 A5 have \( a_r(0) = x \) using restrict_if by simp
moreover from A3 A4 A5 have \( \forall k \in n. a_r(\text{succ}(k)) = f(a_r(k)) \)
using succ_ineq restrict_if by auto
ultimately show thesis by simp
qed

Existence and uniqueness of finite inductive sequences. The proof is by induction and the next lemma is the inductive step.

lemma indseq_exun_ind:
assumes A1: \( f: X \rightarrow X \) and A2: \( x \in X \) and A3: \( n \in \text{nat} \) and
A4: \( \exists ! a. a: \text{succ}(n) \rightarrow X \land a(0) = x \land ( \forall k \in n. a(\text{succ}(k)) = f(a(k))) \)
shows
\( \exists ! a. a: \text{succ}(\text{succ}(n)) \rightarrow X \land a(0) = x \land ( \forall k \in \text{succ}(n). a(\text{succ}(k)) = f(a(k))) \)
proof
fix a b assume
A5: \( a: \text{succ}(\text{succ}(n)) \rightarrow X \land a(0) = x \land ( \forall k \in \text{succ}(n). a(\text{succ}(k)) = f(a(k))) \) and
A6: \( b: \text{succ}(\text{succ}(n)) \rightarrow X \land b(0) = x \land ( \forall k \in \text{succ}(n). b(\text{succ}(k)) = f(b(k))) \)
show a = b
proof -
let a_r = restrict(a, \text{succ}(n))
let b_r = restrict(b, \text{succ}(n))
note A1 A2 A3 A5
moreover have a_r = restrict(a, \text{succ}(n)) by simp
ultimately have I:
\[ a_r: \text{succ}(n) \rightarrow X \land a_r(0) = x \land ( \forall k \in n. a_r(\text{succ}(k)) = f(a_r(k))) \]
by (rule indseq_restrict)
note A1 A2 A3 A6
moreover have b_r = restrict(b, \text{succ}(n)) by simp
ultimately have
\[ b_r: \text{succ}(n) \rightarrow X \land b_r(0) = x \land ( \forall k \in n. b_r(\text{succ}(k)) = f(b_r(k))) \]
by (rule indseq_restrict)
with A4 I have II: \( a_r = b_r \) by blast
from A3 have succ\( (n) \in \text{nat} \) by simp
moreover from A5 A6 have
a: \( \text{succ}(\text{succ}(n)) \rightarrow X \) and b: \( \text{succ}(\text{succ}(n)) \rightarrow X \)
by auto
moreover note II
moreover
have T: n ∈ succ(n) by simp
then have a_r(n) = a(n) and b_r(n) = b(n) using restrict
by auto
with A5 A6 II T have a(succ(n)) = b(succ(n)) by simp
ultimately show a = b by (rule finseq_restr_eq)
qed

next show
∃ a. a: succ(succ(n)) → X ∧ a(0) = x ∧
( ∀k∈succ(n). a(succ(k)) = f(a(k)) )

proof -
from A4 obtain a where III: a: succ(n) → X and IV: a(0) = x
and V: ∀k∈n. a(succ(k)) = f(a(k)) by auto
let b = a ∪ {(succ(n), f(a(n)))}
from A1 III have
VI: b : succ(succ(n)) → X and
VII: ∀k ∈ succ(n). b(k) = a(k) and
VIII: b(succ(n)) = f(a(n))
using apply_functype finseq_extend by auto
from A3 have 0 ∈ succ(n) using empty_in_every_succ by simp
with IV VII have IX: b(0) = x by auto
{ fix k assume k ∈ succ(n)
then have k∈n ∨ k = n by auto
moreover
{ assume A7: k ∈ n
with A3 VII have b(succ(k)) = a(succ(k))
using succ_ineq by auto
also from A7 V VII have a(succ(k)) = f(b(k)) by simp
finally have b(succ(k)) = f(b(k)) by simp
}
moreover
{ assume A8: k = n
with VIII have b(succ(k)) = f(a(k)) by simp
with A8 VII VIII have b(succ(k)) = f(b(k)) by simp
ultimately have b(succ(k)) = f(b(k)) by auto
}
then have ∀k ∈ succ(n). b(succ(k)) = f(b(k)) by simp
with VI IX show thesis by auto
qed

The next lemma combines indseq_exun0 and indseq_exun_ind to show the
existence and uniqueness of finite sequences defined by induction.

lemma indseq_exun:
  assumes A1: f: X→X and A2: x∈X and A3: n ∈ nat
  shows
∃! a. a: succ(n) → X ∧ a(0) = x ∧ ( ∀k∈n. a(succ(k)) = f(a(k)) )
proof -
  note A3
moreover from A1 A2 have
∃! a. a: succ(0) → X ∧ a(0) = x ∧ ( ∀k∈0. a(succ(k)) = f(a(k)) )
using indseq_exun0 by simp
moreover from A1 A2 have ∀ k ∈ nat.
( ∃! a. a: succ(k) → X ∧ a(0) = x ∧
( ∀ i∈k. a(succ(i)) = f(a(i)) ) ) →
( ∃! a. a: succ(succ(k)) → X ∧ a(0) = x ∧
( ∀ i∈succ(k). a(succ(i)) = f(a(i)) ) )
using indseq_exun_ind by simp
ultimately show
∃! a. a: succ(n) → X ∧ a(0) = x ∧ ( ∀ k∈n. a(succ(k)) = f(a(k)) )
by (rule ind_on_nat)
qed

We are now ready to prove the main theorem about finite inductive sequences.

theorem fin_indseq_props:
  assumes A1: f: X→X and A2: x∈X and A3: n ∈ nat and
  A4: a = InductiveSequenceN(x,f,n)
  shows
  a: succ(n) → X a(0) = x ∀ k∈n. a(succ(k)) = f(a(k))
proof -
  let i = THE a. a: succ(n) → X ∧ a(0) = x ∧
  ( ∀ k∈n. a(succ(k)) = f(a(k)) )
  from A1 A2 A3 have
  ∃! a. a: succ(n) → X ∧ a(0) = x ∧ ( ∀ k∈n. a(succ(k)) = f(a(k)) )
  using indseq_exun by simp
  then have
  i: succ(n) → X ∧ i(0) = x ∧ ( ∀ k∈n. i(succ(k)) = f(i(k)) )
  by (rule theI)
  moreover from A1 A4 have a = i
  using InductiveSequenceN_def func1_1_L1 by simp
  ultimately show
  a: succ(n) → X a(0) = x ∀ k∈n. a(succ(k)) = f(a(k))
  by auto
qed

Since we have uniqueness we can show the inverse of fin_indseq_props: a sequence that satisfies the inductive sequence properties listed there is the inductively defined sequence.

lemma is_fin_indseq:
  assumes n ∈ nat f: X→X x∈X and
  a: succ(n) → X a(0) = x ∀ k∈n. a(succ(k)) = f(a(k))
  shows a = InductiveSequenceN(x,f,n)
proof -
  let b = InductiveSequenceN(x,f,n)
  from assms(1,2,3) have
  b: succ(n) → X b(0) = x ∀ k∈n. b(succ(k)) = f(b(k))
  using fin_indseq_props by simp_all
  with assms show thesis using indseq_exun by blast

249
A corollary about the domain of a finite inductive sequence.

corollary fin_indseq_domain:
  assumes A1: f: X → X and A2: x ∈ X and A3: n ∈ nat
  shows domain(InductiveSequenceN(x,f,n)) = succ(n)
proof -
  from assms have InductiveSequenceN(x,f,n) : succ(n) → X
    using fin_indseq_props by simp
  then show thesis using func1_1_L1 by simp
qed

The collection of finite sequences defined by induction is consistent in the sense that the restriction of the sequence defined on a larger set to the smaller set is the same as the sequence defined on the smaller set.

lemma indseq_consistent: assumes A1: f: X → X and A2: x ∈ X and
  A3: i ∈ nat j ∈ nat and A4: i ⊆ j
  shows restrict(InductiveSequenceN(x,f,j),succ(i)) = InductiveSequenceN(x,f,i)
proof -
  let a = InductiveSequenceN(x,f,j)
  let b = restrict(InductiveSequenceN(x,f,j),succ(i))
  let c = InductiveSequenceN(x,f,i)
  from A1 A2 A3 have
    a: succ(j) → X a(0) = x ∀ k ∈ j. a(succ(k)) = f(a(k))
      using fin_indseq_props by auto
  with A3 A4 have
    b: succ(i) → X ∧ b(0) = x ∧ (∀ k ∈ i. b(succ(k)) = f(b(k)))
      using succ_subset restrict_type2 empty_in_every_succ restrict succ_ineq
      by auto
  moreover from A1 A2 A3 have
    c: succ(i) → X ∧ c(0) = x ∧ (∀ k ∈ i. c(succ(k)) = f(c(k)))
      using fin_indseq_props by simp
  moreover from A1 A2 A3 have
    ∃ a. a: succ(i) → X ∧ a(0) = x ∧ (∀ k ∈ i. a(succ(k)) = f(a(k)))
      using indseq_exun by simp
  ultimately show b = c by blast
qed

For any two natural numbers one of the corresponding inductive sequences is contained in the other.

lemma indseq_subsets: assumes A1: f: X → X and A2: x ∈ X and
  A3: i ∈ nat j ∈ nat and A4: a = InductiveSequenceN(x,f,i) b = InductiveSequenceN(x,f,j)
  shows a ⊆ b ∨ b ⊆ a
proof -
  from A3 have i ⊆ j ∨ j ⊆ i using nat_incl_total by simp
  moreover

{ assume \(i \subseteq j\) 
with \(A_1 A_2 A_3 A_4\) have \(\text{restrict}(b,\text{succ}(i)) = a\) 
using \text{indseq\_consistent} by \text{simp} 
moreover have \(\text{restrict}(b,\text{succ}(i)) \subseteq b\) 
using \text{restrict\_subset} by \text{simp} 
ultimately have \(a \subseteq b \lor b \subseteq a\) by \text{simp} } 
moreover 
{ assume \(j \subseteq i\) 
with \(A_1 A_2 A_3 A_4\) have \(\text{restrict}(a,\text{succ}(j)) = b\) 
using \text{indseq\_consistent} by \text{simp} 
moreover have \(\text{restrict}(a,\text{succ}(j)) \subseteq a\) 
using \text{restrict\_subset} by \text{simp} 
ultimately have \(a \subseteq b \lor b \subseteq a\) by \text{simp} } 
ultimately have \(a \subseteq b \lor b \subseteq a\) by \text{auto} 
\text{qed}

The inductive sequence generated by applying a function 0 times is just the singleton list containing the starting point.

\textbf{lemma indseq\_empty:} assumes \(f: X \rightarrow X \ x \in X\) 
shows 
\begin{align*}
\text{InductiveSequenceN}(x,f,0):& \{0\} \rightarrow X \\
\text{InductiveSequenceN}(x,f,0) = & \{\langle 0, x \rangle\}
\end{align*}
\textbf{proof} -
\begin{itemize}
\item let \(a = \text{InductiveSequenceN}(x,f,0)\) 
from \text{assms} have \(\text{a(succ(0))} \in X\) and \(\text{a(0)} = x\) 
using \text{fin\_indseq\_props(1,2)} by \text{simp\_all} 
moreover have \(\text{succ}(0) = \{0\}\) by \text{auto} 
ultimately show \(\text{a:}\{0\} \rightarrow X\) by \text{auto} 
then have \(\exists a = \langle \langle 0, a(0) \rangle \rangle\) using \text{func\_singleton\_pair} 
by \text{simp} 
with \(\langle a(0) = x \rangle\) show \(a = \{\langle 0, x \rangle\}\) by \text{simp} 
\text{qed}
\end{itemize}

The tail of an inductive sequence generated by \(f\) and started from \(x\) is the same as the inductive sequence started from \(f(x)\).

\textbf{lemma indseq\_tail:} assumes \(n \in \text{nat}\) \(f: X \rightarrow X \ x \in X\) 
shows \(\text{Tail(InductiveSequenceN}(x,f,\text{succ}(n)))) = \text{InductiveSequenceN}(f(x),f,n)\) 
\textbf{proof} -
\begin{itemize}
\item let \(a = \text{Tail(InductiveSequenceN}(x,f,\text{succ}(n))))\) 
from \text{assms(2,3)} have \(f(x) \in X\) using \text{apply\_funtype} by \text{simp} 
have \(a: \text{succ}(n) \rightarrow X \ a(0) = f(x)\) and 
\(\forall k \in \text{nat}. \ a(\text{succ}(k)) = f(a(k))\) 
\text{proof} -
\item let \(b = \text{InductiveSequenceN}(x,f,\text{succ}(n))\) 
from \text{assms} have \(I: \text{succ}(n) \in \text{nat}\) \(b: \text{succ}(\text{succ}(n)) \rightarrow X\) 
using \text{fin\_indseq\_props(1)} by \text{simp\_all} 
then show \(\text{Tail(b)}: \text{succ}(n) \rightarrow X\) using \text{tail\_props} by \text{simp} 
from \text{assms(1)} I have \(\forall i. \text{Tail(b)}(0) = b(\text{succ}(0))\) 
using \text{tail\_props empty\_in\_every\_succ} by \text{blast} 
\end{itemize}
The first theorem about properties of infinite inductive sequences: inductive sequence is indeed a sequence (i.e. a function on the set of natural numbers).

**Theorem indseq_seq:** assumes \( A1: f: X \rightarrow X \) and \( A2: x \in X \) shows \( \text{InductiveSequence}(x,f) : nat \rightarrow X \)

**Proof:**
- let \( S = \{ \text{InductiveSequenceN}(x,f,n). n \in nat \} \)
  - fix \( a \) assume \( a \in S \)
    - then obtain \( n \) where \( n \in \text{nat} \) and \( a = \text{InductiveSequenceN}(x,f,n) \)
      by auto
    - with \( A1, A2 \) have \( a : \text{succ}(n) \rightarrow X \) using \( \text{fin_indseq_props} \)
      by simp
    - then have \( \exists A. b : A \rightarrow B \) by auto
  - then have \( \forall a \in S. \exists A. b : A \rightarrow B \) by auto
- moreover
  - fix \( a, b \) assume \( a \in S \) \( b \in S \)
    - then obtain \( i, j \) where \( i \in \text{nat} \) \( j \in \text{nat} \) and \( a = \text{InductiveSequenceN}(x,f,i) \)
      \( b = \text{InductiveSequenceN}(x,f,j) \)
    - by auto
    - with \( A1, A2 \) have \( a \subseteq b \lor b \subseteq a \) using \( \text{indseq_subsets} \) by simp
  - then have \( \forall a \in S. \forall b \in S. a \subseteq b \lor b \subseteq a \) by auto
- ultimately have \( \bigcup S : \text{domain}(\bigcup S) \rightarrow \text{range}(\bigcup S) \)
  - using \( \text{fun_Union} \) by simp
- with \( A1, A2 \) have \( I: \bigcup S : \text{nat} \rightarrow \text{range}(\bigcup S) \)
  - using \( \text{domain_UN} \) \( \text{fin_indseq_domain} \) \( \text{nat_union_succ} \) by simp
- moreover
  - fix \( k \) assume \( A3: k \in \text{nat} \)
    - let \( y = (\bigcup S)(k) \)
    - note \( I, A3 \)
    - moreover have \( y = (\bigcup S)(k) \) by simp
    - ultimately have \( (k,y) \in (\bigcup S) \) by (rule \( \text{func1_I_L5A} \) )

252
then obtain \( n \) where \( n \in \text{nat} \) and \( \text{II}: (k,y) \in \text{InductiveSequenceN}(x,f,n) \)
by auto
with \( A1 \ A2 \) have \( \text{InductiveSequenceN}(x,f,n) : \text{succ}(n) \to X \)
using \text{fin_indseq Props} by simp
with \( \text{II} \) have \( y \in X \) using \text{func1_1_L5} by blast
} then have \( \forall k \in \text{nat}. (\bigcup S)(k) \in X \)
by simp
ultimately have \( \bigcup S : \text{nat} \to X \)
using \text{func1_1_L1A} by blast
then show \( \text{InductiveSequence}(x,f) : \text{nat} \to X \)
using \text{InductiveSequence_def} by simp
qed

Restriction of an inductive sequence to a finite domain is the corresponding
finite inductive sequence.

**Lemma indseq_restr_eq:**
assumes \( A1: f : X \to X \) and \( A2: x \in X \) and \( A3: n \in \text{nat} \)
shows
\( \text{restrict} (\text{InductiveSequence}(x,f), \text{succ}(n)) = \text{InductiveSequenceN}(x,f,n) \)

**Proof:**
- let \( a = \text{InductiveSequence}(x,f) \)
- let \( b = \text{InductiveSequenceN}(x,f,n) \)
- let \( S = \{ \text{InductiveSequenceN}(x,f,n) . n \in \text{nat} \} \)
  from \( A1 \ A2 \ A3 \) have
    \( I: a : \text{nat} \to X \) and \( \text{succ}(n) \subseteq \text{nat} \)
    using \text{indseq_seq succnat_subset_nat} by auto
  then have \( \text{restrict}(a, \text{succ}(n)) : \text{succ}(n) \to X \)
    using \text{restrict_type2} by simp
  moreover from \( A1 \ A2 \ A3 \) have \( b : \text{succ}(n) \to X \)
    using \text{fin_indseq Props} by simp
  moreover
  \( \{ \text{fix } k \text{ assume } A4: k \in \text{succ}(n) \) from \( A1 \ A2 \ A3 \ I \) have
    \( \bigcup S : \text{nat} \to X \) \( b \in S \) \( b : \text{succ}(n) \to X \)
    using \text{InductiveSequence_def fin_indseq Props} by auto
  with \( A4 \) have \( \text{restrict}(a, \text{succ}(n))(k) = b(k) \)
    using \text{fun_Union_apply InductiveSequence_def restrict_if}
    by simp
  \} then have \( \forall k \in \text{succ}(n). \text{restrict}(a, \text{succ}(n))(k) = b(k) \)
    by simp
  ultimately have thesis by (rule \text{func_eq})
qed

The first element of the inductive sequence starting at \( x \) and generated by
\( f \) is indeed \( x \).

**Theorem indseq_valat0:**
assumes \( A1: f : X \to X \) and \( A2: x \in X \)
shows \( \text{InductiveSequence}(x,f)(0) = x \)

**Proof:**
- let \( a = \text{InductiveSequence}(x,f) \)
- let \( b = \text{InductiveSequenceN}(x,f,0) \)
have T: 0 ∈ nat 0 ∈ succ(0) by auto
with A1 A2 have b(0) = x
  using fin_indseq_props by simp
moreover from T have restrict(a,succ(0))(0) = a(0)
  using restrict_if by simp
moreover from A1 A2 T have
  restrict(a,succ(0)) = b
  using indseq_restr_eq by simp
ultimately show a(0) = x by simp
qed

An infinite inductive sequence satisfies the inductive relation that defines it.

theorem indseq_vals:
  assumes A1: f: X→X and A2: x∈X and A3: n ∈ nat
  shows InductiveSequence(x,f)(succ(n)) = f(InductiveSequence(x,f)(n))
proof -
  let a = InductiveSequence(x,f)
  let b = InductiveSequenceN(x,f,succ(n))
  from A3 have T:
    succ(n) ∈ succ(succ(n))
    succ(succ(n)) ∈ nat
    n ∈ succ(succ(n))
    by auto
  then have a(succ(n)) = restrict(a,succ(succ(n)))(succ(n))
    using restrict_if by simp
  also from A1 A2 T have ... = f(restrict(a,succ(succ(n)))(n))
    using indseq_restr_eq fin_indseq_props by simp
  also from T have ... = f(a(n)) using restrict_if by simp
  finally show a(succ(n)) = f(a(n)) by simp
qed

20.2 Images of inductive sequences

In this section we consider the properties of sets that are images of inductive
sequences, that is are of the form \{ f^{(n)}(x) : n ∈ N \} for some x in the domain
of f, where f^{(n)} denotes the n\textsuperscript{th} iteration of the function f. For a function
f : X → X and a point x ∈ X such set is set is sometimes called the orbit
of x generated by f.

The basic properties of orbits.

theorem ind_seq_image: assumes A1: f: X→X and A2: x∈X and
  A3: A = InductiveSequence(x,f)(nat)
  shows x∈A and ∀y∈A. f(y) ∈ A
proof -
  let a = InductiveSequence(x,f)
  from A1 A2 have a : nat → X using indseq_seq
    by simp
with A3 have I: A = \{a(n). n ∈ nat\} using func_imagedef
  by auto hence a(0) ∈ A by auto
with A1 A2 show x∈A using indseq_valat0 by simp
{ fix y assume y∈A
  with I obtain n where II: n ∈ nat and III: y = a(n)
  by auto
  with A1 A2 have a(succ(n)) = f(y)
    using indseq_vals by simp
  moreover from I II have a(succ(n)) ∈ A by auto
  ultimately have f(y) ∈ A by simp
} then show ∀y∈A. f(y) ∈ A by simp
qed

20.3 Subsets generated by a binary operation

In algebra we often talk about sets "generated" by an element, that is sets
of the form (in multiplicative notation) \{a^n|n ∈ Z\}. This is a related to a
general notion of "power" (as in a^n = a · a · · · a) or multiplicity n · a =
a+a+..+a. The intuitive meaning of such notions is obvious, but we need
to do some work to be able to use it in the formalized setting. This sections
is devoted to sequences that are created by repeatedly applying a binary
operation with the second argument fixed to some constant.

Basic properties of sets generated by binary operations.

theorem binop_gen_set:
  assumes A1: f: X×Y → X and A2: x∈X y∈Y and
  A3: a = InductiveSequence(x,Fix2ndVar(f,y))
  shows
  a : nat → X
  a(nat) ∈ Pow(X)
  x ∈ a(nat)
  ∀z ∈ a(nat). Fix2ndVar(f,y)(z) ∈ a(nat)
proof -
let g = Fix2ndVar(f,y)
from A1 A2 have I: g : X→X
  using fix_2nd_var_fun by simp
with A2 A3 show a : nat → X
  using indseq_seq by simp
then show a(nat) ∈ Pow(X) using func1_1_L6 by simp
from A2 A3 I show x ∈ a(nat) using ind_seq_image by blast
from A2 A3 I have
  g : X→X x∈X a(nat) = InductiveSequence(x,g)(nat)
  by auto
then show ∀z ∈ a(nat). Fix2ndVar(f,y)(z) ∈ a(nat)
  by (rule ind_seq_image)
qed

A simple corollary to the theorem binop_gen_set: a set that contains all
iterations of the application of a binary operation exists.
lemma binop_gen_set_ex: assumes A1: \( f: X \times Y \rightarrow X \) and A2: \( x \in X \) \( y \in Y \)
shows \( \{ A \in \text{Pow}(X). \ x \in A \land (\forall z \in A. f(z,y) \in A) \} \neq 0 \)
proof
  -
  let \( a = \text{InductiveSequence}(x,\text{Fix2ndVar}(f,y)) \)
  let \( A = a(\text{nat}) \)
  from A1 A2 have I: \( A \in \text{Pow}(X) \) and \( x \in A \) using binop_gen_set
    by auto
  moreover
  \{ fix \( z \) assume T: \( z \in A \)
    with A1 A2 have \( \text{Fix2ndVar}(f,y)(z) \in A \)
      using binop_gen_set
      by simp
    moreover
    from I T have \( z \in X \) by auto
    with A1 A2 have \( \text{Fix2ndVar}(f,y)(z) = f(z,y) \)
      using fix_var_val
      by simp
    ultimately have \( f(z,y) \in A \) by simp
  } then have \( \forall z \in A. f(z,y) \in A \) by simp
  ultimately show thesis by auto
qed

A more general version of binop_gen_set where the generating binary operation acts on a larger set.

theorem binop_gen_set1: assumes A1: \( f: X \times Y \rightarrow X \) and
  A2: \( X_1 \subseteq X \) and A3: \( x \in X \) \( y \in Y \) and
  A4: \( \forall t \in X_1. f(t,y) \in X_1 \) and
  A5: \( a = \text{InductiveSequence}(x,\text{Fix2ndVar}(\text{restrict}(f,X_1 \times Y),y)) \)
shows
  a : \( \text{nat} \rightarrow X_1 \)
  a(\text{nat}) \in \text{Pow}(X_1)
  x \in a(\text{nat})
  \( \forall z \in a(\text{nat}). \text{Fix2ndVar}(f,y)(z) \in a(\text{nat}) \)
  \( \forall z \in a(\text{nat}). f(z,y) \in a(\text{nat}) \)
proof
  -
  let \( h = \text{restrict}(f,X_1 \times Y) \)
  let \( g = \text{Fix2ndVar}(h,y) \)
  from A2 have \( X_1 \times Y \subseteq X \times Y \) by auto
  with A1 have I: \( h : X_1 \times Y \rightarrow X \)
    using restrict_type2 by simp
  with A3 have II: \( g : X_1 \rightarrow X \) using fix_2nd_var_fun by simp
  from A3 A4 I have \( \forall t \in X_1. g(t) \in X_1 \)
    using restrict fix_var_val by simp
  with II have III: \( g : X_1 \rightarrow X \) using func1_1_L1A by blast
  with A3 A5 show a : \( \text{nat} \rightarrow X_1 \) using indseq_seq by simp
  then show IV: \( a(\text{nat}) \in \text{Pow}(X_1) \) using func1_1_L6 by simp
  from A3 A5 III show x \in a(\text{nat}) using ind_seq_image by blast
  from A3 A5 III have \( g : X_1 \rightarrow X_1 \)
    x \in X_1 \( a(\text{nat}) = \text{InductiveSequence}(x,g)(\text{nat}) \)
    by auto
  then have \( \forall z \in a(\text{nat}). \text{Fix2ndVar}(h,y)(z) \in a(\text{nat}) \)

256
by (rule ind_seq_image)
moreover
{ fix z assume z ∈ a(nat)
  with IV have z ∈ X₁ by auto
  with A1 A2 A3 have g(z) = Fix2ndVar(f,y)(z)
    using fix_2nd_var_restr_comm restrict by simp
} then have ∀ z ∈ a(nat). g(z) = Fix2ndVar(f,y)(z) by simp
ultimately show ∀ z ∈ a(nat). Fix2ndVar(f,y)(z) ∈ a(nat) by simp
moreover
{ fix z assume z ∈ a(nat)
  with A2 IV have z∈X by auto
  with A1 A3 have Fix2ndVar(f,y)(z) = f⟨z,y⟩
    using fix_var_val by simp
} then have ∀ z ∈ a(nat). Fix2ndVar(f,y)(z) = f⟨z,y⟩
  by simp
ultimately show ∀ z ∈ a(nat). f⟨z,y⟩ ∈ a(nat)
  by simp
qed

A generalization of binop_gen_set_ex that applies when the binary operation acts on a larger set. This is used in our Metamath translation to prove the existence of the set of real natural numbers. Metamath defines the real natural numbers as the smallest set that contains 1 and is closed with respect to operation of adding 1.

lemma binop_gen_set_ex1: assumes A1: f: X×Y → X and
  A2: X₁ ⊆ X and A3: x∈X₁ y∈Y and
  A4: ∀ t∈X₁. f(t,y) ∈ X₁
  shows {A ∈ Pow(X₁). x∈A ∧ (∀ z ∈ A. f(z,y) ∈ A) } ≠ 0
proof -
  let a = InductiveSequence(x,Fix2ndVar(restrict(f,X₁×Y),y))
  let A = a(nat)
  from A1 A2 A3 A4 have
    A ∈ Pow(X₁) x ∈ A ∀ z ∈ A. f(z,y) ∈ A
    using binop_gen_set1 by auto
  thus thesis by auto
qed

20.4 Inductive sequences with changing generating function

A seemingly more general form of a sequence defined by induction is a sequence generated by the difference equation \( x_{n+1} = f_n(x_n) \) where \( n \mapsto f_n \) is a given sequence of functions such that each maps \( X \) into itself. For example when \( f_n(x) := x + x_n \) then the equation \( S_{n+1} = f_n(S_n) \) describes the sequence \( n \mapsto S_n = s_0 + \sum_{i=0}^{n} x_n \), i.e. the sequence of partial sums of the sequence \{s_0, x_0, x_1, x_3, ..\}.

The situation where the function that we iterate changes with \( n \) can be derived from the simpler case if we define the generating function appro-
priately. Namely, we replace the generating function in the definitions of InductiveSequenceN by the function $f : X \times n \rightarrow X \times n$, $f(x,k) = \langle f_k(x), k + 1 \rangle$ if $k < n$, $\langle f_k(x), k \rangle$ otherwise. The first notion defines the expression we will use to define the generating function. To understand the notation recall that in standard Isabelle/ZF for a pair $s = \langle x, n \rangle$ we have $\text{fst}(s) = x$ and $\text{snd}(s) = n$.

definition StateTransfFunNMeta(F,n,s) ≡
  if (snd(s) ∈ n) then $\langle F(snd(s))(fst(s)), succ(snd(s)) \rangle$ else s

Then we define the actual generating function on sets of pairs from $X \times \{0,1,...,n\}$.

definition StateTransfFunN(X,F,n) ≡ $\{\langle s, StateTransfFunNMeta(F,n,s) \rangle. s \in X\times\text{succ}(n)\}$

Having the generating function we can define the expression that we can use to define the inductive sequence generates.

definition StatesSeq(x,X,F,n) ≡ InductiveSequenceN($\langle x,0 \rangle$, StateTransfFunN(X,F,n),n)

Finally we can define the sequence given by an initial point $x$, and a sequence $F$ of $n$ functions.

definition InductiveSeqVarFN(x,X,F,n) ≡ $\{\langle k,fst(StatesSeq(x,X,F,n)(k)) \rangle. k \in \text{succ}(n)\}$

The state transformation function ($\text{StateTransfFunN}$ is a function that transforms $X \times n$ into itself.

lemma state_trans_fun: assumes A1: $n \in \text{nat}$ and A2: $F : n \rightarrow (X \rightarrow X)$
  shows $\text{StateTransfFunN}(X,F,n) : X\times\text{succ}(n) \rightarrow X\times\text{succ}(n)$

proof -
  { fix s assume A3: $s \in X\times\text{succ}(n)$
    let x = $\text{fst}(s)$
    let k = $\text{snd}(s)$
    let S = $\text{StateTransfFunNMeta}(F,n,s)$
    from A3 have T: $x \in X \land k \in \text{succ}(n)$ and $\langle x,k \rangle = s$ by auto
    { assume A4: $k \in n$
      with A1 have $\text{succ}(k) \in \text{succ}(n)$ using succ_ineq by simp
      with A2 T A4 have $S \in X\times\text{succ}(n)$ using apply_funtype StateTransfFunNMeta_def by simp }
    with A2 A3 T have $S \in X\times\text{succ}(n)$
      using apply_funtype StateTransfFunNMeta_def by simp }
  then have $\forall s \in X\times\text{succ}(n). \text{StateTransfFunNMeta}(F,n,s) \in X\times\text{succ}(n)$
    by simp
  then have $\{\langle s, \text{StateTransfFunNMeta}(F,n,s) \rangle. s \in X\times\text{succ}(n)\} : X\times\text{succ}(n) \rightarrow X\times\text{succ}(n)$
    by (rule ZF_fun_from_total)

258
then show \( \text{StateTransfFunN}(X,F,n) : X \times \text{succ}(n) \rightarrow X \times \text{succ}(n) \)
using \( \text{StateTransfFunN_def} \) by simp

qed

We can apply \text{fin_indseq_props} to the sequence used in the definition of \text{InductiveSeqVarFN} to get the properties of the sequence of states generated by the \text{StateTransfFunN}.

lemma \text{states_seq_props}:
assumes \( A1: n \in \text{nat} \) and \( A2: F : n \rightarrow (X \rightarrow X) \) and \( A3: x \in X \) and
\( A4: b = \text{StatesSeq}(x,X,F,n) \)
shows
\( b : \text{succ}(n) \rightarrow X \times \text{succ}(n) \)
\( b(0) = \langle x,0 \rangle \)
\( \forall k \in \text{succ}(n). \; \text{snd}(b(k)) = k \)
\( \forall k \in \text{n}. \; b(\text{succ}(k)) = \langle F(k)(\text{fst}(b(k))), \text{succ}(k) \rangle \)

proof -
let \( f = \text{StateTransfFunN}(X,F,n) \)
from \( A1 \) \( A2 \) have I: \( f : X \times \text{succ}(n) \rightarrow X \times \text{succ}(n) \)
using state_trans_fun by simp
moreover from \( A1 \) \( A3 \) have II: \( \langle x,0 \rangle \in X \times \text{succ}(n) \)
using empty_in_every_succ by simp
moreover note \( A1 \)
moreover from \( A4 \) have III: \( b = \text{InductiveSequenceN}(\langle x,0 \rangle,f,n) \)
using StatesSeq_def by simp
ultimately show IV: \( b : \text{succ}(n) \rightarrow X \times \text{succ}(n) \)
by (rule \text{fin_indseq_props})
from I II A1 III show V: \( b(0) = \langle x,0 \rangle \)
by (rule \text{fin_indseq_props})
from I II A1 III have VI: \( \forall k \in \text{n}. \; b(\text{succ}(k)) = f(b(k)) \)
by (rule \text{fin_indseq_props})
\{ fix \( k \)
note I moreover
assume A5: \( k \in \text{n} \) hence \( k \in \text{succ}(n) \) by auto
with IV have b(k) \( \in X \times \text{succ}(n) \) using apply_funtype by simp
moreover have \( f = \{ \langle s, \text{StateTransfFunNMeta}(F,n,s) \rangle. \; s \in X \times \text{succ}(n) \} \)
using StateTransfFunN_def by simp
ultimately have \( f(b(k)) = \text{StateTransfFunNMeta}(F,n,b(k)) \)
by (rule \text{ZF_fun_from_tot_val})
\} then have VII: \( \forall k \in \text{n}. \; f(b(k)) = \text{StateTransfFunNMeta}(F,n,b(k)) \)
by simp
\{ fix \( k \) assume A5: \( k \in \text{succ}(n) \)
note A1 A5 moreover from V have \( \text{snd}(b(0)) = 0 \) by simp
moreover from VI VII have
\( \forall j \in \text{n}. \; \text{snd}(b(j)) = j \rightarrow \text{snd}(b(\text{succ}(j))) = \text{succ}(j) \)
using StateTransfFunNMeta_def by auto
ultimately have \( \text{snd}(b(k)) = k \) by (rule \text{fin_nat_ind})
\} then show \( \forall k \in \text{succ}(n). \; \text{snd}(b(k)) = k \) by simp

259
with VI VII show \( \forall k \in \mathbb{N}. \ b(\text{succ}(k)) = (F(k)(\text{fst}(b(k))))) \), \( \text{succ}(k) \) using StateTransfFunNMeta_def by auto
qed

Basic properties of sequences defined by equation \( x_{n+1} = f_n(x_n) \).

*theorem* fin_indseq_var_f_props:
  assumes A1: \( n \in \mathbb{N} \) and A2: \( x \in X \) and A3: \( F : n \rightarrow (X \rightarrow X) \)
  shows
  \[ a : \text{succ}(n) \rightarrow X \]
  \[ a(0) = x \]
  \[ \forall k \in n. \ a(\text{succ}(k)) = F(k)(a(k)) \]
*proof* -
  let \( f = \text{StateTransfFunN}(X, F, n) \)
  let \( b = \text{StatesSeq}(x, X, F, n) \)
  from A1 A2 A3 have \( b : \text{succ}(n) \rightarrow X \times \text{succ}(n) \)
    using states_seq_props by simp
  then have \( \forall k \in \text{succ}(n). \ b(k) \in X \times \text{succ}(n) \)
    using apply_funtype by simp
  hence \( \forall k \in \text{succ}(n). \ \text{fst}(b(k)) \in X \) by auto
  then have I: \( \{ (k, \text{fst}(b(k))). \ k \in \text{succ}(n) \} : \text{succ}(n) \rightarrow X \)
    by (rule ZF_fun_from_total)
  with A4 show II: \( a : \text{succ}(n) \rightarrow X \) using InductiveSeqVarFN_def
    by simp
  moreover from A1 have \( 0 \in \text{succ}(n) \) using empty_in_every_succ
    by simp
  moreover from A4 have III:
    \( a = \{ (k, \text{fst}(\text{StatesSeq}(x, X, F, n)(k))). \ k \in \text{succ}(n) \} \)
    using InductiveSeqVarFN_def by simp
  ultimately have \( a(0) = \text{fst}(b(0)) \)
    by (rule ZF_fun_from_tot_val)
  with A1 A2 A3 show \( a(0) = x \) using states_seq_props by auto
  \{ fix \( k \)
  assume A5: \( k \in n \)
  with A1 have T1: \( \text{succ}(k) \in \text{succ}(n) \) and T2: \( k \in \text{succ}(n) \)
    using succ_ineq by auto
  from II T1 III have \( a(\text{succ}(k)) = \text{fst}(b(\text{succ}(k))) \)
    by (rule ZF_fun_from_tot_val)
  with A1 A2 A3 A5 have \( a(\text{succ}(k)) = F(k)(\text{fst}(b(k))) \)
    using states_seq_props by simp
  moreover from II T2 III have \( a(k) = \text{fst}(b(k)) \)
    by (rule ZF_fun_from_tot_val)
  ultimately have \( a(\text{succ}(k)) = F(k)(a(k)) \)
    by simp
  \} then show \( \forall k \in n. \ a(\text{succ}(k)) = F(k)(a(k)) \)
    by simp
qed

Uniqueness lemma for sequences generated by equation \( x_{n+1} = f_n(x_n) \):
lemma fin_indseq_var_f_uniq: assumes n ∈ nat x ∈ X F: n → (X → X)
  and a: succ(n) → X a(0) = x ∀ k ∈ n. a(succ(k)) = (F(k))(a(k))
  and b: succ(n) → X b(0) = x ∀ k ∈ n. b(succ(k)) = (F(k))(b(k))
shows a = b
proof -
  have ∀ k ∈ succ(n). a(k) = b(k)
proof -
  let A = {i ∈ succ(succ(n)). ∀ k ∈ i. a(k) = b(k)}
  let m = Maximum(Le, A)
  from assms(1) have I: succ(succ(n)) ∈ nat A ⊆ succ(succ(n)) by auto
  moreover
  from assms(1, 5, 8) have succ(0) ∈ A using empty_in_every_succ succ_ineq
    by simp
  hence II: A ≠ 0 by auto
  ultimately have m ∈ A by (rule nat_max_props)
  moreover have m = succ(n)
proof -
  { assume m ≠ succ(n)
  from I II have III: ∀ k ∈ A. k ≤ m by (rule nat_max_props)
  have succ(m) ∈ A
  proof -
    from m ≠ succ(n) m ∈ A have m ∈ succ(n)
      using mem_succ_not_eq by blast
    from I II have m ∈ nat by (rule nat_max_props)
    from succ(0) ∈ A III have succ(0) ≤ m by blast
    hence m ≠ 0 by auto
    with m ∈ nat obtain k where k ∈ nat m = succ(k)
      using Nat_ZF_1_L3 by auto
    with assms(1) m ∈ succ(n) have k ∈ n using succ_mem by simp
    with assms(6, 9) m = succ(k) m ∈ A have a(m) = b(m) using succ_explained by simp
    with assms(1) m ∈ A m ∈ succ(n) show succ(m) ∈ A
      using succ_explained succ_ineq by blast
  qed
  with III have succ(m) ≤ m by (rule property_holds)
  hence False by auto
  } thus thesis by auto
  qed
  ultimately show thesis by simp
  qed
  with assms(4, 7) show a = b by (rule func_eq)
  qed

A sequence that has the properties of sequences generated by equation
x_{n+1} = f_n(x_n) must be the one generated by this equation.

theorem is_fin_indseq_var_f: assumes n ∈ nat x ∈ X F: n → (X → X)
  and a: succ(n) → X a(0) = x ∀ k ∈ n. a(succ(k)) = (F(k))(a(k))
shows a = InductiveSeqVarFN(x, X, F, n)
proof -

261
let b = InductiveSeqVarFN(x,X,F,n)
from assms(1,2,3) have b: succ(n) → X b(0) = x
and ∀k∈n. b(succ(k)) = F(k)(b(k))
using fin_indseq_var_f_props by simp_all
with assms show thesis by (rule fin_indseq_var_f_uniq)
qed

A consistency condition: if we make the sequence of generating functions shorter, then we get a shorter inductive sequence with the same values as in the original sequence.

lemma fin_indseq_var_f_restrict: assumes
A1: n ∈ nat i ∈ nat x ∈ X F: n → (X→X) G: i → (X→X)
and A2: i ⊆ n and A3: ∀j∈i. G(j) = F(j) and A4: k ∈ succ(i)
shows InductiveSeqVarFN(x,X,G,i)(k) = InductiveSeqVarFN(x,X,F,n)(k)
proof-
let a = InductiveSeqVarFN(x,X,F,n)
let b = InductiveSeqVarFN(x,X,G,i)
from A1 A4 have i ∈ nat k ∈ succ(i) by auto
moreover from A1 have b(0) = a(0)
moreover from A1 A2 A3 have
∀j∈i. b(j) = a(j) → b(succ(j)) = a(succ(j))
using fin_indseq_var_f_props by auto
ultimately show b(k) = a(k)
by (rule fin_nat_ind)
qed

20.5 The Pascal’s triangle

One possible application of the inductive sequences is to define the Pascal’s triangle. The Pascal’s triangle can be defined directly as $P_{n,k} = \binom{n}{k} = \frac{n!}{k!(n-k)!}$ for $n \geq k \geq 0$. Formalizing this definition (or explaining to a 10-years old) is quite difficult as it depends on the definition of factorial and some facts about factorizing natural numbers needed to show that the quotient in $\frac{n!}{k!(n-k)!}$ is always a natural number. Another approach uses induction and the property that each number in the array is the sum of the two numbers directly above it.

To shorten the definition of the function generating the Pascal’s triangle we first define expression for the $k$’th element in the row following given row $r$. The rows are represented as lists, i.e. functions $r : n \rightarrow \mathbb{N}$ (recall that for natural numbers we have $n = \{0, 1, 2, ..., n-1\}$). The value of the next row is 1 at the beginning and equals $r(k-1) + r(k)$ otherwise. A careful reader might wonder why we do not require the values to be 1 on the right boundary of the Pascal’s triangle. We are able to show this as a theorem (see binom_right_boundary below) using the fact that in Isabelle/ZF the value of
a function on an argument that is outside of the domain is the empty set, which is the same as zero of natural numbers.

**definition**

\[
\text{BinomElem}(r,k) \equiv \text{if } k=0 \text{ then } 1 \text{ else } r(\text{pred}(k)) \#+ r(k)
\]

Next we define a function that takes a row in a Pascal’s triangle and returns the next row.

**definition**

\[
\text{GenBinom} \equiv \{ (r, \{ (k, \text{BinomElem}(r,k)) \mid k \in \text{succ}(\text{domain}(r)) \}) \mid r \in \text{NELists}(\text{nat}) \}
\]

The function generating rows of the Pascal’s triangle is indeed a function that maps nonempty lists of natural numbers into nonempty lists of natural numbers.

**lemma** gen_binom_fun: shows GenBinom: \( \text{NELists}(\text{nat}) \to \text{NELists}(\text{nat}) \)

**proof** -

\[
\{ \text{fix } r \text{ assume } r \in \text{NELists}(\text{nat}) \}
\text{then obtain } n \text{ where } n \in \text{nat} \text{ and } r: \text{succ}(n) \to \text{nat} \]
\text{unfolding \text{NELists_def by auto}}
\text{then have domain}(r) = \text{succ}(n) \text{ using \text{func1_1_L1 by simp}}
\text{let } r_1 = \{ (k, \text{BinomElem}(r,k)) \mid k \in \text{succ}(\text{domain}(r)) \}\)
\text{have } \forall k \in \text{succ}(\text{domain}(r)). \text{BinomElem}(r,k) \in \text{nat}
\text{unfolding \text{BinomElem_def by simp}}
\text{then have } r_1: \text{succ}(\text{domain}(r)) \to \text{nat}
\text{by (rule \text{ZF_fun_from_total})}
\text{with } \langle n \in \text{nat} \rangle \langle \text{domain}(r) = \text{succ}(n) \rangle \text{ have } r_1 \in \text{NELists}(\text{nat})
\text{unfolding \text{NELists_def by auto}}
\}
\text{then show thesis using \text{ZF_fun_from_total unfolding \text{GenBinom_def by simp}}}
\]

**qed**

The value of the function GenBinom at a nonempty list \( r \) is a list of length one greater than the length of \( r \).

**lemma** gen_binom_fun_val: assumes \( n \in \text{nat} \) \( r: \text{succ}(n) \to \text{nat} \)
shows GenBinom(\( r \)): \( \text{succ}(\text{succ}(n)) \to \text{nat} \)

**proof** -

\[
\text{let } B = \{ (r, \{ (k, \text{BinomElem}(r,k)) \mid k \in \text{succ}(\text{domain}(r)) \}) \mid r \in \text{NELists}(\text{nat}) \}
\text{let } r_1 = \{ (k, \text{BinomElem}(r,k)) \mid k \in \text{succ}(\text{domain}(r)) \}\)
\text{from assumptions have } r \in \text{NELists}(\text{nat}) \text{ unfolding \text{NELists_def by blast}}
\text{then have } B(r) = r_1 \text{ using \text{ZF_fun_from_tot_val1 by simp}}
\text{have } \forall k \in \text{succ}(\text{domain}(r)). \text{BinomElem}(r,k) \in \text{nat}
\text{unfolding \text{BinomElem_def by simp}}
\text{then have } r_1: \text{succ}(\text{domain}(r)) \to \text{nat}
\text{by (rule \text{ZF_fun_from_total})}
\text{with assumptions(2) } B(r) = r_1 \text{ show thesis}
\text{using \text{func1_1_L1 unfolding \text{GenBinom_def by simp}}}
\]

**qed**

Now we are ready to define the Pascal’s triangle as the inductive sequence
that starts from a singleton list $0 \mapsto 1$ and is generated by iterations of the GenBinom function.

**Definition**

\[
PascalTriangle \equiv \text{InductiveSequence}((\{0,1\}), \text{GenBinom})
\]

The singleton list containing 1 (i.e. the starting point of the inductive sequence that defines the PascalTriangle) is a finite list and the PascalTriangle is a sequence (an infinite list) of nonempty lists of natural numbers.

**Lemma pascal_sequence:**

- shows $\langle 0,1 \rangle \in \text{NELists(nat)}$ and $\text{PascalTriangle} : \text{nat} \to \text{NELists(nat)}$
- using list_len1_singleton(2) gen_binom_fun indseq_seq
- unfolding PascalTriangle_def
- by auto

The GenBinom function creates the next row of the Pascal’s triangle from the previous one.

**Lemma binom_gen:**

- assumes $n \in \text{nat}$
- shows $\text{PascalTriangle}(\text{succ}(n)) = \text{GenBinom}(\text{PascalTriangle}(n))$
- using assms pascal_sequence gen_binom_fun indseq_vals
- unfolding PascalTriangle_def by simp

The $n^{th}$ row of the Pascal’s triangle is a list of $n + 1$ natural numbers.

**Lemma pascal_row_list:**

- assumes $n \in \text{nat}$ shows $\text{PascalTriangle}(n) : \text{succ}(n) \to \text{nat}$
- proof -
  - from assms(1) have $n \in \text{nat}$ and $\text{PascalTriangle}(0) : \text{succ}(0) \to \text{nat}$
    - using gen_binom_fun pascal_sequence(1) indseq_valat0 list_len1_singleton(1)
    - unfolding PascalTriangle_def by auto
  - moreover have $\forall k \in \text{nat}. \text{PascalTriangle}(k) : \text{succ}(k) \to \text{nat} \longrightarrow \text{PascalTriangle}(\text{succ}(k)) : \text{succ}(\text{succ}(k)) \to \text{nat}$
    - proof -
      - \{ fix $k$ assume $k \in \text{nat}$ and $\text{PascalTriangle}(k) : \text{succ}(k) \to \text{nat}$
      - then have $\text{PascalTriangle}(\text{succ}(k)) : \text{succ}(\text{succ}(k)) \to \text{nat}$
        - using gen_binom_fun_val gen_binom_fun pascal_sequence(1) indseq_vals
          - unfolding NELists_def PascalTriangle_def by auto
      \}
    - thus thesis by simp
  - qed
- ultimately show thesis by (rule ind_on_nat)
- qed

In our approach the Pascal’s triangle is a list of lists. The value at index $n \in \mathbb{N}$ is a list of length $n + 1$ (see pascal_row_list above). Hence, the largest index in the domain of this list is $n$. However, we can still show that the value of that list at index $n + 1$ is 0, because in Isabelle/ZF (as well as
in Metamath) the value of a function at a point outside of the domain is the empty set, which happens to be the same as the natural number 0.

**Lemma pascal_val_beyond:** assumes $n \in \text{nat}$
shows $(\text{PascalTriangle}(n))(\text{succ}(n)) = 0$

**Proof -**
- from assms have domain($\text{PascalTriangle}(n)$) = $\text{succ}(n)$
  - using $\text{pascal_row_list func1_1_L1}$ by blast
  - then show thesis using $\text{mem_self apply_0}$
    - by simp

**QED**

For $n > 0$ the Pascal’s triangle values at $(n, k)$ are given by the $\text{BinomElem}$ expression.

**Lemma pascal_row_val:** assumes $n \in \text{nat} \land k \in \text{succ}(\text{succ}(n))$
shows $(\text{PascalTriangle}(\text{succ}(n)))(k) = \text{BinomElem}(\text{PascalTriangle}(n), k)$

**Proof -**
- let $B = \{ \langle r, \{ k, \text{BinomElem}(r, k) \} \rangle \mid k \in \text{succ}(\text{domain}(r)) \}$, $r \in \text{NELists(nat)}$
- let $B_r = \{ \langle k, \text{BinomElem}(r, k) \rangle \mid k \in \text{succ}(\text{succ}(n)) \}$
- from assms(1) have $r \in \text{NELists(nat)}$ and $r : \text{succ}(n) \rightarrow \text{nat}$
  - using $\text{pascal_sequence}(2)$, $\text{apply_funtype pascal_row_list}$
  - by auto
- then have $B(r) = B_r$ using $\text{func1_1_L1} \land \text{ZF_fun_from_tot_val1}$
  - by simp
- moreover from assms(1) have $B(r) = \text{PascalTriangle}(\text{succ}(n))$
  - using $\text{binom_gen unfoldng GenBinom_def}$ by simp
- moreover from assms(2) have $B_r(k) = \text{BinomElem}(r, k)$
  - by (rule $\text{ZF_fun_from_tot_val1}$)
- ultimately show thesis by simp

**QED**

The notion that will actually be used is the binomial coefficient $\binom{n}{k}$ which we define as the value at the right place of the Pascal’s triangle.

**Definition**
$$\text{Binom}(n, k) \equiv (\text{PascalTriangle}(n))(k)$$

Entries in the Pascal’s triangle are natural numbers. Since in Isabelle/ZF the value of a function at a point that is outside of the domain is the empty set (which is the same as zero of natural numbers) we do not need any assumption on $k$.

**Lemma binom_in_nat:** assumes $n \in \text{nat}$
shows $\text{Binom}(n, k) \in \text{nat}$

**Proof -**
- \{ assume $k \in \text{succ}(n)$
  - with assms have $(\text{PascalTriangle}(n))(k) \in \text{nat}$
    - using $\text{pascal_row_list apply_funtype}$ by blast
  \}
- moreover
The top of the Pascal's triangle is equal to 1 (i.e. \( \binom{0}{0} = 1 \)). This is an easy fact that it is useful to have handy as it is at the start of a couple of inductive arguments.

**Lemma binom_zero_zero**: shows \( \binom{0}{0} = 1 \)

using \( \text{gen_binom_fun pascal_sequence(1) indseq_valat0 pair_val} \)
unfolding \( \text{Binom_def PascalTriangle_def} \) by auto

The binomial coefficients are 1 on the left boundary of the Pascal's triangle.

**Theorem binom_left_boundary**: assumes \( n \in \mathbb{N} \) shows \( \binom{n}{0} = 1 \)

proof -

\[
\begin{align*}
\text{assume } n \neq 0 & \quad \text{with } \text{assms obtain } k \text{ where } k \in \mathbb{N} \text{ and } n = \text{succ}(k) \\
& \quad \text{using } \text{Nat_ZF_1_L3 by blast} \\
& \quad \text{then have } \binom{n}{0} = 1 \text{ using } \text{empty_in_every_succ pascal_row_val} \\
& \quad \text{unfolding } \text{BinomElem_def Binom_def by simp}
\end{align*}
\]

then show thesis using binom_zero_zero by blast

**Theorem binom_prop**: assumes \( n \in \mathbb{N} \) \( k \leq n \#+1 \) \( k \not= 0 \) shows \( \binom{n \#+1,k} = \binom{n,k \#-1} \#+ \binom{n,k} \)

proof -

\[
\begin{align*}
\text{let } P &= \text{PascalTriangle} \\
& \quad \text{from } \text{assms(1,2) have } k \in \mathbb{N} \text{ and } k \in \text{succ(succ}(n)) \\
& \quad \text{using } \text{le_in_nat nat_mem_lt(2) by auto} \\
& \quad \text{with } \text{assms(1) have } \binom{n \#+1,k} = \binom{n,k \#-1} \#+ \binom{n,k} \\
& \quad \text{unfolding } \text{Binom_def using pascal_row_val by simp} \\
& \quad \text{also from } \text{assms(3) } <k\in\mathbb{N}> \text{ have} \\
& \quad \text{BinomElem}(P(n),k) = (P(n))(k \#-1) \#+ (P(n))(k) \\
& \quad \text{unfolding } \text{BinomElem_def using pred_minus_one by simp} \\
& \quad \text{also have } (P(n))(k \#-1) \#+ (P(n))(k) = \binom{n,k \#-1} \#+ \binom{n,k} \\
& \quad \text{unfolding } \text{Binom_def by simp}
\end{align*}
\]

finally show thesis by simp

qed

266
A version binom_prop where we write $k + 1$ instead of $k$.

**Lemma binom_prop2**: assumes $n \in \mathbb{N}$ $k \in n \#+ 1$

shows $\text{Binom}(n \#+ 1, k \#+ 1) = \text{Binom}(n, k \#+ 1) \#+ \text{Binom}(n, k)$

**Proof** -

from assms have $k \in \mathbb{N}$ using elem_nat_is_nat(2) by blast

hence $k \#+1 \#- 1 = k$ by simp

moreover from assms have

$\text{Binom}(n \#+ 1, k \#+ 1) = \text{Binom}(n, k \#+1 \#- 1) \#+ \text{Binom}(n, k \#+ 1)$

using succ_inq2 binom_prop by simp

ultimately show thesis by simp

qed

A special case of binom_prop when $n = k + 1$ that helps with the induction step in the proof that the binomial coefficient are 1 on the right boundary of the Pascal’s triangle.

**Lemma binom_prop1**: assumes $n \in \mathbb{N}$

shows $\text{Binom}(n \#+ 1, n \#+ 1) = \text{Binom}(n, n)$

**Proof** -

let $B = \text{Binom}$

from assms have $B(n, n) \in \mathbb{N}$

using pascal_row_list apply_funtype

unfolding Binom_def by blast

from assms have $(\text{PascalTriangle}(n))(\text{succ}(n)) = 0$

using pascal_val_beyond by simp

moreover from assms have $\text{succ}(n) = n \#+ 1$

using succ_add_one(1) by simp

ultimately have $B(n, n \#+ 1) = 0$

unfolding Binom_def by simp

with assms $<B(n, n) \in \mathbb{N}>$ show thesis

using succ_add_one(2) binom_prop add_subtract add_0 add_commute by simp

qed

The binomial coefficients are 1 on the right boundary of the Pascal’s triangle.

**Theorem binom_right_boundary**: assumes $n \in \mathbb{N}$ shows $\text{Binom}(n, n) = 1$

**Proof** -

from assms have $n \in \mathbb{N}$ and $\text{Binom}(0, 0) = 1$

using binom_zero_zero by auto

moreover have

$\forall k \in \mathbb{N}. \text{Binom}(k, k) = 1 \rightarrow \text{Binom}($\text{succ}(k), \text{succ}(k)) = 1$

using binom_prop1 by simp

ultimately show thesis by (rule ind_on_nat)

qed

end
Suppose \( r \) is a linear order on a set \( A \) that has \( n \) elements, where \( n \in \mathbb{N} \).
In the \texttt{FinOrd_ZF} theory we prove a theorem stating that there is a unique order isomorphism between \( n = \{0, 1, \ldots, n-1\} \) (with natural order) and \( A \).
Another way of stating that is that there is a unique way of counting the elements of \( A \) in the order increasing according to relation \( r \). Yet another way of stating the same thing is that there is a unique sorted list of elements of \( A \). We will call this list the \texttt{Enumeration} of \( A \).

### 21.1 Enumerations: definition and notation

In this section we introduce the notion of enumeration and define a proof context (a "locale" in Isabelle terms) that sets up the notation for writing about enumerations.

We define enumeration as the only order isomorphism between a set \( A \) and the number of its elements. We are using the formula \( \bigcup \{x\} = x \) to extract the only element from a singleton. \( \texttt{Le} \) is the (natural) order on natural numbers, defined in \texttt{Nat_ZF} theory in the standard Isabelle library.

**definition**

\[
\text{Enumeration}(A,r) \equiv \bigcup \text{ord_iso}(|A|,\texttt{Le},A,r)
\]

To set up the notation we define a locale \texttt{enums}. In this locale we will assume that \( r \) is a linear order on some set \( X \). In most applications this set will be just the set of natural numbers. Standard Isabelle uses \( \leq \) to denote the "less or equal" relation on natural numbers. We will use the \( \leq \) symbol to denote the relation \( r \). Those two symbols usually look the same in the presentation, but they are different in the source. To shorten the notation the enumeration \( \text{Enumeration}(A,r) \) will be denoted as \( \sigma(A) \). Similarly as in the \texttt{Semigroup} theory we will write \( a \leftarrow x \) for the result of appending an element \( x \) to the finite sequence (list) \( a \). Finally, \( a \sqcup b \) will denote the concatenation of the lists \( a \) and \( b \).

**locale** \texttt{enums} =

\begin{verbatim}
fixes X r
assumes linord: IsLinOrder(X,r)

fixes ler (infix \( \leq \))
defines ler_def[simp]: \( x \leq y \equiv \langle x,y \rangle \in r \)

fixes \( \sigma \)
\end{verbatim}
defines \( \sigma \) def [simp]: \( \sigma(A) \equiv \text{Enumeration}(A, r) \)

fixes append (infix \( \leftarrow \))
defines append_def [simp]: \( a \leftarrow x \equiv \text{Append}(a, x) \)

fixes concat (infixl \( \sqcup \))
defines concat_def [simp]: \( a \sqcup b \equiv \text{Concat}(a, b) \)

21.2 Properties of enumerations

In this section we prove basic facts about enumerations.

A special case of the existence and uniqueness of the order isomorphism for
finite sets when the first set is a natural number.

lemma (in enums) ord_iso_nat_fin: assumes \( A \in \text{FinPow}(X) \) and \( n \in \text{nat} \) and \( A \approx n \)
shows \( \exists ! f. f \in \text{ord_iso}(n, \text{Le}, A, r) \)
using assms NatOrder_ZF_1_L2 linord nat_finpow_nat
fin_ord_iso_ex_uniq by simp

An enumeration is an order isomorhism, a bijection, and a list.

lemma (in enums) enum_props: assumes \( A \in \text{FinPow}(X) \)
shows \( \sigma(A) \in \text{ord_iso}(|A|, \text{Le}, A, r) \)
\( \sigma(A) \in \text{bij}(|A|, A) \)
\( \sigma(A) : |A| \rightarrow A \)
proof -
from assms have \( \text{IsLinOrder}(\text{nat}, \text{Le}) \) and \( |A| \in \text{FinPow}(<\text{nat}) \) and \( A \approx |A| \)
using NatOrder_ZF_1_L2 card_fin_is_nat nat_finpow_nat
by auto
with assms show \( \sigma(A) \in \text{ord_iso}(|A|, \text{Le}, A, r) \)
using linord fin_ord_iso_ex_uniq singleton_extract
Enumeration_def by simp
then show \( \sigma(A) \in \text{bij}(|A|, A) \) and \( \sigma(A) : |A| \rightarrow A \)
using ord_iso_def bij_def surj_def
by auto
qed

A corollary from enum_props. Could have been attached as another assertion,
but this slows down verification of some other proofs.

lemma (in enums) enum_fun: assumes \( A \in \text{FinPow}(X) \)
shows \( \sigma(A) : |A| \rightarrow X \)
proof -
from assms have \( \sigma(A) : |A| \rightarrow A \) and \( A \subseteq X \)
using enum_props FinPow_def by auto
then show \( \sigma(A) : |A| \rightarrow X \) by \( \text{rule func1_1_L1B} \)
qed
If a list is an order isomorphism then it must be the enumeration.

**Lemma (in enums) ord_iso_enum:** assumes A1: A ∈ FinPow(X) and A2: n ∈ nat and A3: f ∈ ord_iso(n,Le,A,r)
shows f = σ(A)
proof -
from A3 have n ≈ A using ord_iso_def eqpoll_def
  by auto
then have A ≈ n by (rule eqpoll_sym)
with A1 A2 have ∃!f. f ∈ ord_iso(n,Le,A,r)
  using ord_iso_nat_fin by simp
with assms ⟨A ≈ n⟩ show f = σ(A)
  using enum_props card_card by blast
qed

What is the enumeration of the empty set?

**Lemma (in enums) empty_enum:** shows σ(0) = 0
proof -
have 0 ∈ FinPow(X) and 0 ∈ nat and 0 ∈ ord_iso(0,Le,0,r)
  using empty_in_finpow empty_ord_iso_empty
  by auto
then show σ(0) = 0 using ord_iso_enum
  by blast
qed

Adding a new maximum to a set appends it to the enumeration.

**Lemma (in enums) enum_append:**
assumes A1: A ∈ FinPow(X) and A2: b ∈ X-A and
A3: ∀ a∈A. a≤b
shows σ(A ∪ {b}) = σ(A)←b
proof -
let f = σ(A) ∪ {{|A|,b}}
from A1 have |A| ∈ nat using card_fin_is_nat
  by simp
from A1 A2 have A ∪ {b} ∈ FinPow(X)
  using singleton_in_finpow union_finpow by simp
moreover from this have |A ∪ {b}| ∈ nat
  using card_fin_is_nat by simp
moreover have f ∈ ord_iso(|A ∪ {b}| , Le, A ∪ {b} ,r)
proof -
from A1 A2 have
  σ(A) ∈ ord_iso(|A|,Le, A,r) and
  |A| /∈ |A| and b /∈ A
  using enum_props mem_not_refl by auto
moreover from ⟨|A| ∈ nat⟩ have
  ∀ k ∈ |A|. ⟨k, |A|⟩ ∈ Le
  using elem_nat_is_nat by blast
moreover from A3 have ∀ a∈A. (a,b) ∈ r by simp
moreover have antisym(Le) and antisym(r)
using \texttt{linord NatOrder_zf_1_L2 IsLinOrder_def} by auto

moreover from A2 \(|A| \in \text{nat}\) have
\(\langle |A|, |A| \rangle \in \text{Le} \) and \(\langle b, b \rangle \in r\)
using \texttt{linord NatOrder_zf_1_L2 IsLinOrder_def}
\texttt{total_is_refl refl_def} by auto

hence \((|A|, |A|) \in \text{Le} \) and \((b, b) \in r\)
by \texttt{(rule ord_iso_extend)}

with A1 A2 show \(f \in \text{ord_iso}(|A| \cup \{b\} , \text{Le} , A \cup \{b\} , r)\)
using \texttt{card_fin_add_one} by simp

ultimately have \(f \in \text{ord_iso}(|A| \cup \{b\} , \text{Le} , A \cup \{b\} , r)\)
by \texttt{(rule ord_iso_extend)}
with A1 A2 show \(f \in \text{ord_iso}(|A| \cup \{b\} , \text{Le} , A \cup \{b\} , r)\)
using \texttt{card_fin_add_one} by simp

ultimately have \(f = \sigma(A \cup \{b\})\)
using \texttt{ord_iso_enum} by simp
moreover have \(\sigma(A) \leftarrow b = f\)
proof -
  have \(\sigma(A) \leftarrow b = \sigma(A) \cup \{(\text{domain} (\sigma(A)), b)\}\)
  using \texttt{Append_def} by simp
  moreover from A1 have \(\text{domain} (\sigma(A)) = |A|\)
  using \texttt{enum_props func1_1_L1} by blast
  ultimately show \(\sigma(A) \leftarrow b = f\) by simp
qeda

ultimately show \(\sigma(A \cup \{b\}) = \sigma(A) \leftarrow b\) by simp
qeda

What is the enumeration of a singleton?

\textbf{lemma (in enums) \texttt{enum_singleton}:}
assumes A1: \(x \in X\) shows \(\sigma(\{x\}) : 1 \rightarrow X \) and \(\sigma(\{x\})(0) = x\)
proof -
  from A1 have
  \(0 \in \text{FinPow}(X)\) and \(x \in (X - 0)\) and \(\forall a \in 0. a \leq x\)
  using \texttt{empty_in_finpow} by auto
  then have \(\sigma(0 \cup \{x\}) = \sigma(0) \leftarrow x\) by \texttt{(rule enum_append)}
  with A1 show \(\sigma(\{x\}) : 1 \rightarrow X \) and \(\sigma(\{x\})(0) = x\)
  using \texttt{empty_enum empty_append1} by auto
qeda

end

\textbf{22 Folding in ZF}

\textbf{theory \texttt{Fold_ZF} imports \texttt{InductiveSeq_ZF}}

\begin{verbatim}
Suppose we have a binary operation \(P : X \times X \rightarrow X\) written multiplicatively as \(P(x, y) = x \cdot y\). In informal mathematics we can take a sequence \(\{x_k\}_{k \in \mathbb{N}}\) of elements of \(X\) and consider the product \(x_0 \cdot x_1 \cdot \ldots \cdot x_n\). To do the same thing
\end{verbatim}
in formalized mathematics we have to define precisely what is meant by that ",...". The definitition we want to use is based on the notion of sequence defined by induction discussed in InductiveSeq_ZF. We don't really want to derive the terminology for this from the word "product" as that would tie it conceptually to the multiplicative notation. This would be awkward when we want to reuse the same notions to talk about sums like \( x_0 + x_1 + \ldots + x_n \).

In functional programming there is something called "fold". Namely for a function \( f \), initial point \( a \) and list \([b, c, d]\) the expression \( \text{fold}(f, a, [b, c, d]) \) is defined to be \( f(f(f(a, b), c), d) \) (in Haskell something like this is called foldl). If we write \( f \) in multiplicative notation we get \( a \cdot b \cdot c \cdot d \), so this is exactly what we need. The notion of folds in functional programming is actually much more general that what we need here (not that I know anything about that). In this theory file we just make a slight generalization and talk about folding a list with a binary operation \( f : X \times Y \rightarrow X \) with \( X \) not necessarily the same as \( Y \).

22.1 Folding in ZF

Suppose we have a binary operation \( f : X \times Y \rightarrow X \). Then every \( y \in Y \) defines a transformation of \( X \) defined by \( T_y(x) = f(x, y) \). In IsarMathLib such transformation is called as \( \text{Fix2ndVar}(f, y) \). Using this notion, given a function \( f : X \times Y \rightarrow X \) and a sequence \( y = \{y_k\}_{k \in \mathbb{N}} \) of elements of \( X \) we can get a sequence of transformations of \( X \). This is defined in Seq2TransSeq below. Then we use that sequence of transformations to define the sequence of partial folds (called FoldSeq) by means of \( \text{InductiveSeqVarFN} \) (defined in InductiveSeq_ZF theory) which implements the inductive sequence determined by a starting point and a sequence of transformations. Finally, we define the fold of a sequence as the last element of the sequence of the partial folds.

Definition that specifies how to convert a sequence \( a \) of elements of \( Y \) into a sequence of transformations of \( X \), given a binary operation \( f : X \times Y \rightarrow X \).

\[
\text{Seq2TrSeq}(f, a) \equiv \{ (k, \text{Fix2ndVar}(f, a(k))) : k \in \text{domain}(a) \}
\]

Definition of a sequence of partial folds.

\[
\text{FoldSeq}(f, x, a) \equiv \text{InductiveSeqVarFN}(x, \text{fstdom}(f), \text{Seq2TrSeq}(f, a), \text{domain}(a))
\]

Definition of a fold.

\[
\text{Fold}(f, x, a) \equiv \text{Last}(\text{FoldSeq}(f, x, a))
\]
If $X$ is a set with a binary operation $f : X \times Y \to X$ then $\text{Seq2TransSeq}(f, a)$ converts a sequence $a$ of elements of $Y$ into the sequence of corresponding transformations of $X$.

**lemma seq2trans_seq_props:**

assumes $A1: n \in \text{nat}$ and $A2: f : X \times Y \to X$ and $A3: a : n \to Y$ and $A4: T = \text{Seq2TrSeq}(f, a)$

shows $T : n \to (X \to X)$ and $\forall k \in n. \forall x \in X. (T(k))(x) = f(x, a(k))$

**proof** -

from $\langle a : n \to Y \rangle$ have $D: \text{domain}(a) = n$ using func1_1_L1 by simp

with $A2$ $A3$ $A4$ show $T : n \to (X \to X)$

using apply_funtype fix_2nd_var fun ZF_fun_from_total Seq2TrSeq_def by simp

with $A4$ $D$ have $I: \forall k \in n. T(k) = \text{Fix2ndVar}(f, a(k))$

using Seq2TrSeq_def ZF_fun_from_tot_val0 by simp

{ fix $k$ $x$ assume $A5: k \in n$ $x \in X$

with $A1$ $A3$ have $a(k) \in Y$ using apply_funtype by auto

with $A2$ $A5$ $I$ have $(T(k))(x) = f(x, a(k))$

using fix_var_val by simp

} thus $\forall k \in n. \forall x \in X. (T(k))(x) = f(x, a(k))$

by simp

qed

Basic properties of the sequence of partial folds of a sequence $a = \{y_k\}_{k \in \{0, \ldots, n\}}$.

**theorem fold_seq_props:**

assumes $A1: n \in \text{nat}$ and $A2: f : X \times Y \to X$ and $A3: y : n \to Y$ and $A4: x \in X$ and $A5: Y \neq 0$ and $A6: F = \text{FoldSeq}(f, x, y)$

shows $F: \text{succ}(n) \to X$

$F(0) = x$ and $\forall k \in n. F(\text{succ}(k)) = f(F(k), y(k))$

**proof** -

let $T = \text{Seq2TrSeq}(f, y)$

from $A1$ $A3$ have $D: \text{domain}(y) = n$

using func1_1_L1 by simp

from $\langle f : X \times Y \to X \rangle$ $\langle Y \neq 0 \rangle$ have $I: \text{fstdom}(f) = X$

using fstdomdef by simp

with $A1$ $A2$ $A3$ $A4$ $A6$ $D$ show $\langle F : \text{succ}(n) \to X \rangle$ $\text{and } F(0) = x$

using seq2trans_seq_props FoldSeq_def fin_indseq_var_f_props by auto

from $A1$ $A2$ $A3$ $A4$ $A6$ $I$ $D$ have $\forall k \in n. F(\text{succ}(k)) = T(k)(F(k))$

using seq2trans_seq_props FoldSeq_def fin_indseq_var_f_props by simp

moreover

{ fix $k$ assume $A5: k \in n$ hence $k \in \text{succ}(n)$ by auto

273
with A1 A2 A3 II A5 have \((T(k))(F(k)) = f(F(k),y(k))\)
using apply_funtype seq2trans_seq_props by simp \}
ultimately show \(\forall k \in n. F(\text{succ}(k)) = f(F(k), y(k))\)
by simp
qed

A consistency condition: if we make the list shorter, then we get a shorter sequence of partial folds with the same values as in the original sequence. This can be proven as a special case of \text{fin_indseq_var_f_restrict} but a proof using \text{fold_seq_props} and induction turns out to be shorter.

\begin{lstlisting}
lemma foldseq_restrict: assumes
  \(n \in \text{nat} \quad k \in \text{succ}(n)\) and
  \(i \in \text{nat} \quad f : X \times Y \to X \quad a : n \to Y \quad b : i \to Y\) and
  \(n \subseteq i \quad \forall j \in n. b(j) = a(j)\) x \(\in X \quad Y \neq 0\)
shows \(\text{FoldSeq}(f,x,b)(k) = \text{FoldSeq}(f,x,a)(k)\)
proof -
  let \(P = \text{FoldSeq}(f,x,a)\)
  let \(Q = \text{FoldSeq}(f,x,b)\)
  from assms have
    \(n \in \text{nat} \quad k \in \text{succ}(n)\)
    \(Q(0) = P(0)\) and
    \(\forall j \in n. Q(j) = P(j) \implies Q(\text{succ}(j)) = P(\text{succ}(j))\)
    using fold_seq_props by auto
  then show \(Q(k) = P(k)\) by (rule \text{fin_nat_ind})
qed
\end{lstlisting}

A special case of \text{foldseq_restrict} when the longer sequence is created from the shorter one by appending one element.

\begin{lstlisting}
corollary fold_seq_append:
  assumes \(n \in \text{nat} \quad f : X \times Y \to X \quad a : n \to Y\) and
  \(x \in X \quad k \in \text{succ}(n) \quad y \in Y\)
shows \(\text{FoldSeq}(f,x,\text{Append}(a,y))(k) = \text{FoldSeq}(f,x,a)(k)\)
proof -
  let \(b = \text{Append}(a,y)\)
  from assms have \(b : \text{succ}(n) \to Y\) \(\forall j \in n. b(j) = a(j)\)
  using append_props by auto
  with assms show thesis using foldseq_restrict by blast
qed
\end{lstlisting}

What we really will be using is the notion of the fold of a sequence, which we define as the last element of (inductively defined) sequence of partial folds. The next theorem lists some properties of the product of the fold operation.

\begin{lstlisting}
theorem fold_props:
  assumes A1: \(n \in \text{nat}\) and
  A2: \(f : X \times Y \to X \quad a : n \to Y\) \(x \in X \quad Y \neq 0\)
shows
  \(\text{Fold}(f,x,a) = \text{FoldSeq}(f,x,a)(n)\) and
  \(\text{Fold}(f,x,a) \in X\)
\end{lstlisting}
proof -
from assms have \( \text{FoldSeq}(f,x,a) : \text{succ}(n) \rightarrow X \)
using \text{fold_seq_props} by simp
with A1 show
Fold(f,x,a) = \text{FoldSeq}(f,x,a)(n) and \( \text{Fold}(f,x,a) \in X \)
using \text{last_seq_elem} \text{apply_functype Fold_def} by auto
qed

A corner case: what happens when we fold an empty list?

\textbf{Theorem fold_empty:} assumes A1: \( f : X \times Y \rightarrow X \) and
A2: a:0→Y x∈X Y≠0
shows \( \text{Fold}(f,x,a) = x \)

proof -
let F = \text{FoldSeq}(f,x,a)
from assms have I:
0 ∈ \text{nat} f : X \times Y \rightarrow X a:0→Y x∈X Y≠0
by auto
then have Fold(f,x,a) = F(0) by (rule fold_props)
moreover
from I have
0 ∈ \text{nat} f : X \times Y \rightarrow X a:0→Y x∈X Y≠0 and
F = \text{FoldSeq}(f,x,a) by auto
then have F(0) = x by (rule fold_seq_props)
ultimately show \( \text{Fold}(f,x,a) = x \) by simp
qed

The next theorem tells us what happens to the fold of a sequence when we
add one more element to it.

\textbf{Theorem fold_append:} assumes A1: n ∈ \text{nat} and
A2: f : X \times Y \rightarrow X and
A3: a:n→Y and A4: x∈X and A5: y∈Y
shows
FoldSeq(f,x,\text{Append}(a,y))(n) = \text{Fold}(f,x,a) and
Fold(f,x,\text{Append}(a,y)) = f(\text{Fold}(f,x,a), y)

proof -
let b = \text{Append}(a,y)
let P = \text{FoldSeq}(f,x,b)
from A5 have I: Y ≠ 0 by auto
with assms show thesis1: P(n) = \text{Fold}(f,x,a)
using \text{fold_seq_append} \text{fold_props} by simp
from assms I have II:
succ(n) ∈ \text{nat} f : X \times Y \rightarrow X
b : succ(n) \rightarrow Y x∈X Y ≠ 0
P = \text{FoldSeq}(f,x,b)
using \text{append_props} by auto
then have
\( \forall k \in \text{succ}(n). P(\text{succ}(k)) = f(P(k), b(k)) \)
by (rule fold_seq_props)
with A3 A5 thesis1 have P(succ(n)) = f(\text{Fold}(f,x,a), y)
using append_props by auto
moreover
from II have P : succ(succ(n)) → X
  by (rule fold_seq Props)
then have Fold(f,x,b) = P(succ(n))
  using last_seq_elem Fold_def by simp
ultimately show Fold(f,x,Append(a,y)) = f(Fold(f,x,a), y)
  by simp
qed

Another way of formulating information contained in fold_append is to start with a longer sequence \( a : n + 1 \to X \) and then detach the last element from it. This provides an identity between the fold of the longer sequence and the value of the folding function on the fold of the shorter sequence and the last element of the longer one.

lemma fold_detach_last:
  assumes \( n \in \text{nat}\ f : X \times Y \to X \ x \in X \ \forall k \in n \ #+ 1 \ q(k) \in Y \)
  shows Fold(f,x,\{\langle k,q(k) \rangle . k \in n \ #+ 1 \}) = f(Fold(f,x,\{\langle k,q(k) \rangle . k \in n \}), q(n))
proof -
  let a = \{\langle k,q(k) \rangle . k \in n \ #+ 1 \}
  let b = \{\langle k,q(k) \rangle . k \in n \}
  from assms have
    Fold(f,x,Append(b,q(n))) = f(Fold(f,x,b), q(n))
      using ZF_fun_from_total fold_append(2) by simp_all
moreover from assms(1,4) have a = Append(b,q(n))
  using set_list_append(4) by simp
ultimately show Fold(f,x,a) = f(Fold(f,x,b), q(n))
  by simp
qed

The tail of the sequence of partial folds defined by the folding function \( f \), starting point \( x \) and a sequence \( y \) is the same as the sequence of partial folds starting from \( f(x,y(0)) \).

lemma fold_seq_detach_first:
  assumes \( n \in \text{nat}\ f : X \times Y \to X \ y : \text{succ}(n) \to Y \ x \in X \)
  shows FoldSeq(f,f(x,y(0)),Tail(y)) = Tail(FoldSeq(f,x,y))
proof -
  let F = FoldSeq(f,x,y)
  let T = Tail(F)
  let S = Seq2TrSeq(f,Tail(y))
  from assms(1,3) have succ(n) \in \text{nat}\ 0 \in \text{succ}(n) \ y(0) \in Y 
    using empty_in_every_succ apply_functype by simp_all
  have n \in \text{nat}\ \forall k \in n. T(succ(k)) = (S(k))(T(k))
    and T:succ(n) \to X \ T(0) = f(x,y(0))
    by simp
  proof -
    from assms(1) show n \in \text{nat} by simp

276
from assms \(\langle \text{suc}(n) \in \text{nat} \rangle\) show \(T: \text{suc}(n) \rightarrow X\)
using \text{fold}_\text{seq}_\text{props}(1) \text{tail}_\text{props}(1) \text{nelist}_\text{vals}_\text{nonempty} \text{ by simp}\nfrom assms(2,4) \(\langle y(0) \in Y \rangle\) show \(f(x,y(0)) \in X\)
using \text{apply}_\text{funttype} \text{ by simp}\nfrom assms(1,2,3) show \(S:n \rightarrow (X \rightarrow X)\)
using \text{tail}_\text{props}(1) \text{seq2trans}_\text{seq}_\text{props} \text{ by simp}\nshow \(T(0) = f(x,y(0))\)
proof -
from \(\langle \text{suc}(n) \in \text{nat} \rangle\) \(\langle 0 \in \text{suc}(n) \rangle\) I
have \(T(0) = F(\text{suc}(0))\)
using \text{tail}_\text{props}(2) \text{ by blast}\nmoreover from assms \(\langle 0 \in \text{suc}(n) \rangle\)
have \(F(\text{suc}(0)) = f(F(0), y(0))\)
using \text{fold}_\text{seq}_\text{props}(3) \text{nelist}_\text{vals}_\text{nonempty} \text{ by blast}\nmoreover from assms have \(F(0) = x\)
using \text{fold}_\text{seq}_\text{props}(2) \text{nelist}_\text{vals}_\text{nonempty} \text{ by blast}\nultimately show \(T(0) = f(x,y(0))\) \text{ by simp}\nqed
show \(\forall k \in n. T(\text{suc}(k)) = (S(k))(T(k))\)
proof -
\{ fix k assume \(k \in n\)
with assms(1) have
\(\text{suc}(k) \in \text{suc}(n)\) \(\langle k \in \text{suc}(n)\rangle\) \(\text{suc}(k) \in \text{suc}(\text{suc}(n))\)
using \text{succ}_\text{ineq} \text{ by auto}\nwith \(\langle \text{suc}(n) \in \text{nat} \rangle\) I have \(T(\text{suc}(k)) = F(\text{suc}(\text{suc}(k)))\)
using \text{tail}_\text{props}(2) \text{ by blast}\nmoreover from assms \(\langle \text{suc}(k) \in \text{suc}(n) \rangle\)
have \(F(\text{suc}(\text{suc}(k))) = f(F(\text{suc}(k)), y(\text{suc}(k)))\)
using \text{fold}_\text{seq}_\text{props}(3) \text{nelist}_\text{vals}_\text{nonempty} \text{ by blast}\nmoreover from assms(1,3) \(\langle k \in n \rangle\) have \(y(\text{suc}(k)) = (\text{Tail}(y))(k)\)
using \text{tail}_\text{props}(2) \text{ by simp}\nmoreover from assms \(\langle k \in n \rangle\) I \(\langle \text{suc}(k) \in \text{suc}(\text{suc}(n)) \rangle\)
have \(f(F(\text{suc}(k)), (\text{Tail}(y))(k)) = (S(k))(F(\text{suc}(k)))\)
using \text{tail}_\text{props}(1) \text{seq2trans}_\text{seq}_\text{props}(2) \text{apply}_\text{funttype} \text{ by simp}\nmoreover from \(\langle \text{suc}(n) \in \text{nat} \rangle\) \(\langle k \in \text{suc}(n) \rangle\)
have \(T(k) = F(\text{suc}(k))\)
using \text{tail}_\text{props}(2) \text{ by blast}\nultimately have \(T(\text{suc}(k)) = (S(k))(T(k))\) \text{ by simp}\n\}
thus thesis by simp\nqed
qed
then have \(T = \text{InductiveSeqVarFN}(f(x,y(0)),X,S,n)\)
by (rule is_fin_indseq_var_f)\nmoreover have \(\text{fstdom}(f) = X\) and \(\text{domain}(\text{Tail}(y)) = n\)
proof -
from assms(2,3) show \(\text{fstdom}(f) = X\)
277
Taking a fold of a sequence \( y \) with a function \( f \) with the starting point \( x \) is the same as the fold starting from \( f(x, y(0)) \) of the tail of \( y \).

**Lemma fold_detach_first:**

Assumes \( n \in \text{nat} \) \( f : X \times Y \rightarrow X \ y : \text{succ}(n) \rightarrow Y \ x \in X \)

shows \( \text{Fold}(f, x, y) = \text{Fold}(f, f(x, y(0)), \text{Tail}(y)) \)

**Proof:**

- From assms have \( \text{FoldSeq}(f, x, y) : \text{succ}(\text{succ}(n)) \rightarrow X \)
  - Using \( \text{fold_seq_props}(1) \) \( \text{nelist_vals_nonempty} \) by simp
  - With assms show thesis
    - Using \( \text{last_tail_last} \) \( \text{fold_seq_detach_first} \) unfolding \( \text{Fold_def} \) by simp

**QED**

### 23 Partitions of sets

**Theory Partitions_ZF imports Finite_ZF FiniteSeq_ZF**

**Begin**

It is a common trick in proofs that we divide a set into non-overlapping subsets. The first case is when we split the set into two nonempty disjoint sets. Here this is modeled as an ordered pair of sets and the set of such divisions of set \( X \) is called \( \text{Bisections}(X) \). The second variation on this theme is a set-valued function (aren’t they all in ZF?) whose values are nonempty and mutually disjoint.

#### 23.1 Bisections

This section is about dividing sets into two non-overlapping subsets.

The set of bisections of a given set \( A \) is a set of pairs of nonempty subsets of \( A \) that do not overlap and their union is equal to \( A \).

**Definition**

\[ \text{Bisections}(X) = \{ p \in \text{Pow}(X) \times \text{Pow}(X). \text{fst}(p) \neq 0 \land \text{snd}(p) \neq 0 \land \text{fst}(p) \cap \text{snd}(p) = 0 \land \text{fst}(p) \cup \text{snd}(p) = X \} \]

Properties of bisections.

**Lemma bisec_props:** Assumes \( (A, B) \in \text{Bisections}(X) \) shows
A \neq 0 \quad B \neq 0 \quad A \subseteq X \quad B \subseteq X \quad A \cap B = 0 \quad A \cup B = X \quad X \neq 0

using assms Bisections_def by auto

Kind of inverse of bisec_props: a pair of nonempty disjoint sets form a bisection of their union.

lemma is_bisec:
assumes A \neq 0 \quad B \neq 0 \quad A \cap B = 0
shows (A,B) \in Bisections(A \cup B) using assms Bisections_def by auto

Bisection of X is a pair of subsets of X.

lemma bisec_is_pair:
assumes Q \in Bisections(X)
sshows Q = \langle \text{fst}(Q), \text{snd}(Q) \rangle
using assms Bisections_def by auto

The set of bisections of the empty set is empty.

lemma bisec_empty:
sows Bisections(0) = 0
using Bisections_def by auto

The next lemma shows what can we say about bisections of a set with another element added.

lemma bisec_add_point:
assumes A1: x \not\in X \quad A2: (A,B) \in Bisections(X \cup \{x\})
sows (A = \{x\} \cup B = \{x\}) \vee ((A - \{x\}, B - \{x\}) \in Bisections(X))
proof -
\{ assume A \neq \{x\} and B \neq \{x\}
    with A2 have A - \{x\} \neq 0 and B - \{x\} \neq 0
    using singl_diff_empty Bisections_def by auto
    moreover have (A - \{x\}) \cup (B - \{x\}) = X
    proof -
      have (A - \{x\}) \cup (B - \{x\}) = (A \cup B) - \{x\}
      by auto
      also from assms have (A \cup B) - \{x\} = X
      using Bisections_def by auto
      finally show thesis by simp
    qed
    moreover from A2 have (A - \{x\}) \cap (B - \{x\}) = 0
    using Bisections_def by auto
    ultimately have (A - \{x\}, B - \{x\}) \in Bisections(X)
    using Bisections_def by auto
  } thus thesis by auto
qed

A continuation of the lemma bisec_add_point that refines the case when the pair with removed point bisects the original set.

lemma bisec_add_point_case3:
assumes A1: (A,B) \in Bisections(X \cup \{x\})
and A2: ⟨A - {x}, B - {x}⟩ ∈ Bisections(X)
s
shows
(⟨A, B - {x}⟩ ∈ Bisections(X) ∧ x ∈ B) ∨
(⟨A - {x}, B⟩ ∈ Bisections(X) ∧ x ∈ A)
proof -
from A1 have x ∈ A ∪ B
  using Bisections_def by auto
hence x ∈ A ∪ B by simp
from A1 have A - {x} = A ∨ B - {x} = B
  using Bisections_def by auto
moreover
{ assume A - {x} = A
  with A2 ⟨x ∈ A ∪ B⟩ have
    ⟨A, B - {x}⟩ ∈ Bisections(X) ∧ x ∈ B
    using singl_diff_eq by simp }
moreover
{ assume B - {x} = B
  with A2 ⟨x ∈ A ∪ B⟩ have
    ⟨A - {x}, B⟩ ∈ Bisections(X) ∧ x ∈ A
    using singl_diff_eq by simp }
ultimately show thesis by auto
qed

Another lemma about bisecting a set with an added point.

lemma point_set_bisec:
  assumes A1: x /∈ X and A2: ⟨{x}, A⟩ ∈ Bisections(X ∪ {x})
  shows A = X and X ≠ 0
proof -
  from A2 have A ⊆ X using Bisections_def by auto
  moreover
  { fix a assume a ∈ X
    with A2 have a ∈ {x} ∪ A using Bisections_def by simp
      with A1 ⟨a ∈ X⟩ have a ∈ A by auto }
  ultimately show A = X by auto
  with A2 show X ≠ 0 using Bisections_def by simp
qed

Yet another lemma about bisecting a set with an added point, very similar
to point_set_bisec with almost the same proof.

lemma set_point_bisec:
  assumes A1: x /∈ X and A2: ⟨A, {x}⟩ ∈ Bisections(X ∪ {x})
  shows A = X and X ≠ 0
proof -
  from A2 have A ⊆ X using Bisections_def by auto
  moreover
  { fix a assume a ∈ X
    with A2 have a ∈ A ∪ {x} using Bisections_def by simp
      with A1 ⟨a ∈ X⟩ have a ∈ A by auto }
  ultimately show A = X by auto
with A2 show X ≠ 0 using Bisections_def by simp

qed

If a pair of sets bisects a finite set, then both elements of the pair are finite.

lemma bisect_fin:
  assumes A1: A ∈ FinPow(X) and A2: Q ∈ Bisections(A)
  shows fst(Q) ∈ FinPow(X) and snd(Q) ∈ FinPow(X)
proof -
  from A2 have ⟨fst(Q), snd(Q)⟩ ∈ Bisections(A)
    using bisec_is_pair by simp
  then have fst(Q) ⊆ A and snd(Q) ⊆ A
    using bisec_props by auto
  with A1 show fst(Q) ∈ FinPow(X) and snd(Q) ∈ FinPow(X)
    using FinPow_def subset_Finite by auto
qed

23.2 Partitions

This section covers the situation when we have an arbitrary number of sets
we want to partition into.

We define a notion of a partition as a set valued function such that the values
for different arguments are disjoint. The name is derived from the fact that
such function "partitions" the union of its arguments. Please let me know
if you have a better idea for a name for such notion. We would prefer to
say "is a partition", but that reserves the letter "a" as a keyword(?) which
causes problems.

definition
  Partition (_ {is partition} [90] 91) where
  P {is partition} ≡ ∀ x ∈ domain(P).
  P(x) ≠ 0 ∧ (∀ y ∈ domain(P). x ≠ y → P(x) ∩ P(y) = 0)

A fact about lists of mutually disjoint sets.

lemma list_partition: assumes A1: n ∈ nat and
  A2: a : succ(n) → X a {is partition}
  shows (∪ i∈n. a(i)) ∩ a(n) = 0
proof -
  { assume (∪ i∈n. a(i)) ∩ a(n) ≠ 0
    then have ∃ x x ∈ (∪ i∈n. a(i)) ∩ a(n)
      by (rule nonempty_has_element)
    then obtain x where x ∈ (∪ i∈n. a(i)) and  I: x ∈ a(n)
      by auto
    then obtain i where i ∈ n and x ∈ a(i) by auto
    with A2 I have False
      using mem_imp_not_eq func1_1_L1 Partition_def
      by auto
  } thus thesis by auto
qed
We can turn every injection into a partition.

**lemma inj_partition:**
assumes $A1: \ b \in \text{inj}(X,Y)$
shows
$\forall x \in X. \ \{\{x, \{b(x)\}\}. \ x \in X\} = \{b(x)\}$ and
$\{\{x, \{b(x)\}\}. \ x \in X\} \ {\text{is partition}}$

**proof** -
let $p = \{\{x, \{b(x)\}\}. \ x \in X\}$
\{ fix $x$ assume $x \in X$
  from $A1$ have $b : X \to Y$ using inj_def
  by simp
  with $\langle x \in X, \ b(x) \rangle \in \text{Pow}(Y)$
  using apply_funtype by simp
} hence $\forall x \in X. \ \{b(x)\} \in \text{Pow}(Y)$ by simp
then have $p : X \to \text{Pow}(Y)$ using ZF_fun_from_total
  by simp
then have $\text{domain}(p) = X$ using func1_1_L1
  by simp
from $\langle p : X \to \text{Pow}(Y) \rangle$ show $I: \ \forall x \in X. \ p(x) = \{b(x)\}$
  using ZF_fun_from_tot_val0 by simp
\{ fix $x$ assume $x \in X$
  with $I$ have $p(x) = \{b(x)\}$ by simp
  hence $p(x) \neq 0$ by simp
  moreover
  \{ fix $t$ assume $t \in X$ and $x \neq t$
    with $A1$ $\langle x \in X, \ b(x) \neq b(t) \rangle$ using inj_def
    by auto
    with $I$ $\langle x \in X, \ t \in X \rangle$ have $p(x) \cap p(t) = 0$
    by auto \}
  ultimately have $p(x) \neq 0 \land (\forall t \in X. \ x\neq t \implies p(x) \cap p(t) = 0)$
  by simp
} with $\langle \text{domain}(p) = X \rangle$ show $\{\{x, \{b(x)\}\}. \ x \in X\} \ {\text{is partition}}$
  using Partition_def by simp
qed

end

24 Quasigroups

theory Quasigroup_ZF imports func1

begin

A quasigroup is an algebraic structure that that one gets after adding (sort of) divisibility to magma. Quasigroups differ from groups in that they are not necessarily associative and they do not have to have the neutral element.
24.1 Definitions and notation

According to Wikipedia there are at least two approaches to defining a quasigroup. One defines a quasigroup as a set with a binary operation, and the other, from universal algebra, defines a quasigroup as having three primitive operations. We will use the first approach.

A quasigroup operation does not have to have the neutral element. The left division is defined as the only solution to the equation \( a \cdot x = b \) (using multiplicative notation). The next definition specifies what does it mean that an operation \( A \) has a left division on a set \( G \).

**definition**

\[
\text{HasLeftDiv}(G,A) \equiv \forall a \in G. \forall b \in G. \exists! x. (x \in G \land A(a,x) = b)
\]

An operation \( A \) has the right inverse if for all elements \( a, b \in G \) the equation \( x \cdot a = b \) has a unique solution.

**definition**

\[
\text{HasRightDiv}(G,A) \equiv \forall a \in G. \forall b \in G. \exists! x. (x \in G \land A(x,a) = b)
\]

An operation that has both left and right division is said to have the Latin square property.

**definition**

\[
\text{HasLatinSquareProp (infix \{has Latin square property on\} 65)} \text{ where}
\]
\[
A \text{ has Latin square property on } G \equiv \text{HasLeftDiv}(G,A) \land \text{HasRightDiv}(G,A)
\]

A quasigroup is a set with a binary operation that has the Latin square property.

**definition**

\[
\text{IsAquasigroup}(G,A) \equiv A : G \times G \to G \land A \text{ has Latin square property on } G
\]

The uniqueness of the left inverse allows us to define the left division as a function. The union expression as the value of the function extracts the only element of the set of solutions of the equation \( x \cdot z = y \) for given \( (x,y) = p \in G \times G \) using the identity \( \bigcup \{x\} = x \).

**definition**

\[
\text{LeftDiv}(G,A) \equiv \{ (p, \bigcup \{ z \in G. A(fst(p),z) = snd(p)) \}. p \in G \times G \}
\]

Similarly the right division is defined as a function on \( G \times G \).

**definition**

\[
\text{RightDiv}(G,A) \equiv \{ (p, \bigcup \{ z \in G. A(z,fst(p)) = snd(p)) \}. p \in G \times G \}
\]

Left and right divisions are binary operations on \( G \).

**lemma** \( \text{lrdiv_binop: assumes IsAquasigroup(G,A) shows} \)

\( \text{LeftDiv}(G,A) : G \times G \to G \) and \( \text{RightDiv}(G,A) : G \times G \to G \)

**proof**

\{ fix p assume p \in G \times G \}
with asms have
\[ \bigcup \{ x \in G. A(\text{fst}(p), x) = \text{snd}(p) \} \in G \quad \text{and} \quad \bigcup \{ x \in G. A(x, \text{fst}(p)) = \text{snd}(p) \} \in G \]
unfolding IsAquasigroup_def HasLatinSquareProp_def HasLeftDiv_def HasRightDiv_def
using ZF1_1_L9(2) by auto
\}
then show \( \text{LeftDiv}(G,A):G \times G \to G \) and \( \text{RightDiv}(G,A):G \times G \to G \)
unfolding LeftDiv_def RightDiv_def using ZF_fun_from_total by auto
qed

We will use multiplicative notation for the quasigroup operation. The right
and left division will be denoted \( a/b \) and \( a \backslash b \), resp.

locale quasigroup0 =
fixes \( G \) \( A \)
assumes qgroupassum: IsAquasigroup\((G,A)\)
fixes qgroper (infixl \( \cdot \) 70)
defines qgroper_def[simp]: \( x \cdot y \equiv A(x,y) \)
fixes leftdiv (infixl \( \backslash \) 70)
defines leftdiv_def[simp]: \( x \backslash y \equiv \text{LeftDiv}(G,A)(x,y) \)
fixes rightdiv (infixl \( \slash \) 70)
defines rightdiv_def[simp]: \( x/\slash y \equiv \text{RightDiv}(G,A)(y,x) \)

The quasigroup operation is closed on \( G \).

lemma (in quasigroup0) qg_op_closed: assumes \( x \in G \) \( y \in G \)
shows \( x \cdot y \in G \)
using qgroupassum assms IsAquasigroup_def apply_funtype by auto

A couple of properties of right and left division:

lemma (in quasigroup0) lrdiv_props: assumes \( x \in G \) \( y \in G \)
shows
\[ \exists z. z \in G \land z \cdot x = y \quad y/x \in G \quad (y/x) \cdot x = y \quad \text{and} \]
\[ \exists z. z \in G \land x \cdot z = y \quad x/y \in G \quad x \cdot (x/y) = y \]
proof -
let \( z_r = \bigcup \{ z \in G. z \cdot x = y \} \)
from qgroupassum have I: RightDiv\((G,A)\):G \times G \to G using lrdiv_binop(2)
by simp
with asms have RightDiv\((G,A)\)(x,y) = \( z_r \)
unfolding RightDiv_def using ZF_fun_from_tot_val by auto
moreover
from qgroupassum asms show \( \exists z. z \in G \land z \cdot x = y \)
unfolding IsAquasigroup_def HasLatinSquareProp_def HasRightDiv_def
by simp
then have \( z_r \cdot x = y \) by (rule ZF1_1_L9)
ultimately show \( (y/x) \cdot x = y \) by simp
let \( z_l = \bigcup \{ z \in G. x \cdot z = y \} \)

284
from qgroupassum have II: LeftDiv(G,A):G×G→G using lrdiv_binop(1)
by simp
with assms have LeftDiv(G,A)(x,y) = z₁
  unfolding LeftDiv_def using ZF_fun_from_tot_val by auto
moreover
from qgroupassum assms show ∃!z. z∈G ∧ x·z = y
  unfolding IsAquasigroup_def HasLatinSquareProp_def HasLeftDiv_def
by simp
then have x·z₁ = y by (rule ZF1_1_L9)
ultimately show x\(x\y) = y by simp
from assms I II show y/x ∈ G and x\y ∈ G using apply_funtype by auto
qed

We can cancel the left element on both sides of an equation.

lemma (in quasigroup0) qg_cancel_left:
assumes x∈G y∈G z∈G and x·y = x·z
shows y=z
using qgroupassum assms qg_op_closed lrdiv_props(4) by blast

We can cancel the right element on both sides of an equation.

lemma (in quasigroup0) qg_cancel_right:
assumes x∈G y∈G z∈G and y·x = z·x
shows y=z
using qgroupassum assms qg_op_closed lrdiv_props(1) by blast

Two additional identities for right and left division:

lemma (in quasigroup0) lrdiv_ident: assumes x∈G y∈G
  shows (y\x)/x = y and x\(x\y) = y
proof -
  from assms have (y\x)/x ∈ G and ((y\x)/x)·x = y·x
    using qg_op_closed lrdiv_props(2,3) by auto
  with assms show (y\x)/x = y using qg_cancel_right by simp
  from assms have x\(x\y) ∈ G and x·(x\(x\y)) = y·y
    using qg_op_closed lrdiv_props(5,6) by auto
  with assms show x\(x\y) = y using qg_cancel_left by simp
qed

end

25 Loops

theory Loop_ZF imports Quasigroup_ZF
begin

This theory specifies the definition and proves basic properites of loops. Loop is very similar to groups, the only property that is missing is associativity of the operation.

285
25.1 Definitions and notation

In this section we define the notions of identity element and left and right inverse.

A loop is a quasigroup with an identity element.

definition IsAloop(G,A) ≡ IsAquasigroup(G,A) ∧ (∃e∈G. ∀x∈G. A(e,x) = x ∧ A(x,e) = x)

The neutral element for a binary operation $A : G \times G \to G$ is defined as the only element $e$ of $G$ such that $A(x,e) = x$ and $A(e,x) = x$ for all $x \in G$.

Note that although the loop definition guarantees the existence of (some) such element(s) at this point we do not know if this element is unique. We can define this notion here but it will become usable only after we prove uniqueness.

definition TheNeutralElement(G,f) ≡ (THE e. e ∈ G ∧ (∀g∈G. f(e,g) = g ∧ f(g,e) = g))

We will reuse the notation defined in the quasigroup0 locale, just adding the assumption about the existence of a neutral element and notation for it.

locale loop0 = quasigroup0 +
assumes ex_ident: ∃e∈G. ∀x∈G. e·x = x ∧ x·e = x
fixes neut (1)
defines neut_def[simp]: 1 ≡ TheNeutralElement(G,A)

In the loop context the pair $(G,A)$ forms a loop.

lemma (in loop0) is_loop: shows IsAloop(G,A)
  unfolding IsAloop_def using ex_ident qgroupassum by simp
If we know that a pair $(G,A)$ forms a loop then the assumptions of the loop0 locale hold.

lemma loop_loop0_valid: assumes IsAloop(G,A) shows loop0(G,A)
  using assms unfolding IsAloop_def loop0_axioms_def quasigroup0_def loop0_def by auto

The neutral element is unique in the loop.

lemma (in loop0) neut_uniq_loop: shows ∃!e. e ∈ G ∧ (∀x∈G. e·x = x ∧ x·e = x)
proof
  from ex_ident show ∃e. e ∈ G ∧ (∀x∈G. e·x = x ∧ x·e = x) by auto
next
  fix e y
  assume e ∈ G ∧ (∀x∈G. e·x = x ∧ x·e = x) y ∈ G ∧ (∀x∈G. y·x = x ∧ x·y = x)
  then have e·y = y and e·y = e by auto
The neutral element as defined in the $\text{loop0}$ locale is indeed neutral.

**Lemma (in loop0) neut_props_loop:** shows $1 \in G$ and $\forall x \in G. \ 1 \cdot x = x \land x \cdot 1 = x$

**Proof**
- let $n = \text{THE e. e} \in G \land (\forall x \in G. e \cdot x = x \land x \cdot e = x)$
- have $1 = \text{TheNeutralElement}(G,A)$ by simp
- then have $1 = n$ unfolding $\text{TheNeutralElement_def}$ by simp
- moreover have $n \in G \land (\forall x \in G. n \cdot x = x \land x \cdot n = x)$ using $\text{neut_uniq_loop}$
- theI by simp
- ultimately show $1 \in G$ and $\forall x \in G. \ 1 \cdot x = x \land x \cdot 1 = x$
- by auto

**QED**

Every element of a loop has unique left and right inverse (which need not be the same). Here we define the left inverse as a function on $G$.

**Definition**

LeftInv$(G,A) \equiv \{(x, \bigcup \{y \in G. \ A(y,x) = \text{TheNeutralElement}(G,A)\}). \ x \in G\}$

Definition of the right inverse as a function on $G$:

**Definition**

RightInv$(G,A) \equiv \{(x, \bigcup \{y \in G. \ A(x,y) = \text{TheNeutralElement}(G,A)\}). \ x \in G\}$

In a loop $G$ right and left inverses are functions on $G$.

**Lemma (in loop0) lr_inv_fun:** shows $\text{LeftInv}(G,A):G \rightarrow G \text{ RightInv}(G,A):G \rightarrow G$

unfolding $\text{LeftInv_def RightInv_def}$
using $\text{neut_props_loop lrdiv_props(1,4) ZF1_1_L9 ZF_fun_from_total}$
by auto

Right and left inverses have desired properties.

**Lemma (in loop0) lr_inv_props:** assumes $x \in G$
shows
- $\text{LeftInv}(G,A)(x) \in G$ (LeftInv$(G,A)(x)) \cdot x = 1$
- $\text{RightInv}(G,A)(x) \in G$ x $(\text{RightInv}(G,A)(x)) \cdot x = 1$

**Proof**
- from assms show $\text{LeftInv}(G,A)(x) \in G$ and $\text{RightInv}(G,A)(x) \in G$
  using $\text{lr_inv_fun apply_functype by auto}$
- from assms have $\exists y. \ y \in G \land y \cdot x = 1$
  using $\text{neut_props_loop(1) lrdiv_props(1)}$ by simp
- then have $(\bigcup \{y \in G. \ y \cdot x = 1\}) \cdot x = 1$
  by (rule $\text{ZF1_1_L9}$)
- with assms show $(\text{LeftInv}(G,A)(x)) \cdot x = 1$
  using $\text{lr_inv_fun(1) ZF_fun_from_tot_val unfolding LeftInv_def by simp}$
- from assms have $\exists y. \ y \in G \land x \cdot y = 1$
  using $\text{neut_props_loop(1) lrdiv_props(4)}$ by simp

287
then have $x \cdot (\bigcup \{ y \in G. \ x \cdot y = 1 \}) = 1$
by (rule ZF1_1_L9)
with assms show $x \cdot \text{RightInv}(G,A)(x) = 1$
using lr_inv_fun(2) ZF_fun_from_tot_val unfolding RightInv_def by simp
qed

code 26 Ordered loops

theory OrderedLoop_ZF imports Loop_ZF Order_ZF
begin

This theory file is about properties of loops (the algebraic structures introduced in IsarMathLib in the Loop_ZF theory) with an additional order relation that is in a way compatible with the loop's binary operation. The oldest reference I have found on the subject is [6].

26.1 Definition and notation

An ordered loop $(G,A)$ is a loop with a partial order relation $r$ that is "translation invariant" with respect to the loop operation $A$.

A triple $(G,A,r)$ is an ordered loop if $(G,A)$ is a loop and $r$ is a relation on $G$ (i.e. a subset of $G \times G$ with is a partial order and for all elements $x, y, z \in G$ the condition $\langle x, y \rangle \in r$ is equivalent to both $\langle A(x, z), A(x, z) \rangle \in r$ and $\langle A(z, x), A(z, x) \rangle \in r$. This looks a bit awkward in the basic set theory notation, but using the additive notation for the group operation and $x \leq y$ to instead of $\langle x, y \rangle \in r$ this just means that $x \leq y$ if and only if $x + z \leq y + z$ and $x \leq y$ if and only if $z + x \leq z + y$.

definition
IsAnOrdLoop($L,A,r$) $\equiv$
IsAloop($L,A$) $\land$ $r \subseteq L \times L$ $\land$ IsPartOrder($L,r$) $\land$ $\forall x \in L. \ \forall y \in L. \ \forall z \in L.$
$\langle A(x,z), A(y,z) \rangle \in r$ $\land$ $\langle A(z,x), A(z,y) \rangle \in r$ $\land$ $((x,y) \in r \iff \langle A(x,z), A(y,z) \rangle \in r)$ $\land$ $((x,y) \in r \iff \langle A(z,x), A(z,y) \rangle \in r)$

We define the set of nonnegative elements in the obvious way as $L^+ = \{ x \in L : 0 \leq x \}$.

definition
Nonnegative($L,A,r$) $\equiv$ \{ $x \in L.$ (TheNeutralElement($L,A$),$x$ $\in r$) \}

The PositiveSet($L,A,r$) is a set similar to Nonnegative($L,A,r$), but without the neutral element.

definition
PositiveSet(L,A,r) ≡ 
{ x ∈ L. ⟨ TheNeutralElement(L,A), x ⟩ ∈ r ∧ TheNeutralElement(L,A) ≠ x }

We will use the additive notation for ordered loops.

locale loop1 = 
fixes L and A and r

assumes ordLoopAssum: IsAnOrdLoop(L,A,r)

fixes neut (0)
defines neut_def[simp]: 0 ≡ TheNeutralElement(L,A)

fixes looper (infixl 69)
defines looper_def[simp]: x + y ≡ A( x,y )

fixes lesseq (infix ≤ 68)
defines lesseq_def[simp]: x ≤ y ≡ ⟨ x,y ⟩ ∈ r

fixes sless (infix < 68)
defines sless_def[simp]: x < y ≡ x≤y ∧ x≠y

fixes nonnegative (L+)
defines nonnegative_def[simp]: L+ ≡ Nonnegative(L,A,r)

fixes positive (L+)
defines positive_def[simp]: L+ ≡ PositiveSet(L,A,r)

fixes leftdiv (- _ + _)
defines leftdiv_def[simp]: -x+y ≡ LeftDiv(L,A)(x,y)

fixes rightdiv (infixl - 69)
defines rightdiv_def[simp]:x-y ≡ RightDiv(L,A)(y,x)

Theorems proven in the loop0 locale are valid in the loop1 locale

sublocale loop1 < loop0 L A looper
using ordLoopAssum loop_loop0_valid unfolding IsAnOrdLoop_def by auto

In this context x ≤ y implies that both x and y belong to L.

lemma (in loop1) lsq_members: assumes x≤y shows x∈L and y∈L
using ordLoopAssum assms IsAnOrdLoop_def by auto

In this context x < y implies that both x and y belong to L.

lemma (in loop1) less_members: assumes x<y shows x∈L and y∈L
using ordLoopAssum assms IsAnOrdLoop_def by auto

In an ordered loop the order is translation invariant.

lemma (in loop1) ord_trans_inv: assumes x≤y and z∈L
shows x+z ≤ y+z and z+x ≤ z+y

289
proof -
  from ordLoopAssum assms have
    \( (\langle x, y \rangle \in r \leftrightarrow \langle A(x, z), A(y, z) \rangle \in r) \land (\langle x, y \rangle \in r \leftrightarrow \langle A(z, x), A(z, y) \rangle \in r) \)
    using lsq_members unfolding IsAnOrdLoop_def by blast
  with assms(1) show \( x+z \leq y+z \) and \( z+x \leq z+y \) by auto
qed

In an ordered loop the strict order is translation invariant.

lemma (in loop1) strict_ord_trans_inv: assumes \( x<y \) \( z \in L \)
  shows \( x+z < y+z \) and \( z+x < z+y \)
proof -
  from assms have \( x+z \leq y+z \) and \( z+x \leq z+y \)
    using ord_trans_inv by auto
  moreover have \( x+z \neq y+z \) and \( z+x \neq z+y \)
    proof -
      \{
        assume \( x+z = y+z \)
        with assms have \( x=y \) using less_members qg_cancel_right by blast
        with assms(1) have False by simp
      \}
      thus \( x+z \neq y+z \) by auto
      \{
        assume \( z+x = z+y \)
        with assms have \( x=y \) using less_members qg_cancel_left by blast
        with assms(1) have False by simp
      \}
      thus \( z+x \neq z+y \) by auto
    qed
  ultimately show \( x+z < y+z \) and \( z+x < z+y \)
    by auto
qed

We can cancel an element from both sides of an inequality on the right side.

lemma (in loop1) ineq_cancel_right: assumes \( x\in L \) \( y\in L \) \( z\in L \) and \( x+z \leq y+z \)
  shows \( x\leq y \)
proof -
  from ordLoopAssum assms(1,2,3) have \( \langle x, y \rangle \in r \leftrightarrow \langle A(x, z), A(y, z) \rangle \in r \)
    unfolding IsAnOrdLoop_def by blast
  with assms(4) show \( x\leq y \) by simp
qed

We can cancel an element from both sides of a strict inequality on the right side.

lemma (in loop1) strict_ineq_cancel_right: assumes \( x\in L \) \( y\in L \) \( z\in L \) and \( x+z < y+z \)
  shows \( x<y \)
    using assms ineq_cancel_right by auto

We can cancel an element from both sides of an inequality on the left side.
lemma (in loop1) ineq_cancel_left: assumes $x \in L$, $y \in L$, $z \in L$ and $z \cdot x \leq z \cdot y$
shows $x \leq y$
proof -
from ordLoopAssum assms(1,2,3) have $(x, y) \in r \iff (A(z, x), A(z, y)) \in r$
unfolding IsAnOrdLoop_def by blast
with assms(4) show $x \leq y$ by simp
qed

We can cancel an element from both sides of a strict inequality on the left side.

lemma (in loop1) strict_ineq_cancel_left:
assumes $x \in L$, $y \in L$, $z \in L$ and $z \cdot x < z \cdot y$
shows $x < y$
using assms ineq_cancel_left by auto

The definition of the nonnegative set in the notation used in the loop1 locale:

lemma (in loop1) nonneg_definition:
shows $x \in L$ $\leftrightarrow$ $0 \leq x$
using ordLoopAssum IsAnOrdLoop_def Nonnegative_def by auto

The nonnegative set is contained in the loop.

lemma (in loop1) nonneg_subset: shows $L^+ \subseteq L$
using Nonnegative_def by auto

The positive set is contained in the loop.

lemma (in loop1) positive_subset: shows $L_+ \subseteq L$
using PositiveSet_def by auto

The definition of the positive set in the notation used in the loop1 locale:

lemma (in loop1) posset_definition:
shows $x \in L_+ \iff (0 \leq x \land x \neq 0)$
using ordLoopAssum IsAnOrdLoop_def PositiveSet_def by auto

Another form of the definition of the positive set in the notation used in the loop1 locale:

lemma (in loop1) posset_definition1:
shows $x \in L_+ \iff 0 < x$
using ordLoopAssum IsAnOrdLoop_def PositiveSet_def by auto

The order in an ordered loop is antisymmetric.

lemma (in loop1) loop_ord_antisym: assumes $x \leq y$ and $y \leq x$
shows $x = y$
proof -
from ordLoopAssum assms have antisym$(r) (x, y) \in r \iff (y, x) \in r$
unfolding IsAnOrdLoop_def IsPartOrder_def by auto

291
then show $x = y$ by (rule Fol1_L4)

qed

The loop order is transitive.

lemma (in loop1) loop_ord_trans: assumes $x \leq y$ and $y \leq z$ shows $x \leq z$
proof -
  from ordLoopAssum assms have $\text{trans}(r)$ and $(x, y) \in r \land (y, z) \in r$
  unfolding IsAnOrdLoop_def IsPartOrder_def by auto
  then have $(x, z) \in r$ by (rule Fol1_L3)
  thus thesis by simp
qed

The loop order is reflexive.

lemma (in loop1) loop_ord_refl: assumes $x \in L$ shows $x \leq x$
using assms ordLoopAssum unfolding IsAnOrdLoop_def IsPartOrder_def refl_def
by simp

A form of mixed transitivity for the strict order:

lemma (in loop1) loop_strict_ord_trans: assumes $x \leq y$ and $y < z$
s Shows $x < z$
proof -
  from assms have $x \leq y$ and $y < z$ by auto
  then have $x < z$ by (rule loop_ord_trans)
  with assms show $x < z$ using loop_ord_antisym by auto
qed

Another form of mixed transitivity for the strict order:

lemma (in loop1) loop_strict_ord_trans1: assumes $x < y$ and $y \leq z$
shows $x < z$
proof -
  from assms have $x \leq y$ and $y \leq z$ by auto
  then have $x < z$ by (rule loop_ord_trans)
  with assms show $x < z$ using loop_ord_antisym by auto
qed

Yet another form of mixed transitivity for the strict order:

lemma (in loop1) loop_strict_ord_trans2: assumes $x < y$ and $y < z$
shows $x < z$
proof -
  from assms have $x \leq y$ and $y \leq z$ by auto
  then have $x < z$ by (rule loop_ord_trans)
  with assms show $x < z$ using loop_ord_antisym by auto
qed

We can move an element to the other side of an inequality. Well, not exactly,
but our notation creates an illusion to that effect.

lemma (in loop1) lsq_other_side: assumes $x \leq y$
shows $0 \leq -x+y \ (-x+y) \in L^+$ $0 \leq y-x \ (y-x) \in L^+$

proof -
from assms have \( x\in L \ y\in L \ 0\in L \ (-x+y) \in L \ \ (y-x) \in L \)
using lsq_members neut_props_loop(1) lrdiv_props(2,5) by auto
then have \( x = x+0 \) and \( y = x+(-x+y) \) using neut_props_loop(2) lrdiv_props(6)
by auto
with assms have \( x+0 \leq x+(-x+y) \) by simp
with \( x\in L \ \langle 0\in L \ \langle (-x+y) \in L \ show \ 0 \leq -x+y \ using \ ineq_cancel_left \)
by simp
then show \( (-x+y) \in L^+ \ using \ nonneg_definition \ by \ simp \)
from \( x\in L \ \langle y\in L \ have \ x = 0+x \ and \ y = (y-x)+x \)
using neut_props_loop(2) lrdiv_props(3) by auto
with assms have \( 0+x \leq (y-x)+x \) by simp
with \( x\in L \ \langle 0\in L \ \langle (y-x) \in L \ show \ 0 \leq y-x \ using \ ineq_cancel_right \)
by simp
then show \( (y-x) \in L^+ \ using \ nonneg_definition \ by \ simp \)
qed

We can move an element to the other side of a strict inequality.

lemma (in loop1) ls_other_side: assumes \( x<y \)
shows \( 0 < -x+y \ (-x+y) \in L^+ \ 0 < y-x \ (y-x) \in L^+ \)

proof -
from assms have \( x\in L \ y\in L \ 0\in L \ (-x+y) \in L \ \ (y-x) \in L \)
using lsq_members neut_props_loop(1) lrdiv_props(2,5) by auto
then have \( x = x+0 \) and \( y = x+(-x+y) \) using neut_props_loop(2) lrdiv_props(6)
by auto
with assms have \( x+0 \leq x+(-x+y) \) by simp
with \( x\in L \ \langle 0\in L \ \langle (-x+y) \in L \ show \ 0 < -x+y \ using \ strict_ineq_cancel_left \)
by simp
then show \( (-x+y) \in L^+ \ using \ posset_definition1 \ by \ simp \)
from \( x\in L \ \langle y\in L \ have \ x = 0+x \ and \ y = (y-x)+x \)
using neut_props_loop(2) lrdiv_props(3) by auto
with assms have \( 0+x < (y-x)+x \) by simp
with \( x\in L \ \langle 0\in L \ \langle (y-x) \in L \ show \ 0 < y-x \ using \ strict_ineq_cancel_right \)
by simp
then show \( (y-x) \in L^+ \ using \ posset_definition1 \ by \ simp \)
qed

We can add sides of inequalities.

lemma (in loop1) add_ineq: assumes \( x\leq y \ z\leq t \)
shows \( x+z \leq y+t \)

proof -
from assms have \( x+z \leq y+z \)
using lsq_members(1) ord_trans_inv(1) by simp
with assms show thesis using lsq_members(2) ord_trans_inv(2) loop_ord_trans
by simp
qed

We can add sides of strict inequalities. The proof uses a lemma that relies
on the antisymmetry of the order relation.

**lemma (in loop1) add_ineq_strict:** assumes \( x < y \) \( z < t \)
shows \( x + z < y + t \)
proof -
  from assms have \( x + z < y + z \)
  using less_members(1) strict_ord_trans_inv(1) by auto
moreover from assms have \( y + z < y + t \)
  using less_members(2) strict_ord_trans_inv(2) by auto
ultimately show thesis by (rule loop_strict_ord_trans2)
qed

We can add sides of inequalities one of which is strict.

**lemma (in loop1) add_ineq_strict1:** assumes \( x \leq y \) \( z < t \)
shows \( x + z < y + t \) and \( z + x < t + y \)
proof -
  from assms have \( x + z \leq y + z \)
  using less_members(1) ord_trans_inv(1) by auto
with assms show \( x + z < y + t \)
  using lsq_members(2) strict_ord_trans_inv(2) loop_strict_ord_trans
  by blast
from assms have \( z + x < t + x \)
  using lsq_members(1) strict_ord_trans_inv(1) by simp
with assms show \( z + x < t + y \)
  using less_members(2) ord_trans_inv(2) loop_strict_ord_trans1
  by blast
qed

Subtracting a positive element decreases the value.

**lemma (in loop1) subtract_pos:** assumes \( x \in L \) \( 0 < y \)
shows \( x - y < x \) and \( -y + x < x \)
proof -
  from assms(2) have \( y \in L \) using less_members(2) by simp
from assms(1) have \( x \leq x \)
  using ordLoopAssum unfolding IsAnOrdLoop_def IsPartOrder_def refl_def
  by simp
with assms(2) have \( x + 0 < x + y \)
  using add_ineq_strict1(1) by simp
with assms \( y \in L \) show \( x - y < x \)
  using neut_props_loop(2) lrdiv_props(3) lrdiv_props(2) strict_ineq_cancel_right
  by simp
from assms(2) \( x \leq x \) have \( 0 + x < y + x \)
  using add_ineq_strict1(2) by simp
with assms \( y \in L \) show \( -y + x < x \)
  using neut_props_loop(2) lrdiv_props(6) lrdiv_props(5) strict_ineq_cancel_left
  by simp
qed

end
27 Semigroups

theory Semigroup_ZF imports Partitions_ZF Fold_ZF Enumeration_ZF

begin

It seems that the minimal setup needed to talk about a product of a sequence is a set with a binary operation. Such object is called "magma". However, interesting properties show up when the binary operation is associative and such algebraic structure is called a semigroup. In this theory file we define and study sequences of partial products of sequences of magma and semigroup elements.

27.1 Products of sequences of semigroup elements

Semigroup is a a magma in which the binary operation is associative. In this section we mostly study the products of sequences of elements of semigroup. The goal is to establish the fact that taking the product of a sequence is distributive with respect to concatenation of sequences, i.e for two sequences $a, b$ of the semigroup elements we have $\prod (a\sqcup b) = (\prod a) \cdot (\prod b)$, where "$a\sqcup b$" is concatenation of $a$ and $b$ ($a++b$ in Haskell notation). Less formally, we want to show that we can discard parantheses in expressions of the form $(a_0 \cdot a_1 \cdot \ldots \cdot a_n) \cdot (b_0 \cdot \ldots \cdot b_k)$.

First we define a notion similar to Fold, except that that the initial element of the fold is given by the first element of sequence. By analogy with Haskell fold we call that Fold1

definition

Fold1(f,a) ≡ Fold(f,a(0),Tail(a))

The definition of the semigr0 context below introduces notation for writing about finite sequences and semigroup products. In the context we fix the carrier and denote it $G$. The binary operation on $G$ is called $f$. All theorems proven in the context semigr0 will implicitly assume that $f$ is an associative operation on $G$. We will use multiplicative notation for the semigroup operation. The product of a sequence $a$ is denoted $\prod a$. We will write $a \xrightarrow{x} \rightarrow$ for the result of appending an element $x$ to the finite sequence (list) $a$. This is a bit nonstandard, but I don’t have a better idea for the "append" notation. Finally, $a \sqcup b$ will denote the concatenation of the lists $a$ and $b$.

locale semigr0 =

fixes G f

assumes assoc_assum: f {is associative on} G

fixes prod (infixl \cdot 72)
defines prod_def [simp]: \( x \cdot y \equiv f(x, y) \)

fixes seqprod (\( \prod \)_71)
defines seqprod_def [simp]: \( \prod a \equiv \text{Fold1}(f, a) \)

fixes append (infix \( \leftarrow \)_72)
defines append_def [simp]: \( a \leftarrow x \equiv \text{Append}(a, x) \)

fixes concat (infixl \( \sqcup \)_69)
defines concat_def [simp]: \( a \sqcup b \equiv \text{Concat}(a, b) \)

The next lemma shows our assumption on the associativity of the semigroup operation in the notation defined in the \( \text{semigr0} \) context.

lemma (in \( \text{semigr0} \)) semigr_assoc:
  assumes \( x \in G, y \in G, z \in G \)
  shows \( x \cdot y \cdot z = x \cdot (y \cdot z) \)
  using assms assoc_assum IsAssociative_def by simp

In the way we define associativity the assumption that \( f \) is associative on \( G \) also implies that it is a binary operation on \( X \).

lemma (in \( \text{semigr0} \)) semigr_binop: shows \( f : G \times G \to G \)
  using assoc_assum IsAssociative_def by simp

Semigroup operation is closed.

lemma (in \( \text{semigr0} \)) semigr_closed:
  assumes \( a \in G, b \in G \)
  shows \( a \cdot b \in G \)
  using assms semigr_binop apply_funtype by simp

Lemma append_1elem written in the notation used in the \( \text{semigr0} \) context.

lemma (in \( \text{semigr0} \)) append_1elem_nice:
  assumes \( n \in \mathbb{N} \text{ and } a : n \to X \text{ and } b : 1 \to X \)
  shows \( a \sqcup b = a \leftarrow b(0) \)
  using assms append_1elem by simp

Lemma concat_init_last_elem rewritten in the notation used in the \( \text{semigr0} \) context.

lemma (in \( \text{semigr0} \)) concat_init_last:
  assumes \( n \in \mathbb{N} \text{ and } k \in \mathbb{N} \text{ and } a : n \to X \text{ and } b : \text{succ}(k) \to X \)
  shows \( (a \sqcup \text{Init}(b)) \leftarrow b(k) = a \sqcup b \)
  using assms concat_init_last_elem by simp

The product of semigroup (actually, magma – we don’t need associativity for this) elements is in the semigroup.

lemma (in \( \text{semigr0} \)) prod_type:
  assumes \( n \in \mathbb{N} \text{ and } a : \text{succ}(n) \to G \)
  shows \( \prod a \in G \)
  proof -
from assms have
  \( \text{succ}(n) \in \text{nat} \) 
  \( f : G \times G \rightarrow G \)
  \( \text{Tail}(a) : n \rightarrow G \)
  using \( \text{semigr_binop} \) \( \text{tail_props} \) by auto
moreover from assms have \( a(0) \in G \) and \( G \neq 0 \)
  using \( \text{empty_in_every_succ} \) \( \text{apply_funtype} \)
  by auto
ultimately show \( (\prod a) \in G \) using \( \text{Fold1_def} \) \( \text{Fold_props} \)
  by simp
qed

What is the product of one element list?

lemma (in \( \text{semigr0} \)) prod_of_1elem: assumes \( A1: a : 1 \rightarrow G \)
shows \( (\prod a) = a(0) \)
proof -
  have \( f : G \times G \rightarrow G \) using \( \text{semigr_binop} \) by simp
  moreover from \( A1 \) have \( \text{Tail}(a) : 0 \rightarrow G \)
    using \( \text{tail_props} \) by blast
  moreover from \( A1 \) have \( a(0) \in G \) and \( G \neq 0 \)
    using \( \text{apply_funtype} \) by auto
  ultimately show \( (\prod a) = a(0) \) using \( \text{fold_empty} \) \( \text{Fold1_def} \)
    by simp
qed

What happens to the product of a list when we append an element to the list?

lemma (in \( \text{semigr0} \)) prod_append: assumes \( A1: n \in \text{nat} \) and
\( A2: a : \text{succ}(n) \rightarrow G \) \( A3: x \in G \)
shows \( (\prod a \leftarrow x) = (\prod a) \cdot x \)
proof -
  from \( A1 \) \( A2 \) have \( I: \text{Tail}(a) : n \rightarrow G \) \( a(0) \in G \)
    using \( \text{tail_props} \) \( \text{empty_in_every_succ} \) \( \text{apply_funtype} \)
    by auto
  from assms have \( (\prod a \leftarrow x) = \text{Fold}(f,a(0),\text{Tail}(a) \leftarrow x) \)
    using \( \text{head_of_append} \) \( \text{tail_append_commute} \) \( \text{Fold1_def} \)
    by simp
  also from \( A1 \) \( A3 \) I have \( ... = (\prod a) \cdot x \)
    using \( \text{semigr_binop} \) \( \text{fold_append} \) \( \text{Fold1_def} \)
    by simp
  finally show thesis by simp
qed

The main theorem of the section: taking the product of a sequence is distributive with respect to concatenation of sequences. The proof is by induction on the length of the second list.

theorem (in \( \text{semigr0} \)) prod_conc_distr:
  assumes \( A1: n \in \text{nat} \) \( k \in \text{nat} \) and
\( A2: a : \text{succ}(n) \rightarrow G \) \( b : \text{succ}(k) \rightarrow G \)
shows \( (\prod a) \cdot (\prod b) = \prod (a \sqcup b) \)
proof -
from A1 have k ∈ nat by simp
moreover have ∀b ∈ succ(0) → G. (∏ a) · (∏ b) = (∏ (a ⊔ b))
proof -
  { fix b assume A3: b : succ(0) → G
    with A1 A2 have
    succ(n) ∈ nat  a : succ(n) → G  b : 1 → G
    by auto
    then have a ⊔ b = a ← b(0) by (rule append_first)
    with A1 A2 A3 have (∏ a) · (∏ b) = (∏ (a ⊔ b))
    using apply_funtype prod_append semigr_binop prod_of_1elem
    by simp
  } thus thesis by simp
qed
moreover have ∀j ∈ nat. (∀ b ∈ succ(j) → G. (∏ a) · (∏ b) = (∏ (a ⊔ b))) −→ (∀ b ∈ succ(succ(j)) → G. (∏ a) · (∏ b) = (∏ (a ⊔ b)))
proof -
  { fix j assume A4: j ∈ nat
    A5: (∀ b ∈ succ(j) → G. (∏ a) · (∏ b) = (∏ (a ⊔ b)))
    { fix b assume A6: b : succ(succ(j)) → G
      let c = Init(b)
      from A4 A6 have T: b(succ(j)) ∈ G and
      I: c : succ(j) → G and II: b = c ← b(succ(j))
      using apply_funtype init_props by auto
      from A1 A2 A4 A6 have
      succ(n) ∈ nat  succ(j) ∈ nat
      a : succ(n) → G  b : succ(succ(j)) → G
      by auto
      then have III: (a ⊔ c) ← b(succ(j)) = a ⊔ b
      by (rule concat_init_last)
      from A4 I T have (∏ c ← b(succ(j))) = (∏ c) · b(succ(j))
      by (rule prod_append)
      with II have
      (∏ a) · (∏ b) = (∏ a) · ((∏ c) · b(succ(j)))
      by simp
    } moreover from A1 A2 A4 T I have
    (∏ a) ∈ G  (∏ c) ∈ G  b(succ(j)) ∈ G
    using prod_type by auto
    ultimately have
    (∏ a) · (∏ b) = ( (∏ a) · (∏ c) ) · b(succ(j))
    using semigr_assoc by auto
    with A5 I have (∏ a) · (∏ b) = (∏ (a ⊔ c)) · b(succ(j))
    by simp
  } moreover from A1 A2 A4 I have
T1: succ(n) ∈ nat  succ(j) ∈ nat and
a : succ(n) → G  c : succ(j) → G
by auto

298
then have \( \text{Concat}(a, c): \text{succ}(n) #\succ(j) \to G \)
by (rule concat_props)
with A1 A4 T have
\( \text{succ}(n #\succ(j)) \in \text{nat} \)
\( a \sqcup c: \text{succ}(\text{succ}(n #\succ(j))) \to G \)
\( b(\text{succ}(j)) \in G \)
using succ_plus by auto
then have \((\prod (a \sqcup c) \leftarrow b(\text{succ}(j))) = (\prod (a \sqcup c)) \cdot b(\text{succ}(j))\)
by (rule prod_append)
ultimately have \((\prod a) \cdot (\prod b) = (\prod (a \sqcup b))\)
by simp
hence \((\forall b \in \text{succ}(\text{succ}(j)) \to G. (\prod a) \cdot (\prod b) = (\prod (a \sqcup b))\)
by simp
thus thesis by blast
qed
ultimately have \((\forall b \in \text{succ}(k) \to G. (\prod a) \cdot (\prod b) = (\prod (a \sqcup b))\)
by (rule ind_on_nat)
with A2 show \((\prod a) \cdot (\prod b) = (\prod (a \sqcup b))\)
by simp
qed

\( a \cdot b \cdot (c \cdot d) = a \cdot (b \cdot c) \cdot d \) for semigroup elements \( a, b, c, d \in G \). The Commutative semigroups section below contains a couple of rearrangements that need commutativity of the semigroup operation, but this one uses only associativity, so it's here.

**Lemma (in semigr0)** rearr4elem_assoc:
assumes \( a \in G, b \in G, c \in G, d \in G \)
shows \( a \cdot b \cdot (c \cdot d) = a \cdot (b \cdot c) \cdot d \)
proof -
from asms have \( a \cdot b \cdot (c \cdot d) = a \cdot b \cdot c \cdot d \) using semigr_closed semigr_assoc
by simp
with asms(1,2,3) show thesis using semigr_assoc by simp
qed

### 27.2 Products over sets of indices

In this section we study the properties of expressions of the form \( \prod_{i \in \Lambda} a_i = a_{i_0} \cdot a_{i_1} \cdot \ldots \cdot a_{i_{n-1}} \), i.e. what we denote as \( \prod(\Lambda, a) \). \( \Lambda \) here is a finite subset of some set \( X \) and \( a \) is a function defined on \( X \) with values in the semigroup \( G \).

Suppose \( a : X \to G \) is an indexed family of elements of a semigroup \( G \) and \( \Lambda = \{i_0, i_1, \ldots, i_{n-1}\} \subseteq \mathbb{N} \) is a finite set of indices. We want to define \( \prod_{i \in \Lambda} a_i = a_{i_0} \cdot a_{i_1} \cdot \ldots \cdot a_{i_{n-1}} \). To do that we use the notion of Enumeration defined in the Enumeration_ZF theory file that takes a set of indices and lists them in increasing order, thus converting it to list. Then we use the Fold1...
to multiply the resulting list. Recall that in Isabelle/ZF the capital letter "O" denotes the composition of two functions (or relations).

**definition**

\[
\text{SetFold}(f,a,\Lambda,r) = \text{Fold1}(f,a \circ \text{Enumeration}(\Lambda,r))
\]

For a finite subset \(\Lambda\) of a linearly ordered set \(X\) we will write \(\sigma(\Lambda)\) to denote the enumeration of the elements of \(\Lambda\), i.e. the only order isomorphism \(|\Lambda| \to \Lambda\), where \(|\Lambda| \in \mathbb{N}\) is the number of elements of \(\Lambda\). We also define notation for taking a product over a set of indices of some sequence of semigroup elements. The product of semigroup elements over some set \(\Lambda \subseteq X\) of indices of a sequence \(a : X \to G\) (i.e. \(\prod_{i\in\Lambda} a_i\)) is denoted \(\prod(\Lambda,a)\). In the semigr1 context we assume that \(a\) is a function defined on some linearly ordered set \(X\) with values in the semigroup \(G\).

**locale** semigr1 = semigr0 +

```isar
fixes X r
assumes linord: IsLinOrder(X,r)

fixes a
assumes a_is_fun: a : X → G

fixes σ
defines σ_def [simp]: \(\sigma(\Lambda) \equiv \text{Enumeration}(\Lambda,r)\)

fixes setpr (\prod)
defines setpr_def [simp]: \(\prod(\Lambda,b) \equiv \text{SetFold}(f,b,\Lambda,r)\)
```

We can use the enums locale in the semigr0 context.

**lemma** (in semigr1) enums_valid_in_semigr1: shows enums(X,r)

```isar
using linord enums_def by simp
```

Definition of product over a set expressed in notation of the semigr0 locale.

**lemma** (in semigr1) setproddef:

```isar
shows \(\prod(\Lambda,a) = \prod (a \circ \sigma(\Lambda))\)
using SetFold_def by simp
```

A composition of enumeration of a nonempty finite subset of \(\mathbb{N}\) with a sequence of elements of \(G\) is a nonempty list of elements of \(G\). This implies that a product over set of a finite set of indices belongs to the (carrier of) semigroup.

**lemma** (in semigr1) setprod_type: assumes

- \(A1: \Lambda \in \text{FinPow}(X)\) and \(A2: \Lambda\neq\emptyset\)
- shows
  \(\exists n \in \mathbb{N} . |\Lambda| = \text{succ}(n) \land a \circ \sigma(\Lambda) : \text{succ}(n) \to G\)
  \(\text{and } \prod(\Lambda,a) \in G\)

```isar
proof -
```
from assms obtain \( n \) where \( n \in \text{nat} \) and \( |\Lambda| = \text{succ}(n) \)
using card_non_empty_succ by auto
from \( A1 \) have \( \sigma(\Lambda) : |\Lambda| \rightarrow A \)
using enums_valid_in_semigr1 enums.enum_props
by simp
with \( A1 \) have \( a \circ \sigma(\Lambda) : |\Lambda| \rightarrow G \)
using a_is_fun FinPow_def comp_fun_subset
by simp
with \( \langle n \in \text{nat} \rangle \) and \( \langle |\Lambda| = \text{succ}(n) \rangle \) show
\( \exists n \in \text{nat} . |\Lambda| = \text{succ}(n) \land a \circ \sigma(\Lambda) : \text{succ}(n) \rightarrow G \)
by auto
from \( \langle n \in \text{nat} \rangle \) and \( \langle |\Lambda| = \text{succ}(n) \rangle \) and \( \langle a \circ \sigma(\Lambda) : |\Lambda| \rightarrow G \rangle \)
show \( \prod(\Lambda, a) \in G \) using prod_type setproddef
by auto
qed

The enum_append lemma from the Enumeration theory specialized for natural numbers.

\begin{lemma}\text{(in semigr1)}\quad \text{semigr1_enum_append}:
\begin{itemize}
\item [assumes] \( \Lambda \in \text{FinPow}(X) \) and
\item \( n \in X - \Lambda \) and \( \forall k \in \Lambda . \langle k, n \rangle \in r \)
\item shows \( \sigma(\Lambda \cup \{n\}) = \sigma(\Lambda) \leftarrow n \)
\end{itemize}
using assms FinPow_def enums_valid_in_semigr1 enums.enum_append by simp
\end{lemma}

What is product over a singleton?

\begin{lemma}\text{(in semigr1)}\quad \text{gen_prod_singleton}:
\begin{itemize}
\item [assumes] \( A1: x \in X \)
\item shows \( \prod(\{x\}, a) = a(x) \)
\end{itemize}
proof -
from \( A1 \) have \( \sigma(\{x\}) : 1 \rightarrow X \) and \( \sigma(\{x\})(0) = x \)
using enums_valid_in_semigr1 enums.enum_singleton
by auto
then show \( \prod(\{x\}, a) = a(x) \)
using a_is_fun comp_fun setproddef prod_of_1elem
comp_fun_apply by simp
qed

A generalization of prod_append to the products over sets of indices.

\begin{lemma}\text{(in semigr1)}\quad \text{gen_prod_append}:
\begin{itemize}
\item assumes \( A1: \Lambda \in \text{FinPow}(X) \) and \( A2: \Lambda \neq 0 \) and
\item \( A3: n \in X - \Lambda \) and
\item \( A4: \forall k \in \Lambda . \langle k, n \rangle \in r \)
\item shows \( \prod(\Lambda \cup \{n\}, a) = (\prod(\Lambda, a)) \cdot a(n) \)
\end{itemize}
proof -
have \( \prod(\Lambda \cup \{n\}, a) = \prod (a \circ \sigma(\Lambda \cup \{n\})) \)
using setproddef by simp
also from \( A1 \) and \( A3 \) and \( A4 \) have \( \ldots = \prod (a \circ \sigma(\Lambda) \leftarrow n) \)
301
using semigr1_enum_append by simp
also have ... = ∏((a 0 σ)("→" a(n))
proof -
from A1 A3 have
|Λ| ∈ nat and σ(Λ) : |Λ| → X and n ∈ X
using card_fin_is_nat enums_valid_in_semigr1 enums.enum_fun
by auto
then show thesis using a_is_fun list_compose_append
by simp
qed
also from assms have ...
using a_is_fun setprod_type apply_funtype prod_append
by blast
also have ... = (∏(Λ,a)) · a(n)
using SetFold_def by simp
finally show ∏(Λ ∪ {n}, a) = (∏(Λ,a)) · a(n)
by simp
qed

Very similar to gen_prod_append: a relation between a product over a set of indices and the product over the set with the maximum removed.

lemma (in semigr1) gen_product_rem_point:
assumes A1: A ∈ FinPow(X) and
A2: n ∈ A and A4: A - {n} ≠ 0 and
A3: ∀k∈A. ⟨k, n⟩ ∈ r
shows
(∏(A - {n},a)) · a(n) = ∏(A, a)
proof -
let Λ = A - {n}
from A1 A2 have Λ ∈ FinPow(X) and n ∈ X - Λ
using fin_rem_point_fin FinPow_def by auto
with A3 A4 have ∏(Λ ∪ {n}, a) = (∏(Λ,a)) · a(n)
using a_is_fun gen_prod_append by blast
with A2 show thesis using rem_add_eq by simp
qed

27.3 Commutative semigroups

Commutative semigroups are those whose operation is commutative, i.e. 
\( a \cdot b = b \cdot a \). This implies that for any permutation \( s : n \to n \) we have 
\( \prod_{j=0}^n a_j = \prod_{j=0}^n a_{s(j)} \), or, closer to the notation we are using in the semigr0 context, \( \prod a = \prod (a \circ s) \). Maybe one day we will be able to prove this, but for now the goal is to prove something simpler: that if the semigroup operation is commutative taking the product of a sequence is distributive with respect to the operation: 
\( \prod_{j=0}^n (a_j \cdot b_j) = \left( \prod_{j=0}^n a_j \right) \left( \prod_{j=0}^n b_j \right) \). Many of the rearrangements (namely those that don’t use the inverse) proven in the AbelianGroup_ZF theory hold in fact in semigroups. Some of them will
be reproven in this section.

A rearrangement with 3 elements.

**lemma** (in semigr0) rearr3elems:
assumes f {is commutative on} G and a ∈ G b ∈ G c ∈ G
shows a · b · c = a · c · b
using assms semigr_assoc IsCommutative_def by simp

A rearrangement of four elements.

**lemma** (in semigr0) rearr4elems:
assumes A1: f {is commutative on} G and
A2: a ∈ G b ∈ G c ∈ G d ∈ G
shows a · b · (c · d) = a · c · (b · d)
proof -
from A2 have a · b · (c · d) = a · b · c · d
  using semigr_closed semigr_assoc by simp
also have a · b · c · d = a · c · (b · d)
proof -
  from A1 A2 have a · b · c · d = c · (a · b) · d
    using IsCommutative_def semigr_closed by simp
  also from A2 have ... = c · a · b · d
    using semigr_closed semigr_assoc by simp
  also from A1 A2 have ... = a · c · b · d
    using IsCommutative_def semigr_closed by simp
  also from A2 have ... = a · c · (b · d)
    using semigr_closed semigr_assoc by simp
finally show a · b · c · d = a · c · (b · d) by simp
qed

finally show a · b · (c · d) = a · c · (b · d)
by simp
qed

We start with a version of prod_append that will shorten a bit the proof of the main theorem.

**lemma** (in semigr0) shorter_seq: assumes A1: k ∈ nat and
A2: a ∈ succ(succ(k)) → G
shows (∏ a) = (∏ Init(a)) · a(succ(k))
proof -
let x = Init(a)
from assms have
  a(succ(k)) ∈ G and x : succ(k) → G
  using apply_funtype init_props by auto
with A1 have (∏ x←a(succ(k))) = (∏ x) · a(succ(k))
  using prod_append by simp
with assms show thesis using init_props
A lemma useful in the induction step of the main theorem.

**Lemma (in semigr0) prod_distr_ind_step:**

assumes A1: \( k \in \text{nat} \) and
A2: \( a : \text{succ(succ}(k)) \rightarrow G \) and
A3: \( b : \text{succ(succ}(k)) \rightarrow G \) and
A4: \( c : \text{succ(succ}(k)) \rightarrow G \) and
A5: \( \forall j \in \text{succ(succ}(k)). \, c(j) = a(j) \cdot b(j) \)

shows
Init(a) : succ(k) \( \rightarrow G \)
Init(b) : succ(k) \( \rightarrow G \)
Init(c) : succ(k) \( \rightarrow G \)
\( \forall j \in \text{succ}(k). \, \text{Init}(c)(j) = \text{Init}(a)(j) \cdot \text{Init}(b)(j) \)

**Proof:**
- from A1 A2 A3 A4 show
  Init(a) : succ(k) \( \rightarrow G \)
  Init(b) : succ(k) \( \rightarrow G \)
  Init(c) : succ(k) \( \rightarrow G \)
  using init_props by auto
- from A1 have T: succ(k) \( \in \text{nat} \) by simp
- from T A2 have \( \forall j \in \text{succ}(k). \, \text{Init}(a)(j) = a(j) \)
  by (rule init_props)
- moreover from T A3 have \( \forall j \in \text{succ}(k). \, \text{Init}(b)(j) = b(j) \)
  by (rule init_props)
- moreover from T A4 have \( \forall j \in \text{succ}(k). \, \text{Init}(c)(j) = c(j) \)
  by (rule init_props)
- moreover from A5 have \( \forall j \in \text{succ}(k). \, c(j) = a(j) \cdot b(j) \)
  by simp
- ultimately show \( \forall j \in \text{succ}(k). \, \text{Init}(c)(j) = \text{Init}(a)(j) \cdot \text{Init}(b)(j) \)
  by simp

**QED**

For commutative operations taking the product of a sequence is distributive with respect to the operation. This version will probably not be used in applications, it is formulated in a way that is easier to prove by induction. For a more convenient formulation see prod_comm_distrib. The proof by induction on the length of the sequence.

**Theorem (in semigr0) prod_comm_distr:**

assumes A1: \( f \) \{is commutative on\} \( G \) and A2: \( n \in \text{nat} \)
shows \( \forall \ a \ b \ c. \)
\( (a : \text{succ}(n) \rightarrow G \land b : \text{succ}(n) \rightarrow G \land c : \text{succ}(n) \rightarrow G \land \)
\( (\forall j \in \text{succ}(n). \, c(j) = a(j) \cdot b(j))) \rightarrow \)
\( (\prod c) = (\prod a) \cdot (\prod b) \)

**Proof:**
- note A2
- moreover have \( \forall \ a \ b \ c. \)
  \( (a : \text{succ}(0) \rightarrow G \land b : \text{succ}(0) \rightarrow G \land c : \text{succ}(0) \rightarrow G \land \)

304
\[(\forall j \in \text{succ}(0). \ c(j) = a(j) \cdot b(j))) \rightarrow \\
(\prod c) = (\prod a) \cdot (\prod b)\]

proof -

\{ fix a b c \\
assume a : \text{succ}(O) \rightarrow G \land b : \text{succ}(0) \rightarrow G \land c : \text{succ}(0) \rightarrow G \\
(\forall j \in \text{succ}(0). \ c(j) = a(j) \cdot b(j)) \rightarrow \\
I: a : 1 \rightarrow G \land b : 1 \rightarrow G \land c : 1 \rightarrow G \land \\
\text{then have} \}

I: a : 1 \rightarrow G \land b : 1 \rightarrow G \land c : 1 \rightarrow G \land \\
(\prod a) = a(0) \land \prod b = b(0) \land \prod c = c(0) \\
using prod_of_1elem by auto \\
with II have (\prod c) = (\prod a) \cdot (\prod b) by simp  \\
then show thesis using Fold1_def by simp 
qed

moreover have \(\forall k \in \text{nat}.\)

\[(\forall a \ b \ c. \ a \in \text{succ}(k) \rightarrow G \land b \in \text{succ}(k) \rightarrow G \land c \in \text{succ}(k) \rightarrow G \land \\
(\forall j \in \text{succ}(k). \ c(j) = a(j) \cdot b(j))) \rightarrow \\
(\prod c) = (\prod a) \cdot (\prod b)\]

\[(\forall j \in \text{succ}(\text{succ}(k)). \ c(j) = a(j) \cdot b(j))) \rightarrow \\
(\prod c) = (\prod a) \cdot (\prod b)\]

proof

fix k assume k \(\in\) \text{nat} \\
show \(\forall a \ b \ c.\)

\[(\forall a \ b \ c. \ a \in \text{succ}(k) \rightarrow G \land \\
b \in \text{succ}(k) \rightarrow G \land c \in \text{succ}(k) \rightarrow G \land \\
(\forall j \in \text{succ}(k). \ c(j) = a(j) \cdot b(j))) \rightarrow \\
(\prod c) = (\prod a) \cdot (\prod b)\]

proof

assume A3: \(\forall a \ b \ c.\)

\[(\forall a \ b \ c. \ a \in \text{succ}(k) \rightarrow G \land \\
b \in \text{succ}(k) \rightarrow G \land c \in \text{succ}(k) \rightarrow G \land \\
(\forall j \in \text{succ}(k). \ c(j) = a(j) \cdot b(j))) \rightarrow \\
(\prod c) = (\prod a) \cdot (\prod b)\]

\[\text{show} \ \forall a \ b \ c.\]

\[(\forall a \ b \ c. \ a \in \text{succ}(k) \rightarrow G \land \\
b \in \text{succ}(k) \rightarrow G \land c \in \text{succ}(k) \rightarrow G \land \\
(\forall j \in \text{succ}(k). \ c(j) = a(j) \cdot b(j))) \rightarrow \\
(\prod c) = (\prod a) \cdot (\prod b)\]

305
\( (\prod c) = (\prod a) \cdot (\prod b) \)

**proof** -

\{ fix \ a \ b \ c \\
\hspace{1em} assume \\
\hspace{2em} a \in \text{succ}(\text{succ}(k)) \rightarrow G \land \\
\hspace{2em} b \in \text{succ}(\text{succ}(k)) \rightarrow G \land \\
\hspace{2em} c \in \text{succ}(\text{succ}(k)) \rightarrow G \land \\
\hspace{2em} (\forall j \in \text{succ}(\text{succ}(k)). c(j) = a(j) \cdot b(j)) \}

**with** \( \langle k \in \text{nat} \rangle \) have I:

\begin{align*}
\hspace{1em} a & : \text{succ}(\text{succ}(k)) \rightarrow G \\
\hspace{1em} b & : \text{succ}(\text{succ}(k)) \rightarrow G \\
\hspace{1em} c & : \text{succ}(\text{succ}(k)) \rightarrow G \\
\hspace{1em} \text{and} \ II: \forall j \in \text{succ}(\text{succ}(k)). c(j) = a(j) \cdot b(j) \\
\end{align*}

\hspace{1em} \text{by auto}

let x = Init(a)
let y = Init(b)
let z = Init(c)

\hspace{1em} \text{from} \ \langle k \in \text{nat} \rangle \ I \ \text{have} III:

\begin{align*}
\hspace{1em} (\prod a) & = (\prod x) \cdot a(\text{succ}(k)) \\
\hspace{1em} (\prod b) & = (\prod y) \cdot b(\text{succ}(k)) \and \\
\hspace{1em} IV: (\prod c) & = (\prod z) \cdot c(\text{succ}(k)) \\
\end{align*}

\hspace{1em} \text{using shorter_seq} \ \text{by auto}

moreover

\hspace{1em} \text{from} \ \langle k \in \text{nat} \rangle \ I \ II \ \text{have}

\begin{align*}
\hspace{1em} x & : \text{succ}(k) \rightarrow G \\
\hspace{1em} y & : \text{succ}(k) \rightarrow G \\
\hspace{1em} z & : \text{succ}(k) \rightarrow G \and \\
\hspace{1em} \forall j \in \text{succ}(k). z(j) = x(j) \cdot y(j) \\
\end{align*}

\hspace{1em} \text{using prod_distr_ind_step} \ \text{by auto}

\hspace{1em} \text{with} \ A3 \ II \ IV \ \text{have}

\begin{align*}
\hspace{1em} (\prod c) & = (\prod x) \cdot (\prod y) \cdot a(\text{succ}(k)) \cdot b(\text{succ}(k)) \\
\end{align*}

\hspace{1em} \text{by simp}

moreover \text{from} A1 \ \langle k \in \text{nat} \rangle \ I \ III \ \text{have}

\begin{align*}
\hspace{1em} (\prod x) \cdot (\prod y) \cdot a(\text{succ}(k)) \cdot b(\text{succ}(k)) & = \\
\hspace{2em} (\prod a) \cdot (\prod b) \\
\end{align*}

\hspace{1em} \text{using init_props prod_type apply_functype}

\hspace{1em} \text{rearr4elems} \ \text{by simp}

ultimately \text{have} \ (\prod c) = (\prod a) \cdot (\prod b)

\hspace{1em} \text{by simp}

\} \ \text{thus} \ \text{thesis} \ \text{by auto}

\hspace{1em} \text{qed}

\hspace{1em} \text{qed}

\hspace{1em} \text{ultimately} \ \text{show} \ \text{thesis} \ \text{by} \ \text{(rule ind_on_nat)}

\hspace{1em} \text{qed}

A reformulation of \text{prod_comm_distr} that is more convenient in applications.

**theorem** \( \text{in semigr0} \) \text{prod_comm_distrib}:

\hspace{1em} \text{assumes} \ f \ {\text{is commutative on}} \ G \ \text{and} \ n \in \text{nat} \ \text{and}
\[ a : \text{succ}(n) \to G \quad b : \text{succ}(n) \to G \quad c : \text{succ}(n) \to G \text{ and} \]
\[ \forall j \in \text{succ}(n). \ c(j) = a(j) \cdot b(j) \]
shows \((\prod c) = (\prod a) \cdot (\prod b)\)
using assms \(\text{prod_comm_distr}\) by simp

A product of two products over disjoint sets of indices is the product over the union.

**Lemma (in semigr1) prod_bisect:**
assumes \(A1: f \text{ is commutative on } G\) and \(A2: A \in \text{FinPow}(X)\)
shows
\[ \forall P \in \text{Bisections}(A). \ (\prod A, a) = (\prod (\text{fst}(P), a)) \cdot (\prod (\text{snd}(P), a)) \]
proof -
have IsLinOrder(X, r) using linord by simp
moreover have
\[ \forall P \in \text{Bisections}(0). \ (\prod (0, a) = (\prod (\text{fst}(P), a)) \cdot (\prod (\text{snd}(P), a)) \]
using bisec_empty by simp
moreover have \(\forall A \in \text{FinPow}(X). \)
\[ (\forall n \in X - A. \quad (\forall P \in \text{Bisections}(A). \ (\prod A, a) = (\prod (\text{fst}(P), a)) \cdot (\prod (\text{snd}(P), a)) \quad \land \quad (\forall k \in A. \ (k, n) \in r) \quad \to \quad \forall Q \in \text{Bisections}(A \cup \{n\}). \ (\prod (A \cup \{n\}, a) = (\prod (\text{fst}(Q), a)) \cdot (\prod (\text{snd}(Q), a))) \]
proof -
{ fix A assume A \(\in\) \(\text{FinPow}(X)\)
fix n assume n \(\in\) \(X - A\)
have ( \(\forall P \in \text{Bisections}(A)\). \ (\prod (A, a) = (\prod (\text{fst}(P), a)) \cdot (\prod (\text{snd}(P), a)) \)
\land \quad (\forall k \in A. \ (k, n) \in r) \quad \to \quad \forall Q \in \text{Bisections}(A \cup \{n\}). \ (\prod (A \cup \{n\}, a) = (\prod (\text{fst}(Q), a)) \cdot (\prod (\text{snd}(Q), a))) \)
proof -
{ assume I:
\(\forall P \in \text{Bisections}(A). \ (\prod (A, a) = (\prod (\text{fst}(P), a)) \cdot (\prod (\text{snd}(P), a)) \)
and II: \(\forall k \in A. \ (k, n) \in r\)
have \(\forall Q \in \text{Bisections}(A \cup \{n\}).\)
\(\prod (A \cup \{n\}, a) = (\prod (\text{fst}(Q), a)) \cdot (\prod (\text{snd}(Q), a))\)
proof -
{ fix Q assume Q \(\in\) \(\text{Bisections}(A \cup \{n\})\)
let \(Q_0 = \text{fst}(Q)\)
let \(Q_1 = \text{snd}(Q)\)
from \(<A \in \text{FinPow}(X)\> <n \in X - A> \text{ have } A \cup \{n\} \in \text{FinPow}(X)\)
using singleton_in_finpow union_finpow by auto
with \(<Q \in \text{Bisections}(A \cup \{n\})>\) have \(Q_0 \in \text{FinPow}(X)\) \(Q_0 \neq 0\) and \(Q_1 \in \text{FinPow}(X)\) \(Q_1 \neq 0\)
using bisect_fin bisec_is_pair Bisections_def by auto
then have \(\prod (Q_0, a) \in G\) and \(\prod (Q_1, a) \in G\)
using a_is_fun setprod_type by auto
from \(<Q \in \text{Bisections}(A \cup \{n\})>\) \(<A \in \text{FinPow}(X)\> <n \in X-A>\)
have refl(X, r) \(Q_0 \subseteq A \cup \{n\}\ Q_1 \subseteq A \cup \{n\}\)
A ⊆ X and n ∈ X

using linord IsLinOrder_def total_is_refl Bisections_def FinPow_def by auto

from refl(X,r) Q_0 ⊆ A ∪ {n} A ⊆ X n ∈ X II
have III: ∀ k ∈ Q_0. ⟨k, n⟩ ∈ r by (rule refl_add_point)

from refl(X,r) Q_1 ⊆ A ∪ {n} A ⊆ X n ∈ X II
have IV: ∀ k ∈ Q_1. ⟨k, n⟩ ∈ r by (rule refl_add_point)

from n ∈ X - A Q ∈ Bisections(A ∪ {n}) have
Q_0 = \{n\} ∪ Q_1 = \{n\} ∪ (Q_0 - \{n\},Q_1 - \{n\}) ∈ Bisections(A)

using bisec_is_pair bisec_add_point by simp

moreover
{ assume Q_1 = \{n\}
from n ∈ X - A have n ∉ A by auto
moreover
from Q ∈ Bisections(A ∪ \{n\}) have (Q_0,Q_1) ∈ Bisections(A ∪ \{n\})
using bisec_is_pair by simp
with Q_1 = \{n\} have (Q_0, \{n\}) ∈ Bisections(A ∪ \{n\})
by simp
ultimately have Q_0 = A and A ≠ 0
using set_point_bisec by auto

with A ∈ FinPow(X) n ∈ X - A II Q_1 = \{n\}
have \( \prod(A ∪ \{n\},a) = (\prod(Q_0,a)) \cdot (\prod(Q_1,a)) \)
using a_is_fun gen_prod_append gen_prod_singleton
by simp }
moreover
{ assume Q_0 = \{n\}
from n ∈ X - A have n ∈ X by auto
then have \{n\} ∈ FinPow(X) and \{n\} ≠ 0
using singleton_in_finpow by auto
from n ∈ X - A have n ∉ A by auto
moreover
from Q ∈ Bisections(A ∪ \{n\}) have (Q_0,Q_1) ∈ Bisections(A ∪ \{n\})
using bisec_is_pair by simp
with Q_0 = \{n\} have \{n\}, Q_1 ∈ Bisections(A ∪ \{n\})
by simp
ultimately have Q_1 = A and A ≠ 0 using point_set_bisec by auto

with A1 ⟨A ∈ FinPow(X)⟩ n ∈ X - A II
\{n\} ∈ FinPow(X) \{n\} ≠ 0 Q_0 = \{n\}
have \( \prod(A ∪ \{n\},a) = (\prod(Q_0,a)) \cdot (\prod(Q_1,a)) \)
using a_is_fun gen_prod_append gen_prod_singleton
setprod_type IsCommutative_def by auto }
moreover
{ assume A4: (Q_0 - \{n\},Q_1 - \{n\}) ∈ Bisections(A)
with A ∈ FinPow(X) have
Q_0 - \{n\} ∈ FinPow(X) Q_0 - \{n\} ≠ 0 and
Q_1 - \{n\} ∈ FinPow(X) Q_1 - \{n\} ≠ 0

308
using FinPow_def Bisections_def by auto

with \( n \in X - A \) have
\[
\prod (Q_0 - \{n\}, a) \in G \quad \prod (Q_1 - \{n\}, a) \in G \quad \text{and}
\]
\( T\colon a(n) \in G \)
using a_is_fun setprod_type apply_funtype by auto

from \( \langle Q, Q_1 - \{n\} \rangle \in \text{Bisections}(A \cup \{n\}) \) A4 have
\[
(\langle Q_0, Q_1 - \{n\} \rangle \in \text{Bisections}(A) \land n \in Q_1) \lor
(\langle Q_0 - \{n\}, Q_1 \rangle \in \text{Bisections}(A) \land n \in Q_0)
\]
using bisec_is_pair bisec_add_point_case3 by auto

moreover { assume \( \langle Q_0, Q_1 - \{n\} \rangle \in \text{Bisections}(A) \) and \( n \in Q_1 \)
then have \( A \neq 0 \) using bisec_props by simp
with A2 \( \langle A \in \text{FinPow}(X) \rangle \langle n \in X - A \rangle \) I II T IV
\( \langle Q_0, Q_1 - \{n\} \rangle \in \text{Bisections}(A) \)
\( \prod_{\langle Q_1 - \{n\}, a \rangle \in G} Q_1 \in \text{FinPow}(X) \)
\( n \in Q_1 \) \( \langle Q_1 - \{n\} \rangle \neq 0 \)

have \( \prod_{\langle A \cup \{n\}, a \rangle} = (\prod (Q_0, a)) \cdot (\prod (Q_1, a)) \)
using gen_prod_append semigr_assoc gen_product_rem_point by simp }
moreover
\[
\langle Q_0 - \{n\}, Q_1 \rangle \in \text{Bisections}(A) \) and \( n \in Q_0 \)
then have \( A \neq 0 \) using bisec_props by simp
with A1 A2 \( \langle A \in \text{FinPow}(X) \rangle \langle n \in X - A \rangle \) I II III T
\( \langle Q_0 - \{n\}, Q_1 \rangle \in \text{Bisections}(A) \)
\( \prod_{\langle Q_1, a \rangle \in G} Q_1 \in \text{FinPow}(X) \)
\( n \in Q_0 \) \( \langle Q_0 - \{n\} \rangle \neq 0 \)

have \( \prod_{\langle A \cup \{n\}, a \rangle} = (\prod (Q_0, a)) \cdot (\prod (Q_1, a)) \)
using gen_prod_append rearr3elems gen_product_rem_point by simp }
ultimately have
\( \prod (A \cup \{n\}, a) = (\prod (Q_0, a)) \cdot (\prod (Q_1, a)) \)
by auto }

\{ thus thesis by simp qed }
go to thesis by simp
\{ thus thesis by simp qed }
moreover note A2
ultimately show thesis by (rule fin_ind_add_max)
qed

A better looking reformulation of prod_bisect.

theorem (in semigr1) prod_disjoint: assumes
\( A1\colon f \) is commutative on \( G \) and
\( A2\colon A \in \text{FinPow}(X) \) \( A \neq 0 \) and
\( A3\colon B \in \text{FinPow}(X) \) \( B \neq 0 \) and
\( A4\colon A \cap B = 0 \)

309
shows $\prod(A \cup B, a) = (\prod(A, a)) \cdot (\prod(B, a))$

proof -
from A2 A3 A4 have $\langle A, B \rangle \in \text{Bisect}(A \cup B)$
using is_bisec by simp
with A1 A2 A3 show thesis
using a_is_fun union_finpow prod_bisect by simp
qed

A generalization of prod_disjoint.

lemma (in semigr1) prod_list_of_lists: assumes
A1: $f$ \{is commutative on\} $G$ and A2: $n \in \text{nat}$
shows $\forall M \in \text{succ}(n) \rightarrow \text{FinPow}(X)$.
M \{is partition\} $\quad \rightarrow$
$\prod \{\{i, \prod(M(i), a)\}. i \in \text{succ}(n)\} =$
$\prod(\bigcup i \in \text{succ}(n). M(i), a))$

proof -
moreover have $\forall M \in \text{succ}(0) \rightarrow \text{FinPow}(X)$.
M \{is partition\} $\quad \rightarrow$
$\prod \{\{i, \prod(M(i), a)\}. i \in \text{succ}(0)\} = \prod(\bigcup i \in \text{succ}(0). M(i), a))$
using a_is_fun func1_1_L1 Partition_def apply_funtype setprod_type
list_len1_singleton prod_of_1elem
by simp
moreover have $\forall k \in \text{nat}$.
$\forall M \in \text{succ}(k) \rightarrow \text{FinPow}(X)$.
M \{is partition\} $\quad \rightarrow$
$\prod \{\{i, \prod(M(i), a)\}. i \in \text{succ}(k)\} =$
$\prod(\bigcup i \in \text{succ}(k). M(i), a))$

proof -
{ fix $k$ assume $k \in \text{nat}$
 assume A3: $\forall M \in \text{succ}(k) \rightarrow \text{FinPow}(X)$.
M \{is partition\} $\quad \rightarrow$
$\prod \{\{i, \prod(M(i), a)\}. i \in \text{succ}(k)\} =$
$\prod(\bigcup i \in \text{succ}(k). M(i), a))$

have $\forall N \in \text{succ}(\text{succ}(k)) \rightarrow \text{FinPow}(X)$.
N \{is partition\} $\quad \rightarrow$
$\prod \{\{i, \prod(N(i), a)\}. i \in \text{succ}(\text{succ}(k))\} =$
$\prod(\bigcup i \in \text{succ}(\text{succ}(k)). N(i), a))$

proof -
{ fix $N$ assume A4: $N : \text{succ}(\text{succ}(k)) \rightarrow \text{FinPow}(X)$
 assume A5: $N \{is partition\}$
with A4 have I: $\forall i \in \text{succ}(\text{succ}(k)). N(i) \neq 0$
using func1_1_L1 Partition_def by simp
let $b = \{\{i, \prod(N(i), a)\}. i \in \text{succ}(\text{succ}(k))\}$
let $c = \{\{i, \prod(N(i), a)\}. i \in \text{succ}(k)\}$
have II: \(\forall i \in \text{succ}(\text{succ}(k)). \prod(N(i),a) \in G\)

proof

fix i assume \(i \in \text{succ}(\text{succ}(k))\)

with A4 I have \(N(i) \in \text{FinPow}(X)\) and \(N(i) \neq 0\)
using apply_funtype by auto
then show \(\prod(N(i),a) \in G\) using setprod_type
by simp

qed

hence \(\forall i \in \text{succ}(k). \prod(N(i),a) \in G\) by auto
then have \(c : \text{succ}(k) \rightarrow G\) by (rule ZF_fun_from_total)

have \(b = \{(i,\prod(N(i),a)). i \in \text{succ}(\text{succ}(k))\}\)
by simp

with II have \(b = \text{Append}(c,\prod(N(\text{succ}(k)),a))\)
by (rule set_list_append)
with II \(<k \in \text{nat}> <c : \text{succ}(k) \rightarrow G>\)
have \((\prod b) = (\prod c) : (\prod(N(\text{succ}(k)),a))\)
using prod_append by simp
also have ...
= \((\prod(\bigcup i \in \text{succ}(k). N(i),a)) \cdot (\prod(N(\text{succ}(k)),a))\)

proof -
let \(M = \text{restrict}(N,\text{succ}(k))\)

have \(\text{succ}(k) \subseteq \text{succ}(\text{succ}(k))\) by auto
with \(<N : \text{succ}(\text{succ}(k)) \rightarrow \text{FinPow}(X)>\)

have \(M : \text{succ}(k) \rightarrow \text{FinPow}(X)\) and

III: \(\forall i \in \text{succ}(k). M(i) = N(i)\)
using restrict_type2 restrict apply_funtype
by auto

with A5 \(<M : \text{succ}(k) \rightarrow \text{FinPow}(X)>\) have \(M \{\text{is partition}\}\)
using func1_1_L1 Partition_def by simp

with A3 \(<M : \text{succ}(k) \rightarrow \text{FinPow}(X)>\) have
\((\prod \{(i,\prod(M(i),a)). i \in \text{succ}(k)\}) = (\prod(\bigcup i \in \text{succ}(k). M(i),a))\)
by blast

with III show thesis by simp

qed

also have ...
= \((\prod(\bigcup i \in \text{succ}(\text{succ}(k)). N(i),a))\)

proof -
let \(A = \bigcup i \in \text{succ}(k). N(i)\)
let \(B = N(\text{succ}(k))\)

from A4 \(<k \in \text{nat}> \text{have succ}(k) \in \text{nat}\) and

\(\forall i \in \text{succ}(k). N(i) \in \text{FinPow}(X)\)
using apply_funtype by auto
then have \(A \in \text{FinPow}(X)\) by (rule union_fin_list_fin)
moreover from I have \(A \neq 0\) by auto
moreover from A4 I have

\(N(\text{succ}(k)) \in \text{FinPow}(X)\) and \(N(\text{succ}(k)) \neq 0\)
using apply_funtype by auto
moreover from \(<\text{succ}(k) \in \text{nat}>\) A4 A5 have \(A \cap B = 0\)
by (rule list_partition)
moreover note A1
ultimately have \( \prod(A \cup B, a) = (\prod(A, a)) \cdot (\prod(B, a)) \)
using \texttt{prod_disjoint} by \texttt{simp}
moreover have \( A \cup B = (\bigcup i \in \text{succ}(\text{succ}(k)). N(i)) \)
by auto
ultimately show thesis by \texttt{simp}
qed

finally have \( (\prod \{ (i, \prod(N(i), a) \}. i \in \text{succ}(\text{succ}(k))\}) = (\prod(\bigcup i \in \text{succ}(\text{succ}(k)). N(i), a)) \)
by \texttt{simp}
\}
thus thesis by auto
qed
\}
thus thesis by \texttt{simp}
qed
ultimately show thesis by (rule \texttt{ind_on_nat})
qed

A more convenient reformulation of \texttt{prod_list_of_lists}.

\begin{theorem}
\texttt{prod_order_irr}: \begin{align*}
\text{assumes } & A1: f \text{ is commutative on } G \text{ and } \\
A2: & n \in \text{nat} \ n \neq 0 \text{ and } \\
A3: & M : n \to \text{FinPow}(X) \ M \text{ is partition} \\
\text{shows } & (\prod \{ (i, \prod(M(i), a) \}. i \in n\}) = (\prod(\bigcup i \in n. M(i), a)) \\
\text{proof -} & \text{from } A2 \ \text{obtain } k \text{ where } k \in \text{nat} \ \text{and } n = \text{succ}(k) \\
& \text{using } \texttt{Nat_ZF_1_L3} \ \text{by auto} \\
& \text{with } A1 \ A3 \ \text{show thesis using } \texttt{prod_list_of_lists} \\
& \text{by } \texttt{simp} \\
\end{align*}
\end{theorem}

The definition of the product \( \prod(A, a) \equiv \texttt{SetFold}(f, a, A, r) \) of a some (finite) set of semigroup elements requires that \( r \) is a linear order on the set of indices \( A \). This is necessary so that we know in which order we are multiplying the elements. The product over \( A \) is defined so that we have \( \prod_A a = \prod a \circ \sigma(A) \) where \( \sigma : |A| \to A \) is the enumeration of \( A \) (the only order isomorphism between the number of elements in \( A \) and \( A \)), see lemma \texttt{setproddef}. However, if the operation is commutative, the order is irrelevant. The next theorem formalizes that fact stating that we can replace the enumeration \( \sigma(A) \) by any bijection between \(|A|\) and \( A \). In a way this is a generalization of \texttt{setproddef}. The proof is based on application of \texttt{prod_list_of_sets} to the finite collection of singletons that comprise \( A \).

\begin{theorem}
\texttt{prod_order_irr}: \begin{align*}
\text{assumes } & A1: f \text{ is commutative on } G \text{ and } \\
A2: & A \in \text{FinPow}(X) \ A \neq 0 \text{ and } \\
A3: & b \in \text{bij}(|A|, A) \\
\text{shows } & (\prod (a \circ b)) = \prod(A, a) \\
\text{proof -} \\
\end{align*}
\end{theorem}

312
let \( n = |A| \)
let \( M = \{ \langle k, \{ b(k) \} \rangle \mid k \in n \} \)

have \( \prod (a \circ b) = \prod \{ \langle i, \prod (M(i), a) \rangle \mid i \in n \} \)
proof
have \( \forall i \in n. \prod (M(i), a) = (a \circ b)(i) \)
proof
fix \( i \) assume \( i \in n \)

with \( A2 \ A3 <i \in n> \) have \( b(i) \in X \)
using \( \text{bij_def inj_def apply_funtype FinPow_def} \)
by auto
then have \( \prod \{ b(i), a \} = a(b(i)) \)
using \( \text{gen_prod_singleton} \)
by simp
with \( A3 <i \in n> \) have \( \prod \{ b(i), a \} = (a \circ b)(i) \)
using \( \text{bij_def inj_def comp_fun_apply by auto} \)
with \( <i \in n> A3 \) show \( \prod (M(i), a) = (a \circ b)(i) \)
using \( \text{bij_def inj_partition by auto} \)
qed

hence \( \{ \langle i, \prod (M(i), a) \rangle \mid i \in n \} = \{ \langle i, (a \circ b)(i) \rangle \mid i \in n \} \)
by simp
moreover have \( \{ \langle i, (a \circ b)(i) \rangle \mid i \in n \} = a \circ b \)
proof
from \( A3 \) have \( b : n \to A \) using \( \text{bij_def inj_def} \)
moreover from \( A2 \) have \( n \in \text{nat} \) and \( n \neq 0 \)
using \( \text{card_fin_is_nat card_non_empty_non_zero by auto} \)
moreover have \( M : n \to \text{FinPow}(X) \) and \( M \{\text{is partition} \} \)
proof
from \( A2 \ A3 \) have \( \forall k \in n. \{ b(k) \} \in \text{FinPow}(X) \)
using \( \text{bij_def inj_def apply_funtype FinPow_def} \)
\( \text{singleton_in_finpow by auto} \)
then show \( M : n \to \text{FinPow}(X) \) using \( \text{ZF_fun_from_total} \)
by simp
from \( A3 \) show \( M \{\text{is partition} \} \) using \( \text{bij_def inj_partition} \)
by auto
qed
ultimately show thesis by simp
qed
also have \( \ldots = (\prod (\bigcup i \in n. M(i), a)) \)
proof
note \( A1 \)
moreover from \( A2 \) have \( n \in \text{nat} \) and \( n \neq 0 \)
using \( \text{card_fin_is_nat card_non_empty_non_zero by auto} \)
moreover have \( M : n \to \text{FinPow}(X) \) and \( M \{\text{is partition} \} \)
proof
from \( A2 \ A3 \) have \( \forall k \in n. \{ b(k) \} \in \text{FinPow}(X) \)
using \( \text{bij_def inj_def apply_funtype FinPow_def} \)
\( \text{singleton_in_finpow by auto} \)
then show \( M : n \to \text{FinPow}(X) \) using \( \text{ZF_fun_from_total} \)
by simp
from \( A3 \) show \( M \{\text{is partition} \} \) using \( \text{bij_def inj_partition} \)
by auto
qed
ultimately show \( \prod (\bigcup \{ i \in n. M(i), a \}) = (\prod (\bigcup i \in n. M(i), a)) \)
by (rule \( \text{prod_list_of_sets} \))
qed
also from A3 have $(\prod (\bigcup i \in n. M(i),a)) = \prod (A,a)$
using bij_def inj_partition surj_singleton_image
by auto
finally show thesis by simp
qed

Another way of expressing the fact that the product does not depend on the order.

corollary (in semigr1) prod_bij_same:
assumes f {is commutative on} G and
A ∈ FinPow(X) A ≠ 0 and
b ∈ bij(|A|,A) c ∈ bij(|A|,A)
shows $(\prod (a 0 b)) = (\prod (a 0 c))$
using assms prod_order_irr by simp

end

28 Commutative Semigroups

theory CommutativeSemigroup_ZF imports Semigroup_ZF
begin

In the Semigroup theory we introduced a notion of $\text{SetFold}(f,a,\Lambda,r)$ that represents the sum of values of some function $a$ valued in a semigroup where the arguments of that function vary over some set $\Lambda$. Using the additive notation something like this would be expressed as $\sum_{x \in \Lambda} f(x)$ in informal mathematics. This theory considers an alternative to that notion that is more specific to commutative semigroups.

28.1 Sum of a function over a set

The $r$ parameter in the definition of $\text{SetFold}(f,a,\Lambda,r)$ (from Semigroup_ZF) represents a linear order relation on $\Lambda$ that is needed to indicate in what order we are summing the values $f(x)$. If the semigroup operation is commutative the order does not matter and the relation $r$ is not needed. In this section we define a notion of summing up values of some function $a : X \to G$ over a finite set of indices $\Gamma \subseteq X$, without using any order relation on $X$.

We define the sum of values of a function $a : X \to G$ over a set $\Lambda$ as the only element of the set of sums of lists that are bijections between the number of values in $\Lambda$ (which is a natural number $n = \{0, 1, ..., n-1\}$ if $\Lambda$ is finite) and $\Lambda$. The notion of $\text{Fold1}(f,c)$ is defined in Semigroup_ZF as the fold (sum) of the list $c$ starting from the first element of that list. The intention is to use the fact that since the result of summing up a list does not depend on
the order, the set \(\{\text{Fold1}(f, a \circ b). b \in \text{bij}(|A|, A)\}\) is a singleton and we can extract its only value by taking its union.

definition

\[\text{CommSetFold}(f, a, A) = \bigcup \{\text{Fold1}(f, a \circ b). b \in \text{bij}(|A|, A)\}\]

the next locale sets up notation for writing about summation in commutative semigroups. We define two kinds of sums. One is the sum of elements of a list (which are just functions defined on a natural number) and the second one represents a more general notion the sum of values of a semigroup valued function over some set of arguments. Since those two types of sums are different notions they are represented by different symbols. However in the presentations they are both intended to be printed as \(\sum\).

locale commsemigr =

fixes G f

assumes csgassoc: \(f\) {is associative on} \(G\)

assumes csgcomm: \(f\) {is commutative on} \(G\)

fixes csgsum (infixl 69)
defines csgsum_def[simp]: \(x + y \equiv f(x, y)\)

fixes X a
assumes csgaisfun: \(a : X \to G\)

fixes csglistsum (\(\sum\) 70)
defines csglistsum_def[simp]: \(\sum k \equiv \text{Fold1}(f, k)\)

fixes csgsetsum (\(\sum\))
defines csgsetsum_def[simp]: \(\sum (A, h) \equiv \text{CommSetFold}(f, h, A)\)

Definition of a sum of function over a set in notation defined in the commsemigr locale.

lemma (in commsemigr) CommSetFolddef:
shows \(\sum (A, a) = \bigcup \{\sum (a \circ b). b \in \text{bij}(|A|, A)\}\)
using CommSetFold_def by simp

The next lemma states that the result of a sum does not depend on the order we calculate it. This is similar to lemma prod_order_irr in the Semigroup theory, except that the semigr1 locale assumes that the domain of the function we sum up is linearly ordered, while in commsemigr we don’t have this assumption.

lemma (in commsemigr) sum_over_set_bij:
assumes A1: \(A \in \text{FinPow}(X)\) \(A \neq 0\) and A2: \(b \in \text{bij}(|A|, A)\)
shows \(\sum (A, a) = \sum (a \circ b)\)
proof -
have
\( \forall c \in \text{bij}(|A|, A). \ \forall d \in \text{bij}(|A|, A). \ (\sum (a \circ c)) = (\sum (a \circ d)) \)

\text{proof - }
\{ \text{fix } c \ \text{assume } c \in \text{bij}(|A|, A) \\
\text{fix } d \ \text{assume } d \in \text{bij}(|A|, A) \\
\text{let } r = \text{InducedRelation}\left(\text{converse}(c), \text{Le}\right) \\
\text{have semigr1(G,f,A,r,restrict(a, A))} \}

\text{proof - }
\text{have semigr0(G,f) using csgassoc semigr0_def by simp}
\text{moreover from A1 } <c \in \text{bij}(|A|, A)> \text{ have IsLinOrder(A,r)}
\text{using bij_converse_bij card_fin_is_nat}
\text{natord_lin_on_each_nat ind_rel_pres_lin by simp}
\text{moreover from A1 have restrict(a, A) : A } \rightarrow \ G
\text{using FinPow_def csgaisfun restrict_fun by simp}
\text{ultimately show thesis using semigr1_axioms.intro semigr1_def}
\text{by simp}
\text{qed}

\text{moreover have } f \ {\text{is commutative on}} \ G \ \text{using csgcomm}
\text{by simp}
\text{moreover from A1 have } A \in \text{FinPow}(A) \ A \neq 0
\text{using FinPow_def by auto}
\text{moreover note } <c \in \text{bij}(|A|, A) > <d \in \text{bij}(|A|, A)>
\text{ultimately have}
\text{Fold1}(f, \text{restrict}(a, A) \ O c) = \text{Fold1}(f, \text{restrict}(a, A) \ O d)
\text{by (rule semigr1.prod_bij_same)}
\text{hence } (\sum (\text{restrict}(a, A) \ O c)) = (\sum (\text{restrict}(a, A) \ O d))
\text{by simp}
\text{moreover from A1 } <c \in \text{bij}(|A|, A) > <d \in \text{bij}(|A|, A)>
\text{have}
\text{restrict(a, A) \ O c = a \ O c and restrict(a, A) \ O d = a \ O d}
\text{using bij_def surj_def csgaisfun FinPow_def comp_restrict by auto}
\text{ultimately have } (\sum (a \ O c)) = (\sum (a \ O d)) \ \text{by simp}
\text{thus thesis by blast}
\text{qed}

\text{with A2 have } (\bigcup \{\sum (a \ O b). b \in \text{bij}(|A|, A)\}) = (\sum (a \ O b))
\text{by (rule singleton_comprehension)}
\text{then show thesis using CommSetFolddef by simp}
\text{qed}

The result of a sum is in the semigroup. Also, as the second assertion
we show that every semigroup valued function generates a homomorphism
between the finite subsets of a semigroup and the semigroup. Adding an
element to a set corresponds to adding a value.

\text{lemma (in commsemigr) sum_over_set_add_point:}
\text{assumes } A1: A \in \text{FinPow}(X) \ A \neq 0
\text{shows } \sum (A,a) \in G \ \text{and}
\forall x \in X \setminus A. \ \sum (A \cup \{x\},a) = (\sum (A,a)) + a(x)
\text{proof - --

316}
from A1 obtain b where b ∈ bij(|A|, A)
  using fin_bij_card by auto
with A1 have \( \sum (A,a) = \left( \sum (a \circ b) \right) \)
  using sum_over_set_bij by simp
from A1 have |A| ∈ nat using card_fin_is_nat by simp
have semigr0(G,f) using csgassoc semigr0_def by simp
moreover
from A1 obtain n where n ∈ nat and |A| = succ(n)
  using card_non_empty_succ by auto
with A1 havethesis
  using card_fin_add_one by auto
ultimately have Fold1(f,a \circ b) ∈ G by (rule semigr0.prod_type)
with \( \sum (A,a) = \left( \sum (a \circ b) \right) \) show \( \sum (A,a) \in G \)
  by simp
{ fix x assume x ∈ X-A
  with A1 have \( (A \cup \{x\}) \in \text{FinPow}(X) \) and \( A \cup \{x\} \neq 0 \)
    using singleton_in_finpow union_finpow by auto
  moreover have Append(b,x) ∈ bij(|A \cup \{x\}|, A \cup \{x\})
    proof -
      note \(|A| ∈ \text{nat}\)
      moreover from \( x ∈ X-A \) have \( x \notin A \) by simp
      ultimately have Append(b,x) ∈ bij(succ(|A|), A \cup \{x\})
    by (rule bij_append_point)
  with A1 \( x ∈ X-A \) show thesis
  using card_fin_add_one by auto
  qed
  ultimately have \( \left( \sum (A \cup \{x\}, a) \right) = \left( \sum (a \circ \text{Append}(b,x)) \right) \)
    using sum_over_set_bij by simp
  also have ... = \( \left( \sum \text{Append}(a \circ b, a(x)) \right) \)
    proof -
      note \(|A| ∈ \text{nat}\)
      moreover from A1 \( b ∈ \text{bij}(|A|, A) \) have \( b : \{0\} \rightarrow A \) and \( A \subseteq X \)
      using bij_def inj_def FinPow_def comp_fun_subset csgaisfun
      by auto
      ultimately show thesis using list_compose_append
    by simp
  qed
  also have ... = \( \left( \sum (A,a) \right) + a(x) \)
    proof -
      note semigr0(G,f) \( n ∈ \text{nat}\) \( a \circ b : \text{succ}(n) → G \)
      moreover from \( x ∈ X-A \) have \( a(x) ∈ G \)
      using csgaisfun apply_funtype by simp
      ultimately have
Fold1(f,Append(a O b, a(x))) = f(Fold1(f,a O b),a(x))
by (rule semigr0.prod_append)
with A1 ⟨b ∈ bij(|A|,A)⟩ show thesis
using sum_over_set_bij by simp
qed

finally have (∑(A ∪ {x},a)) = (∑(A,a)) + a(x)
by simp

thus ∀x ∈ X-A. ∑(A ∪ {x},a) = (∑(A,a)) + a(x)
by simp
qed

end

29 Monoids

theory Monoid_ZF imports func_ZF Loop_ZF Semigroup_ZF

begin

This theory provides basic facts about monoids.

29.1 Definition and basic properties

In this section we talk about monoids. The notion of a monoid is similar to
the notion of a semigroup except that we require the existence of a neutral
element. It is also similar to the notion of group except that we don’t require
existence of the inverse.

Monoid is a set \( G \) with an associative operation and a neutral element. The
operation is a function on \( G \times G \) with values in \( G \). In the context of ZF set
theory this means that it is a set of pairs \((x,y)\), where \( x \in G \times G \) and \( y \in G \).
In other words the operation is a certain subset of \((G \times G) \times G\). We express
all this by defining a predicate IsAmonoid(\( G, f \)). Here \( G \) is the "carrier" of the
monoid and \( f \) is the binary operation on it.

definition
IsAmonoid(\( G, f \)) ⇔
\( f \) {is associative on} \( G \) ∧
(∃e\( ∈ \) G. (∀g\( ∈ \) G. ( (f(e,g)) = g) ∧ (f(g,e)) = g)))

The next locale called "monoid0" defines a context for theorems that concern
monoids. In this context we assume that the pair \((G, f)\) is a monoid. We will
use the \( ∘ \) symbol to denote the monoid operation (for no particular reason).

locale monoid0 =
fixes G f
assumes monoidAssum: IsAmonoid(G,f)

fixes monoper (infixl ∘ 70)
defines monoper_def [simp]: \( a \oplus b \equiv f(a,b) \)

Propositions proven in the semigr0 locale are valid in the monoid0 locale.

lemma (in monoid0) semigr0_valid_in_monoid0: shows semigr0(G,f)
  using monoidAssum IsAmonoid_def semigr0_def by simp

The result of the monoid operation is in the monoid (carrier).

lemma (in monoid0) group0_1_L1:
  assumes a\(\in\)G b\(\in\)G shows a\(\oplus\)b \(\in\) G
  using assms monoidAssum IsAmonoid_def IsAssociative_def apply_funtype
  by auto

There is only one neutral element in a monoid.

lemma (in monoid0) group0_1_L2:
  shows \(\exists !e. e \in G \land (\forall g \in G. (e \oplus g = g) \land g \oplus e = g)\)
proof
  fix e y
  assume e \(\in\) G \(\land\) (\(\forall\)g\(\in\)G. e \(\oplus\) g = g \(\land\) g \(\oplus\) e = g)
  and y \(\in\) G \(\land\) (\(\forall\)g\(\in\)G. y \(\oplus\) g = g \(\land\) g \(\oplus\) y = g)
  then have y\(\oplus\)e = y y\(\oplus\)e = e by auto
  thus e = y by simp
next from monoidAssum show
  \(\exists e. e \in G \land (\forall g \in G. e \oplus g = g) \land g \oplus e = g)\)
  using IsAmonoid_def by auto
qed

The neutral element is neutral.

lemma (in monoid0) unit_is_neutral:
  assumes A1: e = TheNeutralElement(G,f)
  shows e \(\in\) G \(\land\) (\(\forall\)g\(\in\)G. e \(\oplus\) g = g \(\land\) g \(\oplus\) e = g)
proof -
  let n = THE b. b \(\in\) G \(\land\) (\(\forall\)g\(\in\)G. b \(\oplus\) g = g \(\land\) g \(\oplus\) b = g)
  have \(\exists !b. b \in G \land (\forall g \in G. b \oplus g = g \land g \oplus b = g)\)
    using group0_1_L2 by simp
  then have n\(\in\)G \(\land\) (\(\forall\)g\(\in\)G. n \(\oplus\) g = g \(\land\) g \(\oplus\) n = g)
    by (rule theI)
  with A1 show thesis
    using TheNeutralElement_def by simp
qed

The monoid carrier is not empty.

lemma (in monoid0) group0_1_L3A: shows G\(\neq\)0
proof -
  have TheNeutralElement(G,f) \(\in\) G using unit_is_neutral
    by simp
  thus thesis by auto
qed
The monoid operation is a binary function on the carrier with values in the carrier.

**lemma (in monoid0) monoid_oper_fun:** shows $f: G \times G \rightarrow G$

using monoidAssum unfolding IsAmonoid_def IsAssociative_def by simp

The range of the monoid operation is the whole monoid carrier.

**lemma (in monoid0) group0_1_L3B:** shows $\text{range}(f) = G$

**proof**

- from monoidAssum have $f: G \times G \rightarrow G$
  using IsAmonoid_def IsAssociative_def by simp

then show $\text{range}(f) \subseteq G$

- using func1_1_L5B by simp

show $G \subseteq \text{range}(f)$

**proof**

- fix $g$ assume $A1: g \in G$
  let $e = \text{TheNeutralElement}(G, f)$

from $A1$ have $(e, g) \in G \times G$ $g = f(e, g)$

- using unit_is_neutral by auto

with $\langle f: G \times G \rightarrow G \rangle$ show $g \in \text{range}(f)$

- using func1_1_L5A by blast

qed

Another way to state that the range of the monoid operation is the whole monoid carrier.

**lemma (in monoid0) range_carr:** shows $f(G \times G) = G$

using monoidAssum IsAmonoid_def IsAssociative_def

**proof**

- from monoidAssum have $f: G \times G \rightarrow G$
  using IsAmonoid_def IsAssociative_def by simp

then show $\text{range}(f) \subseteq G$

- using func1_1_L5B by simp

show $G \subseteq \text{range}(f)$

**proof**

- fix $g$ assume $A1: e \in G$

let $e = \text{TheNeutralElement}(G, f)$

- from $A1$ have $(e, g) \in G \times G$ $g = f(e, g)$

- using unit_is_neutral by auto

with $\langle f: G \times G \rightarrow G \rangle$ show $g \in \text{range}(f)$

- using func1_1_L5A by blast

qed

In a monoid any neutral element is the neutral element.

**lemma (in monoid0) group0_1_L4:**

- assumes $A1: e \in G$ $(\forall g \in G. e \circledast g = g \land g \circledast e = g)$

- shows $e = \text{TheNeutralElement}(G, f)$

**proof**

- let $n = \text{THE} b$. $b \in G$ $(\forall g \in G. b \circledast g = g \land g \circledast b = g)$

- have $\exists ! b$. $b \in G$ $(\forall g \in G. b \circledast g = g \land g \circledast b = g)$

- using group0_1_L2 by simp

- moreover note $A1$

- ultimately have $n = e$ by (rule the_equality2)

- then show thesis using TheNeutralElement_def by simp

qed

The next lemma shows that if the if we restrict the monoid operation to a subset of $G$ that contains the neutral element, then the neutral element of the monoid operation is also neutral with the restricted operation.

**lemma (in monoid0) group0_1_L5:**

320
assumes A1: ∀x∈H. ∀y∈H. x⊕y ∈ H
and A2: H ⊆ G
and A3: e = TheNeutralElement(G,f)
and A4: g = restrict(f,H×H)
and A5: e ∈ H
and A6: h ∈ H
shows g⟨e,h⟩ = h ∧ g⟨h,e⟩ = h
proof
- from A4 A6 A5 have
  g⟨e,h⟩ = e ⊕ h ∧ g⟨h,e⟩ = h ⊕ e
  using restrict_if by simp
with A3 A4 A6 A2 show
  IsAmonoid(H,restrict(f,H×H))
using unit_is_neutral by auto
qed

The next theorem shows that if the monoid operation is closed on a subset of G then this set is a (sub)monoid (although we do not define this notion). This fact will be useful when we study subgroups.

theorem (in monoid0) group0_1_T1:
  assumes A1: H {is closed under} f
and A2: H ⊆ G
and A3: TheNeutralElement(G,f) ∈ H
shows IsAmonoid(H,restrict(f,H×H))
proof
- let g = restrict(f,H×H)
  let e = TheNeutralElement(G,f)
  from monoidAssum have f ∈ G×G→G
    using IsAmonoid_def IsAssociative_def by simp
moreover from A2 have H×H ⊆ G×G by auto
moreover from A1 have ∀p ∈ H×H. f(p) ∈ H
  using IsOpClosed_def by auto
ultimately have g ∈ H×H→H
  using func1_2_L4 by simp
moreover have ∀x∈H. ∀y∈H. ∀z∈H.
  g(g(x,y),z) = g(x,g(y,z))
proof
- from A1 have ∀x∈H. ∀y∈H. ∀z∈H.
  g(g(x,y),z) = x⊕y⊕z
  using IsOpClosed_def restrict_if by simp
moreover have ∀x∈H. ∀y∈H. ∀z∈H. x⊕y⊕z = x⊕(y⊕z)
proof
- from monoidAssum have
  ∀x∈G. ∀y∈G. ∀z∈G. x⊕y⊕z = x⊕(y⊕z)
  using IsAmonoid_def IsAssociative_def
  by simp
  with A2 show thesis by auto
  qed
moreover from A1 have
∀x∈H. ∀y∈H. ∀z∈H. x⊙(y⊙z) = g(x,g(y,z))

using IsOpClosed_def restrict_if by simp

ultimately show thesis by simp

qed

moreover have

∃n∈H. (∀h∈H. g(n,h) = h ∧ g(h,n) = h)

proof -

from A1 have ∀x∈H. ∀y∈H. x⊙y ∈ H

using IsOpClosed_def by simp

with A2 A3 have

∀ h∈H. g(e,h) = h ∧ g(h,e) = h

using group0_1_L5 by blast

with A3 show thesis by auto

qed

ultimately show thesis using IsAmonoid_def IsAssociative_def

by simp

qed

Under the assumptions of group0_1_T1 the neutral element of a submonoid is the same as that of the monoid.

lemma group0_1_L6:

assumes A1: IsAmonoid(G,f)

and A2: H {is closed under} f

and A3: H ⊆ G

and A4: TheNeutralElement(G,f) ∈ H

shows TheNeutralElement(H,restrict(f,H×H)) = TheNeutralElement(G,f)

proof -

let e = TheNeutralElement(G,f)

let g = restrict(f,H×H)

from assms have monoid0(H,g)

using monoid0_def monoid0.group0_1_T1

by simp

moreover have

e ∈ H ∧ (∀h∈H. g(e,h) = h ∧ g(h,e) = h)

proof -

{ fix h assume h ∈ H

with assms have

monoid0(G,f) ∀x∈H. ∀y∈H. f(x,y) ∈ H

H ⊆ G e = TheNeutralElement(G,f) g = restrict(f,H×H)

e ∈ H h ∈ H

using monoid0_def IsOpClosed_def by auto

then have g(e,h) = h ∧ g(h,e) = h

by (rule monoid0.group0_1_L5)

} hence ∀h∈H. g(e,h) = h ∧ g(h,e) = h by simp

with A4 show thesis by simp

qed

ultimately have e = TheNeutralElement(H,g)

by (rule monoid0.group0_1_L4)

thus thesis by simp
If a sum of two elements is not zero, then at least one has to be nonzero.

**Lemma** (in monoid0) sum_nonzero_elmnt_nonzero:
assumes \( a \oplus b \neq \text{TheNeutralElement}(G,f) \)
shows \( a \neq \text{TheNeutralElement}(G,f) \lor b \neq \text{TheNeutralElement}(G,f) \)
using assms unit_is_neutral by auto

The monoid operation is associative.

**Lemma** (in monoid0) sum_associative:
assumes \( a \in G \), \( b \in G \), \( c \in G \)
shows \((a \oplus b) \oplus c = a \oplus (b \oplus c)\)
using assms monoidAssum unfolding IsAmonoid_def IsAssociative_def by auto

A simple rearrangement of four monoid elements transferred from the semigr0 locale:

**Lemma** (in monoid0) rearr4elem_monoid:
assumes \( a \in G \), \( b \in G \), \( c \in G \), \( d \in G \)
shows \( a \oplus b \oplus (c \oplus d) = a \oplus (b \oplus c) \oplus d \)
using assms semigr0_valid_in_monoid0 semigr0.rearr4elem_assoc by simp

30 Summing lists in a monoid

theory Monoid_ZF_1 imports Monoid_ZF

begin

This theory consider properties of sums of monoid elements, similar to the ones formalized in the Semigroup_ZF theory for sums of semigroup elements. The main difference is that since each monoid has a neutral element it makes sense to define a sum of an empty list of monoid elements. In multiplicative notation the properties considered here can be applied to natural powers of elements \((x^n, n \in \mathbb{N})\) in group or ring theory or, when written additively, to natural multiplicities \(n \cdot x, n \in \mathbb{N}\).

30.1 Notation and basic properties of sums of lists of monoid elements

In this section we setup a context (locale) with notation for sums of lists of monoid elements and prove basic properties of those sums in terms of that notation.
The locale (context) monoid1 extends the locale monoid0, adding the notation for the neutral element as 0 and the sum of a list of monoid elements. It also defines a notation for natural multiple of an element of a monoid, i.e. \( n \cdot x = x \oplus x \oplus ... \oplus x \) (n times).

```ocaml
locale monoid1 = monoid0 +
  fixes mzero (0)
  defines mzero_def [simp]: 0 = TheNeutralElement(G,f)

fixes listsum (\( \sum \_ 70 \))
defines listsum_def [simp]: \( \sum s = \text{Fold}(f,0,s) \)

fixes nat_mult (infix \( \cdot 72 \))
defines nat_mult_def [simp]: \( n \cdot x = \sum \{ \langle k,x \rangle. k \in n \} \)
```

Let’s recall that the neutral element of the monoid is an element of the monoid (carrier) \( G \) and the monoid operation (\( f \) in our notation) is a function that maps \( G \times G \) to \( G \).

```ocaml
lemma (in monoid1) zero_monoid_oper: shows 0 \in G and \( f:G \times G \to G \)
  using monoidAssum unit_is_neutral unfolding IsAmonoid_def IsAssociative_def
  by simp_all
```

The sum of a list of monoid elements is a monoid element.

```ocaml
lemma (in monoid1) sum_in_mono: assumes n \in nat \( \forall k \in n. q(k) \in G \)
  shows (\( \sum \{ \langle k,q(k) \rangle. k \in n \} \)) \in G
  proof -
    let a = \( \{ \langle k,q(k) \rangle. k \in n \} \)
    from assms have n \in nat f:G \times G \to G a:n \to G 0 \in G G \neq 0
      using zero_monoid_oper ZF_fun_from_total by auto
    then show thesis using fold_props by simp
  qed
```

The reason we start from 0 in the definition of the summation sign in the monoid1 locale is that we want to be able to sum the empty list. Such sum of the empty list is 0.

```ocaml
lemma (in monoid1) sum_empty: assumes s:0\to G shows (\( \sum s \)) = 0
  using assms zero_monoid_oper fold_empty group0_1_L3A by simp
```

For nonempty lists our \( \Sigma \) is the same as Fold1.

```ocaml
lemma (in monoid1) sum_nonempty: assumes n \in nat s:succ(n)\to G
  shows (\( \sum s \)) = Fold(f,s(0),Tail(s))
      (\( \sum s \)) = Fold1(f,s)
  proof -
    from assms have s(0) \in G using empty_in_every_succ apply_funtype
      by simp
    with assms have (\( \sum s \)) = Fold(f,0\oplus s(0),Tail(s))
```

324
We can pull the first component of a sum of a nonempty list of monoid elements before the summation sign.

**Theorem (in monoid1) seq_sum_pull_first0**: Assumes \( n \in \text{nat} \ s: \text{succ}(n) \rightarrow G \) shows \( \sum s = s(0) \oplus (\sum \text{Tail}(s)) \)

**Proof** -

- From \( \text{assms} \) have \( s(0) \in G \) using \( \text{empty_in_every_succ apply_funtype} \) by simp
  
  - Assume \( n=0 \) with \( \text{assms} \) have \( \text{Tail}(s):0 \rightarrow G \) using \( \text{tail_props(1)} \) by blast
  
  - With \( \text{assms} \), have \( (\sum s) = s(0) \oplus (\sum \text{Tail}(s)) \)

  - Using \( \text{sum_nonempty(1) sum_empty zero_monoid OPER(2) group0_1_L3} \)

  - Fold_empty unit_is_neutral by simp

- Moreover
  
  - Assume \( n\neq0 \) with \( \text{assms}(1) \) obtain \( m \) where \( m \in \text{nat} \) and \( n = \text{succ}(m) \)

  - Using \( \text{Nat_ZF_1_L3} \) by blast

  - With \( \text{assms} \), have \( \text{Tail}(s):\text{succ}(m) \rightarrow G \) using \( \text{tail_props(1)} \)

  - By simp

  - Let \( a = \{(0,s(0))\} \)

  - From \( \langle s(0) \in G \rangle \) have \( \text{0\in nat a: succ(0) \rightarrow G} \)

  - Using \( \text{pair_func_singleton succ_explained by simp_all} \)

  - With \( \langle m \in \text{nat} \rangle \ <\langle \text{Tail}(s) : \text{succ}(m) \rightarrow G \rangle \)

  - Have \( f(\text{Fold1(f,a),Fold1(f,Tail(s))}) = \text{Fold1(f,Concat(a,Tail(s)))} \)

  - Using \( \text{semigr0_valid_in_monoid0 semigr0.prod_conc_distr} \)

  - By blast

  - With \( \text{assms} \), have \( \langle a: \text{succ(0) \rightarrow G} \rangle \ <\langle m \in \text{nat} \rangle \ <\langle \text{Tail}(s) : \text{succ}(m) \rightarrow G \rangle \)

  - Have \( (\sum s) = s(0) \oplus (\sum \text{Tail}(s)) \)

  - Using \( \text{semigr0_valid_in_monoid0 semigr0.prod_of_1elem pair_val} \)

  - \( \text{sum_nonempty(2) first_concat_tail by simp} \)

- Ultimately show thesis by auto

**Qed**

The first assertion of the next theorem is similar in content to \( \text{seq_sum_pull_first0} \) formulated in terms of the expression defining the list of monoid elements. The second one shows the dual statement: the last element of a sequence can be pulled out of the sequence and put after the summation sign. So, we are showing here that \( \sum_{k=0}^{n} q_k = q_0 \oplus \sum_{k=0}^{n-1} q_{k+1} = (\sum_{k=0}^{n-1} q_k) \oplus q_n \).

**Theorem (in monoid1) seq_sum_pull_one_elem**: Assumes \( n \in \text{nat} \ \forall k \in n \#+1. q(k) \in G \)
shows
\[(\sum_{k=0}^{n} q(k)). k\in n \#+ 1}) = q(0) \oplus (\sum_{k=0}^{n} q(k)). k\in n)\]
\[(\sum_{k=0}^{n} q(k)). k\in n \#+ 1}) = (\sum_{k=0}^{n} q(k)). k\in n) \oplus q(n)\]

proof -
let \(s = \{(k,q(k)). k\in n \#+ 1}\)
from assms(1) have \(0 \in n \#+ 1\) using empty_in_every_succ succ_add_one(1)
by simp
then have \(s(0) = q(0)\) by (rule ZF_fun_from_tot_val1)
from assms(2) have \(s: n \#+ 1 \rightarrow G\) by (rule ZF_fun_from_total)
with assms(1) \(s(0) = q(0)\) have \((\sum s) = q(0) \oplus (\sum \text{Tail}(s))\)
by simp
moreover from assms have \(\text{Tail}(s) = \{(k,q(k \#+ 1)). k \in n\}\)
using tail_formula by simp
ultimately show \((\sum_{k=0}^{n} q(k)). k\in n \#+ 1}) = q(0) \oplus (\sum_{k=0}^{n} q(k)). k\in n)\)
by simp
from assms show \((\sum_{k=0}^{n} q(k)). k\in n \#+ 1}) = q(0) \oplus (\sum_{k=0}^{n} q(k)). k\in n)\)
using zero_monoid_oper fold_detach_last by simp
qed

The sum of a singleton list is its only element,

\[\text{lemma (in monoid1) seq_sum_singleton: assumes } q(0) \in G\]
\[\text{shows } (\sum_{k=0}^{1} q(k)). k\in 1) = q(0)\]
\[\text{proof -}\]
from assms have \(0 \in \text{nat} \text{ and } \forall k\in 0 \#+ 1. q(k) \in G\) by simp_all
then have \((\sum_{k=0}^{0} q(k)). k\in 0 \#+ 1}) = q(0) \oplus (\sum_{j=0}^{n} q(k \#+ 1)). k\in 0)\)
by (rule seq_sum_pull_one_elem)
with assms show thesis using sum_empty unit_is_neutral by simp
qed

If the monoid operation is commutative, then the sum of a nonempty sequence added to another sum of a nonempty sequence of the same length is equal to the sum of pointwise sums of the sequence elements. This is the same as the theorem prod_comm_distrib from the Semigroup_ZF theory, just written in the notation used in the monoid1 locale.

\[\text{lemma (in monoid1) sum_comm_distrib0:}\]
\[\text{assumes } f \{\text{is commutative on}\} G \text{ n\in \text{nat} and}\]
\[a : n \#+ 1 \rightarrow G \text{ b : n \#+ 1 \rightarrow G \text{ c : n \#+ 1 \rightarrow G and}\}
\[\forall j\in n \#+ 1. c(j) = a(j) \oplus b(j)\]
\[\text{shows } (\sum c) = (\sum a) \oplus (\sum b)\]
\[\text{using assms succ_add_one(1) sum_nonempty}\]
\[\text{semigr0.valid_in_monoid0 semigr0.prod_comm_distrib by simp}\]

Another version of \(\text{sum_comm_distrib0}\) written in terms of the expressions defining the sequences, shows that for commutative monoids we have \(\sum_{k=0}^{n-1} q(k) \oplus p(k) = (\sum_{k=0}^{n-1} q(k)) \oplus (\sum_{k=0}^{n-1} q(k))\).
theorem (in monoid1) sum_comm_distrib:
assumes f {is commutative on} G
\forall k \in \mathbb{N}. p(k) \in G \\forall k \in \mathbb{N}. q(k) \in G
shows
(\sum\{\langle k, p(k) \oplus q(k) \rangle. k \in \mathbb{N}\}) = (\sum\{\langle k, p(k) \rangle. k \in \mathbb{N}\}) \oplus (\sum\{\langle k, q(k) \rangle. k \in \mathbb{N}\})

proof -
let a = \{\langle k, p(k) \rangle. k \in \mathbb{N}\}
let b = \{\langle k, q(k) \rangle. k \in \mathbb{N}\}
let c = \{\langle k, p(k) \oplus q(k) \rangle. k \in \mathbb{N}\}
{ assume n=0
then have (\sum c) = (\sum a) \oplus (\sum b)
using sum_empty unit_is_neutral by simp
}
moreover
{ assume n\neq 0
with assms(2) obtain m where m\in \mathbb{N} and n = m \# 1
using nat_not0_succ by blast
from assms(3,4) have a:n\rightarrow G b:n\rightarrow G c:n\rightarrow G
using group0_1_L1 ZF_fun_from_total by simp_all
with assms(1) \langle m\in \mathbb{N}\rangle \langle n = m \# 1 \rangle have
f {is commutative on} G m\in \mathbb{N} and
a:m \# 1\rightarrow G b:m \# 1\rightarrow G c:m \# 1 \rightarrow G
by simp_all
moreover have \forall k \in m \# 1. c(k) = a(k) \oplus b(k)
proof -
{ fix k assume k \in m \# 1
with \langle n = m \# 1 \rangle have k\in\mathbb{N} by simp
then have c(k) = a(k) \oplus b(k)
using ZF_fun_from_tot_val1 by simp_all
} thus thesis by simp
qed
ultimately have (\sum c) = (\sum a) \oplus (\sum b)
using sum_comm_distrib0 by simp
}
ultimately show thesis by blast
qed

30.2 Multiplying monoid elements by natural numbers

A special case of summing (or, using more notation-neutral term folding) a list of monoid elements is taking a natural multiple of a single element. This can be applied to various monoids embedded in other algebraic structures. For example a ring is a monoid with addition as the operation, so the notion of natural multiple directly transfers there. Another monoid in a ring is formed by its multiplication operation. In that case the natural multiple maps into natural powers of a ring element.

The zero’s multiple of a monoid element is its neutral element.
lemma (in monoid1) nat_mult_zero: shows 0·x = 0 using sum_empty by simp

Any multiple of a monoid element is a monoid element.

lemma (in monoid1) nat_mult_type: assumes n∈nat x∈G
  shows n·x ∈ G using assms sum_in_mono by simp

Taking one more multiple of x adds x.

lemma (in monoid1) nat_mult_add_one: assumes n∈nat x∈G
  shows (n #+ 1)·x = n·x ⊕ x and (n #+ 1)·x = x ⊕ n·x
proof -
  from assms(2) have I: ∀k∈n #+ 1. x ∈ G by simp
  with assms(1) have (∑{(k,x). k ∈ n #+ 1}) = x ⊕ (∑{(k,x). k∈n})
    by (rule seq_sum_pull_one_elem)
  thus (n #+ 1)·x = x ⊕ n·x by simp
  from assms(1) I have (∑{(k,x). k∈n #+ 1}) = (∑{(k,x). k∈n}) ⊕ x
    by (rule seq_sum_pull_one_elem)
  thus (n #+ 1)·x = n·x ⊕ x by simp
qed

One element of a monoid is that element.

lemma (in monoid1) nat_mult_one: assumes x∈G
  shows 1·x = x
proof -
  from assms have (0 #+ 1)·x = 0·x ⊕ x using nat_mult_add_one(1) by blast
  with assms show thesis using nat_mult_zero unit_is_neutral by simp
qed

Multiplication of x by a natural number induces a homomorphism between
natural numbers with addition and and the natural multiples of x.

lemma (in monoid1) nat_mult_add: assumes n∈nat m∈nat x∈G
  shows (n #+ m)·x = n·x ⊕ m·x
proof -
  from assms have m∈nat and (n #+ 0)·x = n·x ⊕ 0·x
    using nat_mult_type unit_is_neutral nat_mult_zero by simp_all
  moreover from assms(1,3)
    have ∀k∈nat. (n #+ k)·x = n·x ⊕ k·x → (n #+ (k #+ 1))·x = n·x ⊕ (k #+ 1)·x
      using nat_mult_type nat_mult_add_one(1) sum_associative by simp
  ultimately show thesis by (rule ind_on_nat1)
qed

end

31 Groups - introduction

theory Group_ZF imports Monoid_ZF
begin
This theory file covers basics of group theory.

### 31.1 Definition and basic properties of groups

In this section we define the notion of a group and set up the notation for discussing groups. We prove some basic theorems about groups.

To define a group we take a monoid and add a requirement that the right inverse needs to exist for every element of the group.

**definition**

\[
\text{IsAgroup}(G,f) \equiv (\text{IsAmonoid}(G,f) \land (\forall g \in G. \exists b \in G. f(g,b) = \text{TheNeutralElement}(G,f)))
\]

We define the group inverse as the set \( \{ (x,y) \in G \times G : x \cdot y = e \} \), where \( e \) is the neutral element of the group. This set (which can be written as \( (-1)^{-1}\{e\} \)) is a certain relation on the group (carrier). Since, as we show later, for every \( x \in G \) there is exactly one \( y \in G \) such that \( x \cdot y = e \) this relation is in fact a function from \( G \) to \( G \).

**definition**

\[
\text{GroupInv}(G,f) \equiv \{ (x,y) \in G \times G. f(x,y) = \text{TheNeutralElement}(G,f) \}
\]

We will use the multiplicative notation for groups. The neutral element is denoted 1.

**locale** group0 =

fixes \( G \)

fixes \( P \)

assumes groupAssum: IsAgroup(G,P)

fixes neut (1)

defines neut_def[simp]: \( 1 \equiv \text{TheNeutralElement}(G,P) \)

fixes groper (infixl \( \cdot \)) 70

defines groper_def[simp]: \( a \cdot b \equiv P(a,b) \)

fixes inv (\( _{-1} \)) [90] 91

defines inv_def[simp]: \( x^{-1} \equiv \text{GroupInv}(G,P)(x) \)

First we show a lemma that says that we can use theorems proven in the monoid context (locale).

**lemma** (in group0) group0_2_L1: shows monoid0(G,P)

using groupAssum IsAgroup_def monoid0_def by simp

The theorems proven in the monoid context are valid in the group0 context.

**sublocale** group0 < monoid: monoid0 G P groper

unfolding groper_def using group0_2_L1 by auto
In some strange cases Isabelle has difficulties with applying the definition of a group. The next lemma defines a rule to be applied in such cases.

**lemma definition_of_group:** assumes IsAmonoid(G,f)
and ∀ g ∈ G. ∃ b ∈ G. f(g,b) = TheNeutralElement(G,f)
shows IsAgroup(G,f)
using assms IsAgroup_def by simp

A technical lemma that allows to use 1 as the neutral element of the group without referencing a list of lemmas and definitions.

**lemma (in group0) group0_2_L2:** shows 1 ∈ G ∧ (∀ g ∈ G. 1 · g = g ∧ g · 1 = g)
using group0_2_L1 monoid.unit_is_neutral by simp

The group is closed under the group operation. Used all the time, useful to have handy.

**lemma (in group0) group_op_closed:** assumes a ∈ G b ∈ G
shows a · b ∈ G using assms monoid.group0_1_L1 by simp

The group operation is associative. This is another technical lemma that allows to shorten the list of referenced lemmas in some proofs.

**lemma (in group0) group_oper_assoc:** assumes a ∈ G b ∈ G c ∈ G
shows a · (b · c) = a · b · c
using groupAssum assms IsAgroup_def IsAmonoid_def IsAssociative_def group_op_closed by simp

The group operation maps $G \times G$ into $G$. It is convenient to have this fact easily accessible in the group0 context.

**lemma (in group0) group_oper_fun:** shows $P : G \times G \rightarrow G$
using groupAssum IsAgroup_def IsAmonoid_def IsAssociative_def by simp

The definition of a group requires the existence of the right inverse. We show that this is also the left inverse.

**theorem (in group0) group0_2_T1:**
assumes A1: $g \in G$ and A2: $b \in G$ and A3: $g \cdot b = 1$
sows $b \cdot g = 1$

**proof** -
from A2 groupAssum obtain c where I: $c \in G \land b \cdot c = 1$
using IsAgroup_def by auto
then have $c \in G$ by simp
have $1 \in G$ using group0_2_L2 by simp
with A1 A2 I have $b \cdot g = b \cdot (g \cdot (b \cdot c))$
using group_op_closed group0_2_L2 group_oper_assoc by simp
also from A1 A2 $\langle c \in G \rangle$ have $b \cdot (g \cdot (b \cdot c)) = b \cdot (g \cdot b \cdot c)$
using group_oper_assoc by simp
also from A3 A2 I have b·(g·b·c) = 1 using group0_2_L2 by simp 
finally show b·g = 1 by simp 
qed 

For every element of a group there is only one inverse. 

lemma (in group0) group0_2_L4: 
  assumes A1: x∈G shows ∃!y. y∈G ∧ x·y = 1 
proof 
  from A1 groupAssum show ∃y. y∈G ∧ x·y = 1 
    using IsAgroup_def by auto 
  fix y n 
  assume A2: y∈G ∧ x·y = 1 and A3:n∈G ∧ x·n = 1 show y=n 
  proof - 
    from A1 A2 have T1: y·x = 1 
      using group0_2_T1 by simp 
    from A2 A3 have y = y·(x·n) 
      using group0_2_L2 by simp 
    also from A1 A2 A3 have ... = (y·x)·n 
      using group_oper_assoc by blast 
    also from T1 A3 have ... = n 
      using group0_2_L2 by simp 
    finally show y=n by simp 
  qed 
  qed 

The group inverse is a function that maps G into G. 

theorem group0_2_T2: 
  assumes A1: IsAgroup(G,f) shows GroupInv(G,f) : G→G 
proof - 
  have GroupInv(G,f) ⊆ G×G using GroupInv_def by auto 
  moreover from A1 have 
    ∀x∈G. ∃y. y∈G ∧ (x,y) ∈ GroupInv(G,f) 
      using group0_def group0.group0_2_L4 GroupInv_def by simp 
  ultimately show thesis using func1_1_L11 by simp 
  qed 

We can think about the group inverse (the function) as the inverse image of the neutral element. Recall that in Isabelle f−1(A) denotes the inverse image of the set A. 

theorem (in group0) group0_2_T3: shows P−{1} = GroupInv(G,P) 
proof - 
  from groupAssum have P : G×G → G 
    using IsAgroup_def IsAmonoid_def IsAssociative_def by simp 
  then show P−{1} = GroupInv(G,P) 
    using func1_1_L14 GroupInv_def by auto 
  qed 

The inverse is in the group. 

331
lemma (in group0) inverse_in_group: assumes A1: \( x \in G \) shows \( x^{-1} \in G \)
proof
  from groupAssum have GroupInv\((G,P)\) : \( G \rightarrow G \) using group0_2_T2 by simp
  with A1 show thesis using apply_type by simp
qed

The notation for the inverse means what it is supposed to mean.

lemma (in group0) group0_2_L6:
  assumes A1: \( x \in G \) shows \( xx^{-1} = 1 \land x^{-1}x = 1 \)
proof
  from groupAssum have GroupInv\((G,P)\) : \( G \rightarrow G \)
    using group0_2_T2 by simp
  with A1 have \( \langle x,x^{-1} \rangle \in \text{GroupInv}(G,P) \)
    using apply_Pair by simp
  then show \( xx^{-1} = 1 \) using GroupInv_def by simp
  with A1 show \( x^{-1}x = 1 \) using inverse_in_group group_oper_assoc group0_2_L7
    by blast
qed

The next two lemmas state that unless we multiply by the neutral element, the result is always different than any of the operands.

lemma (in group0) group0_2_L7:
  assumes A1: \( a \in G \) and A2: \( b \in G \) and A3: \( a \cdot b = a \)
  shows \( b = 1 \)
proof
  from A3 have \( a^{-1} \cdot (a \cdot b) = a^{-1}a \) by simp
  with A1 A2 show thesis using
    inverse_in_group group_oper_assoc group0_2_L6 group0_2_L2
    by simp
qed

See the comment to group0_2_L7.

lemma (in group0) group0_2_L8:
  assumes A1: \( a \in G \) and A2: \( b \in G \) and A3: \( a \cdot b = b \)
  shows \( a = 1 \)
proof
  from A3 have \( (a \cdot b) \cdot b^{-1} = b \cdot b^{-1} \) by simp
  with A1 A2 have \( a \cdot (b \cdot b^{-1}) = b \cdot b^{-1} \) using
    inverse_in_group group_oper_assoc by simp
  with A1 A2 show thesis
    using group0_2_L6 group0_2_L2 by simp
qed

The inverse of the neutral element is the neutral element.

lemma (in group0) group_inv_of_one: shows \( 1^{-1} = 1 \)
  using group0_2_L7 inverse_in_group group0_2_L6 group0_2_L7 by blast
if \( a^{-1} = 1 \), then \( a = 1 \).
lemma (in group0) group0_2_L8A:
assumes A1: a ∈ G and A2: a⁻¹ = 1
shows a = 1
proof -
  from A1 have a·a⁻¹ = 1 using group0_2_L6 by simp
  with A1 A2 show a = 1 using group0_2_L2 by simp
qed

If a is not a unit, then its inverse is not a unit either.

lemma (in group0) group0_2_L8B:
assumes a ∈ G and a ≠ 1
shows a⁻¹ ≠ 1 using assms group0_2_L8A by auto

If a⁻¹ is not a unit, then a is not a unit either.

lemma (in group0) group0_2_L8C:
assumes a ∈ G and a⁻¹ ≠ 1
shows a ≠ 1
  using assms group0_2_L8A group_inv_of_one by auto

If a product of two elements of a group is equal to the neutral element then
they are inverses of each other.

lemma (in group0) group0_2_L9:
assumes A1: a ∈ G and A2: b ∈ G and A3: a·b = 1
shows a = b⁻¹ and b = a⁻¹
proof -
  from A3 have a·b⁻¹ = 1·b⁻¹ by simp
  with A1 A2 have a·(b·b⁻¹) = 1·b⁻¹ using
    inverse_in_group group_oper_assoc by simp
  with A1 A2 show a = b⁻¹ using
    group0_2_L6 inverse_in_group group0_2_L2 by simp
  from A3 have a⁻¹·(a·b) = a⁻¹·1 by simp
  with A1 A2 show b = a⁻¹ using
    inverse_in_group group_oper_assoc group0_2_L6 group0_2_L2
    by simp
qed

It happens quite often that we know what is (have a meta-function for) the
right inverse in a group. The next lemma shows that the value of the group
inverse (function) is equal to the right inverse (meta-function).

lemma (in group0) group0_2_L9A:
assumes A1: ∀g∈G. b(g) ∈ G ∧ g·b(g) = 1
shows ∀g∈G. b(g) = g⁻¹
proof
  fix g assume g∈G
  moreover from A1 ⟨g∈G⟩ have b(g) ∈ G by simp
  moreover from A1 ⟨g∈G⟩ have g·b(g) = 1 by simp
  ultimately show b(g) = g⁻¹ by (rule group0_2_L9)
qed
What is the inverse of a product?

lemma (in group0) group_inv_of_two:
  assumes A1: a∈G and A2: b∈G
  shows  b⁻¹·a⁻¹ = (a·b)⁻¹
proof -
  from A1 A2 have
  b⁻¹∈G  a⁻¹∈G  a·b∈G  b⁻¹·a⁻¹ ∈ G
    using inverse_in_group group_op_closed
    by auto
  from A1 A2  b⁻¹∈G  a⁻¹∈G  have a·b·(b⁻¹·a⁻¹) = a·(b⁻¹·a⁻¹)
    using group_oper_assoc by simp
  moreover from A2  b⁻¹∈G  a⁻¹∈G  have b·(b⁻¹·a⁻¹) = b·b⁻¹·a⁻¹
    using group_oper_assoc by simp
  moreover from A2  a⁻¹∈G  have b·b⁻¹·a⁻¹ = a⁻¹
    using group0_2_L6 group0_2_L2 by simp
  ultimately have a·b·(b⁻¹·a⁻¹) = a·a⁻¹
    by simp
  with A1 have a·b·(b⁻¹·a⁻¹) = 1
    using group0_2_L6 by simp
  with  a·b∈G  b⁻¹·a⁻¹ ∈ G  show b⁻¹·a⁻¹ = (a·b)⁻¹
    using group0_2_L9 by simp
qed

What is the inverse of a product of three elements?

lemma (in group0) group_inv_of_three:
  assumes A1: a∈G  b∈G  c∈G
  shows (a·b·c)⁻¹ = c⁻¹·(a·b)⁻¹
  (a·b·c)⁻¹ = c⁻¹·(b⁻¹·a⁻¹)
  (a·b·c)⁻¹ = c⁻¹·b⁻¹·a⁻¹
proof -
  from A1 have T:
  a·b∈G  a⁻¹∈G  b⁻¹ ∈ G  c⁻¹∈G
    using group_op_closed inverse_in_group by auto
  with A1 show
  (a·b·c)⁻¹ = c⁻¹·(a·b)⁻¹ and (a·b·c)⁻¹ = c⁻¹·(b⁻¹·a⁻¹)
    using group_inv_of_two by auto
  with T show (a·b·c)⁻¹ = c⁻¹·b⁻¹·a⁻¹ using group_oper_assoc
    by simp
qed

The inverse of the inverse is the element.

lemma (in group0) group_inv_of_inv:
  assumes a∈G
  shows a = (a⁻¹)⁻¹
using assms inverse_in_group group0_2_L6 group0_2_L9 by simp

Group inverse is nilpotent, therefore a bijection and involution.

lemma (in group0) group_inv_bij:

334
shows \( \text{GroupInv}(G,P) \circ \text{GroupInv}(G,P) = id(G) \) and \( \text{GroupInv}(G,P) \in \text{bij}(G,G) \)

**proof** -

- have \( I: \text{GroupInv}(G,P): G \rightarrow G \) using groupAssum group0_2_T2 by simp
- then have \( \text{GroupInv}(G,P) \circ \text{GroupInv}(G,P): G \rightarrow G \) and \( id(G): G \rightarrow G \)
  - using comp_fun id_type by auto
- moreover
  - \{ fix \( g \) assume \( g \in G \) with I have \( (\text{GroupInv}(G,P) \circ \text{GroupInv}(G,P))(g) = id(G)(g) \)
    - using comp_fun_apply group_inv_of_inv id_conv by simp
  \}
- hence \( \forall g \in G. (\text{GroupInv}(G,P) \circ \text{GroupInv}(G,P))(g) = id(G)(g) \)
  - by (rule func_eq)
- with I show \( \text{GroupInv}(G,P) \in \text{bij}(G,G) \) using nilpotent_imp_bijective by simp
- with \( \text{GroupInv}(G,P) \circ \text{GroupInv}(G,P) = id(G) \) show \( \text{GroupInv}(G,P) = \text{converse}(\text{GroupInv}(G,P)) \) using comp_id_conv by simp

**qed**

A set comprehension form of the image of a set under the group inverse.

**lemma** (in group0) ginv_image: assumes \( V \subseteq G \)
  shows \( \text{GroupInv}(G,P)(V) \subseteq G \) and \( \text{GroupInv}(G,P)(V) = \{ g^{-1}. g \in V \} \)

**proof** -

- from assms have \( I: \text{GroupInv}(G,P)(V) = \{ \text{GroupInv}(G,P)(g). g \in V \} \)
  - using groupAssum group0_2_T2 func_imagedef by blast
- thus \( \text{GroupInv}(G,P)(V) = \{ g^{-1}. g \in V \} \) by simp
- show \( \text{GroupInv}(G,P)(V) \subseteq G \) using groupAssum group0_2_T2 func1_1_L6(2) by blast

**qed**

Inverse of an element that belongs to the inverse of the set belongs to the set.

**lemma** (in group0) ginv_image_el: assumes \( V \subseteq G \) \( g \in \text{GroupInv}(G,P)(V) \)
  shows \( g^{-1} \in V \)

**proof** -

- using assms ginv_image group_inv_of_inv by auto

For the group inverse the image is the same as inverse image.

**lemma** (in group0) inv_image_vimage: shows \( \text{GroupInv}(G,P)(V) = \text{GroupInv}(G,P)-(V) \)
  using group_inv_bij vimage_converse by simp

If the unit is in a set then it is in the inverse of that set.

**lemma** (in group0) neut_inv_neut: assumes \( A \subseteq G \) and \( 1 \in A \)
  shows \( 1 \in \text{GroupInv}(G,P)(A) \)

**proof** -

- have \( \text{GroupInv}(G,P): G \rightarrow G \) using groupAssum group0_2_T2 by simp
- with assms have \( 1^{-1} \in \text{GroupInv}(G,P)(A) \) using func_imagedef by auto
- then show thesis using group_inv_of_one by simp

335
The group inverse is onto.

**Lemma (in group0) group_inv_surj**: shows $\text{GroupInv}(G,P)(G) = G$

using `group_inv_bij bij_def surj_range_image_domain` by auto

If $a^{-1} \cdot b = 1$, then $a = b$.

**Lemma (in group0) group0_2_L11**: assumes $A1: a \in G \quad b \in G$ and $A2: a^{-1} \cdot b = 1$

shows $a = b$

proof -
  from $A1 \ A2$ have $a^{-1} \in G \quad b \in G \quad a^{-1} \cdot b = 1$
  using `inverse_in_group` by auto
  then have $b = (a^{-1})^{-1}$ by (rule group0_2_L9)
  with $A1$ show $a = b$ using `group_inv_of_inv` by simp

qed

If $a \cdot b^{-1} = 1$, then $a = b$.

**Lemma (in group0) group0_2_L11A**: assumes $A1: a \in G \quad b \in G$ and $A2: a \cdot b^{-1} = 1$

shows $a = b$

proof -
  from $A1 \ A2$ have $a \in G \quad b^{-1} \in G \quad a \cdot b^{-1} = 1$
  using `inverse_in_group` by auto
  then have $a = (b^{-1})^{-1}$ by (rule group0_2_L9)
  with $A1$ show $a = b$ using `group_inv_of_inv` by simp

qed

If if the inverse of $b$ is different than $a$, then the inverse of $a$ is different than $b$.

**Lemma (in group0) group0_2_L11B**: assumes $A1: a \in G \quad A2: b^{-1} \neq a$

shows $a^{-1} \neq b$

proof -
  { assume $a^{-1} = b$
    then have $(a^{-1})^{-1} = b^{-1}$ by simp
    with $A1 \ A2$ have False using `group_inv_of_inv` by simp
  } then show $a^{-1} \neq b$ by auto

qed

What is the inverse of $ab^{-1}$?

**Lemma (in group0) group0_2_L12**: assumes $A1: a \in G \quad b \in G$

shows $(a \cdot b^{-1})^{-1} = b \cdot a^{-1}$

$(a^{-1} \cdot b)^{-1} = b^{-1} \cdot a$

proof -
from A1 have
  \((a \cdot b)^{-1} = (b^{-1})^{-1}, a^{-1}\) and \((a^{-1} \cdot b)^{-1} = b^{-1} \cdot (a^{-1})^{-1}\)
  using `inverse_in_group` `group_inv_of_two` by auto
with A1 show \((a \cdot b)^{-1} = b^{-1} \cdot a^{-1}\) \((a^{-1} \cdot b)^{-1} = b^{-1} \cdot a^{-1}\)
  using `group_inv_of_inv` by auto
qed

A couple useful rearrangements with three elements: we can insert a \(b \cdot b^{-1}\) between two group elements (another version) and one about a product of an element and inverse of a product, and two others.

lemma (in group0) `group0_2_L14A`:
  assumes `A1: a \in G \ b \in G \ c \in G`
  shows
  \(a \cdot c^{-1} = (a \cdot b^{-1}) \cdot (b \cdot c^{-1})\)
  \(a^{-1} \cdot c = (a^{-1} \cdot b) \cdot (b^{-1} \cdot c)\)
  \(a \cdot (b \cdot c)^{-1} = a \cdot c^{-1} \cdot b^{-1}\)
  \(a \cdot (b^{-1} \cdot c)^{-1} = a \cdot b \cdot c^{-1}\)
  \((a \cdot b^{-1} \cdot c)^{-1} = c \cdot b^{-1} \cdot a^{-1}\)
  \(a \cdot b^{-1} \cdot (c \cdot b^{-1}) = a\)
  \(a \cdot (b \cdot c)^{-1} = a \cdot b\)
proof-
  from A1 have T:
    \(a^{-1} \in G \ b^{-1} \in G \ c^{-1} \in G\)
    \(a^{-1} \cdot b \in G \ a \cdot b^{-1} \in G \ a \cdot b \in G\)
    \(c \cdot b^{-1} \in G \ b \cdot c \in G\)
    using `inverse_in_group` `group_op_closed` by auto
  from A1 T have
    \(a \cdot c^{-1} = a \cdot (b^{-1} \cdot b) \cdot c^{-1}\)
    \(a^{-1} \cdot c = a^{-1} \cdot (b \cdot b^{-1}) \cdot c\)
    using `group0_2_L2` `group0_2_L6` by auto
  with A1 T show
    \(a \cdot c^{-1} = a \cdot (b^{-1} \cdot (b \cdot c^{-1}))\)
    \(a^{-1} \cdot c = (a^{-1} \cdot (b^{-1} \cdot (b^{-1} \cdot c)))\)
    using `group_oper_assoc` by auto
  from A1 have \(a \cdot (b \cdot c)^{-1} = a \cdot (c^{-1} \cdot b^{-1})\)
    using `group_inv_of_two` by simp
  with A1 T show \(a \cdot (b \cdot c)^{-1} = a \cdot c^{-1} \cdot b^{-1}\)
    using `group_oper_assoc` by simp
  from A1 T show \(a \cdot (b \cdot c)^{-1} = a \cdot b \cdot c^{-1}\)
    using `group_oper_assoc` by simp
  from A1 T show \((a^{-1} \cdot b^{-1} \cdot c)^{-1} = c \cdot b^{-1} \cdot a^{-1}\)
    using `group_inv_of_three` `group_inv_of_inv` by simp
  from T have \(a \cdot b \cdot c^{-1} \cdot (c \cdot b^{-1}) = a \cdot b \cdot (c^{-1} \cdot (c \cdot b^{-1}))\)
    using `group_oper_assoc` by simp
  also from A1 T have \(a \cdot b \cdot b^{-1}\)
    using `group_oper_assoc` `group0_2_L6` `group0_2_L2`
    by simp
also from A1 T have ... = \( a \cdot (b \cdot b^{-1}) \)
using group_oper_assoc by simp
also from A1 have ... = \( a \)
using group0_2_L6 group0_2_L2 by simp
finally show \( a \cdot (b \cdot c) \cdot c^{-1} = a \cdot (b \cdot (c \cdot c^{-1})) \)
using group_oper_assoc by simp
also from A1 T have ... = \( a \cdot b \)
using group0_2_L6 group0_2_L2 by simp
finally show \( a \cdot (b \cdot c) \cdot c^{-1} = a \cdot b \)
by simp
qed

A simple equation to solve

**lemma (in group0) simple_equation0:**

assumes \( a \in G \ b \in G \ c \in G \ a \cdot b^{-1} = c^{-1} \)
shows \( c = b \cdot a^{-1} \)

**proof -**
from assms(4) have \( (a \cdot b^{-1})^{-1} = (c^{-1})^{-1} \) by simp
with assms(1,2,3) show \( c = b \cdot a^{-1} \) using group0_2_L12(1) group_inv_of_inv
by simp
qed

Another simple equation

**lemma (in group0) simple_equation1:**

assumes \( a \in G \ b \in G \ c \in G \ a^{-1} \cdot b = c^{-1} \)
shows \( c = b^{-1} \cdot a \)

**proof -**
from assms(4) have \( (a^{-1} \cdot b)^{-1} = (c^{-1})^{-1} \) by simp
with assms(1,2,3) show \( c = b^{-1} \cdot a \) using group0_2_L12(2) group_inv_of_inv
by simp
qed

Another lemma about rearranging a product of four group elements.

**lemma (in group0) group0_2_L15:**

assumes A1: \( a \in G \ b \in G \ c \in G \ d \in G \)
shows \( (a \cdot b) \cdot (c \cdot d)^{-1} = a \cdot (b \cdot d^{-1})^{-1} \cdot (a \cdot c^{-1}) \)

**proof -**
from A1 have T1:
\( d^{-1} \in G \ c^{-1} \in G \ a \cdot b \in G \ a \cdot (b \cdot d^{-1}) \in G \)
using inverse_in_group group_op_closed
by auto
with A1 have \( (a \cdot b) \cdot (c \cdot d)^{-1} = (a \cdot b) \cdot (d^{-1} \cdot c^{-1}) \)
using group_inv_of_two by simp
also from A1 T1 have ... = \( a \cdot (b \cdot d^{-1}) \cdot c^{-1} \)
using group_oper_assoc by simp
also from A1 T1 have ... = \( a \cdot (b \cdot d^{-1}) \cdot a^{-1} \cdot (a \cdot c^{-1}) \)
using group0_2_L14A by blast
finally show thesis by simp

338
We can cancel an element with its inverse that is written next to it.

**lemma (in group0) inv_cancel_two:**

assumes $A1: a \in G \ b \in G$

shows $a \cdot b \cdot b^{-1} = a$

proof -

from $A1$ have

$a \cdot b^{-1} = a \cdot (b^{1} \cdot b)$

$a^{-1} \cdot (a \cdot b) = b$

$a \cdot (a^{-1} \cdot b) = b$

using inverse_in_group group_oper_assoc by auto

with $A1$ show $a \cdot b \cdot b^{-1} = a$

using group0_2_L6 group0_2_L2 by auto

qed

Another lemma about cancelling with two group elements.

**lemma (in group0) group0_2_L16A:**

assumes $A1: a \in G \ b \in G$

shows $a \cdot (b \cdot a)^{-1} = b$

proof -

from $A1$ have $(b \cdot a)^{-1} = a^{-1} \cdot b^{-1} \ b^{-1} \in G$

using group_inv_of_two inverse_in_group by auto

with $A1$ show $a \cdot (b \cdot a)^{-1} = b$

using inv_cancel_two by simp

qed

Some other identities with three elements and cancelling.

**lemma (in group0) cancel_middle:**

assumes $a \in G \ b \in G \ c \in G$

shows $(a \cdot b)^{-1} \cdot (a \cdot c) = b^{-1} \cdot c$

$(a \cdot b) \cdot (c \cdot b)^{-1} = a \cdot c^{-1}$

$a^{-1} \cdot (a \cdot b \cdot c)^{-1} = b$

$a \cdot (b^{-1} \cdot c) = a \cdot b$

$a \cdot b^{-1} \cdot (b^{-1} \cdot c) = a \cdot c^{-1}$

proof -

from assms have $(a \cdot b)^{-1} \cdot (a \cdot c) = b^{-1} \cdot (a^{-1} \cdot (a \cdot c))$

using group_inv_of_two inverse_in_group group_oper_assoc group_op_closed by auto

with assms(1,3) show $(a \cdot b)^{-1} \cdot (a \cdot c) = b^{-1} \cdot c$

using inv_cancel_two(3) by simp
from assms have \((a \cdot b) \cdot (c \cdot b)^{-1} = a \cdot (b^{-1} \cdot c^{-1})\)
  using group_inv_of_two inverse_in_group group_oper_assoc group_op_closed
by auto
with assms show \((a \cdot b) \cdot (c \cdot b)^{-1} = a \cdot c^{-1}\)
  using inverse_in_group inv_cancel_two(4)
by simp
from assms have \(a^{-1} \cdot (a \cdot b \cdot c) \cdot c^{-1} = \) \(a^{-1} \cdot (a^{-1} \cdot a) \cdot (b \cdot c)\)
  using inverse_in_group group_oper_assoc group_op_closed by auto
with assms show \(a^{-1} \cdot (a \cdot b \cdot c) \cdot c^{-1} = b\)
  using group0_2_L6 group0_2_L2 by simp
moreover from assms have \(a^{-1} \cdot (b^{-1} \cdot c^{-1}) = a \cdot b \cdot (c^{-1})\)
  using inverse_in_group group_oper_assoc group_op_closed
by simp
moreover from assms have \(a^{-1} \cdot (b^{-1} \cdot c^{-1}) = a \cdot (b^{-1} \cdot c^{-1})\)
  using inverse_in_group group_oper_assoc
by simp
ultimately show \(a^{-1} \cdot (b^{-1} \cdot c^{-1}) = a \cdot c^{-1}\)
  using group0_2_L6 group0_2_L2
by simp
qed

Adding a neutral element to a set that is closed under the group operation
results in a set that is closed under the group operation.

lemma (in group0) group0_2_L17:
  assumes \(H \subseteq G\)
  and \(H \{ \text{is closed under} \} P\)
  shows \((H \cup \{1\}) \{ \text{is closed under} \} P\)
  using assms IsOpClosed_def group0_2_L2
by auto

We can put an element on the other side of an equation.

lemma (in group0) group0_2_L18:
  assumes \(A1: a \in G \text{ b} \in G\)
  and \(A2: c = a \cdot b\)
  shows \(c \cdot b^{-1} = a \cdot a^{-1} \cdot c = b\)
proof-
  from \(A2 \ A1\) have \(c \cdot b^{-1} = a \cdot (b^{-1} \cdot c)\)
  using inverse_in_group group_oper_assoc
by auto
moreover from \(A1\) have \(a \cdot (b^{-1} \cdot c) = a \cdot (a^{-1} \cdot a) \cdot b\)
  using group0_2_L6 group0_2_L2
by auto
ultimately show \(c \cdot b^{-1} = a \cdot a^{-1} \cdot c = b\)
  by auto
qed

We can cancel an element on the right from both sides of an equation.

lemma (in group0) cancel_right: assumes \(a \in G \text{ b} \in G\)
  and \(a \cdot b = c \cdot b\)
  shows \(a = c\)
proof -
  from assms \(4\) have \(a \cdot b \cdot b^{-1} = c \cdot b \cdot b^{-1}\)
  by simp
with assms \((1, 2, 3)\) show thesis using inv_cancel_two \(2\)
  by simp
We can cancel an element on the left from both sides of an equation.

**lemma** (in group0) **cancel_left**: assumes \( a \in G \) \( b \in G \) \( c \in G \) \( a \cdot b = a \cdot c \) shows \( b = c \)

**proof**
- from assms(4) have \( a^{-1} \cdot (a \cdot b) = a^{-1} \cdot (a \cdot c) \) by simp
  with assms(1,2,3) show thesis using inv_cancel_two(3) by simp

**qed**

Multiplying different group elements by the same factor results in different group elements.

**lemma** (in group0) **group0_2_L19**:  
assumes \( A1: a \in G \) \( b \in G \) \( c \in G \) and \( A2: a \neq b \) shows \( a \cdot c \neq b \cdot c \) and \( c \cdot a \neq c \cdot b \)

**proof**
- \{ assume \( a \cdot c = b \cdot c \) \( \lor \) \( c \cdot a = c \cdot b \)  
  then have \( a \cdot c \cdot c^{-1} = b \cdot c \cdot c^{-1} \lor c^{-1} \cdot (c \cdot a) = c^{-1} \cdot (c \cdot b) \)  
  by auto  
  with \( A1 \) \( A2 \) have False using inv_cancel_two by simp  
\} then show \( a \cdot c \neq b \cdot c \) and \( c \cdot a \neq c \cdot b \) by auto

**qed**

### 31.2 Subgroups

There are two common ways to define subgroups. One requires that the group operation is closed in the subgroup. The second one defines subgroup as a subset of a group which is itself a group under the group operations. We use the second approach because it results in shorter definition.

The rest of this section is devoted to proving the equivalence of these two definitions of the notion of a subgroup.

A pair \((H,P)\) is a subgroup if \(H\) forms a group with the operation \(P\) restricted to \(H \times H\). It may be surprising that we don’t require \(H\) to be a subset of \(G\). This however can be inferred from the definition if the pair \((G,P)\) is a group, see lemma **group0_3_L2**.

**definition**

\[ \text{IsAsubgroup}(H,P) \equiv \text{IsAgroup}(H, \text{restrict}(P,H \times H)) \]

The group is its own subgroup.

**lemma** (in group0) **group_self_subgroup**: shows IsAsubgroup(G,P)  
using groupAssum group_oper_fun restrict_domain  
unfolding IsAsubgroup_def by simp

Formally the group operation in a subgroup is different than in the group as they have different domains. Of course we want to use the original operation...
with the associated notation in the subgroup. The next couple of lemmas will allow for that.

The next lemma states that the neutral element of a subgroup is in the subgroup and it is both right and left neutral there. The notation is very ugly because we don’t want to introduce a separate notation for the subgroup operation.

**Lemma (group0_3_L1):**

**Assumes A1:** IsAsSubgroup\((H,f)\) and A2: \(n = \text{TheNeutralElement}(H,\text{restrict}(f,H \times H))\)

**Shows** \(n \in H\)

\[
\forall h \in H. \text{restrict}(f,H \times H)(n, h) = h \\
\forall h \in H. \text{restrict}(f,H \times H)(h, n) = h
\]

**Proof**

- Let \(b = \text{restrict}(f,H \times H)\)
- Let \(e = \text{TheNeutralElement}(H,\text{restrict}(f,H \times H))\)
- From A1 have \(\text{group0}(H,b)\) using IsAsSubgroup_def group0_def by simp
- Then have I:
  - \(e \in H \land (\forall h \in H. (b(e,h) = h \land b(h,e) = h))\)
  - by (rule group0.group0_2_L2)
- With A2 show \(n \in H\) by simp
- From A2 I show \(\forall h \in H. b(n,h) = h\) and \(\forall h \in H. b(h,n) = h\)
  - by auto

**Qed**

A subgroup is contained in the group.

**Lemma (in group0) (group0_3_L2):**

**Assumes A1:** IsAsSubgroup\((H,P)\)

**Shows** \(H \subseteq G\)

**Proof**

- Fix \(h\) assume \(h \in H\)
- Let \(b = \text{restrict}(P,H \times H)\)
- Let \(n = \text{TheNeutralElement}(H,\text{restrict}(P,H \times H))\)
- From A1 have \(b \in H \times H \rightarrow H\) using IsAsSubgroup_def IsAgroup_def IsAmonoid_def IsAssociative_def by simp
- Moreover from A1 \(\langle h \in H \rangle\) have \(\langle n, h \rangle \in H \times H\)
  - using group0_3_L1 by simp
- Moreover from A1 \(\langle h \in H \rangle\) have \(h = b(n,h)\)
  - using group0_3_L1 by simp
- Ultimately have \(\langle \langle n, h \rangle, h \rangle \in b\)
  - using func1_1_L5A by blast
- Then have \(\langle \langle n, h \rangle, h \rangle \in P\) using restrict_subset by auto
- Moreover from groupAssum have \(P : G \times G \rightarrow G\)
  - using IsAgroup_def IsAmonoid_def IsAssociative_def by simp
- Ultimately show \(h \in G\) using func1_1_L5
  - by blast

342
The group's neutral element (denoted $1$ in the group0 context) is a neutral element for the subgroup with respect to the group action.

**lemma (in group0) group0_3_L3:**
- assumes IsAsubgroup(H,P)
- shows $\forall h \in H. \ 1 \cdot h = h \land h \cdot 1 = h$
  - using assms groupAssum group0_3_L2 group0_2_L2
  - by auto

The neutral element of a subgroup is the same as that of the group.

**lemma (in group0) group0_3_L4:**
- assumes A1: IsAsubgroup(H,P)
- shows TheNeutralElement(H,restrict(P,H×H)) = 1
  - proof -
    - let $n = \text{TheNeutralElement}(H,\text{restrict}(P,H×H))$
    - from A1 have $n \in H$ using group0_3_L1 by simp
    - with groupAssum A1 have $n \in G$ using group0_3_L2 by auto
    - with A1 ‹$n \in H› show thesis using
      - group0_3_L1 restrict_if group0_2_L7 by simp
  - qed

The neutral element of the group (denoted 1 in the group0 context) belongs to every subgroup.

**lemma (in group0) group0_3_L5:**
- assumes A1: IsAsubgroup(H,P)
- shows $1 \in H$
  - proof -
    - from A1 show $1 \in H$ using group0_3_L1 group0_3_L4
    - by fast
  - qed

Subgroups are closed with respect to the group operation.

**lemma (in group0) group0_3_L6:**
- assumes A1: IsAsubgroup(H,P)
  - and A2: a∈H b∈H
- shows a·b ∈ H
  - proof -
    - let $f = \text{restrict}(P,H×H)$
    - from A1 have monoid0(H,f) using
      - IsAsubgroup_def IsAgroup_def monoid0_def by simp
    - with A2 have $f ((a,b)) \in H$ using monoid0.group0_1_L1
      - by blast
    - with A2 show $a \cdot b \in H$ using restrict_if by simp
  - qed

A preliminary lemma that we need to show that taking the inverse in the subgroup is the same as taking the inverse in the group.

**lemma group0_3_L7A:**
- assumes A1: IsAgroup(G,f)
  - and A2: IsAsubgroup(H,f) and A3: $g = \text{restrict}(f,H×H)$

343
shows \( \text{GroupInv}(G,f) \cap H \times H = \text{GroupInv}(H,g) \)

**proof** -

let \( e = \text{TheNeutralElement}(G,f) \)

let \( e_1 = \text{TheNeutralElement}(H,g) \)

from A1 have group0(G,f) using group0_def by simp

from A2 A3 have group0(H,g)
    using IsAsubgroup_def group0_def by simp

from \( \langle \text{group0}(G,f) \rangle A2 A3 \) have GroupInv(G,f) = \( f-\{e_1\} \)
    using group0/group0_3_L4 group0/group0_2_T3 by simp

moreover have \( g-\{e_1\} = f-\{e_1\} \cap H \times H \)

**proof** -

from A1 have \( f \in G \times G \rightarrow G \)
    using IsAgroup_def IsAmonoid_def IsAssociative_def by simp

moreover from A2 \( \langle \text{group0}(G,f) \rangle \) have \( H \times H \subseteq G \times G \)
    using group0/group0_3_L2 by auto

ultimately show \( g-\{e_1\} = f-\{e_1\} \cap H \times H \)
    using A3 func1_2_L1 by simp

qed

moreover from A3 \( \langle \text{group0}(H,g) \rangle \) have GroupInv(H,g) = \( g-\{e_1\} \)
    using group0/group0_2_T3 by simp

ultimately show thesis by simp

qed

Using the lemma above we can show the actual statement: taking the inverse in the subgroup is the same as taking the inverse in the group.

**theorem** (in group0) group0_3_T1:

assumes A1: IsAsubgroup(H,P)

and A2: \( g = \text{restrict}(P,H \times H) \)

shows GroupInv(H,g) = \( \text{restrict}(\text{GroupInv}(G,P),H) \)

**proof** -

from groupAssum have GroupInv(G,P) : \( G \rightarrow G \)
    using group0/2_T2 by simp

moreover from A1 A2 have GroupInv(H,g) : \( H \rightarrow H \)
    using IsAsubgroup_def group0/2_T2 by simp

moreover from A1 have \( H \subseteq G \)
    using group0/3_L2 by simp

moreover from groupAssum A1 A2 have
    GroupInv(G,P) \( \cap \) \( H \times H = \text{GroupInv}(H,g) \)
    using group0/3_L7A by simp

ultimately show thesis
    using func1_2_L3 by simp

qed

A slightly weaker, but more convenient in applications, reformulation of the above theorem.

**theorem** (in group0) group0_3_T2:

assumes IsAsubgroup(H,P)
and \( g = \text{restrict}(P, H \times H) \)
shows \( \forall h \in H. \text{GroupInv}(H, g)(h) = h^{-1} \)
using assms group0_3_T1 restrict_if by simp

Subgroups are closed with respect to taking the group inverse.

**Theorem (in group0) group0_3_T3A:**
assumes A1: IsAsubgroup(H,P) and A2: \( h \in H \)
shows \( h^{-1} \in H \)
proof -
let \( g = \text{restrict}(P, H \times H) \)
from A1 have \( \text{GroupInv}(H, g) \in H \rightarrow H \)
using IsAsubgroup_def group0_2_T2 by simp
with A2 have \( \text{GroupInv}(H, g)(h) \in H \)
using apply_type by simp
with A1 A2 show \( h^{-1} \in H \) using group0_3_T2 by simp
qed

The next theorem states that a nonempty subset of a group \( G \) that is closed under the group operation and taking the inverse is a subgroup of the group.

**Theorem (in group0) group0_3_T3:**
assumes A1: \( H \neq 0 \)
and A2: \( H \subseteq G \)
and A3: \( H \) {is closed under} \( P \)
and A4: \( \forall x \in H. x^{-1} \in H \)
shows IsAsubgroup(H,P)
proof -
let \( g = \text{restrict}(P, H \times H) \)
let \( n = \text{TheNeutralElement}(H, g) \)
from A3 have I: \( \forall x \in H. \forall y \in H. x \cdot y \in H \)
using IsOpClosed_def by simp
from A1 obtain \( x \) where \( x \in H \) by auto
with A4 I A2 have \( 1 \in H \)
using group0_2_L6 by blast
with A3 A2 have T2: \( \text{IsAmonoid}(H, g) \)
using monoid.group0_1_T1 by simp
moreover have \( \forall h \in H. \exists b \in H. g(h, b) = n \)
proof
fix \( h \) assume \( h \in H \)
with A4 A2 have \( h \cdot h^{-1} = 1 \)
using group0_2_L6 by auto
moreover from groupAssum A2 A3 \( \langle 1 \in H \rangle \) have \( 1 = n \)
using IsAgroup_def group0_1_L6 by auto
moreover from A4 \( \langle h \in H \rangle \) have \( g(h, h^{-1}) = h \cdot h^{-1} \)
using restrict_if by simp
ultimately have \( g(h, h^{-1}) = n \) by simp
with A4 \( \langle h \in H \rangle \) show \( \exists b \in H. g(h, b) = n \) by auto
qed
ultimately show IsAsubgroup(H,P) using
IsAsubgroup_def IsAgroup_def by simp

The singleton with the neutral element is a subgroup.

corollary (in group0) unit_singl_subgr: shows IsAsubgroup({1},P)
  using group0_2_L2 group_inv_of_one group0_3_T3

Intersection of subgroups is a subgroup. This lemma is obsolete and should be replaced by subgroup_inter.

lemma group0_3_L7: assumes A1: IsAgroup(G,f) and A2: IsAsubgroup(H1,f) and A3: IsAsubgroup(H2,f) shows IsAsubgroup(H1∩H2,restrict(f,H1×H1))

Intersection of subgroups is a subgroup. This lemma is obsolete and should be replaced by subgroup_inter.

lemma subgroup_inter: assumes H≠0 and ∀H∈H. IsAsubgroup(H,P) shows IsAsubgroup(∪H,P)
fix \( H \) assume \( H: H \)
with \text{assms(2)} have \( 1: H \) using group0_3_L5 by auto
\}
then have \( \cap H \neq 0 \) using \text{assms(1)} by auto moreover
\{
fix \( t \) assume \( t: \cap H \)
then have \( \forall H \in H. \ t:H \) by auto
with \text{assms} have \( t: G \) using group0_3_L2 by blast
\}
then have \( \cap H \subseteq G \) by auto moreover
\{
fix \( x \ y \) assume \( xy: x: \cap H \) \( y: \cap H \)
\{
fix \( J \) assume \( J: J: H \)
with \( xy \) have \( x: J \) \( y: J \) by auto
with \( J \) have \( P(x, y): J \) using \text{assms(2)} group0_3_L6 by auto
\}
then have \( P(x, y): \cap H \) using \text{assms(1)} by auto
\}
then have \( \cap H \) \{is closed under\} \( P \) unfolding \text{IsOpClosed_def} by simp
moreover
\{
fix \( x \) assume \( x: x: \cap H \)
\{
fix \( J \) assume \( J: J: H \)
with \( x \) have \( x: J \) by auto
with \( J \) \text{assms(2)} have \( x^{-1} \in J \) using group0_3_T3A by auto
\}
then have \( x^{-1} \in \cap H \) using \text{assms(1)} by auto
\}
then have \( \forall x \in \cap H. \ x^{-1} \in \cap H \) by simp
ultimately show thesis using group0_3_T3 by auto
qed

The range of the subgroup operation is the whole subgroup.

\text{lemma} image_subgr_op: \text{assumes} \ A1: \text{IsAsubgroup}(H, P)
\text{shows} \ restrict(P, H \times H)(H \times H) = H
\text{proof} -
from \ A1 have \text{monoid0}(H, \text{restrict}(P, H \times H))
using \text{IsAsubgroup_def} \text{IsAgroup_def} \text{monoid0_def}
by \text{simp}
then show thesis by (rule \text{monoid0.range_carr})
qed

If we restrict the inverse to a subgroup, then the restricted inverse is onto the subgroup.

\text{lemma} (in \text{group0}) restr_inv_onto: \text{assumes} \ A1: \text{IsAsubgroup}(H, P)
\text{shows} \ restrict(\text{GroupInv}(G, P), H)(H) = H
\text{proof} -
from A1 have \( \text{GroupInv}(H, \text{restrict}(P, H \times H))(H) = H \)
using IsAsubgroup_def group0_def group0.group_inv_surj
by simp

with A1 show thesis using group0_3_T1 by simp

qed

A union of two subgroups is a subgroup iff one of the subgroups is a subset of the other subgroup.

lemma (in group0) union_subgroups:
assumes IsAsubgroup\((H_1, P)\) and IsAsubgroup\((H_2, P)\)
shows IsAsubgroup\((H_1 \cup H_2, P)\) \iff \((H_1 \subseteq H_2 \lor H_2 \subseteq H_1)\)

proof
  assume \( H_1 \subseteq H_2 \lor H_2 \subseteq H_1 \) show IsAsubgroup\((H_1 \cup H_2, P)\) by auto

next
  assume IsAsubgroup\((H_1 \cup H_2, P)\) show \( H_1 \subseteq H_2 \lor H_2 \subseteq H_1 \) proof
  { assume \( H_1 \subseteq H_2 \)
    then obtain \( x \) where \( x \in H_1 \) and \( x \notin H_2 \) by auto
    with assms(1) have \( x^{-1} \in H_1 \) using group0_3_T3A by simp
    { fix \( y \) assume \( y \in H_2 \)
      let \( z = x \cdot y \)
      from \( \langle x \in H_1 \rangle \cdot \langle y \in H_2 \rangle \) have \( x \in H_1 \cup H_2 \) and \( y \in H_1 \cup H_2 \) by auto
      with \( \langle \text{IsAsubgroup}(H_1 \cup H_2, P) \rangle \) have \( z \in H_1 \cup H_2 \) using group0_3_L6 proof
        from assms \( \langle x \in H_1 \cup H_2 \rangle \cdot \langle y \in H_2 \rangle \) have \( x \cdot y \cdot z \in G \)
        using group0_3_T3A group0_3_L2 by auto
        then have \( z \cdot y^{-1} = x \) and \( x^{-1} \cdot z = y \) using inv_cancel_two(2,3) by auto
      auto
      { assume \( z \in H_2 \)
        with \( \langle \text{IsAsubgroup}(H_2, P) \rangle \cdot \langle y^{-1} \in H_2 \rangle \) have \( z \cdot y^{-1} \in H_2 \) using group0_3_L6 by simp
      } hence \( z \notin H_2 \) by auto
      with assms(1) \( \langle x^{-1} \in H_1 \rangle \cdot \langle z \in H_1 \cup H_2 \rangle \) have \( x^{-1} \cdot z \in H_1 \) using group0_3_L6 by simp
    } hence \( H_2 \subseteq H_1 \) by blast
  } thus thesis by blast

qed

Transitivity for ”is a subgroup of” relation. The proof (probably) uses the lemma restrict_restrict from standard Isabelle/ZF library which states that \( \text{restrict}(\text{restrict}(f, A), B) = \text{restrict}(f, A \cap B) \). That lemma is added to the simplifier, so it does not have to be referenced explicitly in the proof.
below.

**Lemma subgroup_transitive:**
assumes IsAgroup(G₁,P) IsAsubgroup(G₂,P) IsAsubgroup(G₁,restrict(P,G₂×G₂))
sows IsAsubgroup(G₁,P)
proof -
from assms(2) have group0(G₂,restrict(P,G₂×G₂)) unfolding IsAsubgroup_def
group0_def by simp
with assms(3) have G₁≤G₂ using group0.group0_3_L2 by simp
hence G₂×G₂ ∩ G₁×G₁ = G₁×G₁ by auto
with assms(3) show IsAsubgroup(G₁,P) unfolding IsAsubgroup_def by simp
qed

**31.3 Groups vs. loops**

We defined groups as monoids with the inverse operation. An alternative way of defining a group is as a loop whose operation is associative.

Groups have left and right division.

**Lemma (in group0) gr_has_lr_div:** shows HasLeftDiv(G,P) and HasRightDiv(G,P)
proof -
{ fix x y assume x∈G y∈G
  then have x⁻¹·y ∈ G ∧ x·(x⁻¹·y) = y using group_op_closed inverse_in_group
  inv-cancel_two(4)
  by simp
  hence ∃z. z∈G ∧ x·z = y by auto
  moreover
  { fix z₁ z₂ assume z₁∈G ∧ x·z₁ = y and z₂∈G ∧ x·z₂ = y
    with ⟨x∈G⟩ have z₁ = z₂ using cancel_left by blast
  }
  ultimately have ∃!z. z∈G ∧ x·z = y by auto
} then show HasLeftDiv(G,P) unfolding HasLeftDiv_def by simp
{ fix x y assume x∈G y∈G
  then have y·x⁻¹ ∈ G ∧ (y·x⁻¹)·x = y using group_op_closed inverse_in_group
  inv-cancel_two(1)
  by simp
  hence ∃z. z∈G ∧ z·x = y by auto
  moreover
  { fix z₁ z₂ assume z₁∈G ∧ z₁·x = y and z₂∈G ∧ z₂·x = y
    with ⟨x∈G⟩ have z₁ = z₂ using cancel_right by blast
  }
  ultimately have ∃!z. z∈G ∧ z·x = y by auto
} then show HasRightDiv(G,P) unfolding HasRightDiv_def by simp
qed

A group is a quasigroup and a loop.

**Lemma (in group0) group_is_loop:** shows IsAquasigroup(G,P) and IsAloop(G,P)
proof -
  show IsAquasigroup(G,P) unfolding IsAquasigroup_def HasLatinSquareProp_def
An associative loop is a group.

**Theorem assoc_loop_is_gr:** assumes IsAloop(G,P) and P {is associative on} G shows IsAgroup(G,P)

**Proof** -

from assms(1) have \( \exists e \in G. \forall x \in G. P(e, x) = x \land P(x, e) = x \)
unfolding IsAloop_def by simp

with assms(2) have IsAmonoid(G,P) unfolding IsAmonoid_def by simp

{ fix x assume \( x \in G \)
  let \( y = \text{RightInv}(G, P)(x) \)
  from assms(1) \(<x \in G> \) have \( y \in G \) and \( P(x, y) = \text{TheNeutralElement}(G, P) \)
  using loop_loop0_valid loop0.lr_inv_props(3, 4) by auto
  hence \( \exists y \in G. P(x, y) = \text{TheNeutralElement}(G, P) \) by auto
}

with \(<\text{IsAmonoid}(G,P)>\) show IsAgroup(G,P) unfolding IsAgroup_def by simp

qed

For groups the left and right inverse are the same as the group inverse.

**Lemma (in group0) lr_inv_gr_inv:**

shows LeftInv(G,P) = GroupInv(G,P) and RightInv(G,P) = GroupInv(G,P)

**Proof** -

have LeftInv(G,P):G\rightarrow G using group_is_loop loop_loop0_valid loop0.lr_inv_fun(1)
by simp

moreover from groupAssum have GroupInv(G,P):G\rightarrow G using group0_2_T2
by simp

moreover

{ fix x assume \( x \in G \)
  let \( y = \text{LeftInv}(G, P)(x) \)
  from \(<x \in G>\) have \( y \in G \) and \( y \cdot x = 1 \)
  using group_is_loop(2) loop_loop0_valid loop0.lr_inv_props(1, 2)
  by auto
  with \(<x \in G>\) have \( \text{LeftInv}(G, P)(x) = \text{GroupInv}(G, P)(x) \) using group0_2_L9(1)
  by simp
}

ultimately show LeftInv(G,P) = GroupInv(G,P) using func_eq by blast

have RightInv(G,P):G\rightarrow G using group_is_loop loop_loop0_valid loop0.lr_inv_fun(2)
by simp

moreover from groupAssum have GroupInv(G,P):G\rightarrow G using group0_2_T2
by simp

moreover

{ fix x assume \( x \in G \)
  let \( y = \text{RightInv}(G, P)(x) \)
  from \(<x \in G>\) have \( y \in G \) and \( x \cdot y = 1 \)
  using group_is_loop(2) loop_loop0_valid loop0.lr_inv_props(3, 4)
  by auto
}

350
In this theory we consider right and left translations and odd functions.

32.1 Translations

In this section we consider translations. Translations are maps \( T : G \rightarrow G \) of the form \( T_g(a) = g \cdot a \) or \( T_g(a) = a \cdot g \). We also consider two-dimensional translations \( T_g : G \times G \rightarrow G \times G \), where \( T_g(a, b) = (a \cdot g, b \cdot g) \) or \( T_g(a, b) = (g \cdot a, g \cdot b) \).

For an element \( a \in G \) the right translation is defined a function (set of pairs) such that its value (the second element of a pair) is the value of the group operation on the first element of the pair and \( g \). This looks a bit strange in the raw set notation, when we write a function explicitly as a set of pairs and value of the group operation on the pair \( \langle a, b \rangle \) as \( P(a, b) \) instead of the usual infix \( a \cdot b \) or \( a + b \).

**definition**

\[
\text{RightTranslation}(G, P, g) \equiv \{(a, b) \in G \times G. \ P(a, g) = b}\]

A similar definition of the left translation.

**definition**

\[
\text{LeftTranslation}(G, P, g) \equiv \{(a, b) \in G \times G. \ P(g, a) = b}\]

Translations map \( G \) into \( G \). Two dimensional translations map \( G \times G \) into itself.

**lemma** (in group0) group0_5_L1: assumes A1: \( g \in G \) shows \( \text{RightTranslation}(G, P, g) : G \rightarrow G \) and \( \text{LeftTranslation}(G, P, g) : G \rightarrow G \)

**proof** -
from A1 have \( \forall a \in G. \ a \cdot g \in G \) and \( \forall a \in G. \ g \cdot a \in G \)
using group_oper_fun apply_funtype by auto
then show
RightTranslation(G,P,g) : G\to G
LeftTranslation(G,P,g) : G\to G
using RightTranslation_def LeftTranslation_def func1_1_L11A by auto
qed

The values of the translations are what we expect.

lemma (in group0) group0_5_L2: assumes g\in G a\in G
shows RightTranslation(G,P,g)(a) = a\cdot g
LeftTranslation(G,P,g)(a) = g\cdot a
using assms group0_5_L1 RightTranslation_def LeftTranslation_def
func1_1_L11B by auto

Composition of left translations is a left translation by the product.

lemma (in group0) group0_5_L4: assumes A1: g\in G h\in G a\in G and
A2: T_g = LeftTranslation(G,P,g) T_h = LeftTranslation(G,P,h)
shows T_g(T_h(a)) = g\cdot h\cdot a
T_g(T_h(a)) = LeftTranslation(G,P,g\cdot h)(a)
proof -
from A1 have I: h\cdot a \in G g\cdot h \in G
using group_oper_fun apply_funtype by auto
with A1 A2 show T_g(T_h(a)) = g\cdot h\cdot a
using group0_5_L2 group_oper_assoc by simp
with A1 A2 I show T_g(T_h(a)) = LeftTranslation(G,P,g\cdot h)(a)
using group0_5_L2 group_oper_assoc by simp
qed

Composition of right translations is a right translation by the product.

lemma (in group0) group0_5_L5: assumes A1: g\in G h\in G a\in G and
A2: T_g = RightTranslation(G,P,g) T_h = RightTranslation(G,P,h)
shows T_g(T_h(a)) = a\cdot h\cdot g
T_g(T_h(a)) = RightTranslation(G,P,h\cdot g)(a)
proof -
from A1 have I: a\cdot h\cdot g \in G
using group_oper_fun apply_funtype by auto
with A1 A2 show T_g(T_h(a)) = a\cdot h\cdot g
using group0_5_L2 group_oper_assoc by simp
with A1 A2 I show T_g(T_h(a)) = RightTranslation(G,P,h\cdot g)(a)
using group0_5_L2 group_oper_assoc by simp
qed

Point free version of group0_5_L4 and group0_5_L5.
lemma (in group0) trans_comp: assumes g∈G h∈G shows
  \( \text{RightTranslation}(G, P, g) \circ \text{RightTranslation}(G, P, h) = \text{RightTranslation}(G, P, h \cdot g) \)
  \( \text{LeftTranslation}(G, P, g) \circ \text{LeftTranslation}(G, P, h) = \text{LeftTranslation}(G, P, g \cdot h) \)

proof -
  let \( T_g = \text{RightTranslation}(G, P, g) \)
  let \( T_h = \text{RightTranslation}(G, P, h) \)
  from assms have \( T_g : G \to G \) and \( T_h : G \to G \)
    using group0_5_L1 by auto
  then have \( T_g \circ T_h : G \to G \)
    using comp_fun by simp
  moreover from assms have \( \text{RightTranslation}(G, P, h \cdot g) : G \to G \)
    using group_op_closed group0_5_L1 by simp
  moreover from assms \( T_h : G \to G \) have
    \( \forall a \in G. (T_g \circ T_h)(a) = \text{RightTranslation}(G, P, h \cdot g)(a) \)
    using comp_fun_apply group0_5_L5 by simp
  ultimately show \( T_g \circ T_h = \text{RightTranslation}(G, P, h \cdot g) \)
    by (rule func_eq)
next
  let \( T_g = \text{LeftTranslation}(G, P, g) \)
  let \( T_h = \text{LeftTranslation}(G, P, h) \)
  from assms have \( T_g : G \to G \) and \( T_h : G \to G \)
    using group0_5_L1 by auto
  then have \( T_g \circ T_h : G \to G \)
    using comp_fun by simp
  moreover from assms have \( \text{LeftTranslation}(G, P, g \cdot h) : G \to G \)
    using group_op_closed group0_5_L1 by simp
  moreover from assms \( T_h : G \to G \) have
    \( \forall a \in G. (T_g \circ T_h)(a) = \text{LeftTranslation}(G, P, g \cdot h)(a) \)
    using comp_fun_apply group0_5_L4 by simp
  ultimately show \( T_g \circ T_h = \text{LeftTranslation}(G, P, g \cdot h) \)
    by (rule func_eq)
qed

The image of a set under a composition of translations is the same as the image under translation by a product.

lemma (in group0) trans_comp_image: assumes A1: g∈G h∈G and
  A2: \( T_g = \text{LeftTranslation}(G, P, g) \) \( T_h = \text{LeftTranslation}(G, P, h) \)
shows \( T_g(T_h(A)) = \{a \cdot h \cdot g. a \in A\} \)

proof -
  from A2 have \( T_g(T_h(A)) = (T_g \circ T_h)(A) \)
    using image_comp by simp
  with assms show thesis using trans_comp by simp
qed

Another form of the image of a set under a composition of translations

lemma (in group0) group0_5_L6:
  assumes A1: g∈G h∈G and A2: A⊆G and
  A3: \( T_g = \text{RightTranslation}(G, P, g) \) \( T_h = \text{RightTranslation}(G, P, h) \)
shows \( T_g(T_h(A)) = \{a \cdot h \cdot g. a \in A\} \)

proof -
  from A2 have \( \forall a \in A. a \in G \) by auto
from A1 A3 have \(T_g : G \to G\) \(T_h : G \to G\)
using group0_5_L1 by auto
with asms \(\forall a \in A. a \in G\) show
\(T_g(T_h(A)) = \{a \cdot h \cdot g. a \in A\}\)
using func1_1_L15C group0_5_L5 by auto
qed

The translation by neutral element is the identity on group.

**Lemma (in group0) trans_neutral:** shows
RightTranslation(G,P,1) = id(G) and LeftTranslation(G,P,1) = id(G)
proof -
have RightTranslation(G,P,1):G \to G and \(\forall a \in G.\) RightTranslation(G,P,1)(a) = a
using group0_2_L2 group0_5_L1 group0_5_L2 by auto
then show RightTranslation(G,P,1) = id(G) by (rule indentity_fun)
have LeftTranslation(G,P,1):G \to G and \(\forall a \in G.\) LeftTranslation(G,P,1)(a) = a
using group0_2_L2 group0_5_L1 group0_5_L2 by auto
then show LeftTranslation(G,P,1) = id(G) by (rule indentity_fun)
qed

Translation by neutral element does not move sets.

**Lemma (in group0) trans_neutral_image:** assumes \(V \subseteq G\)
shows RightTranslation(G,P,1)(V) = V and LeftTranslation(G,P,1)(V) = V
using asms trans_neutral image_id_same by auto

Composition of translations by an element and its inverse is identity.

**Lemma (in group0) trans_comp_id:** assumes \(g \in G\) shows
RightTranslation(G,P,g) \O RightTranslation(G,P,g^{-1}) = id(G) and
RightTranslation(G,P,g^{-1}) \O RightTranslation(G,P,g) = id(G) and
LeftTranslation(G,P,g) \O LeftTranslation(G,P,g^{-1}) = id(G) and
LeftTranslation(G,P,g^{-1}) \O LeftTranslation(G,P,g) = id(G)
using asms inverse_in_group trans_comp group0_2_L6 trans_neutral by auto

Translations are bijective.

**Lemma (in group0) trans_bij:** assumes \(g \in G\) shows
RightTranslation(G,P,g) \in bij(G,G) and LeftTranslation(G,P,g) \in bij(G,G)
proof-
from asms have
RightTranslation(G,P,g):G \to G and
RightTranslation(G,P,g^{-1}):G \to G and
RightTranslation(G,P,g) \O RightTranslation(G,P,g^{-1}) = id(G)
RightTranslation(G,P,g^{-1}) \O RightTranslation(G,P,g) = id(G)
using inverse_in_group group0_5_L1 trans_comp_id by auto
then show RightTranslation(G,P,g) \in bij(G,G) using fg_imp_bijection
by simp

354
from asms have
  LeftTranslation(G,P,g) : G → G and
  LeftTranslation(G,P,g⁻¹) : G → G and
  LeftTranslation(G,P,g) O LeftTranslation(G,P,g⁻¹) = id(G)
  LeftTranslation(G,P,g⁻¹) O LeftTranslation(G,P,g) = id(G)
  using inverse_in_group group0_5_L1 trans_comp_id by auto
  then show LeftTranslation(G,P,g) ∈ bij(G,G) using fg_imp_bijective
  by simp
qed

Converse of a translation is translation by the inverse.

lemma (in group0) trans_conv_inv: assumes g∈G shows
  converse(RightTranslation(G,P,g)) = RightTranslation(G,P,g⁻¹) and
  converse(LeftTranslation(G,P,g)) = LeftTranslation(G,P,g⁻¹) and
  LeftTranslation(G,P,g) = converse(RightTranslation(G,P,g⁻¹)) and
  RightTranslation(G,P,g) = converse(RightTranslation(G,P,g⁻¹))
proof -
  from asms have
    RightTranslation(G,P,g) ∈ bij(G,G) RightTranslation(G,P,g⁻¹) ∈ bij(G,G)
    and
    LeftTranslation(G,P,g) ∈ bij(G,G) LeftTranslation(G,P,g⁻¹) ∈ bij(G,G)
    using trans_bij inverse_in_group by auto
  moreover from asms have
    RightTranslation(G,P,g⁻¹) O RightTranslation(G,P,g) = id(G) and
    LeftTranslation(G,P,g⁻¹) O LeftTranslation(G,P,g) = id(G) and
    LeftTranslation(G,P,g⁻¹) O LeftTranslation(G,P,g) = id(G) and
    RightTranslation(G,P,g⁻¹) O LeftTranslation(G,P,g) = id(G)
    using trans_comp_id by auto
  ultimately show
    converse(RightTranslation(G,P,g)) = RightTranslation(G,P,g⁻¹) and
    converse(LeftTranslation(G,P,g)) = LeftTranslation(G,P,g⁻¹) and
    LeftTranslation(G,P,g) = converse(RightTranslation(G,P,g⁻¹)) and
    RightTranslation(G,P,g) = converse(RightTranslation(G,P,g⁻¹))
    using comp_id_conv by auto
qed

The image of a set by translation is the same as the inverse image by by the
inverse element translation.

lemma (in group0) trans_image_vimage: assumes g∈G shows
  LeftTranslation(G,P,g)(A) = LeftTranslation(G,P,g⁻¹)(A) and
  RightTranslation(G,P,g)(A) = RightTranslation(G,P,g⁻¹)(A)
  using asms trans_conv_inv vimage_converse by auto

Another way of looking at translations is that they are sections of the group
operation.

lemma (in group0) trans_eq_section: assumes g∈G shows
  RightTranslation(G,P,g) = Fix2ndVar(P,g) and
  LeftTranslation(G,P,g) = Fix1stVar(P,g)
proof -
let T = RightTranslation(G,P,g)
let F = Fix2ndVar(P,g)
from assms have T: G → G and F: G → G
  using group0_5_L1 group_oper_fun fix_2nd_var_fun by auto
moreover from assms have ∀a∈G. T(a) = F(a)
  using group0_5_L2 group_oper_fun fix_var_val by simp
ultimately show T = F by (rule func_eq)
next
let T = LeftTranslation(G,P,g)
let F = Fix1stVar(P,g)
from assms have T: G → G and F: G → G
  using group0_5_L1 group_oper_fun fix_1st_var_fun by auto
moreover from assms have ∀a∈G. T(a) = F(a)
  using group0_5_L2 group_oper_fun fix_var_val by auto
ultimately show T = F by (rule func_eq)
qed

A lemma demonstrating what is the left translation of a set

lemma (in group0) ltrans_image: assumes A1: V ⊆ G and A2: x∈G
  shows LeftTranslation(G,P,x)(V) = {x·v. v∈V}
proof -
  from assms have LeftTranslation(G,P,x)(V) = {LeftTranslation(G,P,x)(v). v∈V}
    using group0_5_L1 func_imagedef by blast
  moreover from assms have ∀v∈V. LeftTranslation(G,P,x)(v) = x·v
    using group0_5_L2 by auto
  ultimately show thesis by auto
qed

A lemma demonstrating what is the right translation of a set

lemma (in group0) rtrans_image: assumes A1: V ⊆ G and A2: x∈G
  shows RightTranslation(G,P,x)(V) = {v·x. v∈V}
proof -
  from assms have RightTranslation(G,P,x)(V) = {RightTranslation(G,P,x)(v). v∈V}
    using group0_5_L1 func_imagedef by blast
  moreover from assms have ∀v∈V. RightTranslation(G,P,x)(v) = v·x
    using group0_5_L2 by auto
  ultimately show thesis by auto
qed

Right and left translations of a set are subsets of the group. Interestingly, we do not have to assume the set is a subset of the group.

lemma (in group0) lrtrans_in_group: assumes x∈G
  shows LeftTranslation(G,P,x)(V) ⊆ G and RightTranslation(G,P,x)(V) ⊆ G
proof -
  from assms have LeftTranslation(G,P,x):G→G and RightTranslation(G,P,x):G→G

356
using group0_5_L1 by auto
then show \( \text{LeftTranslation}(G,P,x)(V) \subseteq G \) and \( \text{RightTranslation}(G,P,x)(V) \subseteq G \)
using func1_1_L6 by auto
qed

A technical lemma about solving equations with translations.

lemma (in group0) ltrans_inv_in: assumes A1: \( V \subseteq G \) and A2: \( y \in G \) and A3: \( x \in \text{LeftTranslation}(G,P,y)(\text{GroupInv}(G,P)(V)) \)
shows \( y \in \text{LeftTranslation}(G,P,x)(V) \)
proof -
have \( x \in G \)
proof -
  from A2 have \( \text{LeftTranslation}(G,P,y):G \rightarrow G \)
  using group0_5_L1 by simp
then have \( \text{LeftTranslation}(G,P,y)(\text{GroupInv}(G,P)(V)) \subseteq G \)
using func1_1_L6 by simp
with A3 show \( x \in G \) by auto
qed
have \( \exists v \in V. x = y \cdot v^{-1} \)
proof -
  have \( \text{GroupInv}(G,P): G \rightarrow G \)
  using groupAssum group0_2_T2 by simp
with assms obtain \( z \) where \( z \in \text{GroupInv}(G,P)(V) \) and \( x = y \cdot z \)
using func1_1_L6 ltrans_image by auto
with A1 \( \langle \text{GroupInv}(G,P): G \rightarrow G \rangle \) show thesis using func_imagedef by auto
qed
then obtain \( v \) where \( v \in V \) and \( x = y \cdot v^{-1} \) by auto
with A1 A2 have \( y = x \cdot v \) using inv_cancel_two by auto
with assms \( \langle x \in G \rangle \langle v \in V \rangle \) show thesis using ltrans_image by auto
qed

We can look at the result of interval arithmetic operation as union of left translated sets.

lemma (in group0) image_ltrans_union: assumes \( A \subseteq G \) \( B \subseteq G \) shows
\( (\langle P \{\text{lifted to subsets of} G\} \rangle(A,B) = (\bigcup_{a \in A.} \text{LeftTranslation}(G,P,a)(B)) \)
proof
from assms have I: \( (\langle P \{\text{lifted to subsets of} G\} \rangle(A,B) = \{a \cdot b. \langle a,b \rangle \in A \times B\} \)
using group_oper_fun lift_subsets_explained by simp
\{ fix c assume c \in \( (\langle P \{\text{lifted to subsets of} G\} \rangle(A,B) \)
with I obtain \( a \) \( b \) where \( c = a \cdot b \) and \( a \in A \) \( b \in B \) by auto
hence \( c \in \{a \cdot b. \ b \in B\} \) by auto
moreover from assms \( \langle a \in A \rangle \) have \( \text{LeftTranslation}(G,P,a)(B) = \{a \cdot b. \ b \in B\} \) using ltrans_image by auto
ultimately have \( c \in \text{LeftTranslation}(G,P,a)(B) \) by simp
with \( \langle a \in A \rangle \) have \( c \in (\bigcup_{a \in A.} \text{LeftTranslation}(G,P,a)(B)) \) by auto
\} thus \( (\langle P \{\text{lifted to subsets of} G\} \rangle(A,B) \subseteq (\bigcup_{a \in A.} \text{LeftTranslation}(G,P,a)(B)) \) by auto

357
\{ \text{fix } c \text{ assume } c \in \left( \bigcup_{a \in A} \text{ LeftTranslation}(G,P,a)(B) \right) \\
\text{then obtain } a \text{ where } a \in A \text{ and } c \in \text{ LeftTranslation}(G,P,a)(B) \}

by auto

moreover from \text{assms } \langle a \in A \rangle \text{ have } \text{ LeftTranslation}(G,P,a)(B) = \{a \cdot b. b \in B\}

using \text{ltrans_image} \text{ by auto}

ultimately obtain \ b \text{ where } b \in B \text{ and } c = a \cdot b \text{ by auto}

\}

\text{thus } \left( \bigcup_{a \in A} \text{ LeftTranslation}(G,P,a)(B) \right) \subseteq \left( \bigcup_{b \in B} \text{ RightTranslation}(G,P,b)(A) \right)

by auto

\}

\text{thus } \left( \bigcup_{a \in A} \text{ LeftTranslation}(G,P,a)(B) \right) \subseteq \left( \bigcup_{b \in B} \text{ RightTranslation}(G,P,b)(A) \right)

by auto

\}

\text{then obtain } b \text{ where } b \in B \text{ and } c \in \text{ RightTranslation}(G,P,b)(A)

by auto

moreover from \text{assms } \langle b \in B \rangle \text{ have } \text{ RightTranslation}(G,P,b)(A) = \{a \cdot b. a \in A\}

using \text{rtrans_image} \text{ by auto}

ultimately obtain \ a \text{ where } a \in A \text{ and } c = a \cdot b \text{ by auto}

\}

\text{thus } \left( \bigcup_{b \in B} \text{ RightTranslation}(G,P,b)(A) \right) \subseteq \left( \bigcup_{a \in A} \text{ LeftTranslation}(G,P,a)(B) \right)

by auto

qed

The right translation version of image_ltrans_union The proof follows the same schema.

\text{lemma (in group0) image_rtrans_union: assumes } A \subseteq G, B \subseteq G \text{ shows}

\left( \bigcup_{b \in B} \text{ RightTranslation}(G,P,b)(A) \right) = \left( \bigcup_{a \in A} \text{ LeftTranslation}(G,P,a)(B) \right)

proof

\text{from assms have I: } \left( \bigcup_{b \in B} \text{ RightTranslation}(G,P,b)(A) \right) = \{a \cdot b. \langle a,b \rangle \in A \times B\}

using \text{group_oper_fun lift_subsets_explained} \text{ by simp}

\{ \text{fix } c \text{ assume } c \in \left( \bigcup_{b \in B} \text{ RightTranslation}(G,P,b)(A) \right) \}

\text{with I obtain } a \text{ where } c = a \cdot b \text{ and } a \in A, b \in B \text{ by auto}

hence } c \in \{a \cdot b. a \in A\} \text{ by auto}

moreover from \text{assms } \langle b \in B \rangle \text{ have}

\text{RightTranslation}(G,P,b)(A) = \{a \cdot b. a \in A\} \text{ using rtrans_image} \text{ by auto}

ultimately have } c \in \text{ RightTranslation}(G,P,b)(A) \text{ by simp}

with } \langle b \in B \rangle \text{ have } c \in \left( \bigcup_{b \in B} \text{ RightTranslation}(G,P,b)(A) \right) \text{ by auto}

\}

\text{thus } \left( \bigcup_{b \in B} \text{ RightTranslation}(G,P,b)(A) \right) \subseteq \left( \bigcup_{a \in A} \text{ LeftTranslation}(G,P,a)(B) \right)

by auto

\}

\text{then obtain } b \text{ where } b \in B \text{ and } c \in \text{ RightTranslation}(G,P,b)(A)

by auto

moreover from \text{assms } \langle b \in B \rangle \text{ have } \text{ RightTranslation}(G,P,b)(A) = \{a \cdot b. a \in A\}

using \text{rtrans_image} \text{ by auto}

ultimately obtain } a \text{ where } a \in A \text{ and } c = a \cdot b \text{ by auto}

\}

\text{thus } \left( \bigcup_{b \in B} \text{ RightTranslation}(G,P,b)(A) \right) \subseteq \left( \bigcup_{a \in A} \text{ LeftTranslation}(G,P,a)(B) \right)

by auto

qed

If the neutral element belongs to a set, then an element of group belongs the translation of that set.

\text{lemma (in group0) neut_trans_elem:}

\text{assumes } A1: A \subseteq G, g \in G \text{ and } A2: 1 \in A

\text{shows } g \in \text{ LeftTranslation}(G,P,g)(A) \\text{ and } g \in \text{ RightTranslation}(G,P,g)(A)

proof -

358
from assms have \( g \cdot 1 \in \text{LeftTranslation}(G, P, g)(A) \)
using \ltrans_image\ by auto
with \( A_1 \) show \( g \in \text{LeftTranslation}(G, P, g)(A) \)
using \group0_2_L2\ by simp
from assms have \( 1 \cdot g \in \text{RightTranslation}(G, P, g)(A) \)
using \rtrans_image\ by auto
with \( A_1 \) show \( g \in \text{RightTranslation}(G, P, g)(A) \)
using \group0_2_L2\ by simp
qed

The neutral element belongs to the translation of a set by the inverse of an element that belongs to it.

**lemma (in group0) elem_trans_neut:**

**assumes**

\( A_1: A \subseteq G \) and \( A_2: g \in A \)

**shows**

\( 1 \in \text{LeftTranslation}(G, P, g^{-1})(A) \)
\( 1 \in \text{RightTranslation}(G, P, g^{-1})(A) \)

**proof**

from assms have \( g^{-1} \in G \) using \inverse_in_group\ by auto
with assms have \( g^{-1} \cdot g \in \text{LeftTranslation}(G, P, g^{-1})(A) \)
using \ltrans_image\ by auto
moreover from assms have \( g^{-1} \cdot g = 1 \) using \group0_2_L6\ by auto
ultimately show \( 1 \in \text{LeftTranslation}(G, P, g^{-1})(A) \)
by simp
from \( g^{-1} \) assms have \( g \cdot g^{-1} \in \text{RightTranslation}(G, P, g^{-1})(A) \)
using \rtrans_image\ by auto
moreover from assms have \( g \cdot g^{-1} = 1 \) using \group0_2_L6\ by auto
ultimately show \( 1 \in \text{RightTranslation}(G, P, g^{-1})(A) \)
by simp
qed

### 32.2 Odd functions

This section is about odd functions.

Odd functions are those that commute with the group inverse: \( f(a^{-1}) = (f(a))^{-1} \).

**definition**

\[
\text{IsOdd}(G, P, f) \equiv (\forall a \in G. f(\text{GroupInv}(G, P)(a)) = \text{GroupInv}(G, P)(f(a)))
\]

Let’s see the definition of an odd function in a more readable notation.

**lemma (in group0) group0_6_L1:**

**shows**

\( \text{IsOdd}(G, P, p) \leftrightarrow (\forall a \in G. p(a^{-1}) = (p(a))^{-1}) \)
using \IsOdd_def\ by simp

We can express the definition of an odd function in two ways.

**lemma (in group0) group0_6_L2:**

**assumes**

\( A_1: p : G \rightarrow G \)

**shows**

\( (\forall a \in G. p(a^{-1}) = (p(a))^{-1}) \leftrightarrow (\forall a \in G. (p(a^{-1}))^{-1} = p(a)) \)

**proof**

assume \( \forall a \in G. p(a^{-1}) = (p(a))^{-1} \)
with \( A_1 \) show \( \forall a \in G. (p(a^{-1}))^{-1} = p(a) \)
using \apply_funtype group_inv_of_inv\ by simp
next assume \( A_2: \forall a \in G. (p(a^{-1}))^{-1} = p(a) \)
{ fix a assume \( a \in G \) with \( A1 \) \( A2 \) have
\[
 p(a) \in G \text{ and } ((p(a))^{-1})^{-1} = (p(a))^{-1}
\]
using apply_funtype inverse_in_group by auto
then have \( p(a) = (p(a))^{-1} \)
using group_inv_of_inv by simp
} then show \( \forall a \in G. p(a^{-1}) = (p(a))^{-1} \) by simp
qed

32.3 Subgroups and interval arithmetic

The section Binary operations in the func_ZF theory defines the notion of 
"lifting operation to subsets". In short, every binary operation \( f : X \times X \rightarrow X \) on a set \( X \) defines an operation on the subsets of \( X \) defined by
\[
 F(A,B) = \{ f(x,y) | x \in A, y \in B \}.
\]
In the group context using multiplicative notation we can write this as \( H \cdot K = \{ x \cdot y | x \in A, y \in B \} \). Similarly we 
can define \( H^{-1} = \{ x^{-1} | x \in H \} \). In this section we study properties of these 
derived operations and how they relate to the concept of subgroups.

The next locale extends the groups0 locale with notation related to interval 
arithmetic.

locale group4 = group0 +
fixes sdot (infixl · 70)
defines sdot_def [simp]: \( A \cdot B \equiv (P \text{ lifted to subsets of } G)\langle A,B \rangle \)

fixes sinv (_−1 [90] 91)
defines sinv_def[simp]: \( A^{-1} \equiv \text{ GroupInv}(G,P)(A) \)

The next lemma shows a somewhat more explicit way of defining the product 
of two subsets of a group.

lemma (in group4) interval_prod: assumes \( A \subseteq G \) \( B \subseteq G \)
shows \( A \cdot B = \{ x \cdot y. \langle x,y \rangle \in A \times B \} \)
using assms group_oper_fun lift_subsets_explained by auto

Product of elements of subsets of the group is in the set product of those 
subsets

lemma (in group4) interval_prod_el: assumes \( A \subseteq G \) \( B \subseteq G \) \( x \in A \) \( y \in B \)
shows \( x \cdot y \in A \cdot B \)
using assms interval_prod by auto

An alternative definition of a group inverse of a set.

lemma (in group4) interval_inv: assumes \( A \subseteq G \)
shows \( A^{-1} = \{ x^{-1} | x \in A \} \)
proof -
from groupAssum have \( \text{ GroupInv}(G,P) : G \rightarrow G \) using group0_2_T2 by simp
with assms show \( A^{-1} = \{ x^{-1} | x \in A \} \) using func_imagedef by simp
qed
Group inverse of a set is a subset of the group. Interestingly we don’t need to assume the set is a subset of the group.

**Lemma (in group4) interval_inv_cl**: shows \( A^{-1} \subseteq G \)

**Proof** -
- from groupAssum have GroupInv(G,P):G→G using group0_2_T2 by simp
- then show \( A^{-1} \subseteq G \) using func1_1_L6(2) by simp

**Qed**

The product of two subsets of a group is a subset of the group.

**Lemma (in group4) interval_prod_closed**: assumes \( A \subseteq G \) \( B \subseteq G \)
shows \( A \cdot B \subseteq G \)

**Proof**
- fix \( z \) assume \( z \in A \cdot B \)
  - with assms obtain \( x \) \( y \) where \( x \in A \) \( y \in B \) \( z = x \cdot y \) using interval_prod by auto
  - with assms show \( z \in G \) using group_op_closed by auto

**Qed**

The product of sets operation is associative.

**Lemma (in group4) interval_prod_assoc**: assumes \( A \subseteq G \) \( B \subseteq G \) \( C \subseteq G \)
shows \( A \cdot B \cdot C = A \cdot (B \cdot C) \)

**Proof** -
- from groupAssum have (P {lifted to subsets of} G) {is associative on} Pow(G)
  - unfolding IsAgroup_def IsAmonoid_def using lift_subset_assoc by simp
  - with assms show thesis unfolding IsAssociative_def by auto

**Qed**

A simple rearrangement following from associativity of the product of sets operation.

**Lemma (in group4) interval_prod_rearr1**: assumes \( A \subseteq G \) \( B \subseteq G \) \( C \subseteq G \) \( D \subseteq G \)
shows \( A \cdot B \cdot (C \cdot D) = A \cdot (B \cdot C) \cdot D \)

**Proof** -
- from assms(1,2) have \( A \cdot B \subseteq G \) using interval_prod_closed by simp
  - with assms(3,4) have \( A \cdot B \cdot (C \cdot D) = A \cdot B \cdot C \cdot D \)
  - using interval_prod_assoc by simp
  - also from assms(1,2,3) have \( A \cdot B \cdot C \cdot D = A \cdot (B \cdot C) \cdot D \)
  - using interval_prod_assoc by simp
  - finally show thesis by simp

**Qed**

A subset \( A \) of the group is closed with respect to the group operation iff \( A \cdot A \subseteq A \).

**Lemma (in group4) subset_gr_op_cl**: assumes \( A \subseteq G \)
shows \( (A \ {\text{is closed under}} \ P) \iff A \cdot A \subseteq A \)

**Proof**
- assume \( A \ {\text{is closed under}} \ P \)
  - { fix \( z \) assume \( z \in A \cdot A \)
with assms obtain \( x \) \( y \) where \( x \in A \) \( y \in A \) and \( z = x \cdot y \) using interval_prod by auto
with \( \langle A \text{ is closed under} \rangle \ P \) have \( z \in A \) unfolding IsOpClosed_def by simp
} thus \( A \cdot A \subseteq A \) by auto
next
assume \( A \cdot A \subseteq A \)
{ fix \( x \) \( y \)
assume \( x \in A \) \( y \in A \)
with assms have \( x \cdot y \in A \cdot A \) using interval_prod by auto
with \( \langle A \cdot A \subseteq A \rangle \) have \( x \cdot y \in A \) by auto
} then show \( A \text{ is closed under} \ P \) unfolding IsOpClosed_def by simp qed

Inverse and square of a subgroup is this subgroup.

lemma (in group4) subgroup_inv_sq: assumes IsAsubgroup\( (H,P) \)
shows \( H^{-1} = H \) and \( H \cdot H = H \)
proof
from assms have \( H \subseteq G \) using group0_3_L2 by simp
with assms show \( H^{-1} \subseteq H \) using interval_inv group0_3_T3A by auto
{ fix \( x \)
assume \( x \in H \)
with assms have \( (x^{-1})^{-1} \in \{y^{-1} . y \in H\} \) using group0_3_T3A by auto
moreover from \( \langle x \in H \rangle \) \( \langle H \subseteq G \rangle \) have \( (x^{-1})^{-1} = x \) using group_inv_of_inv
by auto
ultimately have \( x \in \{y^{-1} . y \in H\} \) by auto
with \( \langle H \subseteq G \rangle \) have \( x \in H^{-1} \) using interval_inv by simp
} thus \( H \subseteq H^{-1} \) by auto
from assms have \( H \subseteq G \) unfolding IsOpClosed_def by simp
with assms have \( H \cdot H \subseteq H \) using subset_gr_op_cl group0_3_L2 by simp
moreover
{ fix \( x \)
assume \( x \in H \)
with assms have \( x \in G \) using group0_3_L2 by auto
from assms \( \langle H \subseteq G \rangle \) \( \langle x \in H \rangle \) have \( x \cdot 1 \in H \cdot H \) using group0_3_L5 interval_prod by auto
with \( \langle x \in G \rangle \) have \( x \in H \cdot H \) using group0_2_L2 by simp
} hence \( H \subseteq H \cdot H \) by auto
ultimately show \( H \cdot H = H \) by auto
qed

Inverse of a product two sets is a product of inverses with the reversed order.

lemma (in group4) interval_prod_inv: assumes \( A \subseteq G \) \( B \subseteq G \)
saves \( A \cdot B)^{-1} = \{(y^{-1}.x^{-1} . (x,y) \in A \times B)\}
\( (A \cdot B)^{-1} = \{y^{-1} . x^{-1} . (x,y) \in A \times B\} \)
\( (A \cdot B)^{-1} = (B^{-1}) . (A^{-1}) \)
proof
from assms have \( (A \cdot B) \subseteq G \) using interval_prod_closed by simp
then have I: \( (A \cdot B)^{-1} = \{z^{-1} . z \in A \cdot B\} \) using interval_inv by simp
show II: \( (A \cdot B)^{-1} = \{(x \cdot y)^{-1} . (x,y) \in A \times B\} \)

362
proof
{ fix p assume p ∈ (A·B)^{-1} 
  with I obtain z where p·z = 1 and z ∈ A·B by auto 
  with assms obtain x y where ⟨x, y⟩ ∈ A·B and z = x·y using interval_prod 
  by auto 
  with p·z = 1 have p∈{(x·y)^{-1}.⟨x, y⟩ ∈ A·B} by auto 
} thus (A·B)^{-1} ⊆ {(x·y)^{-1}.⟨x, y⟩ ∈ A·B} by blast
{ fix p assume p∈{(x·y)^{-1}.⟨x, y⟩ ∈ A·B} 
  then obtain x y where x ∈ A and y ∈ B and p·x = (x·y)^{-1} by auto 
  with assms (A·B) ⊆ G' have p∈(A·B)^{-1} using interval_prod interval_inv 
  by auto 
} thus {(x·y)^{-1}.⟨x, y⟩ ∈ A·B} ⊆ (A·B)^{-1} by blast
qed

have {(x·y)^{-1}.⟨x, y⟩ ∈ A·B} = {y^{-1}·x^{-1}.⟨x, y⟩ ∈ A·B} 
proof
{ fix p assume p∈{(x·y)^{-1}.⟨x, y⟩ ∈ A·B} 
  then obtain x y where x ∈ A and y ∈ B and p·x = (x·y)^{-1} by auto 
  with assms have y^{-1}·x^{-1} = (x·y)^{-1} using group_inv_of_two by auto 
  with p·(x·y)^{-1} have p = y^{-1}·x^{-1} by simp 
  with x ∈ A and y ∈ B have p∈{y^{-1}·x^{-1}.⟨x, y⟩ ∈ A·B} by auto 
} thus {(x·y)^{-1}.⟨x, y⟩ ∈ A·B} ⊆ {y^{-1}·x^{-1}.⟨x, y⟩ ∈ A·B} by blast
{ fix p assume p∈{y^{-1}·x^{-1}.⟨x, y⟩ ∈ A·B} 
  then obtain x y where x ∈ A and y ∈ B and p·y·x = (x·y)^{-1} by auto 
  with assms have p = (x·y)^{-1} using group_inv_of_two by auto 
  with x ∈ A and y ∈ B have p∈{(x·y)^{-1}.⟨x, y⟩ ∈ A·B} by auto 
} thus {y^{-1}·x^{-1}.⟨x, y⟩ ∈ A·B} ⊆ {(x·y)^{-1}.⟨x, y⟩ ∈ A·B} by blast
qed

with II show III: (A·B)^{-1} = {y^{-1}·x^{-1}.⟨x, y⟩ ∈ A·B} by simp 
have {y^{-1}·x^{-1}.⟨x, y⟩ ∈ A·B} = (B^{-1})·(A^{-1}) 
proof
{ fix p assume p∈{y^{-1}·x^{-1}.⟨x, y⟩ ∈ A·B} 
  then obtain x y where x ∈ A and y ∈ B and p·y·x = (x·y)^{-1} by auto 
  with assms have y^{-1} ∈ (B^{-1}) and x^{-1} ∈ (A^{-1}) 
  using interval_inv by auto 
  with p·y·x = (x·y)^{-1} have p∈{(B^{-1})·(A^{-1}) using interval_inv_cl interval_prod 
  by auto 
} thus {y^{-1}·x^{-1}.⟨x, y⟩ ∈ A·B} ⊆ (B^{-1})·(A^{-1}) by blast
{ fix p assume p∈(B^{-1})·(A^{-1}) 
  then obtain y x where y ∈ B and x ∈ A and p·y·x = (x·y)^{-1} using interval_inv_cl interval_prod by auto 
  with assms obtain x₁ y₁ where x₁ ∈ A and y₁ ∈ B and x₁·y₁ = x·y using interval_inv 
  by auto 
  with p·y·x = (x·y)^{-1} have p∈{y^{-1}·x^{-1}.⟨x, y⟩ ∈ A·B} by auto 
} thus (B^{-1})·(A^{-1}) ⊆ {y^{-1}·x^{-1}.⟨x, y⟩ ∈ A·B} by blast
qed

with III show (A·B)^{-1} = (B^{-1})·(A^{-1}) by simp

qed

If H, K are subgroups then H·K is a subgroup iff H·K = K·H.
theorem (in group4) prod_subgr_subgr:
assumes IsAsubgroup(H,P) and IsAsubgroup(K,P)
sows IsAsubgroup(H·K,P) ⟷ H·K = K·H
proof
assume IsAsubgroup(H·K,P)
then have (H·K)⁻¹ = H·K using subgroup_inv_sq(1) by simp
with assms show H·K = K·H using group0_3_L2 interval_prod_inv subgroup_inv_sq(1)
by auto
next
from assms have H⊆G and K⊆G using group0_3_L2 by auto
have I: H·K ≠ 0
proof
let x = 1 let y = 1
from assms have x·y ∈ (H·K) using group0_3_L5 group0_3_L2 interval_prod
by auto
thus thesis by auto
qed
from H·K = K·H
have II: (H·K){is closed under} P
proof
have (H·K)·(H·K) = H·K
proof
from H⊆G and K⊆G have (H·K)·(H·K) = H·(K·H)·K
using interval_prod_rearr1 by simp
also from H·K = K·H have ... = H·(K·H)·K by simp
also from H⊆G and K⊆G have ... = (H·H)·(K·K)
using interval_prod_rearr1 by simp
also from assms have ... = H·K using subgroup_inv_sq(2) by simp
finally show thesis by simp
qed
with H·K ⊆ G show thesis using subset_gr_op_cl by simp
qed
have IV: ∀x ∈ H·K. x⁻¹ ∈ H·K
proof
{ fix x assume x ∈ H·K
with H⊆G have x⁻¹ ∈ (H·K)⁻¹ using interval_inv by auto
with assms H⊆G and H·K = K·H have x⁻¹ ∈ H·K
using interval_prod_inv subgroup_inv_sq(1) by simp
} thus thesis by auto
qed
from I II III IV show IsAsubgroup(H·K,P) using group0_3_T3 by simp
qed
end
33 Groups - and alternative definition

theory Group_2F_1b imports Group_2F

begin

In a typical textbook a group is defined as a set $G$ with an associative operation such that two conditions hold:

A: there is an element $e \in G$ such that for all $g \in G$ we have $e \cdot g = g$ and $g \cdot e = g$. We call this element a "unit" or a "neutral element" of the group.

B: for every $a \in G$ there exists a $b \in G$ such that $a \cdot b = e$, where $e$ is the element of $G$ whose existence is guaranteed by A.

The validity of this definition is rather dubious to me, as condition A does not define any specific element $e$ that can be referred to in condition B - it merely states that a set of such units $e$ is not empty. Of course it does work in the end as we can prove that the set of such neutral elements has exactly one element, but still the definition by itself is not valid. You just can’t reference a variable bound by a quantifier outside of the scope of that quantifier.

One way around this is to first use condition A to define the notion of a monoid, then prove the uniqueness of $e$ and then use the condition B to define groups.

Another way is to write conditions A and B together as follows:

$\exists e \in G \ (\forall g \in G \ e \cdot g = g \land g \cdot e = g) \land (\forall a \in G \exists b \in G \ a \cdot b = e)$.

This is rather ugly.

What I want to talk about is an amusing way to define groups directly without any reference to the neutral elements. Namely, we can define a group as a non-empty set $G$ with an associative operation "\cdot" such that

C: for every $a, b \in G$ the equations $a \cdot x = b$ and $y \cdot a = b$ can be solved in $G$.

This theory file aims at proving the equivalence of this alternative definition with the usual definition of the group, as formulated in Group_2F.thy. The informal proofs come from an Aug. 14, 2005 post by buli on the matematyka.org forum.

33.1 An alternative definition of group

First we will define notation for writing about groups.

We will use the multiplicative notation for the group operation. To do this, we define a context (locale) that tells Isabelle to interpret $a \cdot b$ as the value of function $P$ on the pair $(a, b)$.

locale group2 =
  fixes P
The next theorem states that a set $G$ with an associative operation that satisfies condition C is a group, as defined in IsarMathLib Group_ZF theory.

**Theorem (in group2) altgroup_is_group:**

assumes $A1: G \neq \emptyset$ and $A2: P \{\text{is associative on}\} G$
and $A3: \forall a \in G. \forall b \in G. \exists x \in G. a \cdot x = b$
and $A4: \forall a \in G. \forall b \in G. \exists y \in G. y \cdot a = b$

shows $\text{IsAgroup}(G, P)$

**Proof** -

- From $A1$ obtain $a$ where $a \in G$ by auto.
- With $A3$ obtain $x$ where $x \in G$ and $a \cdot x = a$ by auto.
- From $A4$ obtain $y$ where $y \in G$ and $y \cdot a = a$ by auto.
- Have $I: \forall b \in G. b = b \cdot x \land b = y \cdot b$ by auto.

- Moreover from $A2$ obtain $x, y \in G$ where $y \cdot a = b \cdot x$ and $y \cdot x = y \cdot (a \cdot x)$ using $\text{IsAssociative}(G, P)$ by auto.
- Moreover from $y \cdot x = y \cdot (a \cdot x)$ have $y \cdot (a \cdot x) = y \cdot (a \cdot x)$ by auto.
- Ultimately show $b = b \cdot x$ and $b = y \cdot b$ by simp.

**QED.**

Moreover have $x = y$ by auto.

- From $x \in G$ I have $x = y \cdot x$ by simp.
- Also from $y \in G$ I have $y \cdot x = y$ by simp.
- Finally show $x = y$ by simp.

**QED.**

The converse of altgroup_is_group: in every (classically defined) group condition C holds. In informal mathematics we can say "Obviously condition C
holds in any group.” In formalized mathematics the word “obviously” is not in the language. The next theorem is proven in the context called group0 defined in the theory Group_ZF.thy. Similarly to the group2 that context defines \( a \cdot b \) as \( P(a,b) \) It also defines notation related to the group inverse and adds an assumption that the pair \((G,P)\) is a group to all its theorems. This is why in the next theorem we don’t explicitly assume that \((G,P)\) is a group - this assumption is implicit in the context.

**Theorem (in group0) group_is_altgroup**: shows
\[
\forall a \in G. \forall b \in G. \exists x \in G. a \cdot x = b \quad \text{and} \quad \forall a \in G. \forall b \in G. \exists y \in G. y \cdot a = b
\]

**Proof** -
\[
\{ \text{fix } a \ b \ \text{assume } a \in G \quad b \in G \\
\quad \text{let } x = a^{-1} \cdot b \\
\quad \text{let } y = b \cdot a^{-1} \\
\quad \text{from } a \in G \quad b \in G \quad \text{have } \\
\quad \quad x \in G \quad y \in G \quad \text{and} \quad a \cdot x = b \quad y \cdot a = b \\
\quad \text{using } \text{inverse_in_group} \quad \text{group_op_closed} \quad \text{inv_cancel_two} \\
\quad \text{by auto} \\
\quad \text{hence } \exists x \in G. a \cdot x = b \quad \text{and} \quad \exists y \in G. y \cdot a = b \quad \text{by auto} \\
\}
\]
thus
\[
\forall a \in G. \forall b \in G. \exists x \in G. a \cdot x = b \quad \text{and} \\
\forall a \in G. \forall b \in G. \exists y \in G. y \cdot a = b \\
\text{by auto}
\]
qed

**end**

34 **Abelian Group**

**theory AbelianGroup_ZF imports Group_ZF**

**begin**

A group is called “abelian” if its operation is commutative, i.e. \( P(a,b) = P(b,a) \) for all group elements \( a, b \), where \( P \) is the group operation. It is customary to use the additive notation for abelian groups, so this condition is typically written as \( a + b = b + a \). We will be using multiplicative notation though (in which the commutativity condition of the operation is written as \( a \cdot b = b \cdot a \)), just to avoid the hassle of changing the notation we used for general groups.

34.1 **Rearrangement formulae**

This section is not interesting and should not be read. Here we will prove formulas is which right hand side uses the same factors as the left hand side, just in different order. These facts are obvious in informal math sense, but
Isabelle prover is not able to derive them automatically, so we have to prove them by hand.

Proving the facts about associative and commutative operations is quite tedious in formalized mathematics. To a human the thing is simple: we can arrange the elements in any order and put parentheses wherever we want, it is all the same. However, formalizing this statement would be rather difficult (I think). The next lemma attempts a quasi-algorithmic approach to this type of problem. To prove that two expressions are equal, we first strip one from parentheses, then rearrange the elements in proper order, then put the parentheses where we want them to be. The algorithm for rearrangement is easy to describe: we keep putting the first element (from the right) that is in the wrong place at the left-most position until we get the proper arrangement. As far removing parentheses is concerned Isabelle does its job automatically.

lemma (in group0) group0_4_L2:
  assumes A1:P {is commutative on} G
  and A2:a∈G b∈G c∈G d∈G E∈G F∈G
  shows (a·b)·(c·d)·(E·F) = (a·(d·F))·(b·(c·E))
proof -
  from A2 have (a·b)·(c·d)·(E·F) = a·b·c·d·E·F
    using group_op_closed group_oper_assoc by simp
  also have a·b·c·d·E·F = a·d·F·b·c·E
proof -
  from A2 have a·b·c·d·E·F = F·(a·b·c·d·E)
    using IsCommutative_def group_op_closed by simp
  also from A2 have F·(a·b·c·d·E) = F·a·b·c·d·E
    using group_op_closed group_oper_assoc by simp
  also from A1 A2 have F·a·b·c·d·E = d·(F·a·b·c)·E
    using IsCommutative_def group_op_closed by simp
  also from A2 have d·(F·a·b·c)·E = d·F·a·b·c·E
    using group_op_closed group_oper_assoc by simp
  also from A1 A2 have d·F·a·b·c·E = a·(d·F)·b·c·E
    using IsCommutative_def group_op_closed by simp
  also from A2 have a·(d·F)·b·c·E = a·d·F·b·c·E
    using group_op_closed group_oper_assoc by simp
  finally show thesis by simp
qed
also from A2 have a·d·F·b·c·E = (a·(d·F))·(b·(c·E))
  using group_op_closed group_oper_assoc by simp
finally show thesis by simp

qed

Another useful rearrangement.

lemma (in group0) group0_4_L3:
  assumes A1: P {is commutative on} G
  and A2: a ∈ G b ∈ G and A3: c ∈ G d ∈ G E ∈ G F ∈ G
  shows a · b · ((c · d)⁻¹ · (E · F)⁻¹) = (a · (E · c)⁻¹) · (b · (F · d)⁻¹)
proof -
  from A3 have T1: c⁻¹ ∈ G d⁻¹ ∈ G E⁻¹ ∈ G F⁻¹ ∈ G (c · d)⁻¹ ∈ G (E · F)⁻¹ ∈ G
    using inverse_in_group group_op_closed by auto
  from A2 T1 have a · b · ((c · d)⁻¹ · (E · F)⁻¹) = a · b · (c · d)⁻¹ · (E · F)⁻¹
    using group_op_closed group_oper_assoc by simp
  also from A2 A3 have
    a · b · ((c · d)⁻¹ · (E · F)⁻¹) = a · b · (c · d)⁻¹ · (E · F)⁻¹
    using group_inv_of_two by simp
  finally show thesis by simp

qed

Some useful rearrangements for two elements of a group.

lemma (in group0) group0_4_L4:
  assumes A1: P {is commutative on} G
  and A2: a ∈ G b ∈ G
  shows b⁻¹ · a⁻¹ = a⁻¹ · b⁻¹
proof -
  from A2 have T1: b⁻¹ ∈ G a⁻¹ ∈ G using inverse_in_group by auto
    with A1 show b⁻¹ · a⁻¹ = a⁻¹ · b⁻¹ using IsCommutative_def by simp
    with A2 show (a · b)⁻¹ = a⁻¹ · b⁻¹ using group_inv_of_two by simp
  from A2 T1 have (a · b)⁻¹ = (b⁻¹)⁻¹ · a⁻¹
    using group_inv_of_two by simp
    with A1 A2 T1 show (a · b)⁻¹ = a⁻¹ · b
      using group_inv_of_inv IsCommutative_def by simp

qed

Another bunch of useful rearrangements with three elements.

lemma (in group0) group0_4_L4A:
  assumes A1: P {is commutative on} G
and A2: a\in G \quad b\in G \quad c\in G

shows
\[ a\cdot b\cdot c = c\cdot a\cdot b \]
\[ a^{-1}\cdot (b^{-1}\cdot c^{-1})^{-1} = (a\cdot (b\cdot c)^{-1})^{-1} \]
\[ a\cdot (b\cdot c)^{-1} = a\cdot b^{-1}\cdot c^{-1} \]
\[ a\cdot b^{-1}\cdot c^{-1} = a\cdot c^{-1}\cdot b^{-1} \]

proof -

from A1 A2 have a\cdot b\cdot c = c\cdot (a\cdot b)
  using IsCommutative_def group_op_closed
  by simp

with A2 show a\cdot b\cdot c = c\cdot a\cdot b using
  group_op_closed group_oper_assoc
  by simp

from A2 have T:
  \[ b^{-1}\in G \quad c^{-1}\in G \quad b^{-1}\cdot c^{-1} \in G \quad a\cdot b \in G \]
  using inverse_in_group group_op_closed
  by auto

with A1 A2 show a^{-1}\cdot (b^{-1}\cdot c^{-1})^{-1} = (a\cdot (b\cdot c)^{-1})^{-1}
  using group_inv_of_two IsCommutative_def
  by simp

from A1 A2 T have a\cdot (b\cdot c)^{-1} = a\cdot (b^{-1}\cdot c^{-1})
  using group_inv_of_two IsCommutative_def by simp

with A2 T show a\cdot (b\cdot c)^{-1} = a\cdot b^{-1}\cdot c^{-1}
  using group_oper_assoc by simp

from A1 A2 T have a\cdot (b\cdot c)^{-1} = a\cdot (b^{-1}\cdot (c^{-1})^{-1})
  using group_inv_of_two IsCommutative_def by simp

with A2 T show a\cdot (b\cdot c)^{-1} = a\cdot b^{-1}\cdot c
  using group_oper_assoc group_inv_of_inv by simp

from A1 A2 T have a\cdot b^{-1}\cdot c^{-1} = a\cdot (c^{-1}\cdot b^{-1})
  using group_oper_assoc group_oper_assoc by simp

with A2 T show a\cdot b^{-1}\cdot c^{-1} = a\cdot c^{-1}\cdot b^{-1}
  using group_oper_assoc by simp

qed

Another useful rearrangement.

lemma (in group0) group0_4_L4B:
  assumes P \{ is commutative on \} G
  and a\in G \quad b\in G \quad c\in G
  shows a\cdot b^{-1}\cdot (b\cdot c^{-1}) = a\cdot c^{-1}
  using assms inverse_in_group group_op_closed
  group0_4_L4 group_oper_assoc inv_cancel_two by simp

A couple of permutations of order for three elements.

lemma (in group0) group0_4_L4C:
  assumes A1: P \{ is commutative on \} G
  and A2: a\in G \quad b\in G \quad c\in G
  shows
  a\cdot b\cdot c = c\cdot a\cdot b
proof -
  from A1 A2 show I: \(a \cdot b \cdot c = c \cdot a \cdot b\)
  using group0_4_L4A by simp
  also from A1 A2 have c \cdot a \cdot b = a \cdot c \cdot b
  using IsCommutative_def by simp
  finally show \(a \cdot b \cdot c = c \cdot a \cdot b\)
  by simp

from A2 I show \(a \cdot b \cdot c = c \cdot a \cdot b\)
  using group_oper_assoc by simp
also from A1 A2 have \(c \cdot a \cdot b = c \cdot b \cdot a\)
  using IsCommutative_def by simp
also from A2 have \(c \cdot (a \cdot b) = c \cdot (b \cdot a)\)
  using group_oper_assoc by simp
finally show \(a \cdot b \cdot c = c \cdot b \cdot a\)
  by simp

Some rearrangement with three elements and inverse.

lemma (in group0) group0_4_L4D:
  assumes A1: \(P \text{ is commutative on } G\)
  and A2: \(a, b, c \in G\)
  shows \(a^{-1} \cdot b^{-1} \cdot c = c \cdot a^{-1} \cdot b^{-1}\)
proof -
  from A2 have T: \(a^{-1} \in G, b^{-1} \in G, c^{-1} \in G\)
  using inverse_in_group by auto
  with A1 A2 show \(a^{-1} \cdot b^{-1} \cdot c = c \cdot a^{-1} \cdot b^{-1}\)
  using group0_4_L4A by auto
from A1 A2 T show \((a^{-1} \cdot b^{-1})^{-1} = a \cdot b^{-1} \cdot c^{-1}\)
  using group_inv_of_three group_inv_of_inv group0_4_L4C
  by simp

Another rearrangement lemma with three elements and equation.

lemma (in group0) group0_4_L5: assumes A1: \(P \text{ is commutative on } G\)
  and A2: \(a, b, c \in G\)
  and A3: \(c = a \cdot b^{-1}\)
  shows \(a = b \cdot c\)
proof -
  from A2 A3 show \(c \cdot (b^{-1})^{-1} = a\)
  using inverse_in_group group0_2_L18

371
by simp
with A1 A2 show thesis using
group_inv_of_inv IsCommutative_def by simp
qed

In abelian groups we can cancel an element with its inverse even if separated
by another element.

lemma (in group0) group0_4_L6A: assumes A1: P {is commutative on} G
and A2: a∈G b∈G
shows
a·b·a⁻¹ = b
a⁻¹·b·a = b
a⁻¹·(b·a) = b
a·(b·a⁻¹) = b
proof -
  from A1 A2 have
    a·b·a⁻¹ = a⁻¹·a·b
    using inverse_in_group group0_4_L4A by blast
  also from A2 have ... = b
    using group0_2_L6 group0_2_L2 by simp
  finally show a·b·a⁻¹ = b by simp
from A1 A2 have
  a⁻¹·b·a = a·a⁻¹·b
  using inverse_in_group group0_4_L4A by blast
also from A2 have ... = b
  using group0_2_L6 group0_2_L2 by simp
finally show a⁻¹·b·a = b by simp
moreover from A2 have a⁻¹·b·a = a⁻¹·(b·a)
  using inverse_in_group group_oper_assoc by simp
ultimately show a⁻¹·(b·a) = b by simp
from A1 A2 have a·(b·a⁻¹) = b
  using inverse_in_group IsCommutative_def inv_cancel_two
  by simp
qed

Another lemma about cancelling with two elements.

lemma (in group0) group0_4_L6AA:
  assumes A1: P {is commutative on} G and A2: a∈G b∈G
  shows a·b⁻¹·a⁻¹ = b⁻¹
  using assms inverse_in_group group0_4_L6A
  by auto

Another lemma about cancelling with two elements.

lemma (in group0) group0_4_L6AB:
  assumes A1: P {is commutative on} G and A2: a∈G b∈G
  shows
    a·(a·b)⁻¹ = b⁻¹
    a·(b·a⁻¹) = b
proof -
from A2 have \((a \cdot b)^{-1} = a \cdot (b^{-1} \cdot a^{-1})\)
    using group_inv_of_two by simp
also from A2 have ... = \(a \cdot b^{-1} \cdot a^{-1}\)
    using inverse_in_group group_oper_assoc by simp
also from A1 A2 have ... = \(b^{-1}\)
    using group0_4_L6AA by simp
finally show \(a \cdot (b \cdot a^{-1}) = a \cdot (a^{-1} \cdot b)\)
    using inverse_in_group IsCommutative_def by simp
also from A2 have ...
    using inverse_in_group group_oper_assoc group0_2_L6 group0_2_L2
    by simp
finally show \(a \cdot (b \cdot a^{-1}) = b\) by simp
qed

Another lemma about cancelling with two elements.

**lemma** (in group0) group0_4_L6AC:
assumes \(P\) {is commutative on} \(G\) and \(a \in G\) \(b \in G\)
sows \(a \cdot (a \cdot b \cdot c) \cdot a^{-1} = b\)
using assms inverse_in_group group0_4_L6AB group_inv_of_inv
by simp

In abelian groups we can cancel an element with its inverse even if separated
by two other elements.

**lemma** (in group0) group0_4_L6B: assumes A1: \(P\) {is commutative on} \(G\)
and A2: \(a \in G\) \(b \in G\) \(c \in G\)
sows \(a \cdot b \cdot c \cdot a^{-1} = b \cdot c\)
\(a^{-1} \cdot b \cdot c \cdot a = b \cdot c\)
proof -
    from A2 have
        \(a \cdot b \cdot c \cdot a^{-1} = a \cdot (b \cdot c) \cdot a^{-1}\)
        \(a^{-1} \cdot b \cdot c \cdot a = a^{-1} \cdot (b \cdot c) \cdot a\)
    using group_op_closed group_oper_assoc inverse_in_group
    by auto
with A1 A2 show
    \(a \cdot b \cdot c \cdot a^{-1} = b \cdot c\)
    \(a^{-1} \cdot b \cdot c \cdot a = b \cdot c\)
    using group_op_closed group0_4_L6A
    by auto
qed

In abelian groups we can cancel an element with its inverse even if separated
by three other elements.

**lemma** (in group0) group0_4_L6C: assumes A1: \(P\) {is commutative on} \(G\)
and A2: \(a \in G\) \(b \in G\) \(c \in G\) \(d \in G\)
sows \(a \cdot b \cdot c \cdot d \cdot a^{-1} = b \cdot c \cdot d\)
proof -
from A2 have \( a \cdot b \cdot c \cdot d \cdot a^{-1} = a \cdot (b \cdot c \cdot d) \cdot a^{-1} \)
  using group_op_closed group_oper_assoc
  by simp
with A1 A2 show thesis
  using group_op_closed group0_4_L6A
  by simp
qed

Another couple of useful rearrangements of three elements and cancelling.

lemma (in group0) group0_4_L6D:
  assumes A1: \( P \) (is commutative on) \( G \)
  and A2: \( a \in G \ b \in G \ c \in G \)
  shows \( a \cdot b \cdot c \cdot a^{-1} = a \cdot (b \cdot c \cdot a^{-1}) \cdot b \cdot c \)
proof -
  from A2 have T:
    \( a^{-1} \in G \ b^{-1} \in G \ c^{-1} \in G \)
    \( a \cdot b \in G \ a^{-1} \cdot b^{-1} \in G \ c^{-1} \cdot a^{-1} \in G \ c^{-1} \cdot a^{-1} \cdot b \)
    \( \text{using inverse_in_group group_op_closed by auto} \)
with A1 A2 show \( a \cdot b \cdot c \cdot a^{-1} = a \cdot (b \cdot c \cdot a^{-1}) \cdot b \cdot c \)
  using group0_4_L6B IsCommutative_def by simp
from A2 T have \( (a \cdot c)^{-1} \cdot (b \cdot c) = c^{-1} \cdot a^{-1} \cdot b \cdot c \)
  using group_inv_of_two group_oper_assoc by simp
also from A1 A2 T have \( ... = a^{-1} \cdot b \)
  using group0_4_L6B by simp
finally show \( (a \cdot c)^{-1} \cdot (b \cdot c) = a^{-1} \cdot b \)
  by simp
from A1 A2 T show \( a \cdot (b \cdot (c \cdot a^{-1} \cdot b^{-1})) = c \)
  using group_oper_assoc group0_4_L6A group0_4_L6A
  by simp
from T have \( a \cdot b \cdot c \cdot a^{-1} = a \cdot b \cdot (c^{-1} \cdot (c \cdot a^{-1})) \)
  using group_oper_assoc by simp
also from A1 A2 T have \( ... = b \)
  using group_oper_assoc group0_2_L6 group0_2_L2 group0_4_L6A
  by simp
finally show \( a \cdot b \cdot c \cdot (c \cdot a^{-1}) = b \text{ by simp} \)
qed

Another useful rearrangement of three elements and cancelling.

lemma (in group0) group0_4_L6E:
  assumes A1: \( P \) (is commutative on) \( G \)
  and A2: \( a \in G \ b \in G \ c \in G \)
  shows \( a \cdot b \cdot (a \cdot c)^{-1} = b \cdot c \)
proof -

from A2 have T: \( b^{-1} \in G \) \( c^{-1} \in G \)
using inverse_in_group by auto
with A1 A2 have
\[
a \cdot (b^{-1})^{-1} \cdot (a \cdot (c^{-1})^{-1})^{-1} = c^{-1} \cdot (b^{-1})^{-1}
\]
using group0_4_L6D by simp
with A1 A2 T show \( a \cdot b \cdot (a \cdot c)^{-1} = b^{-1} \cdot c^{-1} \)
using group_inv_of_inv IsCommutative_def by simp
qed

A rearrangement with two elements and cancelling, special case of group0_4_L6D when \( c = b^{-1} \).

lemma (in group0) group0_4_L6F:
assumes A1: P \{is commutative on\} G
and A2: \( a \in G \) \( b \in G \)
shows \( a \cdot b^{-1} \cdot (a \cdot b)^{-1} = b^{-1} \cdot b^{-1} \)
proof -
from A1 A2 have \( a \cdot b \cdot c \cdot d = d \cdot (a \cdot b) \cdot c \)
using IsCommutative_def group_op_closed by simp
also from A2 have \( a \cdot b \cdot c \cdot d = d \cdot a \cdot b \cdot c \)
using group_op_closed group_oper_assoc by simp
also from A1 A2 have \( a \cdot b \cdot c \cdot d = a \cdot d \cdot b \cdot c \)
using IsCommutative_def group_op_closed by simp
finally show \( a \cdot b \cdot c \cdot d = a \cdot d \cdot b \cdot c \)
by simp
from A1 A2 have \( a \cdot b \cdot c \cdot d = c \cdot (a \cdot b) \cdot d \)
using IsCommutative_def group_op_closed by simp
also from A2 have \( a \cdot b \cdot c \cdot d = c \cdot a \cdot b \cdot d \)

Some other rearrangements with four elements. The algorithm for proof as in group0_4_L2 works very well here.

lemma (in group0) rearr_ab_gr_4_elemA:
assumes A1: P \{is commutative on\} G
and A2: \( a \in G \) \( b \in G \) \( c \in G \) \( d \in G \)
shows 
\[
a \cdot b \cdot c \cdot d = a \cdot d \cdot b \cdot c
\]
proof -
from A1 A2 have \( a \cdot b \cdot c \cdot d = d \cdot (a \cdot b) \cdot c \)
using IsCommutative_def group_op_closed by simp
also from A2 have \( a \cdot b \cdot c \cdot d = d \cdot a \cdot b \cdot c \)
using group_op_closed group_oper_assoc by simp
also from A1 A2 have \( a \cdot b \cdot c \cdot d = a \cdot d \cdot b \cdot c \)
using IsCommutative_def group_op_closed by simp
finally show \( a \cdot b \cdot c \cdot d = a \cdot d \cdot b \cdot c \)
by simp
from A1 A2 have \( a \cdot b \cdot c \cdot d = c \cdot (a \cdot b) \cdot d \)
using IsCommutative_def group_op_closed by simp
also from A2 have \( a \cdot b \cdot c \cdot d = c \cdot a \cdot b \cdot d \)
using group_op_closed group_oper_assoc
by simp
also from A1 A2 have ... = a·c·b·d
  using IsCommutative_def group_op_closed
  by simp
also from A2 have ... = a·c·(b·d)
  using group_op_closed group_oper_assoc
  by simp
finally show a·b·c·d = a·c·(b·d)
  by simp
qed

Some rearrangements with four elements and inverse that are applications
of rearr_ab_gr_4_elem

lemma (in group0) rearr_ab_gr_4_elemB:
  assumes A1: P {is commutative on} G
  and A2: a∈G b∈G c∈G d∈G
  shows
  a·b⁻¹·c⁻¹·d⁻¹ = a·d⁻¹·b⁻¹·c⁻¹
  a·b·c·d⁻¹ = a·c⁻¹·b·c
  a·b·c⁻¹·d⁻¹ = a·c⁻¹·(b·d⁻¹)
proof -
  from A2 have T: b⁻¹ ∈ G c⁻¹ ∈ G d⁻¹ ∈ G
    using inverse_in_group by auto
  with A1 A2 show
    a·b⁻¹·c⁻¹·d⁻¹ = a·d⁻¹·b⁻¹·c⁻¹
    a·b·c·d⁻¹ = a·c⁻¹·b·c
    a·b·c⁻¹·d⁻¹ = a·c⁻¹·(b·d⁻¹)
    using rearr_ab_gr_4_elemA by auto
qed

Some rearrangement lemmas with four elements.

lemma (in group0) group0_4_L7:
  assumes A1: P {is commutative on} G
  and A2: a∈G b∈G c∈G d∈G
  shows
    a·b·c·d⁻¹ = a·d⁻¹·b·c
    a·d·(b·d·(c·d))⁻¹ = a·(b·c)⁻¹·d⁻¹
    a·(b·c)·d = a·b·d·c
proof -
  from A2 have T:
    b·c ∈ G d⁻¹ ∈ G b⁻¹∈G c⁻¹∈G
d⁻¹,b ∈ G c⁻¹·d ∈ G (b·c)⁻¹ ∈ G
  b·d ∈ G b·d·c ∈ G (b·d·c)⁻¹ ∈ G
  a·d ∈ G b·c ∈ G
  using group_op_closed inverse_in_group
  by auto
  with A1 A2 have a·b·c·d⁻¹ = a·(d⁻¹·b·c)
    using group_oper_assoc group0_4_L4A by simp
also from A2 T have \( a \cdot (d^{-1} \cdot b \cdot c) = a \cdot d^{-1} \cdot b \cdot c \)
using group_op_assoc by simp
finally show \( a \cdot b \cdot c \cdot d^{-1} = a \cdot d^{-1} \cdot b \cdot c \) by simp
from A2 T have \( a \cdot d \cdot (b \cdot (c \cdot d))^{-1} = a \cdot d \cdot (d^{-1} \cdot (b \cdot d \cdot c)^{-1}) \)
using group_op_assoc group_inv_of_two by simp
also from A2 T have \( \ldots = a \cdot (b \cdot d \cdot c)^{-1} \)
using group_op_assoc inv_cancel_two by simp
also from A1 A2 have \( \ldots = a \cdot (d \cdot (b \cdot c))^{-1} \)
using IsCommutative_def group_op_assoc by simp
also from A2 T have \( \ldots = a \cdot ((b \cdot c)^{-1} \cdot d^{-1}) \)
using group_inv_of_two by simp
also from A2 T have \( \ldots = a \cdot (b \cdot c)^{-1} \cdot d^{-1} \)
using group_op_assoc by simp
finally show \( a \cdot d \cdot (b \cdot (c \cdot d))^{-1} = a \cdot (b \cdot c)^{-1} \cdot d^{-1} \)
by simp
from A2 have \( a \cdot (b \cdot c) \cdot d = a \cdot (b \cdot (c \cdot d)) \)
using group_op_closed group_op_assoc by simp
also from A1 A2 have \( \ldots = a \cdot (b \cdot (d \cdot c)) \)
using IsCommutative_def group_op_closed by simp
also from A2 have \( \ldots = a \cdot b \cdot d \cdot c \)
using group_op_closed group_op_assoc by simp
finally show \( a \cdot (b \cdot c) \cdot d = a \cdot b \cdot d \cdot c \) by simp
qed

Some other rearrangements with four elements.

**lemma** (in group0) group0_4_L8:
assumes A1: \( P \) (is commutative on) \( G \)
and A2: \( a \in G \ b \in G \ c \in G \ d \in G \)
shows
\[
a \cdot (b \cdot c)^{-1} = (a \cdot d^{-1} \cdot c^{-1}) \cdot (d \cdot b^{-1})
\]
\[
a \cdot b \cdot (c \cdot d) = c \cdot a \cdot (b \cdot d)
\]
\[
a \cdot b \cdot (c \cdot d) = a \cdot c \cdot (b \cdot d)
\]
\[
a \cdot (b \cdot c)^{-1} \cdot d = a \cdot b \cdot d \cdot c^{-1}
\]
\[
(a \cdot b) \cdot (c \cdot d)^{-1} \cdot (d \cdot b^{-1})^{-1} = a \cdot c^{-1}
\]
**proof**
from A2 have T:
\( b \cdot c \in G \ a \cdot b \in G \ d^{-1} \in G \ b^{-1} \in G \ c^{-1} \in G \)
\( d^{-1} \cdot b \in G \ c^{-1} \cdot d \in G \ (b \cdot c)^{-1} \in G \)
\( a \cdot b \in G \ (c \cdot d)^{-1} \in G \ (b \cdot d^{-1})^{-1} \in G \ d \cdot b^{-1} \in G \)
using group_op_closed inverse_in_group by auto
from A2 have \( a \cdot (b \cdot c)^{-1} = a \cdot c^{-1} \cdot b^{-1} \) using group0_2_L14A by blast
moreover from A2 have \( a \cdot c^{-1} = (a \cdot d^{-1}) \cdot (d \cdot c^{-1}) \) using group0_2_L14A by blast
ultimately have \( a \cdot (b \cdot c)^{-1} = (a \cdot d^{-1}) \cdot (d \cdot c^{-1}) \cdot b^{-1} \) by simp
with A1 A2 T have \( a \cdot (b \cdot c)^{-1} = a \cdot d^{-1} \cdot (c^{-1} \cdot d) \cdot b^{-1} \)
using IsCommutative_def by simp
with A2 T show \( a \cdot (b \cdot c)^{-1} = (a \cdot d^{-1} \cdot c^{-1}) \cdot (d \cdot b^{-1}) \)
using group_op_closed group_op_assoc by simp

377
from A2 T have \( a \cdot b \cdot (c \cdot d) = a \cdot b \cdot c \cdot d \)
using group_oper_assoc by simp
also have \( a \cdot b \cdot c \cdot d = c \cdot a \cdot b \cdot d \)
proof -
  from A1 A2 have \( a \cdot b \cdot c \cdot d = c \cdot (a \cdot b) \cdot d \)
  using IsCommutative_def group_op_closed by simp
also from A2 have \( \ldots = c \cdot a \cdot b \cdot d \)
  using group_op_closed group_oper_assoc by simp
finally show thesis by simp
qed
also from A2 have \( c \cdot a \cdot b \cdot d = c \cdot a \cdot (b \cdot d) \)
using group0_4_L7 by simp
finally show \( a \cdot b \cdot (c \cdot d) = a \cdot c \cdot (b \cdot d) \)
by simp
also from A1 A2 T have \( \ldots = (a \cdot b) \cdot (c^{-1} \cdot (d^{-1} \cdot (d \cdot b^{-1}))) \)
using group_inv_of_two group0_2_L12 IsCommutative_def by simp
also from T have \( \ldots = (a \cdot b) \cdot (c^{-1} \cdot (d^{-1} \cdot (d \cdot b^{-1}))) \)
using group_oper_assoc group0_4_L8A: assumes A1: P (is commutative on) G
and A2: \( a \in G \quad b \in G \quad c \in G \quad d \in G \)
sows
\( a \cdot b^{-1} \cdot (c \cdot d^{-1}) = a \cdot c \cdot (b^{-1} \cdot d^{-1}) \)
\( a \cdot b^{-1} \cdot (c \cdot d^{-1}) = a \cdot c \cdot b^{-1} \cdot d^{-1} \)
proof -
  from A2 have \( T: a \cdot b^{-1} \in G \quad c \in G \quad d^{-1} \in G \)
  using inverse_in_group by auto
with A1 show \( a \cdot b^{-1} \cdot (c \cdot d^{-1}) = a \cdot c \cdot (b^{-1} \cdot d^{-1}) \)
  by (rule group0_4_L8)
with A2 T show \( a \cdot b^{-1} \cdot (c \cdot d^{-1}) = a \cdot c \cdot b^{-1} \cdot d^{-1} \)
  using group_op_closed group_oper_assoc
Some rearrangements with an equation.

**Lemma (in group0) group0_4_L9:**

- Assumes $A1: P$ (is commutative on) $G$
- and $A2: a \in G$, $b \in G$, $c \in G$, $d \in G$
- and $A3: a = b \cdot c^{-1} \cdot d^{-1}$

**shows**

- $d = b \cdot a^{-1} \cdot c^{-1}$
- $d = a^{-1} \cdot b \cdot c^{-1}$
- $b = a \cdot d \cdot c$

**Proof**

- From $A2$ have $T:
  - $a^{-1} \in G$, $c^{-1} \in G$, $d^{-1} \in G$, $b \cdot c^{-1} \in G$
- Using `group_op_closed inverse_in_group` by auto
- With $A2$ $A3$ have $a \cdot (d^{-1})^{-1} = b \cdot c^{-1}$
- Using `group0_2_L18` by simp
- With $A2$ have $b \cdot c^{-1} = a \cdot d$
- Using `group_inv_of_inv` by simp
- With $A2$ $T$ have $I: a^{-1} \cdot (b \cdot c^{-1}) = d$
- Using `group0_2_L18` by simp
- With $A1$ $A2$ $T$ show
  - $d = b \cdot a^{-1} \cdot c^{-1}$
  - $d = a^{-1} \cdot b \cdot c^{-1}$
- Using `group_oper_assoc IsCommutative_def` by auto
- From $A3$ have $a \cdot d \cdot c = (b \cdot c^{-1} \cdot d^{-1}) \cdot d \cdot c$ by simp
- Also from $A2$ $T$ have $... = b \cdot c^{-1} \cdot (d^{-1} \cdot d) \cdot c$
- Using `group_oper_assoc` by simp
- Also from $A2$ $T$ have $... = b \cdot c^{-1} \cdot c$
- Using `group0_2_L6` `group0_2_L2` by simp
- Also from $A2$ $T$ have $... = b \cdot (c^{-1} \cdot c)$
- Using `group_oper_assoc` by simp
- Also from $A2$ have $... = b$
- Using `group0_2_L6` `group0_2_L2` by simp
- Finally have $a \cdot d \cdot c = b$ by simp
- Thus $b = a \cdot d \cdot c$ by simp

**qed**

**end**

### 35 Groups 2

The `theory Group_ZF_2 imports AbelianGroup_ZF func_ZF EquivClass1` begins

This theory continues `Group_ZF.thy` and considers lifting the group struc-
ture to function spaces and projecting the group structure to quotient spaces, in particular the quotient group.

### 35.1 Lifting groups to function spaces

If we have a monoid (group) $G$ than we get a monoid (group) structure on a space of functions valued in in $G$ by defining $(f \cdot g)(x) := f(x) \cdot g(x)$. We call this process "lifting the monoid (group) to function space". This section formalizes this lifting.

The lifted operation is an operation on the function space.

**Lemma (in monoid0) Group_ZF_2_1_L0A:**

Assume $A1: F = f \{\text{lifted to function space over} \} X$

shows $F : (X \rightarrow G) \times (X \rightarrow G) \rightarrow (X \rightarrow G)$

**Proof**

- From $monoidAssum$ have $f : G \times G \rightarrow G$
  - Using $IsAmonoid_def$ $IsAssociative_def$ by simp
  - With $A1$ show thesis
    - Using $func_ZF_1_L3$ $group0_1_L3B$ by auto

**QED**

The result of the lifted operation is in the function space.

**Lemma (in monoid0) Group_ZF_2_1_L0:**

Assume $A1: F = f \{\text{lifted to function space over} \} X$

and $A2: \text{s:X} \rightarrow \text{G r:X} \rightarrow \text{G}$

shows $F \langle s,r \rangle : X \rightarrow G$

**Proof**

- From $A1$ have $F : (X \rightarrow G) \times (X \rightarrow G) \rightarrow (X \rightarrow G)$
  - Using $Group_ZF_2_1_L0A$ by simp
  - With $A2$ show thesis using $apply_funtype$
    - By simp

**QED**

The lifted monoid operation has a neutral element, namely the constant function with the neutral element as the value.

**Lemma (in monoid0) Group_ZF_2_1_L1:**

Assume $A1: F = f \{\text{lifted to function space over} \} X$

and $A2: E = \text{ConstantFunction(X,TheNeutralElement(G,f))}$

shows $E : X \rightarrow G \land (\forall s:X \rightarrow G. \ F( E,s ) = s \land F( s,E ) = s)$

**Proof**

- From $A2$ show $T1:E : X \rightarrow G$
  - Using $unit_is_neutral$ $func1_3_L1$ by simp
  - Show $\forall s:X \rightarrow G. \ F( E,s ) = s \land F( s,E ) = s$
    - Proof
      - Fix $s$ assume $A3:s:X \rightarrow G$
      - From $monoidAssum$ have $T2:f : G \times G \rightarrow G$
using IsAmonoid_def IsAssociative_def by simp
from A3 A1 T1 have
  F( E,s) : X→G F( s,E) : X→G s : X→G
  using Group_ZF_2_1_L0 by auto
moreover from T2 A1 T1 A2 A3 have
  ∀x∈X. (F( E,s))(x) = s(x)
  ∀x∈X. (F( s,E))(x) = s(x)
  using func_ZF_1_L4 group0_1_L3B func1_3_L2
apply_type unit_is_neutral by auto
ultimately show
  F( E,s) = s ∧ F( s,E) = s
  using fun_extension_iff by auto
qed

Monoids can be lifted to a function space.

lemma (in monoid0) Group_ZF_2_1_T1:
  assumes A1: F = f {lifted to function space over} X
  shows IsAmonoid(X→G,F)
proof -
  from monoidAssum A1 have
    T1:monoid0(G,f)
    and T2:monoid0(X→G,F)
  using monoid0_def by auto
moreover from A1 have
  ∃ E ∈ X→G. ∀ s ∈ X→G. F( E,s) = s ∧ F( s,E) = s
  using Group_ZF_2_1_L1 by blast
ultimately show thesis using IsAmonoid_def
  by simp
qed

The constant function with the neutral element as the value is the neutral
element of the lifted monoid.

lemma Group_ZF_2_1_L2:
  assumes A1: IsAmonoid(G,f)
  and A2: F = f {lifted to function space over} X
  and A3: E = ConstantFunction(X,TheNeutralElement(G,f))
  shows E = TheNeutralElement(X→G,F)
proof -
  from A1 A2 have
    T1:monoid0(G,f) and T2:monoid0(X→G,F)
  using monoid0_def by auto
from T1 A2 A3 have
  E : X→G ∧ (∀s∈X→G. F( E,s) = s ∧ F( s,E) = s)
  using monoid0.Group_ZF_2_1_L1 by simp
with T2 show thesis
  using monoid0.Group0_1_L4 by auto
qed
The lifted operation acts on the functions in a natural way defined by the monoid operation.

**Lemma (in monoid0) lifted_val:**
- Assumes $F = f$ (lifted to function space over) $X$
- and $s : X \rightarrow G$
- and $r : X \rightarrow G$
- and $x \in X$
- shows $(F(s, r))(x) = s(x) \oplus r(x)$
  - using monoidAssum assms IsAmonoid_def IsAssociative_def
  - group0_1_L3B func_ZF_1_L4
  - by auto

The lifted operation acts on the functions in a natural way defined by the group operation. This is the same as `lifted_val`, but in the group0 context.

**Lemma (in group0) Group_ZF_2_1_L3:**
- Assumes $F = P$ (lifted to function space over) $X$
- and $s : X \rightarrow G$
- and $r : X \rightarrow G$
- and $x \in X$
- shows $(F(s, r))(x) = s(x) \cdot r(x)$
  - using assms group0_2_L1 monoid0.lifted_val by simp

In the group0 context we can apply theorems proven in monoid0 context to the lifted monoid.

**Lemma (in group0) Group_ZF_2_1_L4:**
- Assumes $A1: F = P$ (lifted to function space over) $X$
- shows monoid0($X \rightarrow G, F$)
  - proof -
    - from $A1$ show thesis
      - using group0_2_L1 monoid0.Group_ZF_2_1_T1 monoid0_def
      - by simp
  - qed

The composition of a function $f : X \rightarrow G$ with the group inverse is a right inverse for the lifted group.

**Lemma (in group0) Group_ZF_2_1_L5:**
- Assumes $A1: F = P$ (lifted to function space over) $X$
- and $A2: s : X \rightarrow G$
- and $A3: i = \text{GroupInv}(G,P) \circ s$
- shows $i : X \rightarrow G$ and $F(s,i) = \text{TheNeutralElement}(X \rightarrow G,F)$
  - proof -
    - let $E = \text{ConstantFunction}(X,1)$
    - have $E : X \rightarrow G$
      - using group0_2_L2 func1_3_L1 by simp
    - moreover from groupAssum $A2$ $A3$ $A1$ have
      - $F(s,i) : X \rightarrow G$ using group0_2_T2 comp_fun
        - Group_ZF_2_1_L4 monoid0.group0_1_L1
        - by simp
    - moreover from groupAssum $A2$ $A3$ $A1$ have
      - $\forall x \in X. \ (F(s,i))(x) = E(x)$

382
Groups can be lifted to the function space.

theorem (in group0) Group_ZF_2_1_T2:
assumes 
A1: F = P \{lifted to function space over\} X 
shows IsAgroup(X→G,F)
proof-
  from A1 have IsAmonoid(X→G,F)
    using group0_2_L1 monoid0.Group_ZF_2_1_T1
    by simp
  moreover have ∀ s∈X→G. ∃ i∈X→G. F( s, i) = TheNeutralElement(X→G,F)
    proof
      fix s assume A2: s : X→G
      let i = GroupInv(G,P) O s
      from groupAssum A2 have i:X→G
        using group0_2_T2 comp_fun by simp
      moreover from A1 A2 have
        F( s, i) = TheNeutralElement(X→G,F)
        using Group_ZF_2_1_L5
        by fast
      ultimately show ∃ i∈X→G. F( s, i) = TheNeutralElement(X→G,F)
        by auto
    qed
  ultimately show thesis using IsAgroup_def
    by simp
qed

The propositions proven in the group0 context are valid in the same context when applied to the function space with the lifted group operation.

lemma (in group0) group0_valid_fun_space:
  shows group0(X→G,P \{lifted to function space over\} X)
  using Group_ZF_2_1_T2 unfolding group0_def by simp

What is the group inverse for the lifted group?

lemma (in group0) Group_ZF_2_1_L6:
assumes A1: F = P \{lifted to function space over\} X 
shows ∀ s∈(X→G). GroupInv(X→G,F)(s) = GroupInv(G,P) 0 s
proof -
from A1 have group0(X→G,F)
  using group0_def Group_ZF_2_1_T2
  by simp
moreover from A1 have ∀s∈X→G. GroupInv(G,P) O s : X→G ∧
  F⟨s,GroupInv(G,P) O s⟩ = TheNeutralElement(X→G,F)
  using Group_ZF_2_1_L5 by simp
ultimately have ∀s∈X→G. GroupInv(G,P) O s = GroupInv(X→G,F)(s)
  by (rule group0.group0_2_L9A)
thus thesis by simp
qed

What is the value of the group inverse for the lifted group?

corollary (in group0) lift_gr_inv_val:
  assumes F = P {lifted to function space over} X and
  s : X→G and x∈X
  shows (GroupInv(X→G,F)(s))(x) = (s(x))⁻¹
  using groupAssum assms Group_ZF_2_1_L6 group0_2_T2 comp_fun_apply
  by simp

What is the group inverse in a subgroup of the lifted group?

lemma (in group0) Group_ZF_2_1_L6A:
  assumes A1: F = P {lifted to function space over} X and
  A2: IsAsubgroup(H,F) and
  A3: g = restrict(F,H×H) and
  A4: s∈H
  shows GroupInv(H,g)(s) = GroupInv(G,P) O s
proof -
  from A1 have T1: group0(X→G,F)
    using group0_def Group_ZF_2_1_T2
    by simp
  with A2 A3 A4 have GroupInv(H,g)(s) = GroupInv(X→G,F)(s)
    using group0.group0_3_T1 restrict by simp
  moreover from T1 A1 A2 A4 have
    GroupInv(X→G,F)(s) = GroupInv(G,P) O s
    using group0.group0_3_L2 Group_ZF_2_1_L6 by blast
  ultimately show thesis by simp
qed

The neutral element of a subgroup of the lifted group is the constant function
with value equal to the neutral element of the group.

lemma (in group0) lift_group_subgr_neut:
  assumes F = P {lifted to function space over} X and IsAsubgroup(H,F)
  shows TheNeutralElement(H,restrict(F,H×H)) = ConstantFunction(X,1)
proof -
  from assms have
    TheNeutralElement(H,restrict(F,H×H)) = TheNeutralElement(X→G,F)
    using group0_valid_fun_space group0.group0_3_L4 by blast
  also from groupAssum assms(1) have ... = ConstantFunction(X,1)
If a group is abelian, then its lift to a function space is also abelian.

**Lemma (in group0) GroupZF_2_1_L7:**

- Assumes A1: \( F = P \) \( \text{liffted to function space over} \) X
- and A2: \( P \) \( \text{is commutative on} \) G
- shows \( F \) \( \text{is commutative on} \) (\( X \rightarrow G \))

**Proof:**

- From A1 A2 have \( F \) \( \text{is commutative on} \) (\( X \rightarrow \text{range}(P) \))
- using group_oper_fun funcZF_2_L2
- by simp
- moreover from groupAssum have \( \text{range}(P) = G \)
- using group0_2_L1 monoid0.group0_1_L3B
- by simp
- ultimately show thesis by simp

**QED**

### 35.2 Equivalence relations on groups

The goal of this section is to establish that (under some conditions) given an equivalence relation on a group or (monoid) we can project the group (monoid) structure on the quotient and obtain another group.

The neutral element class is neutral in the projection.

**Lemma (in monoid0) GroupZF_2_2_L1:**

- Assumes A1: \( \text{equiv}(G,r) \) and A2: Congruent2(r,f)
- and A3: \( F = \text{ProjFun2}(G,r,f) \)
- and A4: \( e = \text{TheNeutralElement}(G,f) \)
- shows \( r\{e\} \in G//r \land (\forall c \in G//r. F(r\{e\},c) = c \land F(c,r\{e\}) = c) \)

**Proof:**

- From A4 show T1: \( r\{e\} \in G//r \)
- using unit_is_neutral quotientI
- by simp
- show \( \forall c \in G//r. F(r\{e\},c) = c \land F(c,r\{e\}) = c \)
- **Proof:**
- fix \( c \) assume A5: \( c \in G//r \)
- then obtain \( g \) where D1: \( g \in G \land c = r\{g\} \)
- using quotient_def by auto
- with A1 A2 A3 A4 D1 show \( F(r\{e\},c) = c \land F(c,r\{e\}) = c \)
- using unit_is_neutral EquivClass_1_L10
- by simp
- **QED**
The projected structure is a monoid.

**Theorem (in monoid0) Group_ZF_2_2_T1:**

- **Assumptions:**
  - \( A1: \equiv(G,r) \) and \( A2: \text{Congruent2}(r,f) \)
  - \( A3: F = \text{ProjFun2}(G,r,f) \)

- **Shows:** \( \text{IsAmonoid}(G//r,F) \)

**Proof:**

1. Let \( E = r\{\text{TheNeutralElement}(G,f)\} \)
2. From \( A1 \ A2 \ A3 \) have
   - \( E \in G//r \wedge (\forall c \in G//r. F( E, c) = c \land F( c, E) = c) \)
   - Using `Group_ZF_2_2_L1` by simp
3. Hence
   - \( \exists E \in G//r. \forall c \in G//r. F( E, c) = c \land F( c, E) = c \)
   - By auto
4. With `monoidAssum A1 A2 A3` show thesis
   - Using `IsAmonoid_def` `EquivClass_2_T2` by simp

**QED**

The class of the neutral element is the neutral element of the projected monoid.

**Lemma (in group0) Group_ZF_2_2_L1:**

- **Assumptions:**
  - \( A1: \text{IsAmonoid}(G,f) \)
  - \( A2: \equiv(G,r) \) and \( A3: \text{Congruent2}(r,f) \)
  - \( A4: a \in G \) \( b \in G \)

- **Shows:** \( r\{e\} = \text{TheNeutralElement}(G//r,F) \)

**Proof:**

1. From \( A1 \ A2 \ A3 \ A4 \) show thesis
   - Using `EquivClass_1_L10` by simp

**QED**

The projected operation can be defined in terms of the group operation on representants in a natural way.

**Lemma (in group0) Group_ZF_2_2_L2:**

- **Assumptions:**
  - \( A1: \equiv(G,r) \) and \( A2: \text{Congruent2}(r,P) \)
  - \( A3: F = \text{ProjFun2}(G,r,P) \)
  - \( A4: a \in G \) \( b \in G \)

- **Shows:** \( F( r\{a\}, r\{b\} ) = r\{ a \cdot b \} \)

**Proof:**

1. From \( A1 \ A2 \ A3 \ A4 \) show thesis
   - Using `EquivClass_1_L10` by simp

**QED**

386
The class of the inverse is a right inverse of the class.

**Lemma (in group0) GroupZF_2_2_L3:**

assumes $A1: \equiv(G,r)$ and $A2: \text{Congruent2}(r,P)$
and $A3: F = \text{ProjFun2}(G,r,P)$
and $A4: a \in G$

shows $F(r\{a\}, r\{a^{-1}\}) = \text{TheNeutralElement}(G//r, F)$

**Proof** -

from $A1$ $A2$ $A3$ $A4$ have

$F(r\{a\}, r\{a^{-1}\}) = r\{1\}$

using $\text{inverse_in_group}$ GroupZF_2_2_L2 group0_2_L6

by simp

with $\text{groupAssum}$ $A1$ $A2$ $A3$ show thesis

using $\text{IsAgroup_def}$ GroupZF_2_2_L1 by simp

**QED**

The group structure can be projected to the quotient space.

**Theorem (in group0) GroupZF_3_T2:**

assumes $A1: \equiv(G,r)$ and $A2: \text{Congruent2}(r,P)$

shows $\text{IsAgroup}(G//r, \text{ProjFun2}(G,r,P))$

**Proof** -

let $F = \text{ProjFun2}(G,r,P)$

let $E = \text{TheNeutralElement}(G//r, F)$

from $\text{groupAssum}$ $A1$ $A2$ have $\text{IsAmonoid}(G//r,F)$

using $\text{IsAgroup_def}$ monoid0_def monoid0.GroupZF_2_2_T1

by simp

moreover have

$\forall c \in G//r. \exists b \in G//r. F(c,b) = E$

**Proof**

fix $c$ assume $A3: c \in G//r$

then obtain $g$ where $D1: g \in G \quad c = r\{g\}$

using $\text{quotient_def}$ by auto

let $b = r\{g^{-1}\}$

from $D1$ have $b \in G//r$

using $\text{inverse_in_group}$ quotientI

by simp

moreover from $A1$ $A2$ $D1$ have

$F(c,b) = E$

using GroupZF_2_2_L3 by simp

ultimately show $\exists b \in G//r. F(c,b) = E$

by auto

**QED**

ultimately show thesis

using $\text{IsAgroup_def}$ by simp

**QED**

The group inverse (in the projected group) of a class is the class of the inverse.

**Lemma (in group0) GroupZF_2_2_L4:**

387
assumes A1: \text{equiv}(G,r) \text{ and } 
A2: \text{Congruent}_2(r,P) \text{ and } 
A3: F = \text{ProjFun}_2(G,r,P) \text{ and } 
A4: a \in G 
shows r\{a^{-1}\} = \text{GroupInv}(G/r,F)(r\{a\}) 

proof - 
from A1 A2 A3 have group0(G/r,F) 
using Group_ZF_3_T2 group0_def by simp 
moreover from A4 have 
r\{a\} \in G/r \ r\{a^{-1}\} \in G/r 
using inverse_in_group quotientI by auto 
moreover from A1 A2 A3 A4 have 
F\langle r\{a\},r\{a^{-1}\}\rangle = \text{TheNeutralElement}(G/r,F) 
using Group_ZF_2_2_L3 by simp 
ultimately show thesis 
by (rule group0.group0_2_L9) 

35.3 Normal subgroups and quotient groups 

If $H$ is a subgroup of $G$, then for every $a \in G$ we can consider the sets 
\{a \cdot h.h \in H\} and \{h \cdot a.h \in H\} (called a left and right "coset of $H", \text{resp.}) 
These sets sometimes form a group, called the "quotient group". This section 
discusses the notion of quotient groups. 

A normal subgroup $N$ of a group $G$ is such that $aba^{-1}$ belongs to $N$ if 
a \in G, b \in N. 

definition 
\text{IsAnormalSubgroup}(G,P,N) \equiv \text{IsAsubgroup}(N,P) \land 
(\forall n \in N. \forall g \in G. P( P( g,n ),\text{GroupInv}(G,P)(g) ) \in N) 

Having a group and a normal subgroup $N$ we can create another group 
consisting of equivalence classes of the relation $a \sim b \equiv a \cdot b^{-1} \in N$. We 
will refer to this relation as the quotient group relation. The classes of this 
relation are in fact cosets of subgroup $H$. 

definition 
\text{QuotientGroupRel}(G,P,H) \equiv 
\{ (a,b) \in G \times G. P( a, \text{GroupInv}(G,P)(b) ) \in H\} 

Next we define the operation in the quotient group as the projection of the 
group operation on the classes of the quotient group relation. 

definition 
\text{QuotientGroupOp}(G,P,H) \equiv 
\text{ProjFun}_2(G,\text{QuotientGroupRel}(G,P,H),P) 

Definition of a normal subgroup in a more readable notation. 

lemma (in group0) Group_ZF_2_4_L0: 
assumes IsAnormalSubgroup(G,P,H)
and \( g \in G \) \( n \in H \)
shows \( g \cdot n \cdot g^{-1} \in H \)
using assms IsAnormalSubgroup_def by simp

The quotient group relation is reflexive.

**lemma (in group0) Group_ZF_2_4_L1:**
assumes \( \text{IsAsubgroup}(H,P) \)
shows \( \text{refl}(G, \text{QuotientGroupRel}(G,P,H)) \)
using assms group0_2_L6 group0_3_L5
  QuotientGroupRel_def refl_def by simp

The quotient group relation is symmetric.

**lemma (in group0) Group_ZF_2_4_L2:**
assumes \( A1: \text{IsAsubgroup}(H,P) \)
shows \( \text{sym}(\text{QuotientGroupRel}(G,P,H)) \)
proof -
{  
  fix \( a \) \( b \) assume \( A2: (a,b) \in \text{QuotientGroupRel}(G,P,H) \)
  with \( A1 \) have \( (a \cdot b^{-1})^{-1} \in H \)
    using QuotientGroupRel_def group0_3_T3A
    by simp
  moreover from \( A2 \) have \( (a \cdot b^{-1})^{-1} = b \cdot a^{-1} \)
    using QuotientGroupRel_def group0_2_L12
    by simp
  ultimately have \( b \cdot a^{-1} \in H \) by simp
  with \( A2 \) have \( (b,a) \in \text{QuotientGroupRel}(G,P,H) \)
    using QuotientGroupRel_def by simp
}
then show thesis using symI by simp
qed

The quotient group relation is transitive.

**lemma (in group0) Group_ZF_2_4_L3A:**
assumes \( A1: \text{IsAsubgroup}(H,P) \) and
\( A2: (a,b) \in \text{QuotientGroupRel}(G,P,H) \) and
\( A3: (b,c) \in \text{QuotientGroupRel}(G,P,H) \)
shows \( (a,c) \in \text{QuotientGroupRel}(G,P,H) \)
proof -
let \( r = \text{QuotientGroupRel}(G,P,H) \)
from \( A2 \) \( A3 \) have \( T1: a \in G \) \( b \in G \) \( c \in G \)
  using QuotientGroupRel_def by auto
from \( A1 \) \( A2 \) \( A3 \) have \( (a \cdot b^{-1}) \cdot (b \cdot c^{-1}) \in H \)
  using QuotientGroupRel_def group0_3_L6
  by simp
moreover from \( T1 \) have \( a \cdot c^{-1} = (a \cdot b^{-1}) \cdot (b \cdot c^{-1}) \)
  using group0_2_L14A by blast
ultimately have \( a \cdot c^{-1} \in H \)
  by simp
with T1 show thesis using QuotientGroupRel_def
  by simp
qed

The quotient group relation is an equivalence relation. Note we do not need
the subgroup to be normal for this to be true.

lemma (in group0) Group_ZF_2_4_L3: assumes A1: IsAsubgroup(H,P)
  shows equiv(G,QuotientGroupRel(G,P,H))
proof -
  let r = QuotientGroupRel(G,P,H)
  from A1 have
    ∀ a b c. ⟨⟨a, b⟩, ⟨b, c⟩⟩ ∈ r → ⟨a, c⟩ ∈ r
  using Group_ZF_2_4_L3A by blast
  then have trans(r)
    using Fol1_L2 by blast
  with A1 show thesis
    using Group_ZF_2_4_L1 Group_ZF_2_4_L2
    QuotientGroupRel_def equiv_def
    by auto
qed

The next lemma states the essential condition for congruency of the group
operation with respect to the quotient group relation.

lemma (in group0) Group_ZF_2_4_L4:
  assumes A1: IsAnormalSubgroup(G,P,H)
  and A2: ⟨⟨a1,a2⟩⟩ ∈ QuotientGroupRel(G,P,H)
  and A3: ⟨⟨b1,b2⟩⟩ ∈ QuotientGroupRel(G,P,H)
  shows ⟨⟨a1·b1, a2·b2⟩⟩ ∈ QuotientGroupRel(G,P,H)
proof -
  from A2 A3 have T1:
    a1∈G a2∈G b1∈G b2∈G
    a1·b1 ∈ G a2·b2 ∈ G
    b1·b2⁻¹ ∈ H a1·a2⁻¹ ∈ H
  using QuotientGroupRel_def group0_2_L1 monoid0.group0_1_L1
  by auto
  with A1 show thesis using
    IsAnormalSubgroup_def group0_3_L6 group0_2_L15
    QuotientGroupRel_def by simp
qed

If the subgroup is normal, the group operation is congruent with respect to
the quotient group relation.

lemma Group_ZF_2_4_L5A:
  assumes IsAgroup(G,P)
  and IsAnormalSubgroup(G,P,H)
  shows Congruent2(QuotientGroupRel(G,P,H),P)
  using assms group0_def group0.Group_ZF_2_4_L4 Congruent2_def
  by simp

390
The quotient group is indeed a group.

**Theorem** Group\_ZF\_2\_4\_T1:

assumes IsAgroup(G,P) and IsAnormalSubgroup(G,P,H)

shows IsAgroup(G//QuotientGroupRel(G,P,H),QuotientGroupOp(G,P,H))

using assms group0_def group0.Group\_ZF\_2\_4\_L3 IsAnormalSubgroup_def

Group\_ZF\_2\_4\_L5A group0.Group\_ZF\_3\_T2 QuotientGroupOp_def

by simp

The class (coset) of the neutral element is the neutral element of the quotient group.

**Lemma** Group\_ZF\_2\_4\_L5B:

assumes IsAgroup(G,P) and IsAnormalSubgroup(G,P,H)

and r = QuotientGroupRel(G,P,H)

and e = TheNeutralElement(G,P)

shows r\{e\} = TheNeutralElement(G//r,QuotientGroupOp(G,P,H))

using assms IsAnormalSubgroup_def group0_def

IsAgroup_def group0.Group\_ZF\_2\_4\_L3 Group\_ZF\_2\_4\_L5A

QuotientGroupOp_def Group\_ZF\_2\_2\_L1

by simp

A group element is equivalent to the neutral element iff it is in the subgroup we divide the group by.

**Lemma** (in group0) Group\_ZF\_2\_4\_L5C: assumes a\in G

shows ⟨a,1⟩ ∈ QuotientGroupRel(G,P,H) iff a\in H

using assms QuotientGroupRel_def group_inv_of_one group0_2_L2

by auto

A group element is in H iff its class is the neutral element of G/H.

**Lemma** (in group0) Group\_ZF\_2\_4\_L5D:

assumes A1: IsAnormalSubgroup(G,P,H) and

A2: a\in G and

A3: r = QuotientGroupRel(G,P,H) and

A4: TheNeutralElement(G//r,QuotientGroupOp(G,P,H)) = e

shows r\{a\} = e iff ⟨a,1⟩ \in r

proof

assume r\{a\} = e

with groupAssum assms have

r\{1\} = r\{a\} and I: equiv(G,r)

using Group\_ZF\_2\_4\_L5B IsAnormalSubgroup_def Group\_ZF\_2\_4\_L3

by auto

with A2 have ⟨1,a⟩ ∈ r using eq_equiv_class

by simp

with I show ⟨a,1⟩ ∈ r using equiv_is_sym

next assume ⟨a,1⟩ ∈ r

moreover from A1 A3 have equiv(G,r)

using IsAnormalSubgroup_def Group\_ZF\_2\_4\_L3

by simp

391
ultimately have $r(a) = r(1)$
  using equiv_class_eq by simp
with groupAssum A1 A3 A4 show $r(a) = e$
  using Group_ZF_2_4_L5E by simp
qed

The class of $a \in G$ is the neutral element of the quotient $G/H$ iff $a \in H$.

**lemma (in group0) Group_ZF_2_4_L5E:**

assumes IsAnormalSubgroup(G,P,H) and
a∈G and $r = \text{QuotientGroupRel}(G,P,H)$ and
TheNeutralElement(G//r,QuotientGroupOp(G,P,H)) = e
shows $r\{a\} = e \longleftrightarrow a \in H$
  using assms Group_ZF_2_4_L5C Group_ZF_2_4_L5D
by simp

Essential condition to show that every subgroup of an abelian group is normal.

**lemma (in group0) Group_ZF_2_4_L5:**

assumes A1: $P$ {is commutative on} $G$
and A2: IsAsubgroup(H,P)
and A3: $g \in G$ $h \in H$
shows $g \cdot h \cdot g^{-1} \in H$
proof -
  from A2 A3 have T1:$h \in G$ $g^{-1} \in G$
    using group0_3_L2 inverse_in_group by auto
  with A3 A1 have $g \cdot h \cdot g^{-1} = g^{-1} \cdot g \cdot h$
    using group0_4_L4A by simp
  with A3 T1 show thesis using
    group0_2_L6 group0_2_L2 by simp
qed

Every subgroup of an abelian group is normal. Moreover, the quotient group is also abelian.

**lemma Group_ZF_2_4_L6:**

assumes A1: IsAgroup(G,P)
and A2: $P$ {is commutative on} $G$
and A3: IsAsubgroup(H,P)
shows IsAnormalSubgroup(G,P,H)
QuotientGroupOp(G,P,H) {is commutative on} (G//QuotientGroupRel(G,P,H))
proof -
  from A1 A2 A3 show T1: IsAnormalSubgroup(G,P,H) using
    group0_def IsAnormalSubgroup_def group0.Group_ZF_2_4_L5
    by simp
  let $r = \text{QuotientGroupRel}(G,P,H)$
  from A1 A3 T1 have equiv(G,r) Congruent2(r,P)
    using group0_def group0.Group_ZF_2_4_L3 Group_ZF_2_4_L5A
    by auto

392
with A2 show
  QuotientGroupOp(G,P,H) {is commutative on} (G//QuotientGroupRel(G,P,H))
  using EquivClass_2_T1 QuotientGroupOp_def
  by simp

qed

The group inverse (in the quotient group) of a class (coset) is the class of
the inverse.

lemma (in group0) Group_ZF_2_4_L7:
  assumes IsAnormalSubgroup(G,P,H)
  and a∈G and r = QuotientGroupRel(G,P,H)
  and F = QuotientGroupOp(G,P,H)
  shows r{a⁻¹} = GroupInv(G//r,F)(r{a})
  using groupAssum assms IsAnormalSubgroup_def Group_ZF_2_4_L3
  Group_ZF_2_4_L5A QuotientGroupOp_def Group_ZF_2_2_L4
  by simp

35.4 Function spaces as monoids

On every space of functions \( \{ f : X \rightarrow X \} \) we can define a natural monoid
structure with composition as the operation. This section explores this fact.

The next lemma states that composition has a neutral element, namely the
identity function on \( X \) (the one that maps \( x \in X \) into itself).

lemma Group_ZF_2_5_L1: assumes A1: F = Composition(X)
  shows \( \exists I \in (X \rightarrow X). \forall f \in (X \rightarrow X). F( I,f) = f \land F( f,I) = f \)
proof-
  let I = id(X)
  from A1 have \( I \in X \rightarrow X \land (\forall f \in (X \rightarrow X). F( I,f) = f \land F( f,I) = f) \)
    using id_type func_ZF_6_L1A by simp
  thus thesis by auto
qed

The space of functions that map a set \( X \) into itself is a monoid with com-
position as operation and the identity function as the neutral element.

lemma Group_ZF_2_5_L2: shows
  IsAmonoid(X→X,Composition(X))
  id(X) = TheNeutralElement(X→X,Composition(X))
proof -
  let I = id(X)
  let F = Composition(X)
  show IsAmonoid(X→X,Composition(X))
    using func_ZF_5_L5 Group_ZF_2_5_L1 IsAmonoid_def
    by auto
  then have monoid0(X→X,F)
    using monoid0_def by simp
  moreover have

393
\[ I \in X \rightarrow X \land (\forall f \in (X \rightarrow X). F(I,f) = f \land F(f,I) = f) \]

using \texttt{id_type funcZF.6.L1A} by \texttt{simp}

ultimately show \( I = \text{TheNeutralElement}(X \rightarrow X, F) \)

using \texttt{monoid0.group0_1_L4} by auto

qed

end

36 Groups 3

theory \texttt{GroupZF.3} imports \texttt{GroupZF.2 Finite1}

begin

In this theory we consider notions in group theory that are useful for the construction of real numbers in the \texttt{RealZF.x} series of theories.

36.1 Group valued finite range functions

In this section show that the group valued functions \( f : X \rightarrow G \), with the property that \( f(X) \) is a finite subset of \( G \), is a group. Such functions play an important role in the construction of real numbers in the \texttt{RealZF} series.

The following proves the essential condition to show that the set of finite range functions is closed with respect to the lifted group operation.

lemma (in group0) \texttt{GroupZF.3_1_L1}:
assumes \( A1: F = \text{P (lifted to function space over)} X \)
and \( A2: s \in \text{FinRangeFunctions}(X,G) \ r \in \text{FinRangeFunctions}(X,G) \)
shows \( F(s,r) \in \text{FinRangeFunctions}(X,G) \)

proof -
  let \( q = F(s,r) \)
  from \( A2 \) have \( \{s(x). x \in X\} \in \text{Fin}(G) \)
    \( \{r(x). x \in X\} \in \text{Fin}(G) \)
  using \texttt{Finite1.L18} by auto
  with \( A1 \) have \( q : X \rightarrow G \)
    using \texttt{group0_2.L1 monoid0.GroupZF.2_1.L0} by simp
  moreover have \( q(X) \in \text{Fin}(G) \)
  proof -
    from \( A2 \) have \( \{s(x). x \in X\} \in \text{Fin}(G) \)
      \( \{r(x). x \in X\} \in \text{Fin}(G) \)
    using \texttt{Finite1_L18} by auto
    with \( A1 \ T1 \ T2 \) show thesis using
      \texttt{group_oper_fun Finite1.L15 GroupZF.2_1.L3 func_imagedef} by simp
  qed

ultimately show thesis using \texttt{FinRangeFunctions_def}
The set of group valued finite range functions is closed with respect to the lifted group operation.

**Lemma (in group0) Group_ZF_3_1_L2:**
assumes \( A1: F = P \text{ lifted to function space over } X \)
shows \( \text{FinRangeFunctions}(X,G) \text{ is closed under } F \)

**Proof:**
- let \( A = \text{FinRangeFunctions}(X,G) \)
  from \( A1 \) have \( \forall x \in A. \forall y \in A. F(x,y) \in A \)
  using Group_ZF_3_1_L1 by simp
then show thesis using IsOpClosed_def by simp
qed

A composition of a finite range function with the group inverse is a finite range function.

**Lemma (in group0) Group_ZF_3_1_L3:**
assumes \( A1: s \in \text{FinRangeFunctions}(X,G) \)
shows \( \text{GroupInv}(G,P) \circ s \in \text{FinRangeFunctions}(X,G) \)

using groupAssum assms group0_2_T2 Finite1_L20 by simp

The set of finite range functions is a subgroup of the lifted group.

**Theorem Group_ZF_3_1_T1:**
assumes \( A1: \text{IsAgroup}(G,P) \)
and \( A2: F = P \text{ lifted to function space over } X \)
and \( A3: X \neq 0 \)
shows \( \text{IsAsubgroup}(\text{FinRangeFunctions}(X,G),F) \)

**Proof:**
- let \( e = \text{TheNeutralElement}(G,P) \)
- let \( S = \text{FinRangeFunctions}(X,G) \)
from \( A1 \) have \( \text{T1: group0}(G,P) \) using group0_def
  by simp
with \( A1 \) \( A2 \) have \( \text{T2: group0}(X \rightarrow G,F) \)
  using group0.Group_ZF_2_1_T2 group0_def
  by simp
moreover have \( S \neq 0 \)
proof -
  from \( \text{T1} \) \( A3 \) have
    \( \text{ConstantFunction}(X,e) \in S \)
    using group0.group0_2_L1 monoid0.unit_is_neutral
    Finite1_L17 by simp
  thus thesis by auto
qed
moreover have \( S \subseteq X \rightarrow G \)
  using FinRangeFunctions_def by auto
moreover from \( A2 \) \( \text{T1} \) have
  \( S \text{ is closed under } F \)
using group0.Group_ZF_3_1_L2
by simp
moreover from A1 A2 T1 have
\( \forall s \in S. \text{GroupInv}(X\to G,F)(s) \in S \)
using FinRangeFunctions_def group0.Group_ZF_2_1_L6
group0.Group_ZF_3_1_L3 by simp
ultimately show thesis
using group0.group0_3_T3 by simp
qed

36.2 Almost homomorphisms

An almost homomorphism is a group valued function defined on a monoid
\( M \) with the property that the set \( \{ f(m+n) - f(m) - f(n) \}_{m,n \in M} \) is finite.
This term is used by R. D. Arthan in "The Eudoxus Real Numbers". We
use this term in the general group context and use the A’Campo’s term
"slopes" (see his "A natural construction for the real numbers") to mean
an almost homomorphism mapping integers into themselves. We consider
almost homomorphisms because we use slopes to define real numbers in the
Real_ZF_x series.

\( \text{HomDiff} \) is an acronym for "homomorphism difference". This is the expression
\( s(mn)(s(m)s(n))^{-1} \), or \( s(m+n) - s(m) - s(n) \) in the additive notation.
It is equal to the neutral element of the group if \( s \) is a homomorphism.

definition
\( \text{HomDiff}(G,f,s,x) \equiv \)
f\( \langle s(s(\langle \text{fst}(x),\text{snd}(x) \rangle) ,
\langle \text{GroupInv}(G,F)(s(\langle \text{fst}(x),\text{snd}(x) \rangle) ,\text{fst}(x) ,\text{snd}(x) \rangle) \rangle) \rangle \)

Almost homomorphisms are defined as those maps \( s : G \to G \) such that the
homomorphism difference takes only finite number of values on \( G \times G \).

definition
\( \text{AlmostHoms}(G,f) \equiv \)
\( \{ s \in G\to G. \langle \text{HomDiff}(G,f,s,x) . x \in G\times G \} \in \text{Fin}(G) \} \)

\( \text{AlHomOp1}(G,f) \) is the group operation on almost homomorphisms defined
in a natural way by \( (s \cdot r)(n) = s(n) \cdot r(n) \). In the terminology defined in
func1.thy this is the group operation \( f \) (on \( G \)) lifted to the function space
\( G \to G \) and restricted to the set \( \text{AlmostHoms}(G,f) \).

definition
\( \text{AlHomOp1}(G,f) \equiv \)
restrict(f \( \{ \text{lifted to function space over} \} \ G,\)
\( \text{AlmostHoms}(G,f)\times\text{AlmostHoms}(G,f) \))

We also define a composition (binary) operator on almost homomorphisms
in a natural way. We call that operator \( \text{AlHomOp2} \) - the second operation on
almost homomorphisms. Composition of almost homomorphisms is used to define multiplication of real numbers in \textit{RealZF} series.

\textbf{Definition}

\[ \text{AlHomOp2}(G,f) \equiv \text{restrict}(\text{Composition}(G), \text{AlmostHoms}(G,f) \times \text{AlmostHoms}(G,f)) \]

This lemma provides more readable notation for the \textit{HomDiff} definition. Not really intended to be used in proofs, but just to see the definition in the notation defined in the group0 locale.

\textbf{Lemma (in group0) HomDiff_notation:}

\begin{align*}
\text{HomDiff}(G,P,s,(m,n)) &= s(m \cdot n) \cdot (s(m) \cdot s(n))^{-1} \\
\text{using} \quad \text{HomDiff_def by simp} 
\end{align*}

The next lemma shows the set from the definition of almost homomorphism in a different form.

\textbf{Lemma (in group0) GroupZF3_2_L1A:}

\[ \{\text{HomDiff}(G,P,s,x) : x \in G \times G \} = \{ s(m \cdot n) \cdot (s(m) \cdot s(n))^{-1} : (m,n) \in G \times G \} \]

\text{proof -}

\begin{align*}
\text{have } \forall m \in G. \forall n \in G. \text{HomDiff}(G,P,s,(m,n)) &= s(m \cdot n) \cdot (s(m) \cdot s(n))^{-1} \\
\text{using } \text{HomDiff_notation by simp} \\
\text{then show thesis by (rule ZF1_1_L4A)}
\end{align*}

\text{qed}

Let’s define some notation. We inherit the notation and assumptions from the group0 context (locale) and add some. We will use AH to denote the set of almost homomorphisms. \(\sim\) is the inverse (negative if the group is the group of integers) of almost homomorphisms, \((\sim p)(n) = p(n)^{-1}\). \(\delta\) will denote the homomorphism difference specific for the group \((\text{HomDiff}(G,f))\). The notation \(s \approx r\) will mean that \(s, r\) are almost equal, that is they are in the equivalence relation defined by the group of finite range functions (that is a normal subgroup of almost homomorphisms, if the group is abelian). We show that this is equivalent to the set \(\{s(n) \cdot r(n)^{-1} : n \in G\}\) being finite. We also add an assumption that the \(G\) is abelian as many needed properties do not hold without that.

\textbf{locale group1 = group0 +}

\begin{align*}
\text{assumes } \text{isAbelian: P \{is commutative on\} G} \\
\text{fixes } AH \\
\text{defines } AH_def [simp]: AH \equiv \text{AlmostHoms}(G,P) \\
\text{fixes Op1} \\
\text{defines Op1_def [simp]: Op1 \equiv AlHomOp1(G,P)} \\
\text{fixes Op2} \\
\text{defines Op2_def [simp]: Op2 \equiv AlHomOp2(G,P)}
\end{align*}
fixes FR
defines FR_def [simp]: FR ≡ FinRangeFunctions(G,G)

fixes neg (\sim [90] [91])
defines neg_def [simp]: \sim s ≡ GroupInv(G,P) O s

fixes δ
defines δ_def [simp]: δ(s,x) ≡ HomDiff(G,P,s,x)

fixes AHprod (infix · 69)
defines AHprod_def [simp]: s · r ≡ AlHomOp1(G,P) ⟨s,r⟩

fixes AHcomp (infix ◦ 70)
defines AHcomp_def [simp]: s ◦ r ≡ AlHomOp2(G,P)⟨s,r⟩

fixes AlEq (infix ≃ 68)
defines AlEq_def [simp]: s ≃ r ≡ ⟨s,r⟩ ∈ QuotientGroupRel(AH,Op1,FR)

HomDiff is a homomorphism on the lifted group structure.

lemma (in group1) Group_ZF_3_2_L1:
  assumes A1: s:G→G r:G→G
  and A2: x ∈ G×G
  and A3: F = P {lifted to function space over} G
  shows δ(F⟨s,r⟩,x) = δ(s,x) · δ(r,x)
proof -
  let p = F⟨s,r⟩
  from A2 obtain m n where
    D1: x = ⟨m,n⟩ m∈G n∈G
    by auto
  then have T1:m·n ∈ G
    using group0_2_L1 monoid0.group0_1_L1 by simp
  with A1 D1 have T2:
    s(m)∈G s(n)∈G r(m)∈G
    r(n)∈G s(m·n)∈G r(m·n)∈G
    using apply_funtype by auto
  from A3 A1 have T3:p : G→G
    using group0_2_L1 monoid0.Group_ZF_2_1_L0 by simp
  from D1 T3 have
    δ(p,x) = p(m·n)·((p(n))⁻¹·(p(m))⁻¹)
    using HomDiff_notation apply_funtype group_inv_of_two
    by simp
  also from A3 A1 D1 T1 isAbelian T2 have...
    δ(s,x) · δ(r,x)
    using Group_ZF_2_1_L3 group0_4_L3 HomDiff_notation
    by simp
  finally show thesis by simp
qed

The group operation lifted to the function space over G preserves almost
homomorphisms.

**lemma (in group1) Group_ZF_3_2_L2:** assumes A1: \( s \in AH \) \( r \in AH \) and A2: \( F = P \text{ (lifted to function space over) } G \) shows \( F( s, r) \in AH \)

**proof** -
- let \( p = F( s, r) \)
- from A1 A2 have \( p : G \to G \)
  - using \( \text{AlmostHoms_def group0_2_L1 monoid0.Group_ZF_2_1_L0} \)
    - by simp
- moreover have \( \{ \delta(p,x). x \in G \times G \} \in \text{Fin}(G) \)
  - proof -
    - from A1 have \( \{ \delta(s,x). x \in G \times G \} \in \text{Fin}(G) \)
      - \( \{ \delta(r,x). x \in G \times G \} \in \text{Fin}(G) \)
      - using \( \text{AlmostHoms_def} \) by auto
    - with \( \text{groupAssum A1 A2 show thesis} \)
      - using \( \text{IsAgroup_def IsAmonoid_def IsAssociative_def} \)
        - \( \text{Finitel_L15 AlmostHoms_def Group_ZF_3_2_L1} \)
          - by auto
    - qed
  - ultimately show thesis using \( \text{AlmostHoms_def} \)
    - by simp
  - qed

The set of almost homomorphisms is closed under the lifted group operation.

**lemma (in group1) Group_ZF_3_2_L3:**
- assumes \( F = P \text{ (lifted to function space over) } G \)
  - shows \( AH \text{ (is closed under) } F \)
- using \( \text{assms IsOpClosed_def Group_ZF_3_2_L2} \) by simp

The terms in the homomorphism difference for a function are in the group.

**lemma (in group1) Group_ZF_3_2_L4:**
- assumes \( s:G \to G \) and \( m \in G \) \( n \in G \)
  - shows
    - \( m \cdot n \in G \)
    - \( s(m \cdot n) \in G \)
    - \( s(m) \in G \) \( s(n) \in G \)
    - \( \delta(s,(m,n)) \in G \)
    - \( s(m) \cdot s(n) \in G \)
- using \( \text{assms group_op_closed inverse_in_group} \)
  - apply \( \text{funtyle HomDiff_def} \) by auto

It is handy to have a version of \( \text{Group_ZF_3_2_L4} \) specifically for almost homomorphisms.

**corollary (in group1) Group_ZF_3_2_L4A:**
- assumes \( s \in AH \) and \( m \in G \) \( n \in G \)
  - shows \( m \cdot n \in G \)
\[ s(m \cdot n) \in G \]
\[ s(m) \in G \quad s(n) \in G \]
\[ \delta(s, (m, n)) \in G \]
\[ s(m) \cdot s(n) \in G \]

Using assms AlmostHoms_def Group_ZF_3_2_L4 by auto

The terms in the homomorphism difference are in the group, a different form.

**Lemma (in group1) Group_ZF_3_2_L4B**:

**Assumes** A1: \( s \in AH \) and A2: \( x \in G \times G \)

**Shows** \( \text{fst}(x) \cdot \text{snd}(x) \in G \)
\[ s(\text{fst}(x) \cdot \text{snd}(x)) \in G \]
\[ s(\text{fst}(x)) \in G \quad s(\text{snd}(x)) \in G \]
\[ \delta(s, x) \in G \]
\[ s(\text{fst}(x)) \cdot s(\text{snd}(x)) \in G \]

**Proof** -

let \( m = \text{fst}(x) \)
let \( n = \text{snd}(x) \)
from A1 A2 show
\[ m \cdot n \in G \quad s(m \cdot n) \in G \]
\[ s(m) \in G \quad s(n) \in G \]
\[ s(m) \cdot s(n) \in G \]
using Group_ZF_3_2_L4A by auto
from A1 A2 have \( \delta(s, (m, n)) \in G \) using Group_ZF_3_2_L4A by simp
moreover from A2 have \( (m, n) = x \) by auto
ultimately show \( \delta(s, x) \in G \) by simp

qed

What are the values of the inverse of an almost homomorphism?

**Lemma (in group1) Group_ZF_3_2_L5**:

**Assumes** \( s \in AH \) and \( n \in G \)

**Shows** \( (\sim s)(n) = (s(n))^{-1} \)

using assms AlmostHoms_def comp_fun_apply by auto

Homomorphism difference commutes with the inverse for almost homomorphisms.

**Lemma (in group1) Group_ZF_3_2_L6**:

**Assumes** A1: \( s \in AH \) and A2: \( x \in G \times G \)

**Shows** \( \delta((\sim s, x) = (\delta(s, x))^{-1} \)

**Proof** -

let \( m = \text{fst}(x) \)
let \( n = \text{snd}(x) \)
have \( \delta((\sim s, x) = (\sim s)(m \cdot n) \cdot ((\sim s)(m) \cdot (\sim s)(n))^{-1} \)
using HomDiff_def by simp
from A1 A2 isAbelian show thesis
The inverse of an almost homomorphism maps the group into itself.

**Lemma (in group1) Group_ZF_3_2_L7:**

assumes $s \in AH$

shows $\sim s : G \to G$

using groupAssum assms AlmostHoms_def group0_2_T2 comp_fun by auto

The inverse of an almost homomorphism is an almost homomorphism.

**Lemma (in group1) Group_ZF_3_2_L8:**

assumes $A1: F = P \langle\text{lifted to function space over}\rangle G$ and $A2: s \in AH$

shows $\text{GroupInv}(G\to G,F)(s) \in AH$

**Proof:**

from $A2$ have $\{\delta(s,x). x \in G\times G\} \in \text{Fin}(G)$

using AlmostHoms_def by simp

with groupAssum have $\text{GroupInv}(G,P)\{\delta(s,x). x \in G\times G\} \in \text{Fin}(G)$

using group0_2_T2 Finite1_L6A by blast

moreover have $\text{GroupInv}(G,P)\{\delta(s,x). x \in G\times G\} = \{((\delta(s,x))^{-1}. x \in G\times G\}$

proof -

from groupAssum have $\text{GroupInv}(G,P) : G \to G$

using group0_2_T2 by simp

moreover from $A2$ have $\forall x\in G\times G. \delta(s,x)\in G$

using Group_ZF_3_2_L4B by simp

ultimately show thesis

using func1_1_L17 by simp

qed

ultimately have $\{((\delta(s,x))^{-1}. x \in G\times G\} \in \text{Fin}(G)$

by simp

moreover from $A2$ have $\{((\delta(s,x))^{-1}. x \in G\times G\} = \{\delta(\sim s,x). x \in G\times G\}$

using Group_ZF_3_2_L6 by simp

ultimately have $\{\delta(\sim s,x). x \in G\times G\} \in \text{Fin}(G)$

by simp

with $A2$ groupAssum $A1$ show thesis

using Group_ZF_3_2_L7 AlmostHoms_def Group_ZF_2_1_L6 by simp

qed

The function that assigns the neutral element everywhere is an almost homomorphism.
lemma (in group1) Group_ZF_3_2_L9: shows ConstantFunction(G,1) ∈ AH and AH≠0
proof -
  let z = ConstantFunction(G,1)
  have G×G≠0 using group0_2_L1 monoid0.group0_1_L3A
    by blast
  moreover have ∀x∈G×G. δ(z,x) = 1
    proof
      fix x assume A1:x ∈ G × G
      then obtain m n where x = ⟨ m,n⟩ m∈G n∈G
        by auto
      then show δ(z,x) = 1
        using group0_2_L1 monoid0.group0_1_L1
          func1_3_L2 HomDiff_def group0_2_L2
          group_inv_of_one
        by simp
    qed
  ultimately have \{δ(z,x). x ∈ G×G\} = \{1\} by (rule ZF1_1_L5)
  then show z ∈ AH using group0_2_L2 Finite1_L16
    func1_3_L1 group0_2_L2 AlmostHoms_def by simp
  then show AH≠0 by auto
qed

If the group is abelian, then almost homomorphisms form a subgroup of the lifted group.

lemma Group_ZF_3_2_L10: assumes A1: IsAgroup(G,P)
  and A2: P {is commutative on} G
  and A3: F = P {lifted to function space over} G
shows IsAsubgroup(AlmostHoms(G,P),F)
proof -
  let AH = AlmostHoms(G,P)
  from A2 A1 have T1: group1(G,P)
    using group1_axioms.intro group0_def group1_def
    by simp
  from A1 A3 have group0(G→G,F)
    using group0_def group0.Group_ZF_2_1_T2 by simp
  moreover from T1 have AH≠0
    using group1.Group_ZF_3_2_L9 by simp
  moreover have T2:AH ⊆ G→G
    using AlmostHoms_def by auto
  moreover from T1 A3 have
    AH {is closed under} F
    using group1.Group_ZF_3_2_L3 by simp
  moreover from T1 A3 have
    ∀s∈AH. GroupInv(G→G,F)(s) ∈ AH
    using group1.Group_ZF_3_2_L8 by simp
  ultimately show IsAsubgroup(AlmostHoms(G,P),F)
    using group0.group0_3_T3 by simp
qed
If the group is abelian, then almost homomorphisms form a group with the first operation, hence we can use theorems proven in group0 context applied to this group.

lemma (in group1) Group_ZF_3_2_L10A:
  shows IsAgroup(AH,Op1) group0(AH,Op1)
  using groupAssum isAbelian Group_ZF_3_2_L10 IsAsubgroup_def
  AlHomOp1_def group0_def by auto

The group of almost homomorphisms is abelian

lemma Group_ZF_3_2_L11: assumes A1: IsAgroup(G,f) and A2: f {is commutative on} G shows IsAgroup(AlmostHoms(G,f),AlHomOp1(G,f)) AlHomOp1(G,f) {is commutative on} AlmostHoms(G,f)
proof-
  let AH = AlmostHoms(G,f)
  let F = f {lifted to function space over} G
  from A1 A2 have IsAsubgroup(AH,F)
  using Group_ZF_3_2_L10 by simp
  then show IsAgroup(AH,AlHomOp1(G,f))
  using IsAsubgroup_def AlHomOp1_def by simp
  from A1 have F : (G→G)×(G→G)→(G→G)
  using IsAgroup_def monoid0_def monoid0.Group_ZF_2_1_L0A
  by simp
  moreover have AH ⊆ G→G
  using AlmostHoms_def by auto
  moreover from A1 A2 have F {is commutative on} (G→G)
  using group0_def group0.Group_ZF_2_1_L7
  by simp
  ultimately show AlHomOp1(G,f) {is commutative on} AH
  using func_ZF_4_L1 AlHomOp1_def by simp
 qed

The first operation on homomorphisms acts in a natural way on its operands.

lemma (in group1) Group_ZF_3_2_L12:
  assumes s∈AH r∈AH and n∈G
  shows (s•r)(n) = s(n)•r(n)
  using assms AlHomOp1_def restrict AlmostHoms_def Group_ZF_2_1_L0A
  by simp

What is the group inverse in the group of almost homomorphisms?

lemma (in group1) Group_ZF_3_2_L13:
  assumes A1: s∈AH
  shows GroupInv(AH,Op1)(s) = GroupInv(G,P) O s
  GroupInv(AH,Op1)(s) ∈ AH
GroupInv(G,P) O s ∈ AH

proof -

let F = P \{lifted to function space over\} G
from groupAssum isAbelian have IsAsubgroup(AH,F)
using Group_ZF_3_2_L10 by simp
with A1 show I: GroupInv(AH,Op1)(s) = GroupInv(G,P) O s
using A1HomOp1_def Group_ZF_2_1_L6A by simp
from A1 show GroupInv(AH,Op1)(s) ∈ AH
using Group_ZF_3_2_L10A group0.inverse_in_group by simp
with I show GroupInv(G,P) O s ∈ AH by simp
qed

The group inverse in the group of almost homomorphisms acts in a natural
way on its operand.

lemma (in group1) Group_ZF_3_2_L14:
assumes s ∈ AH and n ∈ G
shows (GroupInv(AH,Op1)(s))(n) = (s(n))⁻¹
using isAbelian assms Group_ZF_3_2_L13 AlmostHoms_def comp_fun_apply
by auto

The next lemma states that if s,r are almost homomorphisms, then s · r⁻¹
is also an almost homomorphism.

lemma Group_ZF_3_2_L15: assumes IsAgroup(G,f)
and f {is commutative on} G
and AH = AlmostHoms(G,f) Op1 = AlHomOp1(G,f)
and s ∈ AH r ∈ AH
shows Op1⟨s, r⟩ ∈ AH
GroupInv(AH,Op1)(r) ∈ AH
Op1⟨s, GroupInv(AH,Op1)(r)⟩ ∈ AH
using assms group0_def group1_axioms.intro group1_def
  group1.Group_ZF_3_2_L10A group0.group0_2_L1
  monoid0.group0_1_L1 group0.inverse_in_group by auto

A version of Group_ZF_3_2_L15 formulated in notation used in group1 con-
text. States that the product of almost homomorphisms is an almost homo-
morphism and the the product of an almost homomorphism with a (point-
wise) inverse of an almost homomorphism is an almost homomorphism.

corollary (in group1) Group_ZF_3_2_L16: assumes s ∈ AH r ∈ AH
shows s · r ∈ AH s · (~ r) ∈ AH
using assms isAbelian group0_def group1_axioms group1_def
Group_ZF_3_2_L15 Group_ZF_3_2_L13 by auto

36.3 The classes of almost homomorphisms

In the Real_ZF series we define real numbers as a quotient of the group of
integer almost homomorphisms by the integer finite range functions. In this
section we setup the background for that in the general group context.
Finite range functions are almost homomorphisms.

lemma (in group1) Group_ZF_3_3_L1: shows FR ⊆ AH
proof
  fix s assume A1:s ∈ FR
  then have T1:{s(n). n ∈ G} ∈ Fin(G)
  {s(fst(x)). x∈G×G} ∈ Fin(G)
  {s(snd(x)). x∈G×G} ∈ Fin(G)
  using Finite1_L18 Finite1_L6B by auto
  have {s(fst(x)·snd(x)). x ∈ G×G} ∈ Fin(G)
  proof
  - have ∀ x∈G×G. fst(x)·snd(x) ∈ G
    using group0_2_L1 monoid0.group0_1_L1 by simp
  moreover from T1 have {s(n). n ∈ G} ∈ Fin(G) by simp
  ultimately show thesis by (rule Finite1_L6B)
  qed
  moreover have
  {(s(fst(x))·s(snd(x)))⁻¹. x∈G×G} ∈ Fin(G)
  proof
  - have ∀ g∈G. g⁻¹ ∈ G using inverse_in_group
    by simp
  moreover from T1 have
  {s(fst(x))·s(snd(x)). x∈G×G} ∈ Fin(G)
  using group_oper_fun Finite1_L15 by simp
  ultimately show thesis
  by (rule Finite1_L6C)
  qed
  ultimately have {δ(s,x). x∈G×G} ∈ Fin(G)
  using HomDiff_def Finite1_L15 group_oper_fun
  by simp
  with A1 show s ∈ AH
  using FinRangeFunctions_def AlmostHoms_def
  by simp
  qed

Finite range functions valued in an abelian group form a normal subgroup
of almost homomorphisms.

lemma Group_ZF_3_3_L2: assumes A1:IsAgroup(G,f)
  and A2:f {is commutative on} G
  shows IsAsubgroup(FinRangeFunctions(G,G),AlHomOp1(G,f))
  IsANormalSubgroup(AlmostHoms(G,f),AlHomOp1(G,f),
  FinRangeFunctions(G,G))
proof
  let H1 = AlmostHoms(G,f)
  let H2 = FinRangeFunctions(G,G)
  let F = f {lifted to function space over} G
  from A1 A2 have T1:group0(G,f)
  monoid0(G,f) group1(G,f)
  using group0_def group0.group0_2_L1

405
The group of almost homomorphisms divided by the subgroup of finite range functions is an abelian group.

theorem (in group1) Group_ZF_3_3_T1:
shows IsAgroup(AH//QuotientGroupRel(AH,Op1,FR),QuotientGroupOp(AH,Op1,FR))
and QuotientGroupOp(AH,Op1,FR) {is commutative on}
(AH//QuotientGroupRel(AH,Op1,FR))
using groupAssum isAbelian Group_ZF_3_3_L2 Group_ZF_3_2_L10A Group_ZF_2_4_T1 Group_ZF_3_2_L11 Group_ZF_3_3_L2 IsAnormalSubgroup_def Group_ZF_2_4_L6 by auto

It is useful to have a direct statement that the quotient group relation is an equivalence relation for the group of AH and subgroup FR.

lemma (in group1) Group_ZF_3_3_L3: shows QuotientGroupRel(AH,Op1,FR) ⊆ AH × AH and equiv(AH,QuotientGroupRel(AH,Op1,FR))
using groupAssum isAbelian QuotientGroupRel_def Group_ZF_3_3_L2 Group_ZF_3_2_L10A group0.Group_ZF_2_4_L3 by auto

The "almost equal" relation is symmetric.

lemma (in group1) Group_ZF_3_3_L3A: assumes A1: s∼r
s shows r∼s
proof -
  let R = QuotientGroupRel(AH,Op1,FR)
  from A1 have equiv(AH,R) and ⟨s,r⟩ ∈ R
    using Group_ZF_3_3_L3 by simp_all
then have \((r,s) \in R\) by (rule equiv_is_sym)
then show \(r\equiv s\) by simp
qed

Although we have bypassed this fact when proving that group of almost homomorphisms divided by the subgroup of finite range functions is a group, it is still useful to know directly that the first group operation on \(A\)H is congruent with respect to the quotient group relation.

**lemma (in group1) Group_ZF_3_3_L4:** shows Congruent2(QuotientGroupRel(AH,Op1,FR),Op1)
using groupAssum isAbelian Group_ZF_3_2_L10A Group_ZF_3_3_L2
Group_ZF_2_4_L5A by simp

The class of an almost homomorphism \(s\) is the neutral element of the quotient group of almost homomorphisms iff \(s\) is a finite range function.

**lemma (in group1) Group_ZF_3_3_L5:** assumes \(s \in AH\) and \(r = \text{QuotientGroupRel}(AH,Op1,FR)\) and \(\text{TheNeutralElement}(AH//r,\text{QuotientGroupOp}(AH,Op1,FR)) = e\)
sows \(r(s) = e \iff s \in FR\)
using groupAssum isAbelian assms Group_ZF_3_2_L11
 group0_def Group_ZF_3_3_L2 group0.Group_ZF_2_4_L5E
by simp

The group inverse of a class of an almost homomorphism \(f\) is the class of the inverse of \(f\).

**lemma (in group1) Group_ZF_3_3_L6:** assumes \(A1: s \in AH\) and \(r = \text{QuotientGroupRel}(AH,Op1,FR)\) and \(F = \text{ProjFun2}(AH,r,Op1)\)
sows \(r\{\sim s\} = \text{GroupInv}(AH//r,F)(r(s))\)
proof -
from groupAssum isAbelian assms have \(r(\text{GroupInv}(AH, Op1)(s)) = \text{GroupInv}(AH//r,F)(r \{s\})\)
using Group_ZF_3_2_L10A Group_ZF_3_3_L2 QuotientGroupOp_def
 group0.Group_ZF_2_4_L7 by simp
with \(A1\) show thesis using Group_ZF_3_2_L13
 by simp
qed

### 36.4 Compositions of almost homomorphisms

The goal of this section is to establish some facts about composition of almost homomorphisms that are needed for the real numbers construction in Real_ZF_x series. In particular we show that the set of almost homomorphisms is closed under composition and that composition is congruent with respect to the equivalence relation defined by the group of finite range functions (a normal subgroup of almost homomorphisms).
The next formula restates the definition of the homomorphism difference to express the value an almost homomorphism on a product.

**Lemma** (in group1) GroupZF_3_4_L1:

assumes \( s \in AH \) and \( m \in G \) \( n \in G \)

shows \( isAbelian \) \( \text{assms} \) GroupZF_3_2_L4A HomDiff_def group0_4_L5

by simp

What is the value of a composition of almost homomorphisms?

**Lemma** (in group1) GroupZF_3_4_L2:

assumes \( s \in AH \) \( r \in AH \) and \( m \in G \)

shows \( (s \circ r)(m) = s(r(m)) \) \( s(r(m)) \in G \)

using \assms\ AlmostHoms_def func_ZF_5_L3 restrict AlHomOp2_def apply_funtype

by auto

What is the homomorphism difference of a composition?

**Lemma** (in group1) GroupZF_3_4_L3:

assumes \( A1: s \in AH \) \( r \in AH \) and \( A2: m \in G \) \( n \in G \)

shows \( \delta(s \circ r, \langle m,n \rangle) = \delta(s, \langle r(m),r(n) \rangle) \cdot \delta(s, \langle r(m) \cdot r(n), \delta(r, \langle m,n \rangle) \rangle) \)

proof

- from \( A1 \) \( A2 \) have \( T1: s(r(m)) \cdot s(r(n)) \in G \)

  \( \delta(s, \langle r(m),r(n) \rangle) \in G \) \( s(\delta(r, \langle m,n \rangle)) \in G \)

  using GroupZF_3_4_L2 AlmostHoms_def apply_funtype

  GroupZF_3_2_L4A group0_2_L1 monoid0.group0_1_L1

  by auto

from \( A1 \) \( A2 \) have \( \delta(s, \langle m,n \rangle) = s(r(m)) \cdot s(r(n))^{-1} \)

using HomDiff_def group0_2_L1 monoid0.group0_1_L1 GroupZF_3_4_L2

GroupZF_3_4_L1 by simp

moreover from \( A1 \) \( A2 \) have \( s(r(m) \cdot r(n)) \cdot \delta(r, \langle m,n \rangle) = s(r(m)) \cdot s(r(n)) \cdot \delta(s, \langle r(m),r(n) \rangle) \)

using GroupZF_3_2_L4A GroupZF_3_4_L1 by auto

moreover from \( T1 \) isAbelian have \( s(r(m)) \cdot s(r(n))^{-1} = s(\delta(r, \langle m,n \rangle) \cdot \delta(r, \langle r(m) \cdot r(n), \delta(r, \langle m,n \rangle) \rangle)\)

using group0_4_L6C by simp

ultimately show thesis by simp

qed

What is the homomorphism difference of a composition (another form)?

Here we split the homomorphism difference of a composition into a product
of three factors. This will help us in proving that the range of homomorphism
difference for the composition is finite, as each factor has finite range.

lemma (in group1) Group_ZF_3_4_L4:
  assumes A1: \( s \in \text{AH} \) r\( \in \text{AH} \) and A2: \( x \in G \times G \)
  and A3:
  \( A = \delta(s,(r(fst(x)),r(snd(x)))) \)
  \( B = s(\delta(r,x)) \)
  \( C = \delta(s,(r(fst(x)) \cdot r(snd(x))),\delta(r,x))) \)
  shows \( \delta(s \circ r,x) = A \cdot B \cdot C \)
proof -
  let \( m = fst(x) \)
  let \( n = snd(x) \)
  note A1
  moreover from A2 have \( m \in G \) \( n \in G \)
    by auto
  ultimately have
    \( \delta(s,\langle m,n \rangle) = \delta(s,\langle r(m),r(n) \rangle) \cdot s(\delta(r,\langle m,n \rangle)) \)
    by (rule Group_ZF_3_4_L3)
  with A1 A2 A3 show thesis
    by auto
qed

The range of the homomorphism difference of a composition of two almost
homomorphisms is finite. This is the essential condition to show that a
composition of almost homomorphisms is an almost homomorphism.

lemma (in group1) Group_ZF_3_4_L5:
  assumes A1: \( s \in \text{AH} \) \( r \in \text{AH} \)
  shows \( \{\delta(\text{Composition}(G) \langle s,r \rangle,x) \cdot x \in G \times G \} \in \text{Fin}(G) \)
proof -
  from A1 have \( \forall x \in G \times G. (r(fst(x)),r(snd(x))) \in G \times G \)
    using Group_ZF_3_2_L4B by simp
  moreover from A1 have \( \{\delta(s,x) \cdot x \in G \times G \} \in \text{Fin}(G) \)
    using AlmostHoms_def by simp
  ultimately have
    \( \{\delta(s,\langle r(fst(x)),r(snd(x)) \rangle) \cdot x \in G \times G \} \in \text{Fin}(G) \)
    by (rule Finite1_L6B)
  moreover have \( \{s(\delta(r,x)) \cdot x \in G \times G \} \in \text{Fin}(G) \)
proof -
  from A1 have \( \forall m \in G. s(m) \in G \)
    using AlmostHoms_def apply_funtype by auto
  moreover from A1 have \( \{\delta(r,x) \cdot x \in G \times G \} \in \text{Fin}(G) \)
    using AlmostHoms_def by simp
  ultimately show thesis
    by (rule Finite1_L6C)
qed
ultimately have
{\delta(s, (r(fst(x)), r(snd(x))))s(\delta(r,x)). x \in G \times G} \in \text{Fin}(G)
using \text{group_oper_fun Finite1_L15 by simp}
moreover have
{\delta(s, (r(fst(x)) \cdot r(snd(x))), \delta(r,x)). x \in G \times G} \in \text{Fin}(G)
proof -
from A1 have
\forall x \in G \times G. ( (r(fst(x)) \cdot r(snd(x))), \delta(r,x)) \in G \times G
using Group_ZF_3_2_L4B by simp
moreover from A1 have
{\delta(s,x). x \in G \times G} \in \text{Fin}(G)
using AlmostHoms_def by simp
ultimately show thesis by (rule Finite1_L6B)
qed
ultimately have
{\delta(s, (r(fst(x)), r(snd(x))))s(\delta(r,x))
\delta(s, (r(fst(x)) \cdot r(snd(x))), \delta(r,x)). x \in G \times G} \in \text{Fin}(G)
using \text{group_oper_fun Finite1_L15 by simp}
moreover from A1 have {\delta(s \circ r,x). x \in G \times G} =
{\delta(s, (r(fst(x)), r(snd(x))))s(\delta(r,x))
\delta(s, (r(fst(x)) \cdot r(snd(x))), \delta(r,x)). x \in G \times G}
using Group_ZF_3_4_L4 by simp
ultimately have {\delta(s \circ r,x). x \in G \times G} \in \text{Fin}(G) by simp
with A1 show thesis using restrict AlHomOp2_def
by simp
qed
Composition of almost homomorphisms is an almost homomorphism.

theorem (in group1) Group_ZF_3_4_T1:
assumes
A1: \( s \in \text{AH} \) \( r \in \text{AH} \)
shows Composition(G)(s,r) \in \text{AH} \ s \circ r \in \text{AH}
proof -
from A1 have \( \langle s,r \rangle \in (G \rightarrow G) \times (G \rightarrow G) \)
using AlmostHoms_def by simp
then have Composition(G)(s,r) : G\rightarrow G
using func_ZF_5_L1 apply_funtype by blast
with A1 show Composition(G)(s,r) \in \text{AH}
using Group_ZF_3_4_L5 AlmostHoms_def
by simp
with A1 show \( s \circ r \in \text{AH} \) using AlHomOp2_def restrict
by simp
qed
The set of almost homomorphisms is closed under composition. The second
operation on almost homomorphisms is associative.

lemma (in group1) Group_ZF_3_4_L6: shows
\( \text{AH \ is closed under} \) Composition(G)
\( \text{AlHomOp2}(G,P) \ \text{is associative on} \ \text{AH} \)
proof -
show $\text{AH}$ is closed under Composition($G$)
using Group_ZF_3_4_T1 IsOpClosed_def by simp
moreover have $\text{AH} \subseteq G \rightarrow G$ using AlmostHoms_def
by auto
moreover have Composition($G$) is associative on $(G \rightarrow G)$
using func_ZF_5_L5 by simp
ultimately show $\text{AlHomOp2}(G,P)$ is associative on $\text{AH}$
using func_ZF_4_L3 AlHomOp2_def by simp
qed

Type information related to the situation of two almost homomorphisms.

lemma (in group1) Group_ZF_3_4_L7:
assumes A1: $s \in \text{AH} \quad r \in \text{AH}$ and A2: $n \in G$
shows $s(n) \in G \quad (r(n))^{-1} \in G \quad s(n) \cdot (r(n))^{-1} \in G \quad s(r(n)) \in G$
proof -
from A1 A2 show $s(n) \in G \quad (r(n))^{-1} \in G \quad s(r(n)) \in G \quad s(n) \cdot (r(n))^{-1} \in G$
using AlmostHoms_def apply_type
by auto
qed

Type information related to the situation of three almost homomorphisms.

lemma (in group1) Group_ZF_3_4_L8:
assumes A1: $s \in \text{AH} \quad r \in \text{AH} \quad q \in \text{AH}$ and A2: $n \in G$
shows $q(n) \in G \quad s(r(n)) \in G \quad r(n) \cdot (q(n))^{-1} \in G \quad s(r(n)) \cdot (q(n))^{-1} \in G \quad \delta(s,(q(n),r(n) \cdot (q(n))^{-1})) \in G$
proof -
from A1 A2 show $q(n) \in G \quad s(r(n)) \in G \quad r(n) \cdot (q(n))^{-1} \in G$
using AlmostHoms_def apply_type
by auto
with A1 A2 show $s(r(n) \cdot (q(n))^{-1}) \in G \quad \delta(s,(q(n),r(n) \cdot (q(n))^{-1})) \in G$
using AlmostHoms_def apply_type Group_ZF_3_2_L4A
by auto
qed

A formula useful in showing that the composition of almost homomorphisms
is congruent with respect to the quotient group relation.

lemma (in group1) Group_ZF_3_4_L9:
assumes A1: s1 ∈ AH r1 ∈ AH s2 ∈ AH r2 ∈ AH
and A2: n∈G
shows (s1\circ r1)(n).((s2\circ r2)(n))^{-1} =
 s1(r2(n)). (s2(r2(n)))^{-1}.s1(r1(n). (r2(n))^{-1}).
\delta(s1,( r2(n),r1(n). (r2(n))^{-1}))
proof -
  from A1 A2 isAbelian have
  (s1\circ r1)(n).((s2\circ r2)(n))^{-1} =
  s1(r2(n). (r1(n). (r2(n))^{-1})).(s2(r2(n)))^{-1}
  using Group_ZF_3_4_L2 Group_ZF_3_4_L7 group0_4_L6A
  group_oper_assoc by simp
  with A1 A2 have (s1\circ r1)(n).((s2\circ r2)(n))^{-1} = s1(r2(n)).
  s1(r1(n). (r2(n))^{-1}).\delta(s1,( r2(n),r1(n). (r2(n))^{-1})).
  (s2(r2(n)))^{-1}
  using Group_ZF_3_4_L8 Group_ZF_3_4_L1 by simp
  with A1 A2 isAbelian show thesis using
  Group_ZF_3_4_L8 group0_4_L7 by simp
qed

The next lemma shows a formula that translates an expression in terms of
the first group operation on almost homomorphisms and the group inverse
in the group of almost homomorphisms to an expression using only the
underlying group operations.

lemma (in group1) Group_ZF_3_4_L10: assumes A1: s ∈ AH r ∈ AH
and A2: n ∈ G
shows (s\cdot (\text{GroupInv}(AH,Op1)(r)))(n) = s(n). (r(n))^{-1}
proof -
  from A1 A2 show thesis using isAbelian Group_ZF_3_2_L13 Group_ZF_3_2_L12 Group_ZF_3_2_L14
  by simp
qed

A neccessary condition for two a. h. to be almost equal.

lemma (in group1) Group_ZF_3_4_L11: assumes A1: s\equiv r
shows \{s(n). (r(n))^{-1}. n\in G\} ∈ Fin(G)
proof -
  from A1 have s\in AH r\in AH
    using QuotientGroupRel_def by auto
  moreover from A1 have
    \{(s\cdot (\text{GroupInv}(AH,Op1)(r)))(n). n\in G\} ∈ Fin(G)
    using QuotientGroupRel_def Finite1_L18 by simp
  ultimately show thesis using
    Group_ZF_3_4_L10 by simp
qed

A sufficient condition for two a. h. to be almost equal.
lemma (in group1) Group_ZF_3_4_L12: assumes A1: \( s \in AH \) \( r \in AH \) and A2: \( \{s(n) \cdot (r(n))^{-1}. n \in G\} \in \text{Fin}(G) \) shows \( s \sim r \)
proof -
  from groupAssum isAbelian A1 A2 show thesis
  using Group_ZF_3_2_L15 AlmostHoms_def
  Group_ZF_3_4_L10 Finite1_L19 QuotientGroupRel_def
  by simp
qed

Another sufficient condition for two a.h. to be almost equal. It is actually just an expansion of the definition of the quotient group relation.

lemma (in group1) Group_ZF_3_4_L12A: assumes s \( \in AH \) \( r \in AH \) and \( s \cdot (\text{GroupInv}(AH,Op1)(r)) \in \text{FR} \)
  shows \( s \sim r \) \( r \sim s \)
proof -
  from asms show \( s \sim r \) using asms QuotientGroupRel_def
  by simp
  then show \( r \sim s \) by (rule Group_ZF_3_3_L3A)
qed

Another necessary condition for two a.h. to be almost equal. It is actually just an expansion of the definition of the quotient group relation.

lemma (in group1) Group_ZF_3_4_L12B: assumes \( s \sim r \)
  shows \( s \cdot (\text{GroupInv}(AH,Op1)(r)) \in \text{FR} \)
  using asms QuotientGroupRel_def
  by simp

The next lemma states the essential condition for the composition of a. h. to be congruent with respect to the quotient group relation for the subgroup of finite range functions.

lemma (in group1) Group_ZF_3_4_L13: assumes A1: \( s1 \sim s2 \) \( r1 \sim r2 \)
  shows \( (s1 \circ r1) \sim (s2 \circ r2) \)
proof -
  have \( \{s1(r2(n)) \cdot (s2(r2(n)))^{-1}. n \in G\} \in \text{Fin}(G) \)
  proof -
    from A1 have \( \forall n \in G. r2(n) \in G \)
      using QuotientGroupRel_def AlmostHoms_def apply_funtype
      by auto
    moreover from A1 have \( \{s1(n) \cdot (s2(n))^{-1}. n \in G\} \in \text{Fin}(G) \)
      using Group_ZF_3_4_L11 by simp
    ultimately show thesis by (rule Finite1_L6B)
  qed
  moreover have \( \{s1(r1(n)) \cdot (r2(n))^{-1}. n \in G\} \in \text{Fin}(G) \)
  proof -
    from A1 have \( \forall n \in G. s1(n) \in G \)
      using QuotientGroupRel_def AlmostHoms_def apply_funtype
      by auto
moreover from A1 have \( \{r_1(n) \cdot (r_2(n))^{-1} \}_n \in \text{Fin}(G) \)
using Group_ZF_3_4_L11 by simp
ultimately show thesis by (rule Finite1_L6C)

qed

ultimately have
\( \{s_1(r_2(n)) \cdot (s_2(r_2(n)))^{-1} \cdot s_1(r_1(n) \cdot (r_2(n))^{-1}) \}_n \in \text{Fin}(G) \)
using group_oper_fun Finite1_L15 by simp
moreover have
\( \{\delta(s_1, r_2(n), r_1(n) \cdot (r_2(n))^{-1}) \}_n \in \text{Fin}(G) \)
proof -
from A1 have \( \forall n \in G. \ (r_2(n), r_1(n) \cdot (r_2(n))^{-1}) \in G \times G \)
using QuotientGroupRel_def Group_ZF_3_4_L7 by auto
moreover from A1 have \( \{\delta(s_1, x) \}_x \in G \times G \) \in \text{Fin}(G) 
using QuotientGroupRel_def AlmostHoms_def by simp
ultimately show thesis by (rule Finite1_L6B)

qed

ultimately have
\( \{s_1(r_2(n)) \cdot (s_2(r_2(n)))^{-1} \cdot s_1(r_1(n) \cdot (r_2(n))^{-1}) \}_n \in \text{Fin}(G) \)
using group_oper_fun Finite1_L15 by simp
with A1 show thesis using
QuotientGroupRel_def Group_ZF_3_4_L9
Group_ZF_3_4_T1 Group_ZF_3_4_L12 by simp

qed

Composition of a. h. to is congruent with respect to the quotient group relation for
the subgroup of finite range functions. Recall that if an operation say "\( \circ \)" on \( X \) is
congruent with respect to an equivalence relation \( R \) then we can define
the operation on the quotient space \( X/R \) by \( [s]_R \circ [r]_R := [s \circ r]_R \)
and this definition will be correct i.e. it will not depend on the choice of
representants for the classes \([x]\) and \([y]\). This is why we want it here.

lemma (in group1) Group_ZF_3_4_L13A: shows
\( \text{Congruent2}(\text{QuotientGroupRel}(AH,Op1,FR),Op2) \)
proof -
show thesis using Group_ZF_3_4_L13 Congruent2_def
by simp

qed

The homomorphism difference for the identity function is equal to the neutral
element of the group (denoted \( e \) in the group1 context).

lemma (in group1) Group_ZF_3_4_L14: assumes A1: \( x \in G \times G \)
shows \( \delta(id(G), x) = 1 \)
proof -
from A1 show thesis using
group0_2_L1 monoid0.group0_1_L1 HomDiff_def id_conv group0_2_L6
by simp

qed
The identity function \((I(x) = x)\) on \(G\) is an almost homomorphism.

**Lemma (in group1) Group_ZF_3_4_L15**: shows \(\text{id}(G) \in \text{AH}\)

**proof** -

- have \(G \times G \neq 0\) using group0_2_L1 groupoid0.group0_1_L3A
  by blast
- then show thesis using Group_ZF_3_4_L14 group0_2_L2
  id_type AlmostHoms_def by simp

qed

Almost homomorphisms form a monoid with composition. The identity function on the group is the neutral element there.

**Lemma (in group1) Group_ZF_3_4_L16**: shows \(\text{IsAmonoid}(\text{AH}, \text{Op2})\)

**proof** -

- let \(i = \text{TheNeutralElement}(G \rightarrow G, \text{Composition}(G))\)
  have \(\text{IsAmonoid}(G \rightarrow G, \text{Composition}(G))\)
  \(\text{monoid0}(G \rightarrow G, \text{Composition}(G))\)
  using monoid0_def Group_ZF_2_5_L2 by auto
- moreover have \(\text{AH} \{\text{is closed under}\} \text{ Composition}(G)\)
  using Group_ZF_3_4_L6 by simp
- moreover have \(\text{AH} \subseteq G \rightarrow G\)
  using AlmostHoms_def by auto
- moreover have \(i \in \text{AH}\)
  using Group_ZF_2_5_L2 Group_ZF_3_4_L15 by simp
- moreover have \(\text{id}(G) = i\)
  using Group_ZF_2_5_L2 by simp
- ultimately show \(\text{IsAmonoid}(\text{AH}, \text{Op2})\)
  \(\text{monoid0}(\text{AH}, \text{Op2})\)
  \(\text{id}(G) = \text{TheNeutralElement}(\text{AH}, \text{Op2})\)
  using monoid0.group0_1_T1 group0_1_L6 AlHomOp2_def monoid0_def
  by auto

qed

We can project the monoid of almost homomorphisms with composition to the group of almost homomorphisms divided by the subgroup of finite range functions. The class of the identity function is the neutral element of the quotient (monoid).

**Theorem (in group1) Group_ZF_3_4_T2**: assumes \(\text{A1: R} = \text{QuotientGroupRel}(\text{AH}, \text{Op1}, \text{FR})\)

**shows** \(\text{IsAmonoid}(\text{AH} / \text{R}, \text{ProjFun2}(\text{AH}, \text{R}, \text{Op2}))\)

**R{id(G)} = TheNeutralElement(AH//R,ProjFun2(AH,R,Op2))**

**proof** -
have group0(AH,Op1) using Group_ZF_3_2_L10A group0_def by simp
with A1 groupAssum isAbelian show
using Group_ZF_3_3_L2 group0.Group_ZF_2_4_L3 Group_ZF_3_4_L13A
  Group_ZF_3_4_L16 monoid0.Group_ZF_2_2_T1 Group_ZF_2_2_L1
by auto
qed

36.5 Shifting almost homomorphisms

In this this section we consider what happens if we multiply an almost
homomorphism by a group element. We show that the resulting function is
also an a. h., and almost equal to the original one. This is used only for
slopes (integer a.h.) in Int_ZF_2 where we need to correct a positive slopes
by adding a constant, so that it is at least 2 on positive integers.

If \( s \) is an almost homomorphism and \( c \) is some constant from the group,
then \( s \cdot c \) is an almost homomorphism.

**lemma** (in group1) Group_ZF_3_5_L1:

assumes A1: \( s \in AH \) and A2: \( c \in G \) and
A3: \( r = \{\langle x, s(x) \cdot c \rangle . \ x \in G\} \)

shows \( \forall x \in G. \ r(x) = s(x) \cdot c \)

using \( \text{isAbelian} \ A1 A2 \)

proof -
  from A1 A2 A3 have I: \( r : G \rightarrow G \)
  using AlmostHoms_def apply_funtype group_op_closed
  ZF_fun_from_total by auto
  with A3 show II: \( \forall x \in G. \ r(x) = s(x) \cdot c \)
  using ZF_fun_from_tot_val by simp
  with isAbelian A1 A2 have III:
  \( \forall p \in G \times G. \ \delta(r,p) = \delta(s,p) \cdot c^{-1} \)
  using group_op_closed AlmostHoms_def apply_funtype
  HomDiff_def group0_4_L7 by auto
  have \( \{\delta(r,p). \ p \in G \times G\} \in \text{Fin}(G) \)
  proof -
    from A1 A2 have \( \{\delta(s,p). \ p \in G \times G\} \in \text{Fin}(G) \)
    using AlmostHoms_def inverse_in_group by auto
    then have \( \{\delta(s,p) \cdot c^{-1}. \ p \in G \times G\} \in \text{Fin}(G) \)
    using group_op_closed group0_4_L7 by auto
    moreover from III have
    \( \{\delta(r,p). \ p \in G \times G\} = \{\delta(s,p) \cdot c^{-1}. \ p \in G \times G\} \)
    using inverse_in_group apply_funtype
    by (rule ZF1_1_L4B)

416
ultimately show thesis by simp

qed

with I show IV: \( r \in AH \) using AlmostHoms_def

by simp

from isAbelian A1 A2 I II have

\[ \forall n \in G. \ s(n) . (r(n))^{-1} = c^{-1} \]

using AlmostHoms_def apply_funtype group0_4_L6AB

by auto

then have \( \{s(n) . (r(n))^{-1}. n \in G\} = \{c^{-1}. n \in G\} \)

by (rule ZF1_1_L4B)

with A1 A2 IV show \( s \sim r \)

using group0_2_L1 monoid0.group0_1_L3A

inverse_in_group Group_ZF_3_4_L12 by simp

qed

end

37 Direct product

theory DirectProduct_ZF imports func_ZF

begin

This theory considers the direct product of binary operations. Contributed by Seo Sanghyeon.

37.1 Definition

In group theory the notion of direct product provides a natural way of creating a new group from two given groups.

Given \((G, \cdot)\) and \((H, \circ)\) a new operation \((G \times H, \times)\) is defined as \((g, h) \times (g', h') = (g \cdot g', h \circ h')\).

\[ \text{definition} \]

\[ \text{DirectProduct}(P, Q, G, H) \equiv \langle x, \langle P(fst(fst(x)),fst(snd(x))), Q(snd(fst(x)),snd(snd(x)))\rangle \rangle. \]

\[ x \in (G \times H) \times (G \times H) \]

We define a context called \text{direct0} which holds an assumption that \(P, Q\) are binary operations on \(G, H\), resp. and denotes \(R\) as the direct product of \((G, P)\) and \((H, Q)\).

locale direct0 =

fixes P Q G H

assumes Pfun: \( P : G \times G \rightarrow G \)

assumes Qfun: \( Q : H \times H \rightarrow H \)

fixes R

defines Rdef [simp]: \( R \equiv \text{DirectProduct}(P, Q, G, H) \)

417
The direct product of binary operations is a binary operation.

**Lemma (in direct0) DirectProduct_ZF_1_L1:**

shows \( R : (G \times H) \times (G \times H) \rightarrow G \times H \)

**Proof** -

from PfUn QfUn have \( \forall x \in (G \times H) \times (G \times H). \)

\( \langle P(fst(fst(x)),fst(snd(x))),Q(snd(fst(x)),snd(snd(x))) \rangle \in G \times H \)

by auto

then show thesis using ZF_fun_from_total DirectProduct_def

by simp

qed

And it has the intended value.

**Lemma (in direct0) DirectProduct_ZF_1_L2:**

shows \( \forall x \in (G \times H). \forall y \in (G \times H). R(x,y) = \langle P(fst(x),fst(y)),Q(snd(x),snd(y)) \rangle \)

using DirectProduct_def DirectProduct_ZF_1_L1 ZF_fun_from_tot_val

by simp

And the value belongs to the set the operation is defined on.

**Lemma (in direct0) DirectProduct_ZF_1_L3:**

shows \( \forall x \in (G \times H). \forall y \in (G \times H). R(x,y) \in G \times H \)

using DirectProduct_ZF_1_L1

by simp

37.2 Associative and commutative operations

If P and Q are both associative or commutative operations, the direct product of P and Q has the same property.

Direct product of commutative operations is commutative.

**Lemma (in direct0) DirectProduct_ZF_2_L1:**

assumes P {is commutative on} G and Q {is commutative on} H

shows R {is commutative on} G×H

**Proof** -

from assms have \( \forall x \in (G \times H). \forall y \in (G \times H). R(x,y) = R(y,x) \)

using DirectProduct_ZF_1_L2 IsCommutative_def by simp

then show thesis using IsCommutative_def by simp

qed

Direct product of associative operations is associative.

**Lemma (in direct0) DirectProduct_ZF_2_L2:**

assumes P {is associative on} G and Q {is associative on} H

shows R {is associative on} G×H

**Proof** -

have \( \forall x \in G \times H. \forall y \in G \times H. \forall z \in G \times H. R(R(x,y),z) = \)

\( P(P(fst(x),fst(y)),fst(z)),Q(Q(snd(x),snd(y)),snd(z)) \)

using DirectProduct_ZF_1_L2 DirectProduct_ZF_1_L3

by auto

moreover have \( \forall x \in G \times H. \forall y \in G \times H. \forall z \in G \times H. R(x,R(y,z)) = \)

418
ordered groups - introduction

theory OrderedGroup_ZF imports Group_ZF_1 AbelianGroup_ZF Finite_ZF_1 OrderedLoop_ZF

begin

This theory file defines and shows the basic properties of (partially or linearly) ordered groups. We show that in linearly ordered groups finite sets are bounded and provide a sufficient condition for bounded sets to be finite. This allows to show in Int_ZF_IML.thy that subsets of integers are bounded iff they are finite. Some theorems proven here are properties of ordered loops rather that groups. However, for now the development is independent from the material in the OrderedLoop_ZF theory, we just import the definitions of NonnegativeSet and PositiveSet from there.

38.1 Ordered groups

This section defines ordered groups and various related notions.

An ordered group is a group equipped with a partial order that is "translation invariant", that is if \( a \leq b \) then \( a \cdot g \leq b \cdot g \) and \( g \cdot a \leq g \cdot b \).

definition
IsAnOrdGroup(G,P,r)
≡
(\text{IsAgroup}(G,P) \land r \subseteq G \times G \land \text{IsPartOrder}(G,r) \land (\forall g \in G. \forall a \ b.
\langle a,b \rangle \in r \rightarrow (\langle P(a,g),P(b,g) \rangle \in r \land (\langle P(g,a),P(g,b) \rangle \in r)))

We also define the absolute value as a ZF-function that is the identity on \( G^+ \) and the group inverse on the rest of the group.

definition
AbsoluteValue(G,P,r)
≡\text{id}(\text{Nonnegative}(G,P,r)) \cup \text{restrict}(\text{GroupInv}(G,P),G - \text{Nonnegative}(G,P,r))

The odd functions are defined as those having property \( f(a^{-1}) = (f(a))^{-1} \). This looks a bit strange in the multiplicative notation, I have to admit. For linearly ordered groups a function \( f \) defined on the set of positive elements...
inquely defines an odd function of the whole group. This function is called an odd extension of \( f \)

**definition**

\[
\text{OddExtension}(G,P,r,f) \equiv (f \cup \{(a, \text{GroupInv}(G,P)(f(\text{GroupInv}(G,P)(a)))) : a \in \text{GroupInv}(G,P)(\text{PositiveSet}(G,P,r))\} \cup \{(\text{TheNeutralElement}(G,P),\text{TheNeutralElement}(G,P))\})
\]

We will use a similar notation for ordered groups as for the generic groups. \( G^+ \) denotes the set of nonnegative elements (that satisfy \( 1 \leq a \)) and \( G_+ \) is the set of (strictly) positive elements. \(-A\) is the set inverses of elements from \( A \). I hope that using additive notation for this notion is not too shocking here. The symbol \( f^o \) denotes the odd extension of \( f \). For a function defined on \( G_+ \) this is the unique odd function on \( G \) that is equal to \( f \) on \( G_+ \).

**locale** group3 =

- fixes \( G \) and \( P \) and \( r \)
- assumes ordGroupAssum: IsAnOrdGroup(G,P,r)
- fixes unit (1)
  - defines unit_def [simp]: \( 1 \equiv \text{TheNeutralElement}(G,P) \)
- fixes groper (infixl \( \cdot \))
  - defines groper_def [simp]: \( a \cdot b \equiv P(a,b) \)
- fixes inv (_\(-^{-1}\))
  - defines inv_def [simp]: \( x^{-1} \equiv \text{GroupInv}(G,P)(x) \)
- fixes leseq (infix \( \leq \))
  - defines leseq_def [simp]: \( a \leq b \equiv (a,b) \in r \)
- fixes sless (infix \(<\))
  - defines sless_def [simp]: \( a < b \equiv a \leq b \land a \neq b \)
- fixes nonnegative (\( G^+ \))
  - defines nonnegative_def [simp]: \( G^+ \equiv \text{Nonnegative}(G,P,r) \)
- fixes positive (\( G_+ \))
  - defines positive_def [simp]: \( G_+ \equiv \text{PositiveSet}(G,P,r) \)
- fixes setinv (_\(-\))
  - defines setninv_def [simp]: \(-A \equiv \text{GroupInv}(G,P)(A) \)
- fixes abs (|_\()\)
  - defines abs_def [simp]: \(|a| \equiv \text{AbsoluteValue}(G,P,r)(a) \)
- fixes oddext (_\^°)
defines oddext_def [simp]: \( f^o \equiv \text{OddExtension}(G,P,r,f) \)

In group3 context we can use the theorems proven in the group0 context.

lemma (in group3) OrderedGroup_ZF_1_L1: shows group0(G,P)
  using ordGroupAssum IsAnOrdGroup_def group0_def by simp

Ordered group (carrier) is not empty. This is a property of monoids, but it
is good to have it handy in the group3 context.

lemma (in group3) OrderedGroup_ZF_1_L1A: shows \( G \neq \emptyset \)
  using OrderedGroup_ZF_1_L1 group0.group0_2_L1 monoid0.group0_1_L3A
  by blast

The next lemma is just to see the definition of the nonnegative set in our
notation.

lemma (in group3) OrderedGroup_ZF_1_L2:
  shows \( g \in G^+ \iff 1 \leq g \)
  using ordGroupAssum IsAnOrdGroup_def Nonnegative_def
  by auto

The next lemma is just to see the definition of the positive set in our notation.

lemma (in group3) OrderedGroup_ZF_1_L2A:
  shows \( g \in G^+ \iff (1 \leq g \land g \neq 1) \)
  using ordGroupAssum IsAnOrdGroup_def PositiveSet_def
  by auto

For total order if \( g \) is not in \( G^+ \), then it has to be less or equal the unit.

lemma (in group3) OrderedGroup_ZF_1_L2B:
  assumes A1: \( r \text{ is total on } G \) and A2: \( a \in G-G^+ \)
  shows \( a \leq 1 \)
proof -
  from A2 have \( a \in G \quad 1 \in G \quad \neg(1 \leq a) \)
    using OrderedGroup_ZF_1_L1 group0.group0_2_L2 OrderedGroup_ZF_1_L2
    by auto
  with A1 show thesis using IsTotal_def by auto
qed

The group order is reflexive.

lemma (in group3) OrderedGroup_ZF_1_L3: assumes \( g \in G \)
  shows \( g \leq g \)
  using ordGroupAssum assms IsAnOrdGroup_def IsPartOrder_def refl_def
  by simp

1 is nonnegative.

lemma (in group3) OrderedGroup_ZF_1_L3A: shows \( 1 \in G^+ \)
  using OrderedGroup_ZF_1_L2 OrderedGroup_ZF_1_L3
  OrderedGroup_ZF_1_L1 group0.group0_2_L2 by simp
In this context $a \leq b$ implies that both $a$ and $b$ belong to $G$.

**Lemma (in group3) OrderedGroup_ZF_1_L4:**

- Assumes $a \leq b$ shows $a \in G$ $b \in G$
- Using ordGroupAssum assms IsAnOrdGroup_def by auto

Similarly in this context $a \leq b$ implies that both $a$ and $b$ belong to $G$.

**Lemma (in group3) less_are_members:**

- Assumes $a < b$ shows $a \in G$ $b \in G$
- Using ordGroupAssum assms IsAnOrdGroup_def by auto

It is good to have transitivity handy.

**Lemma (in group3) Group_order_transitive:**

- Assumes $A1: a \leq b$ $b \leq c$ shows $a \leq c$
- Proof -
  - From ordGroupAssum have $\text{trans}(r)$
  - Using IsAnOrdGroup_def IsPartOrder_def by simp
  - Moreover from $A1$ have $(a,b) \in r$ $\land$ $(b,c) \in r$ by simp
  - Ultimately have $(a,c) \in r$ by (rule Fol1_L3)
  - Thus thesis by simp
- Qed

The order in an ordered group is antisymmetric.

**Lemma (in group3) group_order_antisym:**

- Assumes $A1: a \leq b$ $b \leq a$ shows $a = b$
- Proof -
  - From ordGroupAssum $A1$ have $\text{antisym}(r)$
  - $(a,b) \in r$ $(b,a) \in r$
  - Using IsAnOrdGroup_def IsPartOrder_def by auto
  - Then show $a = b$ by (rule Fol1_L4)
- Qed

Transitivity for the strict order: if $a < b$ and $b \leq c$, then $a < c$.

**Lemma (in group3) OrderedGroup_ZF_1_L4A:**

- Assumes $A1: a < b$ and $A2: b \leq c$
  - Shows $a < c$
- Proof -
  - From $A1$ $A2$ have $a \leq b$ $b \leq c$ by auto
  - Then have $a \leq c$ by (rule Group_order_transitive)
  - Moreover from $A1$ $A2$ have $a \neq c$ using group_order_antisym by auto
  - Ultimately show $a < c$ by simp
- Qed

Another version of transitivity for the strict order: if $a \leq b$ and $b < c$, then $a < c$.

**Lemma (in group3) group_strict_ord_transit:**

- Assumes $A1: a \leq b$ and $A2: b < c$

422
shows $a < c$
proof -  
  from A1 A2 have $a \leq b$  
  b$\leq c$ by auto  
  then have $a < c$ by (rule Group_order_transitive)  
  moreover from A1 A2 have $a \neq c$ using group_order_antisym by auto  
  ultimately show $a < c$ by simp
qed

The order is translation invariant.

lemma (in group3) ord_transl_inv: assumes $a \leq b$  
  c $\in$ G  
  shows $a \cdot c \leq b \cdot c$  
  and  
  $c \cdot a \leq c \cdot b$  
  using ordGroupAssum assms unfolding IsAnOrdGroup_def by auto

Strict order is preserved by translations.

lemma (in group3) group_strict_ord_transl_inv: assumes $a < b$  
  c $\in$ G  
  shows $a \cdot c < b \cdot c$  
  and  
  $c \cdot a < c \cdot b$  
  using assms ord_transl_inv OrderedGroup_ZF_1_L4 OrderedGroup_ZF_1_L1  
  group0.group0_2_L19  
  by auto

If the group order is total, then the group is ordered linearly.

lemma (in group3) group_ord_total_is_lin: assumes $r \text{ is total on } G$  
  shows IsLinOrder(G,r)  
  using assms ordGroupAssum IsAnOrdGroup_def Order_ZF_1_L3  
  by simp

For linearly ordered groups elements in the nonnegative set are greater than  
those in the complement.

lemma (in group3) OrderedGroup_ZF_1_L4B: assumes $r \text{ is total on } G$  
  and $a \in G^+$  
  and $b \in G-G^+$  
  shows $b \leq a$
proof -  
  from assms have $b \leq 1$  
  l$\leq a$  
  using OrderedGroup_ZF_1_L2 OrderedGroup_ZF_1_L2B by auto  
  then show thesis by (rule Group_order_transitive)
qed

If $a \leq 1$ and $a \neq 1$, then $a \in G \setminus G^+$.

lemma (in group3) OrderedGroup_ZF_1_L4C:  
  assumes A1: $a \leq 1$  
  and A2: $a \neq 1$  
  shows $a \in G-G^+$
proof -  
  { assume $a \notin G-G^+$  
    with ordGroupAssum A1 A2 have False  
    using OrderedGroup_ZF_1_L4 OrderedGroup_ZF_1_L2
  }
An element smaller than an element in $G \setminus G^+$ is in $G \setminus G^+$.

**lemma (in group3) OrderedGroup_ZF_1_L4D:**
assumes $A1: a \in G - G^+$ and $A2: b \leq a$
shows $b \in G - G^+$
**proof** -
{ assume $b \not\in G - G^+$
with $A2$ have $1 \leq b$ $b \leq a$
using OrderedGroup_ZF_1_L4 OrderedGroup_ZF_1_L2 by auto
then have $1 \leq a$ by (rule Group order transitive)
with $A1$ have False using OrderedGroup_ZF_1_L2 by simp
} thus thesis by auto
qed

The nonnegative set is contained in the group.

**lemma (in group3) OrderedGroup_ZF_1_L4E:** shows $G^+ \subseteq G$
using OrderedGroup_ZF_1_L2 OrderedGroup_ZF_1_L4 by auto

The positive set is contained in the nonnegative set, hence in the group.

**lemma (in group3) pos_set_in_gr:** shows $G^+ \subseteq G^+$ and $G^+ \subseteq G$
using OrderedGroup_ZF_1_L2A OrderedGroup_ZF_1_L2 OrderedGroup_ZF_1_L4E
by auto

Taking the inverse on both sides reverses the inequality.

**lemma (in group3) OrderedGroup_ZF_1_L5:**
assumes $A1: a \leq b$ shows $b^{-1} \leq a^{-1}$
**proof** -
from $A1$ have $T1: a \in G$ $b \in G$ $a^{-1} \in G$ $b^{-1} \in G$
using OrderedGroup_ZF_1_L4 OrderedGroup_ZF_1_L1
by auto
with $A1$ ordGroupAssum have $a \cdot a^{-1} \leq b \cdot a^{-1}$ using IsAnOrdGroup_def
by simp
with $T1$ ordGroupAssum have $b^{-1} \cdot 1 \leq b^{-1} \cdot (b \cdot a^{-1})$
using OrderedGroup_ZF_1_L1 group0.group0_2_L6 IsAnOrdGroup_def
by simp
with $T1$ show thesis using
OrderedGroup_ZF_1_L1 group0.group0_2_L2 group0.group_oper_assoc
by simp
qed

If an element is smaller that the unit, then its inverse is greater.

**lemma (in group3) OrderedGroup_ZF_1_L5A:**
assumes $A1: a \leq 1$ shows $1 \leq a^{-1}$
**proof** -
from A1 have $1^{-1} \leq a^{-1}$ using OrderedGroup_ZF_1_L5
  by simp
then show thesis using OrderedGroup_ZF_1_L1 group0.group_inv_of_one
  by simp
qed

If an the inverse of an element is greater that the unit, then the element is smaller.

lemma (in group3) OrderedGroup_ZF_1_L5AA:
  assumes A1: $a \in G$ and A2: $1 \leq a^{-1}$
  shows $a \leq 1$
proof -
  from A2 have $(a^{-1})^{-1} \leq 1^{-1}$ using OrderedGroup_ZF_1_L5
    by simp
  with A1 show $a \leq 1$
    using OrderedGroup_ZF_1_L1 group0.group_inv_of_inv group0.group_inv_of_one
    by simp
qed

If an element is nonnegative, then the inverse is not greater that the unit. Also shows that nonnegative elements cannot be negative

lemma (in group3) OrderedGroup_ZF_1_L5AB:
  assumes A1: $1 \leq a$
  shows $a^{-1} \leq 1$ and $\neg(a \leq 1 \land a \neq 1)$
proof -
  from A1 have $a^{-1} \leq 1^{-1}$
    using OrderedGroup_ZF_1_L5 by simp
  then show $a^{-1} \leq 1$ using OrderedGroup_ZF_1_L1 group0.group_inv_of_one
    by simp
  \{ assume $a \leq 1$ and $a \neq 1$
    with A1 have False using group_order_antisym
      by blast
  \} then show $\neg(a \leq 1 \land a \neq 1)$ by auto
qed

If two elements are greater or equal than the unit, then the inverse of one is not greater than the other.

lemma (in group3) OrderedGroup_ZF_1_L5AC:
  assumes A1: $1 \leq a$ $1 \leq b$
  shows $a^{-1} \leq b$
proof -
  from A1 have $a^{-1} \leq 1$ $1 \leq b$
    using OrderedGroup_ZF_1_L5AB by auto
  then show $a^{-1} \leq b$ by (rule Group_order_transitive)
qed

38.2 Inequalities

This section developes some simple tools to deal with inequalities.
Taking negative on both sides reverses the inequality, case with an inverse on one side.

**lemma** (in group3) OrderedGroup_ZF_1_L5AD:

assumes A1: \( b \in G \) and A2: \( a \leq b^{-1} \)

shows \( b \leq a^{-1} \)

**proof** -

from A2 have \((b^{-1})^{-1} \leq a^{-1}\)

using OrderedGroup_ZF_1_L5 by simp

with A1 show \( b \leq a^{-1} \)

using OrderedGroup_ZF_1_L1 group0.group_inv_of_inv by simp

qed

We can cancel the same element on both sides of an inequality.

**lemma** (in group3) OrderedGroup_ZF_1_L5AE:

assumes A1: \( a \in G \) b \( \in G \) c \( \in G \) and A2: \( a \cdot b \leq a \cdot c \)

shows \( b \leq c \)

**proof** -

from ordGroupAssum A1 A2 have \( a^{-1} \cdot (a \cdot b) \leq a^{-1} \cdot (a \cdot c) \)

using OrderedGroup_ZF_1_L1 group0.inverse_in_group IsAnOrdGroup_def

by simp

with A1 show \( b \leq c \)

using OrderedGroup_ZF_1_L1 group0.inv_cancel_two by simp

qed

We can cancel the same element on both sides of an inequality, right side.

**lemma** (in group3) ineq_cancel_right:

assumes a \( \in G \) b \( \in G \) c \( \in G \) and a \( \cdot b \leq c \cdot b \)

shows \( a \leq c \)

**proof** -

from assms(2,4) have \((a \cdot b)^{-1} \leq (c \cdot b)^{-1}\)

using OrderedGroup_ZF_1_L1 group0.inverse_in_group ord_transl_inv(1)

by simp

with assms(1,2,3) show \( a \leq c \) using OrderedGroup_ZF_1_L1 group0.inv_cancel_two(2)

by auto

qed

We can cancel the same element on both sides of an inequality, a version with an inverse on both sides.

**lemma** (in group3) OrderedGroup_ZF_1_L5AF:

assumes A1: \( a \in G \) b \( \in G \) c \( \in G \) and A2: \( a \cdot b^{-1} \leq a \cdot c^{-1} \)

shows \( c \leq b \)

**proof** -

from A1 A2 have \((c^{-1})^{-1} \leq (b^{-1})^{-1}\)

using OrderedGroup_ZF_1_L1 group0.inverse_in_group

426
Taking negative on both sides reverses the inequality, another case with an inverse on one side.

lemma (in group3) OrderedGroup_ZF_1_L5AG:
  assumes A1: a ∈ G and A2: a⁻¹ ≤ b
  shows b⁻¹ ≤ a
proof -
  from A2 have b⁻¹ ≤ (a⁻¹)⁻¹
    using OrderedGroup_ZF_1_L5 by simp
  with A1 show b⁻¹ ≤ a
    using OrderedGroup_ZF_1_L1 group0.group_inv_of_inv by simp
qed

We can multiply the sides of two inequalities.

lemma (in group3) OrderedGroup_ZF_1_L5B:
  assumes A1: a ≤ b and A2: c ≤ d
  shows a · c ≤ b · d
proof -
  from A1 A2 have c ∈ G b ∈ G using OrderedGroup_ZF_1_L4 by auto
  with A1 A2 ordGroupAssum have a · c ≤ b · c b · c ≤ b · d
    using IsAnOrdGroup_def by auto
  then show a · c ≤ b · d by (rule Group_order_transitive)
qed

We can replace first of the factors on one side of an inequality with a greater one.

lemma (in group3) OrderedGroup_ZF_1_L5C:
  assumes A1: c ∈ G and A2: a · c ≤ b and A3: b ≤ b₁
  shows a ≤ b₁ · c
proof -
  from A1 A3 have b · c ≤ b₁ · c
    using OrderedGroup_ZF_1_L3 OrderedGroup_ZF_1_L5B by simp
  with A2 show a ≤ b₁ · c by (rule Group_order_transitive)
qed

We can replace second of the factors on one side of an inequality with a greater one.

lemma (in group3) OrderedGroup_ZF_1_L5D:
  assumes A1: b ∈ G and A2: a ≤ b · c and A3: c ≤ b₁
  shows a ≤ b · b₁
proof -
  from A1 A3 have b · c ≤ b · b₁
    using OrderedGroup_ZF_1_L3 OrderedGroup_ZF_1_L5B by auto
with A2 show \( a \leq b \cdot b_1 \) by (rule Group_order_transitive)  
qed

We can replace factors on one side of an inequality with greater ones.

**lemma** (in group3) OrderedGroup_ZF_1_L5E:  
assumes A1: \( a \leq b \cdot c \) and A2: \( b \leq b_1 \) \( c \leq c_1 \)  
shows \( a \leq b_1 \cdot c_1 \)  
proof -  
from A2 have \( b \cdot c \leq b_1 \cdot c_1 \) using OrderedGroup_ZF_1_L5B  
by simp  
with A1 show \( a \leq b_1 \cdot c_1 \) by (rule Group_order_transitive)  
qed

We don’t decrease an element of the group by multiplying by one that is nonnegative.

**lemma** (in group3) OrderedGroup_ZF_1_L5F:  
assumes A1: \( 1 \leq a \) and A2: \( b \in G \)  
shows \( b \leq a \cdot b \) \( b_1 \leq a \cdot b \)  
proof -  
from ordGroupAssum A1 A2  
have I: \( b \leq b \cdot c \) and II: \( b \leq c \cdot b \)  
using OrderedGroup_ZF_1_L4 OrderedGroup_ZF_1_L5F  
by auto  
from A1 I show \( a \leq c \cdot b \) by (rule Group_order_transitive)  
from A1 II show \( a \leq b \cdot a \) by (rule Group_order_transitive)  
qed

We can multiply the right hand side of an inequality by a nonnegative element.

**lemma** (in group3) OrderedGroup_ZF_1_L5G: assumes A1: \( a \leq b \) and A2: \( 1 \leq c \) shows \( a \leq b \cdot c \) \( a \leq c \cdot b \)  
proof -  
from A1 A2 have I: \( b \leq b \cdot c \) and II: \( c \leq c \cdot b \)  
using OrderedGroup_ZF_1_L4 OrderedGroup_ZF_1_L5F by auto  
from A1 I show \( a \leq b \cdot c \) by (rule Group_order_transitive)  
from A1 II show \( a \leq c \cdot b \) by (rule Group_order_transitive)  
qed

We can put two elements on the other side of inequality, changing their sign.

**lemma** (in group3) OrderedGroup_ZF_1_L5H:  
assumes A1: \( a \in G \) \( b \in G \) and A2: \( a \cdot b^{-1} \leq c \)  
shows \( a \leq c \cdot b \) \( c^{-1} \cdot a \leq b \)  
proof -  
from A2 have T: \( c^{-1} \in G \) \( c^{-1} \in G \)  
using OrderedGroup_ZF_1_L4 OrderedGroup_ZF_1_L1  
group0.inverse_in_group by auto
from ordGroupAssum A1 A2 have a⋅b⁻¹⋅b ≤ c⋅b
using IsAnOrdGroup_def by simp
with A1 show a ≤ c⋅b
using OrderedGroup_ZF_1_L1 group0.inv_cancel_two
by simp
with ordGroupAssum A2 T have c⁻¹⋅a ≤ c⁻¹⋅(c⋅b)
using IsAnOrdGroup_def by simp
with A1 T show c⁻¹⋅a ≤ b
using OrderedGroup_ZF_1_L1 group0.inv_cancel_two
by simp
qed

We can multiply the sides of one inequality by inverse of another.

lemma (in group3) OrderedGroup_ZF_1_L5I:
assumes a≤b and c≤d
shows a⋅d⁻¹ ≤ b⋅c⁻¹
using assms OrderedGroup_ZF_1_L5 OrderedGroup_ZF_1_L5B
by simp

We can put an element on the other side of an inequality changing its sign, version with the inverse.

lemma (in group3) OrderedGroup_ZF_1_L5J:
assumes A1: a∈G b∈G and A2: c≤a⋅b⁻¹
shows c⋅b≤a
proof -
from ordGroupAssum A1 A2 have c⋅b ≤ a⋅b⁻¹⋅b
using IsAnOrdGroup_def by simp
with A1 show c⋅b ≤ a
using OrderedGroup_ZF_1_L1 group0.inv_cancel_two
by simp
qed

We can put an element on the other side of an inequality changing its sign, version with the inverse.

lemma (in group3) OrderedGroup_ZF_1_L5JA:
assumes A1: a∈G b∈G and A2: c≤a⁻¹⋅b
shows a⋅c≤b
proof -
from ordGroupAssum A1 A2 have a⋅c ≤ a⋅(a⁻¹⋅b)
using IsAnOrdGroup_def by simp
with A1 show a⋅c≤b
using OrderedGroup_ZF_1_L1 group0.inv_cancel_two
by simp
qed

A special case of OrderedGroup_ZF_1_L5J where c = 1.

corollary (in group3) OrderedGroup_ZF_1_L5K:
assumes A1: a∈G b∈G and A2: 1 ≤ a⋅b⁻¹
shows \( b \leq a \)

**proof -**

from \( A1 \ A2 \) have \( 1 \cdot b \leq a \)

using OrderedGroup_ZF_1_L5J by simp

with \( A1 \) show \( b \leq a \)

using OrderedGroup_ZF_1_L1 group0.group0_2_L2 by simp

**qed**

A special case of OrderedGroup_ZF_1_L5JA where \( c = 1 \).

**corollary** (in group3) OrderedGroup_ZF_1_L5KA:

assumes \( A1: a \in G \ b \in G \) and \( A2: 1 \leq a^{-1} \cdot b \)

shows \( a \leq b \)

**proof -**

from \( A1 \ A2 \) have \( a \cdot 1 \leq b \)

using OrderedGroup_ZF_1_L5JA by simp

with \( A1 \) show \( a \leq b \)

using OrderedGroup_ZF_1_L1 group0.group0_2_L2 by simp

**qed**

If the order is total, the elements that do not belong to the positive set are negative. We also show here that the group inverse of an element that does not belong to the nonnegative set does belong to the nonnegative set.

**lemma** (in group3) OrderedGroup_ZF_1_L6:

assumes \( A1: r \text{ is total on } G \) and \( A2: a \in G^- \)

shows \( a \leq 1 \ a^{-1} \in G^+ \) restrict\((\text{GroupInv}(G,P),G^-)(a) \in G^+ \)

**proof -**

from \( A2 \) have \( T1: a \in G \ a \in G^- \ 1 \in G \)

using OrderedGroup_ZF_1_L1 group0.group0_2_L2 by auto

with \( A1 \) show \( a \leq 1 \) using OrderedGroup_ZF_1_L2 IsTotal_def by auto

then show \( a^{-1} \in G^+ \) using OrderedGroup_ZF_1_L5A OrderedGroup_ZF_1_L2 by simp

with \( A2 \) show \( \text{restrict}(\text{GroupInv}(G,P),G^-)(a) \in G^+ \)

using restrict by simp

**qed**

If a property is invariant with respect to taking the inverse and it is true on the nonnegative set, than it is true on the whole group.

**lemma** (in group3) OrderedGroup_ZF_1_L7:

assumes \( A1: r \text{ is total on } G \)

and \( A2: \forall a \in G^+, \forall b \in G^+. \ Q(a,b) \)

and \( A3: \forall a \in G, \forall b \in G. \ Q(a,b) \iff Q(a^{-1},b) \)

and \( A4: \forall a \in G, \forall b \in G. \ Q(a,b) \iff Q(a,b^{-1}) \)

and \( A5: a \in G \ b \in G \)

shows \( Q(a,b) \)

**proof -**
{ assume \( A6: \ a \in G^+ \) have \( Q(a,b) \)
  proof -
  { assume \( b \in G^+ \)
    with \( A6 \) \( A2 \) have \( Q(a,b) \) by simp }
    moreover
    { assume \( b \notin G^+ \)
      with \( A1 \) \( A2 \) \( A4 \) \( A5 \) \( A6 \) have \( Q(a,(b^{-1})^{-1}) \)
        using \( \text{OrdGroup\_ZF\_1\_L6} \) \( \text{OrdGroup\_ZF\_1\_L1} \) \( \text{group0\_inverse\_in\_group} \)
          by simp
      with \( A5 \) have \( Q(a,b) \) using \( \text{OrdGroup\_ZF\_1\_L1} \) \( \text{group0\_group\_inv\_of\_inv} \)
        by simp
    }
    ultimately show \( Q(a,b) \) by auto
  qed }
  moreover
  { assume \( a \notin G^+ \)
    with \( A1 \) \( A5 \) have \( T1: \ a^{-1} \in G^+ \) using \( \text{OrdGroup\_ZF\_1\_L6} \) by simp
    have \( Q(a,b) \)
    proof -
      { assume \( b \in G^+ \)
        with \( A2 \) \( A3 \) \( A5 \) \( T1 \) have \( Q((a^{-1})^{-1},b) \)
          using \( \text{OrdGroup\_ZF\_1\_L1} \) \( \text{group0\_inverse\_in\_group} \) by simp
        with \( A5 \) have \( Q(a,b) \) using \( \text{OrdGroup\_ZF\_1\_L1} \) \( \text{group0\_group\_inv\_of\_inv} \)
          by simp
      }
      ultimately show \( Q(a,b) \) by auto
    qed }
  moreover
  { assume \( b \notin G^+ \)
    with \( A1 \) \( A2 \) \( A3 \) \( A4 \) \( A5 \) \( T1 \) have \( Q((a^{-1})^{-1},(b^{-1})^{-1}) \)
      using \( \text{OrdGroup\_ZF\_1\_L6} \) \( \text{OrdGroup\_ZF\_1\_L1} \) \( \text{group0\_inverse\_in\_group} \) by simp
    with \( A5 \) have \( Q(a,b) \) using \( \text{OrdGroup\_ZF\_1\_L1} \) \( \text{group0\_group\_inv\_of\_inv} \)
      by simp
    }
    ultimately show \( Q(a,b) \) by auto
  qed }
ultimately show \( Q(a,b) \) by auto
qed

A lemma about splitting the ordered group “plane” into 6 subsets. Useful for proofs by cases.

lemma (in group3) OrdGroup_6cases: assumes \( A1: \ r \) \{is total on\} \( G \)
  and \( A2: \ a \in G^+ \) \( b \in G \)
shows
\[ \begin{align*}
1 \leq a \land 1 \leq b & \lor a \leq 1 \land b \leq 1 \\
 a \leq 1 \land 1 \leq b \land 1 \leq a \lor a \leq 1 \land 1 \leq b \land a \cdot b \leq 1 \\
1 \leq a \land b \leq 1 \land 1 \leq a \lor 1 \leq a \land b \leq 1 \land a \cdot b \leq 1
\end{align*} \]
proof -
from \( A1 \) \( A2 \) have
\[ \begin{align*}
1 \leq a \lor a \leq 1 \\
1 \leq b \lor b \leq 1 \\
1 \leq a \cdot b \lor a \cdot b \leq 1
\end{align*} \]
using \( \text{OrdGroup\_ZF\_1\_L1} \) \( \text{group0\_op\_closed} \) \( \text{group0\_2\_L2} \)
IsTotal_def by auto
then show thesis by auto
qed

The next lemma shows what happens when one element of a totally ordered group is not greater or equal than another.

lemma (in group3) OrderedGroup_ZF_1_L8:
  assumes A1: r {is total on} G
  and A2: a∈G  b∈G
  and A3: ¬(a≤b)
  shows b ≤ a  a⁻¹ ≤ b⁻¹  a≠b  b<a
proof -
  from A1 A2 A3 show I: b ≤ a using IsTotal_def by auto
  then show a⁻¹ ≤ b⁻¹ using OrderedGroup_ZF_1_L5 by simp
  from A2 have a ≤ a using OrderedGroup_ZF_1_L3 by simp
  with I A3 show a≠b  b < a by auto
qed

If one element is greater or equal and not equal to another, then it is not smaller or equal.

lemma (in group3) OrderedGroup_ZF_1_L8AA:
  assumes A1: a ≤ b and A2: a≠b
  shows ¬(b≤a)
proof -
  { note A1
    moreover assume b≤a
    ultimately have a=b by (rule group_order_antisym)
    with A2 have False by simp
  } thus ¬(b≤a) by auto
qed

A special case of OrderedGroup_ZF_1_L8 when one of the elements is the unit.

corollary (in group3) OrderedGroup_ZF_1_L8A:
  assumes A1: r {is total on} G
  and A2: a∈G and A3: ¬(1≤a)
  shows 1 ≤ a⁻¹  1≠a  a≤1
proof -
  from A1 A2 A3 have I:
    r {is total on} G
    1∈G  a∈G
    ¬(1≤a)
    using OrderedGroup_ZF_1_L1 group0.group0_2_L2 by auto
  then have 1⁻¹ ≤ a⁻¹
    by (rule OrderedGroup_ZF_1_L8)
  then show 1 ≤ a⁻¹

using OrderedGroup_ZF_1_L1 group0.group_inv_of_one by simp
from I show 1≠a by (rule OrderedGroup_ZF_1_L8)
from A1 I show a≤1 using IsTotal_def
by auto
qed

A negative element can not be nonnegative.

lemma (in group3) OrderedGroup_ZF_1_L8B:
assumes A1: a≤1 and A2: a≠1 shows ¬(1≤a)
proof -
{ assume 1≤a
  with A1 have a=1 using group_order_antisym
  by auto
  with A2 have False by simp
} thus thesis by auto
qed

An element is greater or equal than another iff the difference is nonpositive.

lemma (in group3) OrderedGroup_ZF_1_L9:
assumes A1: a∈G b∈G
shows a≤b ↔ a·b⁻¹ ≤ 1
proof
assume a ≤ b
with ordGroupAssum A1 have a·b⁻¹ ≤ b·b⁻¹
  using OrderedGroup_ZF_1_L1 group0.inverse_in_group
  IsAnOrdGroup_def by simp
with A1 show a·b⁻¹ ≤ 1
  using OrderedGroup_ZF_1_L1 group0.group0_2_L6
  by simp
next assume A2: a·b⁻¹ ≤ 1
  with ordGroupAssum A1 have a·b⁻¹·b ≤ 1·b
    using IsAnOrdGroup_def by simp
  with A1 show a ≤ b
    using OrderedGroup_ZF_1_L1
group0.inv_cancel_two group0.group0_2_L2
    by simp
qed

We can move an element to the other side of an inequality.

lemma (in group3) OrderedGroup_ZF_1_L9A:
assumes A1: a∈G b∈G c∈G
shows a·b ≤ c ↔ a ≤ c·b⁻¹
proof
assume a·b ≤ c
with ordGroupAssum A1 have a·b·b⁻¹ ≤ c·b⁻¹
  using OrderedGroup_ZF_1_L1 group0.inverse_in_group IsAnOrdGroup_def
  by simp
with A1 show a ≤ c·b⁻¹
  using OrderedGroup_ZF_1_L1 group0.inv_cancel_two by simp
next assume \( a \leq c \cdot b^{-1} \)
  with ordGroupAssum A1 have \( a \cdot b \leq c \cdot b^{-1} \cdot b \)
  using OrderedGroup_ZF_1_L1 group0.inverse_in_group IsAnOrdGroup_def
  by simp
  with A1 show \( a \cdot b \leq c \cdot b \cdot b^{-1} \)
  using OrderedGroup_ZF_1_L1 group0.inv_cancel_two by simp
qed

A one side version of the previous lemma with weaker assumptions.

lemma (in group3) OrderedGroup_ZF_1_L9B:
  assumes A1: \( a \in G \) \( b \in G \) and A2: \( a \cdot b^{-1} \leq c \)
  shows \( a \leq c \cdot b \)
proof -
  from A1 A2 have \( a \in G \) \( b^{-1} \in G \) \( c \in G \)
  using OrderedGroup_ZF_1_L1 group0.inverse_in_group
  OrderedGroup_ZF_1_L4 by auto
  with A1 A2 show \( a \leq c \cdot b \)
  using OrderedGroup_ZF_1_L9A OrderedGroup_ZF_1_L1
  group0.group_inv_of_inv by simp
qed

We can put an element on the other side of inequality, changing its sign.

lemma (in group3) OrderedGroup_ZF_1_L9C:
  assumes A1: \( a \in G \) \( b \in G \) and A2: \( c \leq a \cdot b \)
  shows \( c \cdot b^{-1} \leq a \)
  \( a^{-1} \cdot c \leq b \)
proof -
  from ordGroupAssum A1 A2 have \( c \cdot b^{-1} \leq a \cdot b^{-1} \)
  \( a^{-1} \cdot c \leq a^{-1} \cdot (a \cdot b) \)
  using OrderedGroup_ZF_1_L1 group0.inverse_in_group IsAnOrdGroup_def
  by auto
  with A1 show \( c \cdot b^{-1} \leq a \)
  \( a^{-1} \cdot c \leq b \)
  using OrderedGroup_ZF_1_L1 group0.inv_cancel_two by auto
qed

If an element is greater or equal than another then the difference is nonnegative.

lemma (in group3) OrderedGroup_ZF_1_L9D: assumes A1: \( a \leq b \)
  shows \( 1 \leq b \cdot a^{-1} \)
proof -
  from \( a \in G \) \( b \in G \) \( a^{-1} \in G \)
  using OrderedGroup_ZF_1_L4 OrderedGroup_ZF_1_L1
  group0.inverse_in_group by auto
  with ordGroupAssum A1 have \( a \cdot a^{-1} \leq b \cdot a^{-1} \)

If an element is greater than another then the difference is positive.

lemma (in group3) OrderedGroup_ZF_1_L9E:
  assumes A1: a ≤ b a ≠ b
  shows 1 ≤ b·a⁻¹ 1 ≠ b·a⁻¹ b·a⁻¹ ∈ G⁺
proof -
  from A1 have T: a∈G b∈G using OrderedGroup_ZF_1_L4
  by auto
  from A1 show I: 1 ≤ b·a⁻¹ using OrderedGroup_ZF_1_L9D
  by simp
  { assume b·a⁻¹ = 1
    with T have a=b
      using OrderedGroup_ZF_1_L1 group0.group0_2_L11A
    by auto
    with A1 have False by simp
  } then show 1 ≠ b·a⁻¹ by auto
  then have b·a⁻¹ ≠ 1 by auto
  with I show b·a⁻¹ ∈ G⁺ using OrderedGroup_ZF_1_L2A
  by simp
qed

If the difference is nonnegative, then a ≤ b.

lemma (in group3) OrderedGroup_ZF_1_L9F:
  assumes A1: a∈G b∈G and A2: 1 ≤ b·a⁻¹
  shows a ≤ b
proof -
  from A1 A2 have 1·a ≤ b
    using OrderedGroup_ZF_1_L4 OrderedGroup_ZF_1_L9A
    by simp
  with A1 show a≤b
    using OrderedGroup_ZF_1_L1 group0.group0_2_L2
    by simp
qed

If we increase the middle term in a product, the whole product increases.

lemma (in group3) OrderedGroup_ZF_1_L10:
  assumes a∈G b∈G and c≤d
  shows a·c·b ≤ a·d·b
using ordGroupAssum assms IsAnOrdGroup_def by simp

A product of (strictly) positive elements is not the unit.

lemma (in group3) OrderedGroup_ZF_1_L11:
  assumes A1: 1≤a 1≤b
and A2: $1 \neq a \ 1 \neq b$
shows $1 \neq a \cdot b$
proof -
from A1 have T1: $a \in G \ b \in G$
  using OrderedGroup_ZF_1_L4 by auto
{ assume $1 = a \cdot b$
  with A1 T1 have $a \leq a \ 1 \leq a$
    using OrderedGroup_ZF_1_L1 group0.group0_2_L9 OrderedGroup_ZF_1_L5AA
    by auto
  then have $a = 1$ by (rule group_order_antisym)
  with A2 have False by simp
} then show $1 \neq a \cdot b$ by auto
qed

A product of nonnegative elements is nonnegative.

lemma (in group3) OrderedGroup_ZF_1_L12:
assumes A1: $1 \leq a \ 1 \leq b$
shows $1 \leq a \cdot b$
proof -
from A1 have $1 \cdot 1 \leq a \cdot b$
  using OrderedGroup_ZF_1_L5B by simp
then show $1 \leq a \cdot b$
  using OrderedGroup_ZF_1_L1 group0.group0_2_L2
  by simp
qed

If $a$ is not greater than $b$, then $1$ is not greater than $b \cdot a^{-1}$.

lemma (in group3) OrderedGroup_ZF_1_L12A:
assumes A1: $a \leq b$
shows $1 \leq b \cdot a^{-1}$
proof -
from A1 have T: $1 \in G \ a \in G \ b \in G$
  using OrderedGroup_ZF_1_L4 OrderedGroup_ZF_1_L1 group0.group0_2_L2
  by auto
with A1 have $1 \cdot a \leq b$
  using OrderedGroup_ZF_1_L1 group0.group0_2_L2
  by simp
with T show $1 \leq b \cdot a^{-1}$ using OrderedGroup_ZF_1_L9A
  by simp
qed

We can move an element to the other side of a strict inequality.

lemma (in group3) OrderedGroup_ZF_1_L12B:
assumes A1: $a \in G \ b \in G$ and A2: $a \cdot b^{-1} < c$
shows $a < c \cdot b$
proof -
from A1 A2 have $a \cdot b^{-1} \cdot b < c \cdot b$
  using group_strict_ord_transl_inv by auto
moreover from A1 have $a \cdot b^{-1} \cdot b = a$

using OrderedGroup_ZF_1_L11 group0.inv_cancel_two
   by simp
ultimately show a < c·b
   by auto
qed

We can multiply the sides of two inequalities, first of them strict and we get
a strict inequality.

lemma (in group3) OrderedGroup_ZF_1_L12C:
   assumes A1: a<b and A2: c≤d
   shows a·c < b·d
proof -
   from A1 A2 have T: a∈G b∈G c∈G d∈G
     using OrderedGroup_ZF_1_L4 by auto
   with ordGroupAssum A2 have a·c ≤ a·d
     using IsAnOrdGroup_def by simp
   moreover from A1 T have a·d < b·d
     using group_strict_ord_transl_inv by simp
   ultimately show a·c < b·d
     by (rule group_strict_ord_transit)
qed

We can multiply the sides of two inequalities, second of them strict and we
get a strict inequality.

lemma (in group3) OrderedGroup_ZF_1_L12D:
   assumes A1: a≤b and A2: c<d
   shows a·c < b·d
proof -
   from A1 A2 have T: a∈G b∈G c∈G d∈G
     using OrderedGroup_ZF_1_L4 by auto
   with A2 have a·c < a·d
     using group_strict_ord_transl_inv by simp
   moreover from ordGroupAssum A1 T have a·d ≤ b·d
     using IsAnOrdGroup_def by simp
   ultimately show a·c < b·d
     by (rule OrderedGroup_ZF_1_L4A)
qed

38.3 The set of positive elements

In this section we study $G_+$ - the set of elements that are (strictly) greater
than the unit. The most important result is that every linearly ordered
group can decomposed into $\{1\}, G_+$ and the set of those elements $a \in G$
such that $a^{-1} \in G_+$. Another property of linearly ordered groups that we
prove here is that if $G_+ \neq \emptyset$, then it is infinite. This allows to show that
nontrivial linearly ordered groups are infinite.

The positive set is closed under the group operation.
lemma (in group3) OrderedGroup_ZF_1_L13: shows $G_+ \{\text{is closed under}\} P$
proof -
{ fix $a \ b$ assume $a \in G_+ \ b \in G_+$
  then have T1: $1 \leq a \cdot b \quad \text{and} \quad 1 \neq a \cdot b$
    using PositiveSet_def OrderedGroup_ZF_1_L11 OrderedGroup_ZF_1_L12
    by auto
  moreover from T1 have $a \cdot b \in G$
    using OrderedGroup_ZF_1_L4 by simp
  ultimately have $a \cdot b \in G_+$ using PositiveSet_def by simp
} then show $G_+ \{\text{is closed under}\} P$ using IsOpClosed_def
  by simp
qed

For totally ordered groups every nonunit element is positive or its inverse is positive.

lemma (in group3) OrderedGroup_ZF_1_L14:
assumes A1: $r \{\text{is total on}\} G$ and A2: $a \in G$
shows $a=1 \lor a \in G_+ \lor a^{-1} \in G_+$
proof -
{ assume A3: $a \neq 1$
  moreover from A1 A2 have $a \leq 1 \lor 1 \leq a$
    using IsTotal_def OrderedGroup_ZF_1_L1 group0.group0_2_L2
    by simp
  moreover from A3 A2 have T1: $a^{-1} \neq 1$
    using OrderedGroup_ZF_1_L1 group0.group0_2_L8B
    by simp
  ultimately have $a^{-1} \in G_+ \lor a \in G_+$
    using OrderedGroup_ZF_1_L5A OrderedGroup_ZF_1_L2A
    by auto
} thus $a=1 \lor a \in G_+ \lor a^{-1} \in G_+$ by auto
qed

If an element belongs to the positive set, then it is not the unit and its inverse does not belong to the positive set.

lemma (in group3) OrderedGroup_ZF_1_L15:
assumes A1: $a \in G_+$ shows $a \neq 1 \quad a^{-1} \notin G_+$
proof -
{ from A1 show T1: $a \neq 1$ using PositiveSet_def by auto
  { assume $a^{-1} \in G_+$
    with A1 have $a \leq 1 \quad 1 \leq a$
      using OrderedGroup_ZF_1_L5AA PositiveSet_def by auto
    then have $a=1$ by (rule group_order_antisym)
    with T1 have False by simp
  } then show $a^{-1} \notin G_+$ by auto
qed

If $a^{-1}$ is positive, then $a$ can not be positive or the unit.

lemma (in group3) OrderedGroup_ZF_1_L16:
assumes $A1: a \in G$ and $A2: a^{-1} \in G_+$ shows $a \neq 1$, $a \not\in G_+$

proof -
from $A2$ have $a^{-1} \neq 1$, $(a^{-1})^{-1} \not\in G_+$
  using OrderedGroup_ZF_1_L15 by auto
with $A1$ show $a \neq 1$, $a \not\in G_+$
  using OrderedGroup_ZF_1_L1, group0.group0_2_L8C, group0.group_inv_of_inv
by auto
qed

For linearly ordered groups each element is either the unit, positive or its inverse is positive.

**lemma** (in group3) OrdGroup_decomp:
assumes $A1: r \text{ is total on } G$ and $A2: a \in G$
shows $\text{Exactly}_1\text{ of}_3\text{ holds } (a=1, a \in G_+, a^{-1} \in G_+)$

proof -
from $A1$ $A2$ have $a=1 \lor a \in G_+ \lor a^{-1} \in G_+$
  using OrderedGroup_ZF_1_L14 by simp
moreover from $A2$ have $a=1 \longrightarrow (a \not\in G_+ \land a^{-1} \not\in G_+)$
  using OrderedGroup_ZF_1_L1, group0.group_inv_of_one
PositiveSet_def by simp
moreover from $A2$ have $a \in G_+ \longrightarrow (a \neq 1 \land a^{-1} \not\in G_+)$
  using OrderedGroup_ZF_1_L15 by simp
moreover from $A2$ have $a^{-1} \in G_+ \longrightarrow (a \neq 1 \land a \not\in G_+)$
  using OrderedGroup_ZF_1_L16 by simp
ultimately show $\text{Exactly}_1\text{ of}_3\text{ holds } (a=1, a \in G_+, a^{-1} \in G_+)$
  by (rule Fol1_L5)
qed

A if $a$ is a nonunit element that is not positive, then $a^{-1}$ is is positive. This is useful for some proofs by cases.

**lemma** (in group3) OrdGroup_cases:
assumes $A1: r \text{ is total on } G$ and $A2: a \in G$ and $A3: a \neq 1$, $a \not\in G_+$
shows $a^{-1} \in G_+$

proof -
from $A1$ $A2$ have $a=1 \lor a \in G_+ \lor a^{-1} \in G_+$
  using OrderedGroup_ZF_1_L14 by simp
with $A3$ show $a^{-1} \in G_+$ by auto
qed

Elements from $G \setminus G_+$ are not greater that the unit.

**lemma** (in group3) OrderedGroup_ZF_1_L17:
assumes $A1: r \text{ is total on } G$ and $A2: a \in G \setminus G_+$
shows $a \leq 1$

proof -
  { assume $a=1$
    with $A2$ have $a \leq 1$ using OrderedGroup_ZF_1_L3 by simp}
  
  { assume $a \neq 1$
    with $A2$ have $a \neq 1$ using OrderedGroup_ZF_1_L3 by simp
    with $A1$ $A2$ have $a^{-1} \in G_+$ using OrderedGroup_ZF_1_L15 by auto
    with $A3$ have $a^{-1} \not\in G_+$ using OrderedGroup_ZF_1_L16 by simp
    ultimately show $a \leq 1$ by (rule Fol1_L5)
  }
  
  ultimately show $a \leq 1$ by (rule Fol1_L5)
  
  { assume $a \neq 1$
    with $A2$ have $a \neq 1$ using OrderedGroup_ZF_1_L3 by simp
    with $A1$ $A2$ have $a^{-1} \in G_+$ using OrderedGroup_ZF_1_L15 by auto
    with $A3$ have $a^{-1} \not\in G_+$ using OrderedGroup_ZF_1_L16 by simp
    ultimately show $a \leq 1$ by (rule Fol1_L5)
  }
  
  ultimately show $a \leq 1$ by (rule Fol1_L5)
moreover
{ assume a≠1
  with A1 A2 have a≤1
    using PositiveSet_def OrderedGroup_ZF_1_L8A
    by auto }
ultimately show a≤1 by auto
qed

The next lemma allows to split proofs that something holds for all a ∈ G into cases a = 1, a ∈ G⁺, −a ∈ G⁺.

lemma (in group3) OrderedGroup_ZF_1_L18:
  assumes A1: r {is total on} G and A2: b∈G
  and A3: Q(1) and A4: ∀ a∈G⁺. Q(a) and A5: ∀ a∈G⁺. Q(a⁻¹)
  shows Q(b)
proof -
  from A1 A2 A3 A4 A5 have Q(b) ∨ Q((b⁻¹)⁻¹)
    using OrderedGroup_ZF_1_L14 by auto
  with A2 show Q(b) using OrderedGroup_ZF_1_L1 group0.group_inv_of_inv
    by simp
qed

All elements greater or equal than an element of G⁺ belong to G⁺.

lemma (in group3) OrderedGroup_ZF_1_L19:
  assumes A1: a ∈ G⁺ and A2: a≤b
  shows b ∈ G⁺
proof -
  from A1 have I: 1≤a and II: a≠1
    using OrderedGroup_ZF_1_L2A by auto
  from I A2 have 1≤b by (rule Group_order_transitive)
  moreover have b≠1
    proof -
      { assume b=1
        with I A2 have 1≤a a≤1
          by auto
        then have 1=a by (rule group_order_antisym)
        with II have False by simp
      } then show b≠1 by auto
    qed
  ultimately show b ∈ G⁺
    using OrderedGroup_ZF_1_L2A by simp
qed

The inverse of an element of G⁺ cannot be in G⁺.

lemma (in group3) OrderedGroup_ZF_1_L20:
  assumes A1: r {is total on} G and A2: a ∈ G⁺
  shows a⁻¹ ∉ G⁺
proof -
  from A2 have a∈G using PositiveSet_def
    by simp

with A1 have Exactly_1_of_3_holds (a=1, a∈G, a⁻¹∈G⁺)
using OrdGroup_decomp by simp
with A2 show a⁻¹ ∉ G⁺ by (rule Fol1_L7)
qed

The set of positive elements of a nontrivial linearly ordered group is not empty.

lemma (in group3) OrderedGroup_ZF_1_L21:
assumes A1: r {is total on} G and A2: G ≠ {1}
shows G⁺ ≠ 0
proof -
have 1 ∈ G using OrderedGroup_ZF_1_L1 group0.group0_2_L2
by simp
with A2 obtain a where a∈G a ≠ 1 by auto
with A1 have a∈G⁺ ∨ a⁻¹∈G⁺
using OrderedGroup_ZF_1_L14 by auto
then show G⁺ ≠ 0 by auto
qed

If b ∈ G⁺, then a < a · b. Multiplying a by a positive element increases a.

lemma (in group3) OrderedGroup_ZF_1_L22:
assumes A1: a∈G b∈G⁺
shows a ≤ a · b a ≠ a · b a · b ∈ G
proof -
from ordGroupAssum A1 have a·1 ≤ a·b
using OrderedGroup_ZF_1_L2A IsAnOrdGroup_def
by simp
with A1 show a≤a·b
using OrderedGroup_ZF_1_L1 group0.group0_2_L2
by simp
then show a·b ∈ G
using OrderedGroup_ZF_1_L4 by simp
{ from A1 have a∈G b∈G
using PositiveSet_def by simp
moreover assume a = a·b
ultimately have b = 1
using OrderedGroup_ZF_1_L1 group0.group0_2_L7
by simp
with A1 have False using PositiveSet_def
by simp
}
} then show a ≠ a·b by auto
qed

If G is a nontrivial linearly ordered group, then for every element of G we can find one in G⁺ that is greater or equal.

lemma (in group3) OrderedGroup_ZF_1_L23:
assumes A1: r {is total on} G and A2: G ≠ {1}
and A3: a∈G

441
shows $\exists b \in G^+. a \leq b$

proof -
{ assume A4: $a \in G^+$ then have $a \leq 1$
  using PositiveSet_def OrderedGroup_ZF_1_L3
  by simp
  with A4 have $\exists b \in G^+. a \leq b$ by auto }

moreover
{ assume $a \notin G^+$
  with A1 A3 have $a \leq 1$ using OrderedGroup_ZF_1_L17
  by simp
  from A1 A2 obtain $b$ where II: $b \in G^+$
    using OrderedGroup_ZF_1_L21 by auto
  then have $1 \leq b$ using PositiveSet_def by simp
  with I have $a \leq b$ by (rule Group_order_transitive)
  with II have $\exists b \in G^+. a \leq b$ by auto }

ultimately show thesis by auto

qed

The $G^+$ is $G_+$ plus the unit.

lemma (in group3) OrderedGroup_ZF_1_L24: shows $G^+ = G_+ \cup \{1\}$
  using OrderedGroup_ZF_1_L2 OrderedGroup_ZF_1_L2A OrderedGroup_ZF_1_L3A
  by auto

What is $-G_+$, really?

lemma (in group3) OrderedGroup_ZF_1_L25: shows $\langle -G_+ \rangle = \{a^{-1}. a \in G_+\}$
  (-G_+) $\subseteq$ G
proof -
{ from ordGroupAssum have I: GroupInv(G,P) : G$\rightarrow$G
  using IsAnOrdGroup_def group0_2_T2 by simp
  moreover have $G_+ \subseteq G$ using PositiveSet_def by auto
  ultimately show $\langle -G_+ \rangle = \{a^{-1}. a \in G_+\}$
  $\langle -G_+ \rangle \subseteq G$
    using func_imagedef func1_1_L6 by auto
}

qed

If the inverse of $a$ is in $G_+$, then $a$ is in the inverse of $G_+$.

lemma (in group3) OrderedGroup_ZF_1_L26: assumes A1: $a \in G$ and A2: $a^{-1} \in G_+$
  shows $a \in \langle -G_+ \rangle$
proof -
{ from A1 have $a^{-1} \in G$ a = $(a^{-1})^{-1}$ using OrderedGroup_ZF_1_L1
  group0.inverse_in_group group0.group_inv_of_inv
  by auto
  with A2 show $a \in \langle -G_+ \rangle$ using OrderedGroup_ZF_1_L25
    by auto
}

qed
If \( a \) is in the inverse of \( G_+ \), then its inverse is in \( G_+ \).

**Lemma** (in group3) **OrderedGroup_ZF_1_L27**:
assumes \( a \in (-G_+) \)
shows \( a^{-1} \in G_+ \)
using assms OrderedGroup_ZF_1_L25 PositiveSet_def
  OrderedGroup_ZF_1_L1 group0.group_inv_of_inv
by auto

A linearly ordered group can be decomposed into \( G_+ \), \( \{1\} \) and \( -G_+ \)

**Lemma** (in group3) **OrdGroup_decomp2**:
assumes \( A1: r \text{ is total on } G \)
shows \( G = G_+ \cup (-G_+) \cup \{1\} \)
\( G_+ \cap (-G_+) = 0 \)
\( 1 \notin G_+ \cup (-G_+) \)
proof -
\{ fix \( a \) assume \( A2: a \in G \)
  with \( A1 \) have \( a \in G_+ \lor a^{-1} \in G_+ \lor a = 1 \)
  using OrderedGroup_ZF_1_L14 by auto
  with \( A2 \) have \( a \in G_+ \lor a \in (-G_+) \lor a = 1 \)
  using OrderedGroup_ZF_1_L26 by auto
  then have \( a \in G_+ \cup (-G_+) \cup \{1\} \)
    by auto
\} then have \( G \subseteq G_+ \cup (-G_+) \cup \{1\} \)
  by auto
moreover have \( G_+ \cup (-G_+) \cup \{1\} \subseteq G \)
  using OrderedGroup_ZF_1_L25 PositiveSet_def
  OrderedGroup_ZF_1_L1 group0.group0_2_L2
by auto
ultimately show \( G = G_+ \cup (-G_+) \cup \{1\} \) by auto
\{ let \( A = G_+ \cap (-G_+) \)
  assume \( A \neq 0 \) by simp
  then obtain \( a \in A \) by blast
  then have \( False \) using OrderedGroup_ZF_1_L15 OrderedGroup_ZF_1_L27
    by auto
\} then show \( G_+ \cap (-G_+) = 0 \) by auto
show \( 1 \notin G_+ \cup (-G_+) \)
  using OrderedGroup_ZF_1_L27
  OrderedGroup_ZF_1_L1 group0.group_inv_of_one
  OrderedGroup_ZF_1_L2A by auto
qed

If \( a \cdot b^{-1} \) is nonnegative, then \( b \leq a \). This maybe used to recover the order from the set of nonnegative elements and serve as a way to define order by prescribing that set (see the "Alternative definitions" section).

**Lemma** (in group3) **OrderedGroup_ZF_1_L28**:
assumes \( A1: a \in G \) \( b \in G \) and \( A2: a \cdot b^{-1} \in G^+ \)
shows \( b \leq a \)

proof -

from A2 have \( 1 \leq a \cdot b^{-1} \) using OrderedGroup_ZF_1_L2
  by simp
with A1 show \( b \leq a \) using OrderedGroup_ZF_1_L5K
  by simp
qed

A special case of OrderedGroup_ZF_1_L28 when \( a \cdot b^{-1} \) is positive.

corollary (in group3) OrderedGroup_ZF_1_L29:
assumes A1: \( a \in G \) \( b \in G \) and A2: \( a \cdot b^{-1} \in G^+ \)
shows \( b \leq a \) \( b \neq a \)
proof -
from A2 have \( 1 \leq a \cdot b^{-1} \) and I: \( a \cdot b^{-1} \neq 1 \)
  using OrderedGroup_ZF_1_L2A by auto
with A1 have \( a \cdot b^{-1} = 1 \) \( a \cdot b^{-1} \in G^+ \) \( (a \cdot b^{-1})^{-1} \in G^+ \)
  using OrderedGroup_ZF_1_L14 by simp
moreover
  { assume \( a \cdot b^{-1} = 1 \)
    then have \( a \cdot b^{-1} \cdot b = 1 \cdot b \) by simp
      with A2 have \( a = b \) \( (a \leq b \land a \neq b) \) \( (b \leq a \land b \neq a) \)
        using OrderedGroup_ZF_1_L1
      group0.inv_cancel_two group0.group0_2_L2 by auto }
moreover
  { assume \( a \cdot b^{-1} \in G^+ \)
    with A2 have \( a = b \) \( (a \leq b \land a \neq b) \) \( (b \leq a \land b \neq a) \)
      using OrderedGroup_ZF_1_L1
  444 group0.group0_2_L6 by auto }

qed

A bit stronger that OrderedGroup_ZF_1_L29, adds case when two elements are equal.

lemma (in group3) OrderedGroup_ZF_1_L30:
assumes a\in G \( b \in G \) and a=b \or a<b \or b<a
shows a\leq b
using assms OrderedGroup_ZF_1_L3 OrderedGroup_ZF_1_L29
by auto

A different take on decomposition: we can have \( a = b \) or \( a < b \) or \( b < a \).

lemma (in group3) OrderedGroup_ZF_1_L31:
assumes A1: \( r \) (is total on) \( G \) and A2: \( a \in G \) \( b \in G \)
shows a\leq b \( \lor \) \( (a \leq b \land a \neq b) \) \( \lor \) \( (b \leq a \land b \neq a) \)
proof -
from A2 have \( a \cdot b^{-1} \in G \) using OrderedGroup_ZF_1_L1
  group0.inverse_in_group group0.group_op_closed
  by simp
with A1 have \( a \cdot b^{-1} = 1 \) \( a \cdot b^{-1} \in G^+ \) \( (a \cdot b^{-1})^{-1} \in G^+ \)
  using OrderedGroup_ZF_1_L14 by simp
moreover
  { assume \( a \cdot b^{-1} = 1 \)
    then have \( a \cdot b^{-1} \cdot b = 1 \cdot b \) by simp
      with A2 have \( a = b \) \( (a \leq b \land a \neq b) \) \( (b \leq a \land b \neq a) \)
        using OrderedGroup_ZF_1_L1
      group0.inv_cancel_two group0.group0_2_L2 by auto }
moreover
  { assume \( a \cdot b^{-1} \in G^+ \)
    with A2 have \( a = b \) \( (a \leq b \land a \neq b) \) \( (b \leq a \land b \neq a) \)
      using OrderedGroup_ZF_1_L1
  444 group0.group0_2_L6 by auto }

Moreover

\{ assume \((a\cdot b^{-1})^{-1} \in G\),
with \(A2\) have \(b\cdot a^{-1} \in G^+\) using OrderedGroup_ZF_1_L1
\quad group0.group0_2_L12 by simp
with \(A2\) have \(a=b \vee (a\leq b \land a\neq b) \vee (b\leq a \land b\neq a)\)
\quad using OrderedGroup_ZF_1_L29 by auto \}
ultimately show \(a=b \vee (a\leq b \land a\neq b) \vee (b\leq a \land b\neq a)\)
\quad by auto
\qed

38.4 Intervals and bounded sets

Intervals here are the closed intervals of the form \(\{x \in G. a \leq x \leq b\}\).

A bounded set can be translated to put it in \(G^+\) and then it is still bounded above.

**Lemma (in group3) OrderedGroup_ZF_2_L1:**

assumes \(A1: \forall g \in A. L \leq g \land g \leq M\)
and \(A2: S = \text{RightTranslation}(G,P,L^{-1})\)
and \(A3: a \in S(A)\)
shows \(a \leq M \cdot L^{-1} \quad 1 \leq a\)

**Proof -**

from \(A3\) have \(A \neq 0\) using func1_1_L13A by fast
then obtain \(g\) where \(g \in A\) by auto
with \(A1\) have \(T1: L \in G \quad M \in G \quad L^{-1} \in G\)
\quad using OrderedGroup_ZF_1_L4 OrderedGroup_ZF_1_L1
\quad group0.inverse_in_group by auto
with \(A2\) have \(S : G \rightarrow G\) using OrderedGroup_ZF_1_L1 group0.group0_5_L1
\quad by simp
moreover from \(A1\) have \(T2: A \subseteq G\) using OrderedGroup_ZF_1_L1_L4 by auto
ultimately have \(S(A) = \{S(b). b \in A\}\) using func_imagedef
\quad by simp
with \(A3\) obtain \(b\) where \(T3: b \in A \quad a = S(b)\) by auto
with \(A1\) ordGroupAssum \(T1\) have \(b \cdot L^{-1} \leq M \cdot L^{-1} \land L \cdot L^{-1} \leq b \cdot L^{-1} \)
\quad using IsAnOrdGroup_def by auto
with \(T3\) \(A2\) \(T1\) \(T2\) show \(a \leq M \cdot L^{-1} \quad 1 \leq a\)
\quad using OrderedGroup_ZF_1_L1 group0.group0_5_L2 group0.group0_2_L6
\quad by auto
\qed

Every bounded set is an image of a subset of an interval that starts at 1.

**Lemma (in group3) OrderedGroup_ZF_2_L2:**

assumes \(A1: \text{IsBounded}(A,r)\)
shows \(\exists B. \exists g \in G^+. \exists T \in G \rightarrow G. A = T(B) \land B \subseteq \text{Interval}(r,1,g)\)

**Proof -**

\{ assume \(A2: A=0\)
\quad let \(B = 0\)
\quad let \(g = 1\)
let $T = \text{ConstantFunction}(G, 1)$
have $g \in G^+$ using OrderedGroup_ZF_1_L3A by simp
moreover have $T : G \rightarrow G$
  using func1_3_L1 OrderedGroup_ZF_1_L1 group0.group0_2_L2 by simp
moreover from $A2$ have $A = T(B)$ by simp
moreover have $B \subseteq \text{Interval}(r, 1, g)$ by simp
ultimately have
  $\exists B. \exists g \in G^+. \exists T \in G \rightarrow G. A = T(B)$
  by auto
moreover
{ assume $A3$: $A \neq 0$
 with $A1$ have $\exists L. \forall x \in A. \ L \leq x \text{ and } \exists U. \forall x \in A. \ x \leq U$
  using IsBounded_def IsBoundedBelow_def IsBoundedAbove_def
  by auto
then obtain $L, U$ where $D1$: $\forall x \in A. \ L \leq x \land x \leq U$
  by auto
with $A3$ have $T1$: $A \subseteq G$ using OrderedGroup_ZF_1_L4 by auto
from $A3$ obtain $a$ where $a \in A$ by auto
with $D1$ have $T2$: $L \leq a \leq U$ by auto
then have $T3$: $L \leq L^{-1} \leq G$ $U \in G$
  using OrderedGroup_ZF_1_L4 OrderedGroup_ZF_1_L1
  group0.inverse_in_group by auto
let $T = \text{RightTranslation}(G, P, L)$
let $B = \text{RightTranslation}(G, P, L^{-1})(A)$
let $g = U \cdot L^{-1}$
have $g \in G^+$
proof -
  from $T2$ have $L \leq U$ using Group_order_transitive by fast
  with ordGroupAssum $T3$ have $L \cdot L^{-1} \leq g$
using IsAnOrdGroup_def by simp
  with $T3$ show thesis using OrderedGroup_ZF_1_L1 group0.group0_2_L6
OrderedGroup_ZF_1_L2 by simp
qed
moreover from $T3$ have $T : G \rightarrow G$
  using OrderedGroup_ZF_1_L1 group0.group0_5_L1
  by simp
moreover have $A = T(B)$
proof -
  from $T3$ $T1$ have $T(B) = \{a \cdot L^{-1} \cdot L. \ a \in A\}$
using OrderedGroup_ZF_1_L1 group0.group0_5_L6
  by simp
moreover from $T3$ $T1$ have $\forall a \in A. \ a \cdot L^{-1} \cdot L = a \cdot (L^{-1} \cdot L)$
using OrderedGroup_ZF_1_L1 group0.group_oper_assoc by auto
ultimately have $T(B) = \{a \cdot (L^{-1} \cdot L). \ a \in A\}$ by simp
with $T3$ have $T(B) = \{a \cdot L^{-1} \cdot L. \ a \in A\}$
using OrderedGroup_ZF_1_L1 group0.group0_2_L6 by simp
moreover from $T1$ have $\forall a \in A. \ a \cdot L^{-1} \cdot L = a$
using OrderedGroup_ZF_1_L1 group0.group0_2_L2 by auto
ultimately show thesis by simp

qed
moreover have $B \subseteq \text{Interval}(r,1,g)$
proof
  fix $y$ assume $A4$: $y \in B$
  let $S = \text{RightTranslation}(G,P,L^{-1})$
  from D1 have $T4$: $\forall x \in A. \ L \leq x \land x \leq U$ by simp
  moreover have $T5$: $S = \text{RightTranslation}(G,P,L^{-1})$
by simp
  moreover from $A4$ have $T6$: $y \in S(A)$ by simp
  ultimately have $y \leq U \cdot L^{-1}$ using OrderedGroup_ZF_2_L1
by blast
  moreover from $T4$ $T5$ $T6$ have $1 \leq y$ by (rule OrderedGroup_ZF_2_L1)
  ultimately show $y \in \text{Interval}(r,1,g)$ using Interval_def by auto
qed
ultimately have $\exists B. \exists g \in G^+. \exists \mathcal{T} : G \rightarrow G. A = \mathcal{T}(B) \land B \subseteq \text{Interval}(r,1,g)$
by auto
ultimately show thesis by auto
qed

If every interval starting at 1 is finite, then every bounded set is finite. I find it interesting that this does not require the group to be linearly ordered (the order to be total).

**Theorem (in group3) OrderedGroup_ZF_2_T1:**
assumes $A1$: $\forall g \in G^+. \ \text{Interval}(r,1,g) \in \text{Fin}(G)$
and $A2$: $\text{IsBounded}(A,r)$
shows $A \in \text{Fin}(G)$
proof
  from $A2$ have $\exists B. \exists g \in G^+. \exists \mathcal{T} : G \rightarrow G. A = \mathcal{T}(B) \land B \subseteq \text{Interval}(r,1,g)$
  using OrderedGroup_ZF_2_L2 by simp
  then obtain $B$ $g$ $\mathcal{T}$ where $D1$: $g \in G^+$ $B \subseteq \text{Interval}(r,1,g)$
  and $D2$: $\mathcal{T} : G \rightarrow G$ $A = \mathcal{T}(B)$ by auto
  from $D1$ $A1$ have $B \in \text{Fin}(G)$ using Fin_subset_lemma by blast
  with $D2$ show thesis using Finite1_L6A by simp
qed

In linearly ordered groups finite sets are bounded.

**Theorem (in group3) ord_group_fin_bounded:**
assumes $r$ {is total on} $G$ and $B \in \text{Fin}(G)$
shows $\text{IsBounded}(B,r)$
using ordGroupAssum assms IsAnOrdGroup_def IsPartOrder_def Finite_ZF_1_T1 by simp

For nontrivial linearly ordered groups if for every element $G$ we can find one in $A$ that is greater or equal (not necessarily strictly greater), then $A$ can neither be finite nor bounded above.

**Lemma (in group3) OrderedGroup_ZF_2_L2A:**
assumes $A_1$: $r$ {is total on} $G$ and $A_2$: $G \neq \{1\}$
and $A_3$: $\forall a \in G. \exists b \in A. a \leq b$
shows
$\forall a \in G. \exists b \in A. a \neq b \land a \leq b$
$\neg \text{IsBoundedAbove}(A, r)$
$\ A \notin \text{Fin}(G)$

proof -
{ fix $a$
  from $A_1$ $A_2$ obtain $c$ where $c \in G_+$
    using OrderedGroup_ZF_1_L21 by auto
  moreover assume $a \in G$
  ultimately have
    $a \cdot c \in G$ and $I: a < a \cdot c$
    using OrderedGroup_ZF_1_L22 by auto
  with $A_3$ obtain $b$ where $II: b \in A$ and $III: a \cdot c \leq b$
    by auto
    moreover from $I$ $III$ have $a < b$ by (rule OrderedGroup_ZF_1_L4A)
    ultimately have $\exists b \in A. a \neq b \land a \leq b$ by auto
} thus $\forall a \in G. \exists b \in A. a \neq b \land a \leq b$ by simp
with ordGroupAssum $A_1$ show
  $\neg \text{IsBoundedAbove}(A, r)$
  $\ A \notin \text{Fin}(G)$
using IsAnOrdGroup_def IsPartOrder_def
OrderedGroup_ZF_1_L1A Order_ZF_3_L14 Finite_ZF_1_1_L3
by auto

qed

Nontrivial linearly ordered groups are infinite. Recall that $\text{Fin}(A)$ is the collection of finite subsets of $A$. In this lemma we show that $G \notin \text{Fin}(G)$, that is that $G$ is not a finite subset of itself. This is a way of saying that $G$ is infinite. We also show that for nontrivial linearly ordered groups $G_+$ is infinite.

theorem (in group3) Linord_group_infinite:
  assumes $A_1$: $r$ {is total on} $G$ and $A_2$: $G \neq \{1\}$
  shows
  $G_+ \notin \text{Fin}(G)$
  $G \notin \text{Fin}(G)$
proof -
  from $A_1$ $A_2$ show $I: G_+ \notin \text{Fin}(G)$
    using OrderedGroup_ZF_1_L23 OrderedGroup_ZF_2_L2A
    by simp
  { assume $G \in \text{Fin}(G)$
    moreover have $G_+ \subseteq G$ using PositiveSet_def by auto
    ultimately have $G_+ \in \text{Fin}(G)$ using Fin_subset_lemma
    by blast
    with $I$ have False by simp
  } then show $G \notin \text{Fin}(G)$ by auto

qed
A property of nonempty subsets of linearly ordered groups that don’t have a maximum: for any element in such subset we can find one that is strictly greater.

**lemma** (in group3) **OrderedGroup_ZF_2_L2B**:

- assumes \( A1: r \{ \text{is total on} \} G \) and \( A2: A \subseteq G \) and
- \( A3: \neg \text{HasAmaximum}(r,A) \) and \( A4: x \in A \)

**shows** \( \exists y \in A. \ x < y \)

**proof** -

- from ordGroupAssum assms have
  - \( \text{antisym}(r) \)
  - \( r \{ \text{is total on} \} G \)
  - \( A \subseteq G \)
  - \( \neg \text{HasAmaximum}(r,A) \)
  - \( x \in A \)
- using IsAnOrdGroup_def IsPartOrder_def by auto

- then have \( \exists y \in A. \ (x,y) \in r \land y \neq x \)
- using Order_ZF_4_L16 by simp

- then show \( \exists y \in A. \ x < y \)
  - by auto

**qed**

In linearly ordered groups \( G \setminus G_+ \) is bounded above.

**lemma** (in group3) **OrderedGroup_ZF_2_L3**:

- assumes \( A1: r \{ \text{is total on} \} G \)
- **shows** IsBoundedAbove(\( G \setminus G_+ \),r)

**proof** -

- from \( A1 \) have \( \forall a \in G \setminus G_+. \ a \leq 1 \)
- using OrderedGroup_ZF_1_L17 by simp

- then show IsBoundedAbove(\( G \setminus G_+,r \))
  - using IsBoundedAbove_def by auto

**qed**

In linearly ordered groups if \( A \cap G_+ \) is finite, then \( A \) is bounded above.

**lemma** (in group3) **OrderedGroup_ZF_2_L4**:

- assumes \( A1: r \{ \text{is total on} \} G \)
- \( A2: A \subseteq G \)
- \( A3: A \cap G_+ \in \text{Fin}(G) \)
- **shows** IsBoundedAbove(\( A,r \))

**proof** -

- have \( A \cap (G \setminus G_+) \subseteq G \setminus G_+ \) by auto

- with \( A1 \) have IsBoundedAbove(\( A \cap (G \setminus G_+),r \))
  - using OrderedGroup_ZF_2_L3 Order_ZF_3_L13 by blast

- moreover from \( A1 \) \( A3 \) have IsBoundedAbove(\( A \cap G_+,r \))
  - using ord_group_fin_bounded IsBounded_def by simp

- moreover from \( A1 \) ordGroupAssum have
  - \( r \{ \text{is total on} \} G \)
  - \( \text{trans}(r) \)
  - \( r \subseteq G \times G \)
  - using IsAnOrdGroup_def IsPartOrder_def by auto

- ultimately have IsBoundedAbove(\( A \cap (G \setminus G_+) \cup A \cap G_+,r \))
  - using Order_ZF_3_L3 by simp

- moreover from \( A2 \) have \( A = A \cap (G \setminus G_+) \cup A \cap G_+ \)

449
If a set $-A \subseteq G$ is bounded above, then $A$ is bounded below.

**lemma** (in group3) OrderedGroup_ZF_2_L5:

assumes $A1: A \subseteq G$ and $A2: \text{IsBoundedAbove}(-A,r)$

shows $\text{IsBoundedBelow}(A,r)$

**proof** -

{ assume $A = 0$ then have $\text{IsBoundedBelow}(A,r)$ using $\text{IsBoundedBelow}_\text{def}$ by auto }

moreover

{ assume $A3: A \neq 0$
  from ordGroupAssum have $I: \text{GroupInv}(G,P) : G \rightarrow G$
    using $\text{IsAnOrdGroup}_\text{def}$ group0_2_T2 by simp
  with $A1$ $A2$ $A3$ obtain $u$ where $D$: $\forall a \in (-A). \ a \leq u$
    using $\text{func1}_1_\text{L15}_A$ $\text{IsBoundedAbove}_\text{def}$ by auto
  { fix $b$ assume $b \in A$
    with $A1$ $I$ $D$ have $b^{-1} \leq u$ and $T$: $b \in G$
      using $\text{func}_\text{imagedef}$ by auto
    then have $u^{-1} \leq (b^{-1})^{-1}$ using $\text{OrderedGroup}_1_\text{L5}$
      by simp
    with $T$ have $u^{-1} \leq b$
      using $\text{OrderedGroup}_1_\text{L1}$ $\text{group}_0$.group_inv_of_inv by simp
    } then have $\forall b \in A. \ (u^{-1}, b) \in r$ by simp
    then have $\text{IsBoundedBelow}(A,r)$
      using $\text{Order}_2_\text{L9}$ by blast }

ultimately show thesis by auto

qed

If $a \leq b$, then the image of the interval $a..b$ by any function is nonempty.

**lemma** (in group3) OrderedGroup_ZF_2_L6:

assumes $a \leq b$ and $f:G \rightarrow G$

shows $f(\text{Interval}(r,a,b)) \neq 0$

using ordGroupAssum assms $\text{OrderedGroup}_1_\text{L4}$

$\text{Order}_2_\text{L6}$ $\text{Order}_2_\text{L2A}$

$\text{IsAnOrdGroup}_\text{def}$ $\text{IsPartOrder}_\text{def}$ $\text{func1}_1_\text{L15}_A$

by auto

end

39 More on ordered groups

theory OrderedGroup_ZF_1 imports OrderedGroup_ZF

begin

In this theory we continue the OrderedGroup_ZF theory development.
39.1 Absolute value and the triangle inequality

The goal of this section is to prove the triangle inequality for ordered groups.

Absolute value maps $G$ into $G$.

**Lemma (in group3) OrderedGroup_ZF_3_L1:**

This shows that $\text{AbsoluteValue}(G,P,r) : G \rightarrow G$.

**Proof** -

Let $f = \text{id}(G^+)$

Let $g = \text{restrict}(\text{GroupInv}(G,P),G-G^+)$

Have $f : G^+ \rightarrow G^+$ using $\text{id_type}$ by simp

Then have $f : G^+ \rightarrow G$ using $\text{OrderedGroup_ZF_1_L4E}$ $\text{fun_weaken_type}$

by blast

Moreover have $g : G-G^+ \rightarrow G$

- From $\text{ordGroupAssum}$ have $\text{GroupInv}(G,P) : G \rightarrow G$
  - using $\text{IsAnOrdGroup_def}$ $\text{group0_2_T2}$ by simp
  - Moreover have $G-G^+ \subseteq G$ by auto
  - Ultimately show thesis using $\text{restrict_type2}$ by simp

Qed

Moreover have $G^+ \cap (G-G^+) = 0$ by blast

Ultimately have $f \cup g : G^+ \cup (G-G^+) \rightarrow G \cup G$

by (rule $\text{fun_disjoint_Un}$)

Moreover have $G^+ \cup (G-G^+) = G$ using $\text{OrderedGroup_ZF_1_L4E}$

by auto

Ultimately show $\text{AbsoluteValue}(G,P,r) : G \rightarrow G$

using $\text{AbsoluteValue_def}$ by simp

Qed

If $a \in G^+$, then $|a| = a$.

**Lemma (in group3) OrderedGroup_ZF_3_L2:**

This assumes $A1: a \in G^+$ shows $|a| = a$.

**Proof** -

From $\text{ordGroupAssum}$ have $\text{GroupInv}(G,P) : G \rightarrow G$

using $\text{IsAnOrdGroup_def}$ $\text{group0_2_T2}$ by simp

With $A1$ show thesis using

$\text{func1_1_L1}$ $\text{OrderedGroup_ZF_1_L4E}$ $\text{fun_disjoint_apply1}$

$\text{AbsoluteValue_def}$ $\text{id_conv}$ by simp

Qed

The absolute value of the unit is the unit. In the additive totation that would be $|0| = 0$.

**Lemma (in group3) OrderedGroup_ZF_3_L2A:**

This shows $|1| = 1$ using $\text{OrderedGroup_ZF_1_L3A}$ $\text{OrderedGroup_ZF_3_L2}$

by simp

If $a$ is positive, then $|a| = a$.

**Lemma (in group3) OrderedGroup_ZF_3_L2B:**
assumes $a \in G$, shows $|a| = a$
using assms PositiveSet_def Nonnegative_def OrderedGroup_ZF_3_L2
by auto

If $a \in G \setminus G^+$, then $|a| = a^{-1}$.

lemma (in group3) OrderedGroup_ZF_3_L3:
assumes $A1: a \in G \setminus G^+$ shows $|a| = a^{-1}$
proof -
  have domain(id($G^+$)) = $G^+$
    using id_type func1_1_L1 by auto
  with $A1$ show thesis using fun_disjoint_apply2 AbsoluteValue_def
    restrict by simp
qed

For elements that not greater than the unit, the absolute value is the inverse.

lemma (in group3) OrderedGroup_ZF_3_L3A:
assumes $A1: a \leq 1$
shows $|a| = a^{-1}$
proof -
  { assume $a=1$
    then have $|a| = a^{-1}$
      using OrderedGroup_ZF_3_L2A OrderedGroup_ZF_1_L1 group0.group_inv_of_one
      by simp }
  moreover
  { assume $a \neq 1$
    with $A1$ have $|a| = a^{-1}$ using OrderedGroup_ZF_1_L4C OrderedGroup_ZF_3_L3
      by simp }
  ultimately show $|a| = a^{-1}$ by blast
qed

In linearly ordered groups the absolute value of any element is in $G^+$.

lemma (in group3) OrderedGroup_ZF_3_L3B:
assumes $A1: r \text{ is total on } G$ and $A2: a \in G$
shows $|a| \in G^+$
proof -
  { assume $a \in G^+$ then have $|a| \in G^+$
    using OrderedGroup_ZF_3_L2 by simp }
  moreover
  { assume $a \notin G^+$
    with $A1$ have $|a| \in G^+$ using OrderedGroup_ZF_3_L3
      OrderedGroup_ZF_1_L6 by simp }
  ultimately show $|a| \in G^+$ by blast
qed

For linearly ordered groups (where the order is total), the absolute value
maps the group into the positive set.

lemma (in group3) OrderedGroup_ZF_3_L3C:
assumes $A1: r \text{ is total on } G$
shows AbsoluteValue($G,P,r$) : $G \rightarrow G^+$

452
proof-
have AbsoluteValue(G,P,r) : G→G using OrderedGroup_ZF_3_L1 by simp
moreover from A1 have T2:
  ∀g∈G. AbsoluteValue(G,P,r)(g) ∈ G^+
  using OrderedGroup_ZF_3_L3B by simp
ultimately show thesis by (rule func1_1_L1A)
qued

If the absolute value is the unit, then the element is the unit.

lemma (in group3) OrderedGroup_ZF_3_L3D:
  assumes A1: a∈G and A2: |a| = 1
  shows a = 1
proof -
  { assume a ∈ G^+
    with A1 have a = 1 using OrderedGroup_ZF_3_L2 by simp }
moreover
  { assume a /∈ G^+
    with A1 A2 have a = 1 using
      OrderedGroup_ZF_3_L3 OrderedGroup_ZF_1_L1 group0.group0_2_L8A
      by auto }
ultimately show a = 1 by blast
qued

In linearly ordered groups the unit is not greater than the absolute value of any element.

lemma (in group3) OrderedGroup_ZF_3_L3E:
  assumes r {is total on} G and a∈G
  shows 1 ≤ |a|
  using assms OrderedGroup_ZF_3_L3B OrderedGroup_ZF_1_L2 by simp

If b is greater than both a and a^−1, then b is greater than |a|.

lemma (in group3) OrderedGroup_ZF_3_L4:
  assumes A1: a≤b and A2: a^−1≤ b
  shows |a|≤ b
proof -
  { assume a∈G^+
    with A1 have |a|≤ b using OrderedGroup_ZF_3_L2 by simp }
moreover
  { assume a∉G^+
    with A1 A2 have |a|≤ b
      using OrderedGroup_ZF_1_L4 OrderedGroup_ZF_3_L3 by simp }
ultimately show |a|≤ b by blast
qued

In linearly ordered groups a ≤ |a|.

lemma (in group3) OrderedGroup_ZF_3_L5:
  assumes A1: r {is total on} G and A2: a∈G
shows $a \leq |a|$

proof -

\[
\{ \text{ assume } a \in G^+ \text{ with } A2 \text{ have } a \leq |a| \} \\
\text{ using } \text{OrderedGroup(ZF)_3_L2 OrderedGroup(ZF)_1_L3 by simp } \}
\]

moreover

\[
\{ \text{ assume } a \notin G^+ \text{ with } A1 A2 \text{ have } a \leq |a| \\
\text{ using } \text{OrderedGroup(ZF)_3_L3B OrderedGroup(ZF)_1_L4B by simp } \}
\]

ultimately show $a \leq |a|$ by blast

qed

$a^{-1} \leq |a|$ (in additive notation it would be $-a \leq |a|$).

\textbf{lemma (in group3) OrderedGroup(ZF)_3_L6:}

\begin{itemize}
  \item \textbf{assumes } A1: $a \in G$ \textbf{shows } $a^{-1} \leq |a|$\end{itemize}

\textbf{proof -}

\[
\{ \text{ assume } a \in G^+ \text{ then have } T1: 1 \leq a \text{ and } T2: |a| = a \text{ using } \text{OrderedGroup(ZF)_1_L2} \\
\text{ OrderedGroup(ZF)_3_L2 by auto } \}
\text{ then have } a^{-1} \leq 1^{-1} \text{ using } \text{OrderedGroup(ZF)_1_L5 by simp } \}
\text{ then have T3: } a^{-1} \leq 1 \text{ using } \text{OrderedGroup(ZF)_1_L1 group0.group_inv_of_one by simp } \}
\text{ from T3 T1 have } a^{-1} \leq a \text{ by (rule Group_order_transitive) } \}
\text{ with T2 have } a^{-1} \leq |a| \text{ by simp } \}
\]

moreover

\[
\{ \text{ assume } A2: a \notin G^+ \text{ from } A1 \text{ have } |a| \in G \\
\text{ using } \text{OrderedGroup(ZF)_3_L1 apply_funtype by auto } \}
\text{ with ordGroupAssum have } |a| \leq |a| \text{ using } \text{IsAnOrdGroup_def IsPartOrder_def refl_def by simp } \}
\text{ with A1 A2 have } a^{-1} \leq |a| \text{ using } \text{OrderedGroup(ZF)_3_L3 by simp } \}
\]

ultimately show $a^{-1} \leq |a|$ by blast

qed

Some inequalities about the product of two elements of a linearly ordered group and its absolute value.

\textbf{lemma (in group3) OrderedGroup(ZF)_3_L6A:}

\begin{itemize}
  \item \textbf{assumes } r \text{ {is total on} } G \text{ and } a \in G \text{ b} \in G \text{ shows } \\
  a \cdot b \leq |a| \cdot |b| \\
  a \cdot b^{-1} \leq |a| \cdot |b| \\
  a^{-1} \cdot b \leq |a| \cdot |b| \\
  a^{-1} \cdot b^{-1} \leq |a| \cdot |b| \\
  \text{ using } \text{assms OrderedGroup(ZF)_3_L5 OrderedGroup(ZF)_3_L6} \\
  \text{ OrderedGroup(ZF)_1_L5B by auto } \}
\end{itemize}

$|a^{-1}| \leq |a|$. 

\textbf{lemma (in group3) OrderedGroup(ZF)_3_L7:}

\begin{itemize}
  \item \textbf{assumes } r \text{ {is total on} } G \text{ and } a \in G \\
\end{itemize}

454
shows $|a^{-1}| \leq |a|$
using assms OrderedGroup_ZF_3_L5 OrderedGroup_ZF_1_L1 group0.group_inv_of_inv
OrderedGroup_ZF_3_L6 OrderedGroup_ZF_3_L4 by simp

$|a^{-1}| = |a|$. 

**lemma** (in group3) OrderedGroup_ZF_3_L7A:
assumes A1: r {is total on} G and A2: a$\in$G
shows $|a^{-1}| = |a|$
**proof**
from A2 have $a^{-1} \in G$ using OrderedGroup_ZF_1_L1 group0.inverse_in_group
by simp
with A1 have $|(a^{-1})^{-1}| \leq |a^{-1}|$ using OrderedGroup_ZF_3_L7 by simp
with A1 A2 have $|a^{-1}| \leq |a|$ |a$| \leq |a^{-1}|$
using OrderedGroup_ZF_1_L1 group0.group_inv_of_inv OrderedGroup_ZF_3_L7
by auto
then show thesis by (rule group_order_antisym)
qed

$|a \cdot b^{-1}| = |b \cdot a^{-1}|$. It doesn’t look so strange in the additive notation:
$|a-b| = |b-a|$. 

**lemma** (in group3) OrderedGroup_ZF_3_L7B:
assumes A1: r {is total on} G and A2: a$\in$G b$\in$G
shows $|a \cdot b^{-1}| = |b \cdot a^{-1}|$
**proof**
from A1 A2 have $(a \cdot b^{-1})^{-1} = a \cdot b^{-1}$ using 
OrderedGroup_ZF_1_L1 group0.inverse_in_group group0.group0_2_L1
monoid0.group0_1_L1 OrderedGroup_ZF_3_L7A by simp
moreover from A2 have $(a \cdot b^{-1})^{-1} = b \cdot a^{-1}$
using OrderedGroup_ZF_1_L1 group0.group0_2_L12 by simp
ultimately show thesis by simp
qed

Triangle inequality for linearly ordered abelian groups. It would be nice to drop commutativity or give an example that shows we can't do that.

**theorem** (in group3) OrdGroup_triangle_ineq:
assumes A1: P {is commutative on} G
and A2: r {is total on} G and A3: a$\in$G b$\in$G
shows $|a \cdot b| \leq |a||b|$
**proof**
from A1 A2 A3 have $a \leq |a| \ b \leq |b| \ a^{-1} \leq |a| \ b^{-1} \leq |b|$
using OrderedGroup_ZF_3_L5 OrderedGroup_ZF_3_L6 by auto
then have $a \cdot b \leq |a||b| \ a^{-1} \cdot b^{-1} \leq |a||b|$
using OrderedGroup_ZF_1_L5B by auto
with A1 A3 show $|a \cdot b| \leq |a||b|$
using OrderedGroup_ZF_1_L1 group0.group0_2_L12 IsCommutative_def
OrderedGroup_ZF_3_L4 by simp

455
We can multiply the sides of an inequality with absolute value.

**lemma (in group3) OrderedGroup_ZF_3_L7C:**

assumes \( P \) {is commutative on} \( G \)
and \( r \) {is total on} \( G \)
\( a \in G \) \( b \in G \)

and \( |a| \leq c \) \( |b| \leq d \)

shows \( |a \cdot b| \leq c \cdot d \)

**proof -**

from \( \text{assms}(1,2,3,4) \) have \( |a \cdot b| \leq |a| \cdot |b| \)
using \( \text{OrdGroup_triangle_ineq} \) by simp

moreover from \( \text{assms}(5,6) \) have \( |a| \cdot |b| \leq c \cdot d \)
using \( \text{OrderedGroup_ZF_1_L5B} \) by simp

ultimately show thesis by (rule Group_order_transitive)

**qed**

A version of the \( \text{OrderedGroup_ZF_3_L7C} \) but with multiplying by the inverse.

**lemma (in group3) OrderedGroup_ZF_3_L7CA:**

assumes \( P \) {is commutative on} \( G \)
and \( r \) {is total on} \( G \)
\( a \in G \) \( b \in G \)

and \( |a| \leq c \) \( |b| \leq d \)

shows \( |a \cdot b^{-1}| \leq c \cdot d \)

using \( \text{assms} \) \( \text{OrderedGroup_ZF_1_L1} \) \( \text{group0.inverse_in_group} \)
\( \text{OrderedGroup_ZF_3_L7A} \) \( \text{OrderedGroup_ZF_3_L7C} \) by simp

**Triangle inequality with three integers.**

**lemma (in group3) OrdGroup_triangle_ineq3:**

assumes \( A1: P \) {is commutative on} \( G \)
and \( A2: r \) {is total on} \( G \) and \( A3: a \in G \) \( b \in G \) \( c \in G \)

shows \( |a \cdot b \cdot c| \leq |a| \cdot |b| \cdot |c| \)

**proof -**

from \( A3 \) have \( T: a \cdot b \in G \) \( |c| \in G \)
using \( \text{OrderedGroup_ZF_1_L1} \) \( \text{group0.group_op_closed} \)
\( \text{OrderedGroup_ZF_3_L1} \) \( \text{apply_funtype} \) by auto

with \( A1 \) \( A2 \) \( A3 \) have \( |a \cdot b \cdot c| \leq |a| \cdot |b| \cdot |c| \)
using \( \text{OrdGroup_triangle_ineq} \) by simp

moreover from \( \text{ordGroupAssum} \) \( A1 \) \( A2 \) \( A3 \) \( T \) have
\( |a \cdot b \cdot c| \leq |a| \cdot |b| \cdot |c| \)
using \( \text{OrdGroup_triangle_ineq} \) \( \text{IsAnOrdGroup_def} \) by simp

ultimately show \( |a \cdot b \cdot c| \leq |a| \cdot |b| \cdot |c| \)
by (rule Group_order_transitive)

**qed**

Some variants of the triangle inequality.

**lemma (in group3) OrderedGroup_ZF_3_L7D:**

assumes \( A1: P \) {is commutative on} \( G \)
and \( A2: r \) {is total on} \( G \) and \( A3: a \in G \) \( b \in G \)

and \( A4: |a \cdot b^{-1}| \leq c \)
shows
|a| ≤ c·|b|
|a| ≤ |b|·c
|c|−1·a ≤ b
a·|c|−1 ≤ b
a ≤ b·c
proof -
from A3 A4 have
T: a·b−1 ∈ G |b| ∈ G c∈G c−1 ∈ G
using OrderedGroup_ZF_1_L1
group0.inverse_in_group group0.group0_2_L1 monoid0.group0_1_L1
OrderedGroup_ZF_3_L1 apply_funtype OrderedGroup_ZF_1_L4
by auto
from A3 have |a| = |a·b−1·b|
using OrderedGroup_ZF_1_L1 group0.inv_cancel_two
by simp
with A1 A2 A3 T have |a| ≤ |a·b−1|·|b|
using OrdGroup_triangle_ineq by simp
with T A4 show |a| ≤ c·|b| using OrderedGroup_ZF_1_L5C
by blast
with T A1 show |a| ≤ |b|·c
using IsCommutative_def by simp
from A2 T have a·b−1 ≤ |a·b−1|
using OrderedGroup_ZF_3_L5 by simp
moreover note A4
ultimately have I: a·b−1 ≤ c
by (rule Group_order_transitive)
with A3 show c−1·a ≤ b
using OrderedGroup_ZF_1_L5H by simp
with A1 A3 T show a·c−1 ≤ b
using IsCommutative_def by simp
from A1 A3 T I show a ≤ b·c
using OrderedGroup_ZF_1_L5H IsCommutative_def
by auto
qed

Some more variants of the triangle inequality.

lemma (in group3) OrderedGroup_ZF_3_L7E:
assumes A1: P {is commutative on} G
and A2: r {is total on} G and A3: a∈G b∈G
and A4: |a·b−1| ≤ c
shows b·c−1 ≤ a
proof -
from A3 have a·b−1 ∈ G
using OrderedGroup_ZF_1_L1
group0.inverse_in_group group0.group_op_closed
by auto
with A2 have |(a·b−1)| = |a·b−1|
using OrderedGroup_ZF_3_L7A by simp
moreover from A3 have \((a \cdot b)^{-1} = b \cdot a^{-1}\)

using OrderedGroup_ZF_1_L1 group0.group0_2_L12
by simp
ultimately have \(|a^{-1}| = |a \cdot b^{-1}|\)
by simp

with A1 A2 A3 A4 show \(b \cdot c^{-1} \leq a\)
using OrderedGroup_ZF_3_L7D by simp

qed

An application of the triangle inequality with four group elements.

lemma (in group3) OrderedGroup_ZF_3_L7F:
assumes A1: \(P \text{ is commutative on } G\)
and A2: \(r \text{ is total on } G\) and
A3: \(a \in G \quad b \in G \quad c \in G \quad d \in G\)
sows \(|a \cdot c^{-1}| \leq |a \cdot b| \cdot |c \cdot d| \cdot |b \cdot d^{-1}|\)
proof -
from A3 have T:
\[a \cdot c^{-1} \in G \quad a \cdot b \in G \quad c \cdot d \in G \quad b \cdot d^{-1} \in G\]
\[(c \cdot d)^{-1} \in G \quad (b \cdot d^{-1})^{-1} \in G\]
using OrderedGroup_ZF_1_L1
  group0.inverse_in_group group0.group_op_closed
by auto
with A1 A2 have \(|(a \cdot b) \cdot (c \cdot d)^{-1} \cdot (b \cdot d^{-1})^{-1}| \leq |a \cdot b| \cdot |c \cdot d| \cdot |b \cdot d^{-1}|\)
using OrdGroup_triangle_ineq3 by simp
moreover from A2 T have \(|(c \cdot d)^{-1}| = |c \cdot d| \quad \text{and} \quad |(b \cdot d^{-1})^{-1}| = |b \cdot d^{-1}|\)
using OrderedGroup_ZF_3_L7A by auto
moreover from A1 A3 have \((a \cdot b) \cdot (c \cdot d)^{-1} \cdot (b \cdot d^{-1})^{-1} = a \cdot c^{-1}\)
using OrderedGroup_ZF_1_L1 group0.group0_4_L8
by simp
ultimately show \(|a \cdot c^{-1}| \leq |a \cdot b| \cdot |c \cdot d| \cdot |b \cdot d^{-1}|\)
by simp

qed

\(|a| \leq L \text{ implies } L^{-1} \leq a\) (it would be \(-L \leq a\) in the additive notation).

lemma (in group3) OrderedGroup_ZF_3_L8:
assumes A1: \(a \in G\) and A2: \(|a| \leq L\)
sows \(L^{-1} \leq a\)
proof -
from A1 have I: \(a^{-1} \leq |a|\) using OrderedGroup_ZF_3_L6 by simp
from I A2 have \(a^{-1} \leq L\) by (rule Group_order_transitive)
then have \(L^{-1} \leq (a^{-1})^{-1}\) using OrderedGroup_ZF_1_L5 by simp
with A1 show \(L^{-1} \leq a\) using OrderedGroup_ZF_1_L1 group0.group_inv_of_inv
  by simp

qed

In linearly ordered groups \(|a| \leq L \text{ implies } a \leq L\) (it would be \(a \leq L\) in the additive notation).

lemma (in group3) OrderedGroup_ZF_3_L8A:
assumes A1: r {is total on} G
and A2: a\in G and A3: |a|\leq L
shows
a\leq L
1\leq L
proof -
  from A1 A2 have I: a \leq |a| using OrderedGroup_ZF_3_L5 by simp
  from I A3 show a\leq L by (rule Group_order_transitive)
  from A1 A2 A3 have 1 \leq |a| \leq |a|\leq L
    using OrderedGroup.ZF_3_L3B OrderedGroup.ZF_1_L2 by auto
  then show 1\leq L by (rule Group_order_transitive)
qed

A somewhat generalized version of the above lemma.

lemma (in group3) OrderedGroup.ZF_3_L8B:
  assumes A1: a\in G and A2: |a|\leq L and A3: 1\leq c
  shows (L\cdot c)^{-1} \leq a
proof -
  from A1 A2 A3 have c^{-1}\cdot L^{-1} \leq 1^{-1}a
    using OrderedGroup.ZF_3_L8 OrderedGroup.ZF_1_L5AB
    OrderedGroup.ZF_1_L5B by simp
  with A1 A2 A3 show (L\cdot c)^{-1} \leq a
    using OrderedGroup.ZF_1_L4 OrderedGroup.ZF_1_L1
    group0.group_inv_of_two group0.group0_2_L2
    by simp
qed

If b is between a and a \cdot c, then b \cdot a^{-1} \leq c.

lemma (in group3) OrderedGroup.ZF_3_L8C:
  assumes A1: a\leq b and A2: c\in G and A3: b\leq c\cdot a
  shows |b\cdot a^{-1}| \leq c
proof -
  from A1 A2 A3 have b\cdot a^{-1} \leq c
    using OrderedGroup.ZF_2_L9C OrderedGroup.ZF_1_L4
    by simp
  moreover have (b\cdot a^{-1})^{-1} \leq c
    proof -
      from A1 have T: a\in G b\in G
        using OrderedGroup.ZF_1_L4 by auto
      with A1 have a\cdot b^{-1} \leq 1
        using OrderedGroup.ZF_1_L9 by blast
      moreover
      from A1 A3 have a\leq c\cdot a
        by (rule Group_order_transitive)
      with ordGroupAssum T have a\cdot a^{-1} \leq c\cdot a\cdot a^{-1}
        using OrderedGroup.ZF_1_L1 group0.inverse_in_group
        IsAnOrdGroup_def by simp
      with T A2 have 1 \leq c
        using OrderedGroup.ZF_1_L1
end
For linearly ordered groups if the absolute values of elements in a set are bounded, then the set is bounded.

**Lemma** (in group3) OrderedGroup_ZF_3_L9:

assumes
A1: \( r \) \( \text{is total on} \) \( G \)
A2: \( A \subseteq G \) and A3: \( \forall a \in A. \ |a| \leq L \)

shows \( \text{IsBounded}(A,r) \)

proof -
from A1 A2 A3 have
\( \forall a \in A. a \leq a \forall a \in A. L^{-1} \leq a \)
using OrderedGroup_ZF_3_L8 OrderedGroup_ZF_3_L8A by auto
then show thesis using IsBoundedAbove_def IsBoundedBelow_def IsBounded_def by auto
qed

A slightly more general version of the previous lemma, stating the same fact for a set defined by separation.

**Lemma** (in group3) OrderedGroup_ZF_3_L9A:

assumes A1: \( r \) \( \text{is total on} \) \( G \)
and A2: \( \forall x \in X. b(x) \in G \) \( \land \ |b(x)| \leq L \)

shows \( \text{IsBounded}() \)

proof -
from A2 have \( \{b(x). x \in X\} \subseteq G \) \( \forall a \in \{b(x). x \in X\}. \ |a| \leq L \)
by auto
with A1 show thesis using OrderedGroup_ZF_3_L9 by blast
qed

A special form of the previous lemma stating a similar fact for an image of a set by a function with values in a linearly ordered group.

**Lemma** (in group3) OrderedGroup_ZF_3_L9B:

assumes A1: \( r \) \( \text{is total on} \) \( G \)
and A2: \( f: X \rightarrow G \) and A3: \( A \subseteq X \)
and A4: \( \forall x \in A. \ |f(x)| \leq L \)

shows \( \text{IsBounded}(f(A),r) \)

proof -
from A2 A3 A4 have \( \forall x \in A. f(x) \in G \) \( \land \ |f(x)| \leq L \)
using apply_funtype by auto
with A1 have IsBounded({f(x). x∈A}, r)
  by (rule OrderedGroup_ZF_3_L9A)
with A2 A3 show IsBounded(f(A), r)
  using func_imagedef by simp
qed

For linearly ordered groups if \( l \leq a \leq u \) then \(|a|\) is smaller than the greater of \(|l|, |u|\).

lemma (in group3) OrderedGroup_ZF_3_L10: assumes A1: r {is total on} G and A2: l ≤ a ≤ u shows |a| ≤ GreaterOf(r, |l|, |u|) proof -
  from A2 have T1: |l| ∈ G |a| ∈ G |u| ∈ G
    using OrderedGroup_ZF_1_L4 OrderedGroup_ZF_3_L1 apply_funtype
    by auto
  { assume A3: a∈G⁺
    with A2 have 1≤a a≤u
      using OrderedGroup_ZF_1_L2 by auto
    then have 1≤u by (rule Group_order_transitive)
    with A2 A3 have |a|≤|u|
      using OrderedGroup_ZF_1_L2 OrderedGroup_ZF_3_L2 by simp
    moreover from A1 T1 have |u| ≤ GreaterOf(r, |l|, |u|)
      using Order_ZF_3_L2 by simp
    ultimately have |a| ≤ GreaterOf(r, |l|, |u|)
      by (rule Group_order_transitive) }
  moreover
  { assume A4: a∉G⁺
    with A2 have T2:
      l∈G |l| ∈ G |a| ∈ G |u| ∈ G a ∈ G-G⁺
      using OrderedGroup_ZF_1_L4 OrderedGroup_ZF_3_L1 apply_funtype
      by auto
    with A2 have 1 ∈ G-G⁺ using OrderedGroup_ZF_1_L4D by fast
    with T2 A2 have |a| ≤ |l|
      using OrderedGroup_ZF_3_L3 OrderedGroup_ZF_1_L5
      by simp
    moreover from A1 T2 have |l| ≤ GreaterOf(r, |l|, |u|)
      using Order_ZF_3_L2 by simp
    ultimately have |a| ≤ GreaterOf(r, |l|, |u|)
      by (rule Group_order_transitive) }
  ultimately show thesis by blast
qed

For linearly ordered groups if a set is bounded then the absolute values are bounded.

lemma (in group3) OrderedGroup_ZF_3_L10A: assumes A1: r {is total on} G
and \(A_2: \text{IsBounded}(A, r)\)

shows \(\exists L. \forall a \in A. \|a\| \leq L\)

proof -

{ assume \(A = 0\) then have thesis by auto }

moreover { assume \(A_3: A \neq 0\)

with \(A_2\) have \(\exists u. \forall g \in A. g \leq u\) and \(\exists l. \forall g \in A. l \leq g\)

using IsBounded_def IsBoundedAbove_def IsBoundedBelow_def by auto

then obtain \(u, l\) where \(\forall g \in A. l \leq g \land g \leq u\)

by auto

with \(A_1\) have \(\forall a \in A. |a| \leq \text{GreaterOf}(r, |l|, |u|)\)

using OrderedGroup_ZF_3_L10 by simp

then have thesis by auto }

ultimately show thesis by blast

qed

A slightly more general version of the previous lemma, stating the same fact for a set defined by separation.

lemma (in group3) OrderedGroup_ZF_3_L11:

assumes \(\text{r} \text{ is total on } G\)

and \(\text{IsBounded}(|\{b(x). x \in X\}|, r)\)

shows \(\exists L. \forall x \in X. |b(x)| \leq L\)

using assms OrderedGroup_ZF_3_L11A by blast

Absolute values of elements of a finite image of a nonempty set are bounded by an element of the group.

lemma (in group3) OrderedGroup_ZF_3_L11A:

assumes \(A_1: \text{r} \text{ is total on } G\)

and \(A_2: X \neq 0\) and \(A_3: \{b(x). x \in X\} \in \text{Fin}(G)\)

shows \(\exists L \in G. \forall x \in X. |b(x)| \leq L\)

proof -

from \(A_1\) \(A_3\) have \(\exists L. \forall x \in X. |b(x)| \leq L\)

using ord_group_fin_bounded OrderedGroup_ZF_3_L11 by simp

then obtain \(L\) where \(I: \forall x \in X. |b(x)| \leq L\)

using OrderedGroup_ZF_3_L11 by auto

from \(A_2\) obtain \(x\) where \(x \in X\) by auto

with \(I\) show thesis using OrderedGroup_ZF_1_L4 by blast

qed

In totally ordered groups the absolute value of a nonunit element is in \(G_+\).

lemma (in group3) OrderedGroup_ZF_3_L12:

assumes \(A_1: \text{r} \text{ is total on } G\)

and \(A_2: a \in G\) and \(A_3: a \neq 1\)

shows \(|a| \in G_+\)

proof -
from A1 A2 have |a| ∈ G 1 ≤ |a|
  using OrderedGroup_ZF_3_L1 apply_funtype
  OrderedGroup_ZF_3_L3B OrderedGroup_ZF_1_L2
  by auto
moreover from A2 A3 have |a| ≠ 1
  using OrderedGroup_ZF_3_L3D by auto
ultimately show |a| ∈ G
  using PositiveSet_def by auto
qed

39.2 Maximum absolute value of a set

Quite often when considering inequalities we prefer to talk about the absolute values instead of raw elements of a set. This section formalizes some material that is useful for that.

If a set has a maximum and minimum, then the greater of the absolute value of the maximum and minimum belongs to the image of the set by the absolute value function.

lemma (in group3) OrderedGroup_ZF_4_L1:
  assumes A: A ⊆ G
  and HasAmaximum(r,A) HasAminimum(r,A)
  and M = GreaterOf(r,|Minimum(r,A)|,|Maximum(r,A)|)
  shows M ∈ AbsoluteValue(G,P,r)(A)
  using ordGroupAssum assms IsAnOrdGroup_def IsPartOrder_def
  Order_ZF_4_L3 Order_ZF_4_L4 OrderedGroup_ZF_3_L1
  func_imagedef GreaterOf_def by auto

If a set has a maximum and minimum, then the greater of the absolute value of the maximum and minimum bounds absolute values of all elements of the set.

lemma (in group3) OrderedGroup_ZF_4_L2:
  assumes A1: r {is total on} G
  and A2: HasAmaximum(r,A) HasAminimum(r,A)
  and A3: a ∈ A
  shows |a| ≤ GreaterOf(r,|Minimum(r,A)|,|Maximum(r,A)|)
proof -
  from ordGroupAssum A2 A3 have
    Minimum(r,A) ≤ a ≤ Maximum(r,A)
    using IsAnOrdGroup_def IsPartOrder_def Order_ZF_4_L3 Order_ZF_4_L4
    by auto
  with A1 show thesis by (rule OrderedGroup_ZF_3_L10)
qed

If a set has a maximum and minimum, then the greater of the absolute value of the maximum and minimum bounds absolute values of all elements of the set. In this lemma the absolute values of elements of a set are represented as the elements of the image of the set by the absolute value function.
lemma (in group3) OrderedGroup_ZF_4_L3:
assumes r: {is total on} G and A ⊆ G and HasAmaximum(r,A) HasAminimum(r,A) and b ∈ AbsoluteValue(G,P,r)(A)
shows b ≤ GreaterOf(r,|Minimum(r,A)|,|Maximum(r,A)|)
using assms OrderedGroup_ZF_3_L1 func_imagedef OrderedGroup_ZF_4_L2 by auto

If a set has a maximum and minimum, then the set of absolute values also has a maximum.

lemma (in group3) OrderedGroup_ZF_4_L4:
assumes A1: r: {is total on} G and A2: A ⊆ G and A3: HasAmaximum(r,A) HasAminimum(r,A)
shows HasAmaximum(r,AbsoluteValue(G,P,r)(A))
proof -
  let M = GreaterOf(r,|Minimum(r,A)|,|Maximum(r,A)|)
  from A2 A3 have M ∈ AbsoluteValue(G,P,r)(A)
    using OrderedGroup_ZF_4_L1 by simp
  moreover from A1 A2 A3 have ∀ b ∈ AbsoluteValue(G,P,r)(A). b ≤ M
    using OrderedGroup_ZF_4_L3 by simp
  ultimately show thesis using HasAmaximum_def by auto
qed

If a set has a maximum and a minimum, then all absolute values are bounded by the maximum of the set of absolute values.

lemma (in group3) OrderedGroup_ZF_4_L5:
assumes A1: r: {is total on} G and A2: A ⊆ G and A3: HasAmaximum(r,A) HasAminimum(r,A) and A4: a ∈ A
shows |a| ≤ Maximum(r,AbsoluteValue(G,P,r)(A))
proof -
  from A2 A4 have |a| ∈ AbsoluteValue(G,P,r)(A)
    using OrderedGroup_ZF_3_L1 func_imagedef by auto
  with ordGroupAssum A1 A2 A3 show thesis using
    IsAnOrdGroup_def IsPartOrder_def OrderedGroup_ZF_4_L4 Order_ZF_4_L3 by simp
qed

39.3 Alternative definitions

Sometimes it is useful to define the order by prescribing the set of positive or nonnegative elements. This section deals with two such definitions. One takes a subset $H$ of $G$ that is closed under the group operation, $1 \notin H$ and for every $a \in H$ we have either $a \in H$ or $a^{-1} \in H$. Then the order is defined as $a \leq b$ iff $a = b$ or $a^{-1}b \in H$. For abelian groups this makes a linearly ordered group. We will refer to order defined this way in the comments as the order defined by a positive set. The context used in this section is
the group0 context defined in Group_ZF theory. Recall that f in that context
denotes the group operation (unlike in the previous sections where the group
operation was denoted P.

The order defined by a positive set is the same as the order defined by a
nonnegative set.

lemma (in group0) OrderedGroup_ZF_5_L1:
  assumes A1: r = {p ∈ G×G. fst(p) = snd(p) ∨ fst(p)−¹.snd(p) ∈ H}
  shows ⟨a,b⟩ ∈ r ↔ a∈G ∧ b∈G ∧ a⁻¹.b ∈ H ∪ {1}
proof
  assume ⟨a,b⟩ ∈ r
  with A1 show a∈G ∧ b∈G ∧ a⁻¹.b ∈ H ∪ {1}
    using group0_2_L6 by auto
next assume a∈G ∧ b∈G ∧ a⁻¹.b ∈ H ∪ {1}
  then have a∈G ∧ b∈G ∧ b=(a⁻¹)⁻¹ ∨ a∈G ∧ b∈G ∧ a⁻¹.b ∈ H
    using inverse_in_group group0_2_L9 by auto
  with A1 show ⟨a,b⟩ ∈ r using group_inv_of_inv
    by auto
qed

The relation defined by a positive set is antisymmetric.

lemma (in group0) OrderedGroup_ZF_5_L2:
  assumes A1: r = {p ∈ G×G. fst(p) = snd(p) ∨ fst(p)−¹.snd(p) ∈ H}
  and A2: ∀a∈G. a≠1 −→ (a∈H) Xor (a⁻¹∈H)
  shows antisym(r)
proof -
  { fix a b assume A3: ⟨a,b⟩ ∈ r ⟨b,a⟩ ∈ r
    with A1 have T: a∈G ∧ b∈G by auto
    assume A4: a≠b
      with A1 A3 have a⁻¹.b ∈ G a⁻¹.b ∈ H (a⁻¹.b)⁻¹ ∈ H
        using inverse_in_group group0_2_L1 monoid0.group0_1_L1 group0_2_L12
        by auto
      with A2 have a⁻¹.b = 1 using Xor_def by auto
      with T A4 have False using group0_2_L11 by auto
    } then have a=b by auto
  } then show antisym(r) by (rule antisymI)
qed

The relation defined by a positive set is transitive.

lemma (in group0) OrderedGroup_ZF_5_L3:
  assumes A1: r = {p ∈ G×G. fst(p) = snd(p) ∨ fst(p)−¹.snd(p) ∈ H}
  and A2: H⊆G H {is closed under} P
  shows trans(r)
proof -
  { fix a b c assume ⟨a,b⟩ ∈ r ⟨b,c⟩ ∈ r
    with A1 have
      a∈G ∧ b∈G ∧ a⁻¹.b ∈ H ∪ {1}
      b∈G ∧ c∈G ∧ b⁻¹.c ∈ H ∪ {1}

using OrderedGroup_ZF_5_L1 by auto 
with A2 have 
I: a∈G b∈G c∈G 
and (a⁻¹·b)·(b⁻¹·c) ∈ H ∪ \{1\} 
using inverse_in_group group_0_2_L17 IsOpClosed_def by auto 
moreover from I have a⁻¹·c = (a⁻¹·b)·(b⁻¹·c) 
by (rule group_0_2_L14A) 
ultimately have (a,c) ∈ G×G a⁻¹·c ∈ H ∪ \{1\} 
by auto 
with A1 have ⟨a,c⟩ ∈ r using OrderedGroup_ZF_5_L1 
by auto 
} then have ∀ a b c. (a, b) ∈ r ∧ (b, c) ∈ r ---→ (a, c) ∈ r 
by blast then show trans(r) by (rule Fol1_L2) qed 

The relation defined by a positive set is translation invariant. With our definition this step requires the group to be abelian.

lemma (in group_0) OrderedGroup_ZF_5_L4: 
assumes A1: r = \{p ∈ G×G. fst(p) = snd(p) ∨ fst(p)⁻¹·snd(p) ∈ H\} 
and A2: P (is commutative on) G 
and A3: (a,b) ∈ r and A4: c ∈ G 
shows (a·c,b·c) ∈ r ∧ (c·a,c·b) ∈ r 
proof 
from A1 A3 A4 have 
I: a∈G b∈G a·c ∈ G b·c ∈ G 
and II: a⁻¹·b ∈ H ∪ \{1\} 
using OrderedGroup_ZF_5_L1 group_op_closed by auto 
with A2 A4 have (a·c)⁻¹·(b·c) ∈ H ∪ \{1\} 
using group_0_4_L6D by simp 
with A1 I show (a·c,b·c) ∈ r using OrderedGroup_ZF_5_L1 
by auto 
with A2 A4 I show (c·a,c·b) ∈ r using IsCommutative_def by simp 
qed 

If H ⊆ G is closed under the group operation 1 \∉ H and for every a ∈ H we have either a ∈ H or a⁻¹ ∈ H, then the relation "≤" defined by a ≤ b ⇔ a⁻¹·b ∈ H orders the group G. In such order H may be the set of positive or nonnegative elements.

lemma (in group_0) OrderedGroup_ZF_5_L5: 
assumes A1: P (is commutative on) G 
and A2: H⊆G H (is closed under) P 
and A3: ∀a∈G. a≠1 ---→ (a∈H) Xor (a⁻¹∈H) 
and A4: r = \{p ∈ G×G. fst(p) = snd(p) ∨ fst(p)⁻¹·snd(p) ∈ H\} 
shows IsAnOrdGroup(G,P,r)
r \{is total on\} G
Nonnegative(G,P,r) = PositiveSet(G,P,r) \cup \{1\}

proof -
from groupAssum A2 A3 A4 have
  IsAgroup(G,P) r \subseteq G \times G IsPartOrder(G,r)
  using refl_def OrderedGroup_ZF_5_L2 OrderedGroup_ZF_5_L3
  IsPartOrder_def by auto
moreover from A1 A4 have
  \forall g \in G. \forall a b. \langle a,b \rangle \in r \rightarrow \langle a \cdot g,b \cdot g \rangle \in r
  \langle g \cdot a,g \cdot b \rangle \in r
  using OrderedGroup_ZF_5_L4 by blast
ultimately show IsAnOrdGroup(G,P,r)
  using IsAnOrdGroup_def by simp
then show Nonnegative(G,P,r) = PositiveSet(G,P,r) \cup \{1\}
  using group3_def group3.OrderedGroup_ZF_1_L24 by simp

{ fix a b
  assume T: a \in G b \in G
  then have T1: a^{-1} \cdot b \in G
    using inverse_in_group group_op_closed by simp
  { assume \langle a,b \rangle /\in r
    with A4 T T1 I
    have \langle b,a \rangle \in r
      using Xor_def group0_2_L12 by simp
  }
  then show \langle a,b \rangle \in r \lor \langle b,a \rangle \in r by auto
  }
then show r \{is total on\} G using IsTotal_def by simp
qed

If the set defined as in OrderedGroup_ZF_5_L4 does not contain the neutral
element, then it is the positive set for the resulting order.

lemma (in group0) OrderedGroup_ZF_5_L6:
  assumes P {is commutative on} G
  and H \subseteq G and 1 \notin H
  and r = \{ p \in G \times G. fst(p) = snd(p) \lor fst(p)^{-1} \cdot snd(p) \in H \}
  shows PositiveSet(G,P,r) = H
  using assms group_inv_of_one group0_2_L2 PositiveSet_def
  by auto

The next definition describes how we construct an order relation from the
prescribed set of positive elements.

definition
  OrderFromPosSet(G,P,H) \equiv
  \{ p \in G \times G. fst(p) = snd(p) \lor P(\text{GroupInv}(G,P)(fst(p)),snd(p)) \in H \}

The next theorem rephrases lemmas OrderedGroup_ZF_5_L5 and OrderedGroup_ZF_5_L6
using the definition of the order from the positive set OrderFromPosSet. To
summarize, this is what it says: Suppose that $H \subseteq G$ is a set closed under that group operation such that $1 \notin H$ and for every nonunit group element $a$ either $a \in H$ or $a^{-1} \in H$. Define the order as $a \leq b$ iff $a = b$ or $a^{-1} \cdot b \in H$. Then this order makes $G$ into a linearly ordered group such $H$ is the set of positive elements (and then of course $H \cup \{1\}$ is the set of nonnegative elements).

**Theorem (in group0)** Group_ord_by_positive_set:
- Assumes $P$ is commutative on $G$
- and $H \subseteq G$ $H$ is closed under $P$
- and $1 \notin H$
- shows $\forall a \in G. a \neq 1 \rightarrow (a \in H)$ xor $(a^{-1} \in H)$

shows $\text{IsAnOrdGroup}(G,P,\text{OrderFromPosSet}(G,P,H))$ $\text{OrderFromPosSet}(G,P,H)$ is total on $G$ $\text{PositiveSet}(G,P,\text{OrderFromPosSet}(G,P,H)) = H$ $\text{Nonnegative}(G,P,\text{OrderFromPosSet}(G,P,H)) = H \cup \{1\}$

using assms OrderFromPosSet_def OrderedGroup_ZF_5_L5 OrderedGroup_ZF_5_L6 by auto

### 39.4 Odd Extensions

In this section we verify properties of odd extensions of functions defined on $G_+$. An odd extension of a function $f : G_+ \rightarrow G$ is a function $f^o : G \rightarrow G$ defined by $f^o(x) = f(x)$ if $x \in G_+$, $f(1) = 1$ and $f^o(x) = (f(x^{-1}))^{-1}$ for $x < 1$. Such function is the unique odd function that is equal to $f$ when restricted to $G_+$.

The next lemma is just to see the definition of the odd extension in the notation used in the group1 context.

**Lemma (in group3)** OrderedGroup_ZF_6_L1:
- shows $f^o = f \cup \{(a, (f(a^{-1}))^{-1}). a \in -G_+ \cup \{(1,1)\}$

using OddExtension_def by simp

A technical lemma that states that from a function defined on $G_+$ with values in $G$ we have $(f(a^{-1}))^{-1} \in G$.

**Lemma (in group3)** OrderedGroup_ZF_6_L2:
- assumes $f : G_+ \rightarrow G$ and $a \in -G_+$
- shows $f(a^{-1}) \in G$ $(f(a^{-1}))^{-1} \in G$

using assms OrderedGroup_ZF_1_L27 apply_funtype OrderedGroup_ZF_1_L1 group0.inverse_in_group by auto

The main theorem about odd extensions. It basically says that the odd extension of a function is what we want to be.

**Lemma (in group3)** odd_ext_props:
assumes $A1: r \text{ is total on } G$ and $A2: f: G_+ \to G$

shows

$f^*: G \to G$

$\forall a \in G_+. \ (f^*)(a) = f(a)$

$\forall a \in (-G_+). \ (f^*)(a) = (f(a^{-1}))^{-1}$

$(f^*)(1) = 1$

proof -

from $A1$ $A2$ have $I: f: G_+ \to G$

$\forall a \in -G_+. \ (f(a^{-1}))^{-1} \in G$

$G_+ \cap (-G_+) = 0$

$1 \notin G_+ \cup (-G_+)$

$f^* = f \cup \{(a, (f(a^{-1}))^{-1}). \ a \in -G_+ \cup \{1,1\}\}$

using OrderedGroup_ZF_6_L2 OrdGroup_decomp2 OrderedGroup_ZF_6_L1 by auto

then have $f^*: G_+ \cup (-G_+) \cup \{1\} \to G \cup G \cup \{1\}$

by (rule func1_1_L11E)

moreover from $A1$ have $G_+ \cup (-G_+) \cup \{1\} = G$

$G \cup \{1\} = G$

using OrdGroup_decomp2 OrderedGroup_ZF_1_L1 group0.group0_2_L2 by auto

ultimately show $f^*: G \to G$ by simp

from $I$ show $\forall a \in G_+. \ (f^*)(a) = f(a)$

by (rule func1_1_L11E)

from $I$ show $\forall a \in (-G_+). \ (f^*)(a) = (f(a^{-1}))^{-1}$

by (rule func1_1_L11E)

from $I$ show $(f^*)(1) = 1$

by (rule func1_1_L11E)

qed

Odd extensions are odd, of course.

*lemma (in group3) oddext_is_odd:*

assumes $A1: r \text{ is total on } G$ and $A2: f: G_+ \to G$

and $A3: a \in G$

shows $(f^*)(a^{-1}) = ((f^*)(a))^{-1}$

proof -

from $A1$ $A3$ have $a \in G_+ \lor a \in (-G_+) \lor a = 1$

using OrdGroup_decomp2 by blast

moreover

{ assume $a \in G_+$

with $A1$ $A2$ have $a^{-1} \in -G_+ \land (f^*)(a) = f(a)$

using OrderedGroup_ZF_1_L25 odd_ext_props by auto

with $A1$ $A2$ have $(f^*)(a^{-1}) = (f((a^{-1})^{-1}))^{-1} \land (f(a))^{-1} = ((f^*)(a))^{-1}$

using odd_ext_props by auto

with $A3$ have $(f^*)(a^{-1}) = ((f^*)(a))^{-1}$

using OrderedGroup_ZF_1_L1 group0.group_inv_of_inv by simp }

469
moreover
\{ assume A4: \( a \in - G_+ \)
with A1 A2 having \( a^\sim \in G_+ \) and \( (f^\sim)(a) = (f(a^\sim))^{-1} \)
using OrderedGroup_ZF_1_L27 odd_ext_props
by auto
with A1 A2 A4 having \( (f^\sim)(a^\sim) = ((f^\sim)(a))^{-1} \)
using odd_ext_props OrderedGroup_ZF_6_L2
OrderedGroup_ZF_1_L1 group0.group_inv_of_inv
by simp \}
moreover
\{ assume a = 1
with A1 A2
have \( (f^\circ)(a^\sim) = ((f^\circ)(a))^{-1} \)
using OrderedGroup_ZF_1_L1 group0.group_inv_of_one
odd_ext_props by simp \}
ultimately show \( (f^\circ)(a^\sim) = ((f^\circ)(a))^{-1} \)
by auto
qed

Another way of saying that odd extensions are odd.

lemma (in group3) oddext_is_odd_alt:
assumes A1: r \{is total on\} \( G \) and A2: \( f: G_+ \to G \)
and A3: \( a \in G \)
shows \( ((f^\sim)(a^\sim))^{-1} = (f^\sim)(a) \)
proof -
from A1 A2 have
\( f^\circ: G \to G \)
\( \forall a \in G . \ (f^\sim)(a^\sim) = ((f^\sim)(a))^{-1} \)
using odd_ext_props oddext_is_odd by auto
then have \( \forall a \in G . \ ((f^\sim)(a^\sim))^{-1} = (f^\sim)(a) \)
using OrderedGroup_ZF_1_L1 group0.group0_6_L2 by simp
with A3 show \( ((f^\sim)(a^\sim))^{-1} = (f^\sim)(a) \) by simp
qed

39.5 Functions with infinite limits

In this section we consider functions \( f: G \to G \) with the property that for \( f(x) \) is arbitrarily large for large enough \( x \). More precisely, for every \( a \in G \) there exist \( b \in G_+ \) such that for every \( x \geq b \) we have \( f(x) \geq a \). In a sense this means that \( \lim_{x \to \infty} f(x) = \infty \), hence the title of this section. We also prove dual statements for functions such that \( \lim_{x \to -\infty} f(x) = -\infty \).

If an image of a set by a function with infinite positive limit is bounded above, then the set itself is bounded above.

lemma (in group3) OrderedGroup_ZF_7_L1:
assumes A1: r \{is total on\} \( G \) and A2: \( G \neq \{1\} \) and
A3: \( f: G \to G \)
and A4: \( \forall a \in G . \ \exists b \in G_+ . \ \forall x . \ b \leq x \longrightarrow a \leq f(x) \) and
If an image of a set defined by separation by a function with infinite positive limit is bounded above, then the set itself is bounded above.

**lemma (in group3)** OrderedGroup_ZF_7_L2:

assumes A1: \( r \text{ (is total on) } G \) and A2: \( G \neq \{1\} \) and A3: \( X \neq 0 \) and A4: \( f : G \rightarrow G \) and A5: \( \forall a \in G. \exists b \in G. \forall y. b \leq y \rightarrow a \leq f(y) \) and A6: \( \forall x \in X. b(x) \in G \land f(b(x)) \leq U \)

shows \( \exists u. \forall x \in X. b(x) \leq u \)

proof -

let \( A = \{b(x). x \in X\} \)

from A6 have I: \( A \subseteq G \) by auto

moreover note assms

moreover have IsBoundedAbove(f(A),r)

proof -

from A4 A6 I have \( \forall z \in f(A). \langle z, U \rangle \in r \)

using func_imagedef by simp

then show IsBoundedAbove(f(A),r) by (rule Order_ZF_3_L10)

qed
ultimately have IsBoundedAbove(A,r) using OrderedGroup_ZF_7_L1
by simp
with A3 have ∃u.∀y∈A. y ≤ u
using IsBoundedAbove_def by simp
then show ∃u.∀x∈X. b(x) ≤ u by auto
qed

If the image of a set defined by separation by a function with infinite negative
limit is bounded below, then the set itself is bounded above. This is dual to
OrderedGroup_ZF_7_L2.

lemma (in group3) OrderedGroup_ZF_7_L3:
assumes A1: r {is total on} G and A2: G ≠ {1} and
A3: X ≠ 0 and A4: f:G→G and
A5: ∀a∈G.∃b∈G+,∀y. b≤y → f(y⁻¹) ≤ a and
A6: ∀x∈X. b(x) ∈ G ∧ L ≤ f(b(x))
shows ∃l.∀x∈X. l ≤ b(x)
proof -
let g = GroupInv(G,P) O f O GroupInv(G,P)
from ordGroupAssum have I: GroupInv(G,P) : G→G
using IsAnOrdGroup_def group0_2_T2 by simp
with A4 I have g : G→G
using comp_fun by blast
moreover have ∀a∈G.∃b∈G+.∀y. b≤y → a ≤ g(y)
proof -
{ fix a assume A7: a∈G
then have a⁻¹ ∈ G
using OrderedGroup_ZF_1_L1 group0.inverse_in_group by simp
with A5 obtain b where
III: b∈G+ and ∀y. b≤y → f(y⁻¹) ≤ a⁻¹
by auto
with II A7 have ∀y. b≤y → a ≤ g(y)
using OrderedGroup_ZF_1_L5AD OrderedGroup_ZF_1_L4 by simp
with III have ∃b∈G+.∀y. b≤y → a ≤ g(y)
by auto
} then show ∀a∈G.∃b∈G+.∀y. b≤y → a ≤ g(y)
by simp
qed
moreover have ∀x∈X. b(x)⁻¹ ∈ G ∧ g(b(x)⁻¹) ≤ L⁻¹
proof-
{ fix x assume x∈X
with A6 have
T: b(x) ∈ G b(x)⁻¹ ∈ G and L ≤ f(b(x))
using OrderedGroup_ZF_1_L1 group0.inverse_in_group by auto
then have \((f(b(x)))^{-1} \leq L^{-1}\)
using OrderedGroup_ZF_1_L5 by simp
moreover from II T have \((f(b(x)))^{-1} = g(b(x)^{-1})\)
using OrderedGroup_ZF_1_L1 group0.group_inv_of_inv
by simp
ultimately have \(g(b(x)^{-1}) \leq L^{-1}\) by simp
with T have \(b(x)^{-1} \in G \land g(b(x)^{-1}) \leq L^{-1}\)
by simp
} then show \(\forall x \in X. b(x)^{-1} \in G \land g(b(x)^{-1}) \leq L^{-1}\)
by simp
qed
ultimately have \(\exists u. \forall x \in X. (b(x))^{-1} \leq u\)
by (rule OrderedGroup_ZF_7_L2)
then have \(\exists u. \forall x \in X. u^{-1} \leq (b(x)^{-1})^{-1}\)
using OrderedGroup_ZF_1_L5 by auto
with A6 show \(\exists l. \forall x \in X. 1 \leq b(x)\)
using OrderedGroup_ZF_1_L1 group0.group_inv_of_inv
by auto
qed

The next lemma combines OrderedGroup_ZF_7_L2 and OrderedGroup_ZF_7_L3
to show that if an image of a set defined by separation by a function with
infinite limits is bounded, then the set itself is bounded.

lemma (in group3) OrderedGroup_ZF_7_L4:
assumes A1: r {is total on} G and A2: G \neq \{1\} and
A3: \(X \neq 0\) and A4: \(f:G \rightarrow G\) and
A5: \(\forall a \in G. \exists b \in G. \forall y. b \leq y \rightarrow a \leq f(y)\) and
A6: \(\forall a \in G. \exists b \in G. \forall y. b \leq y \rightarrow f(y^{-1}) \leq a\) and
A7: \(\forall x \in X. b(x) \in G \land L \leq f(b(x)) \land f(b(x)) \leq U\)
shows \(\exists M. \forall x \in X. |b(x)| \leq M\)
proof -
from A7 have
I: \(\forall x \in X. b(x) \in G \land f(b(x)) \leq U\) and
II: \(\forall x \in X. b(x) \in G \land L \leq f(b(x))\)
by auto
from A1 A2 A3 A4 A5 I have \(\exists u. \forall x \in X. b(x) \leq u\)
by (rule OrderedGroup_ZF_7_L2)
moreover from A1 A2 A3 A4 A5 II have \(\exists l. \forall x \in X. 1 \leq b(x)\)
by (rule OrderedGroup_ZF_7_L3)
ultimately have \(\exists u l. \forall x \in X. 1 \leq b(x) \land b(x) \leq u\)
by auto
with A1 have \(\exists u l. \forall x \in X. |b(x)| \leq \text{GreaterOf}(r, |l|, |u|)\)
using OrderedGroup_ZF_3_L10 by blast
then show \(\exists M. \forall x \in X. |b(x)| \leq M\)
by auto
qed

end
40 Rings - introduction

theory Ring_ZF imports AbelianGroup_ZF

begin

This theory file covers basic facts about rings.

40.1 Definition and basic properties

In this section we define what is a ring and list the basic properties of rings.

We say that three sets \((R, A, M)\) form a ring if \((R, A)\) is an abelian group, \((R, M)\) is a monoid and \(A\) is distributive with respect to \(M\) on \(R\). \(A\) represents the additive operation on \(R\). As such it is a subset of \((R \times R) \times R\) (recall that in ZF set theory functions are sets). Similarly \(M\) represents the multiplicative operation on \(R\) and is also a subset of \((R \times R) \times R\). We don’t require the multiplicative operation to be commutative in the definition of a ring.

definition
\[ \text{IsAring}(R, A, M) \equiv \text{IsAgroup}(R, A) \land (A \text{ is commutative on } R) \land \text{IsAmonoid}(R, M) \land \text{IsDistributive}(R, A, M) \]

We also define the notion of having no zero divisors. In standard notation the ring has no zero divisors if for all \(a, b \in R\) we have \(a \cdot b = 0\) implies \(a = 0\) or \(b = 0\).

definition
\[ \text{HasNoZeroDivs}(R, A, M) \equiv (\forall a \in R. \forall b \in R. M(a, b) = \text{TheNeutralElement}(R, A) \longrightarrow a = \text{TheNeutralElement}(R, A) \lor b = \text{TheNeutralElement}(R, A)) \]

Next we define a locale that will be used when considering rings.

locale ring0 =

fixes \(R\) and \(A\) and \(M\)

assumes ringAssum: IsAring\((R, A, M)\)

fixes ringa (infixl + 90)
defines ringa_def [simp]: \(x + y \equiv A(x, y)\)

fixes ringminus (infixl - 89)
defines ringminus_def [simp]: \((- x) \equiv \text{GroupInv}(R, A)(x)\)

fixes ringsub (infixl - 90)
defines ringsub_def [simp]: \(x - y \equiv x + (- y)\)
fixes ringm (infixl · 95)
defines ringm_def [simp]: \( x \cdot y \equiv M(x,y) \)

fixes ringzero (0)
defines ringzero_def [simp]: \( 0 \equiv \text{TheNeutralElement}(R,A) \)

fixes ringone (1)
defines ringone_def [simp]: \( 1 \equiv \text{TheNeutralElement}(R,M) \)

fixes ringtwo (2)
defines ringtwo_def [simp]: \( 2 \equiv 1+1 \)

fixes ringsq (_\^2 [96] 97)
defines ringsq_def [simp]: \( x^2 \equiv x \cdot x \)

In the \textit{ring0} context we can use theorems proven in some other contexts.

\textbf{lemma (in ring0) RingZF1L1}: shows
\begin{itemize}
\item \text{monoid0}(R,M)
\item \text{group0}(R,A)
\item A \{is commutative on\} R
\end{itemize}
using ringAssum IsAring_def group0_def monoid0_def by auto

The theorems proven in \textit{group0} context (locale) are valid in the \textit{ring0} context when applied to the additive group of the ring.

\textbf{sublocale ring0 < add_group: group0 R A ringzero ringa ringminus}
using RingZF1L1(2) unfolding ringa_def ringminus_def ringzero_def by auto

The theorem proven in the \textit{monoid0} context are valid in the \textit{ring0} context when applied to the multiplicative monoid of the ring.

\textbf{sublocale ring0 < mult_monoid: monoid0 R M ringm}
using RingZF1L1(1) unfolding ringm_def by auto

The additive operation in a ring is distributive with respect to the multiplicative operation.

\textbf{lemma (in ring0) ring_oper_distr}: assumes A1: a\in R  b\in R  c\in R
shows
\begin{itemize}
\item a \cdot (b+c) = a \cdot b + a \cdot c
\item (b+c) \cdot a = b \cdot a + c \cdot a
\end{itemize}
using ringAssum assms IsAring_def IsDistributive_def by auto

Zero and one of the ring are elements of the ring. The negative of zero is zero.

\textbf{lemma (in ring0) RingZF1L2}:
shows 0\in R  1\in R  (-0) = 0
using add_group.group0_2L2 mult_monoid.unit_is_neutral
add_group.group inv_of_one by auto

475
The next lemma lists some properties of a ring that require one element of a ring.

**Lemma** (in `ring0`) `Ring_ZF_1_L3`:

**Assumes**: \( a \in R \)

**Shows**:
- \((-a) \in R\)
- \((-a)) = a\)
- \(a+0 = a\)
- \(0+a = a\)
- \(a\cdot1 = a\)
- \(a\cdot a = 0\)
- \(a\cdot 0 = a\)
- \(2a = a+a\)
- \((-a)+a = 0\)

**Using**:
- `assms add_group.inverse_in_group`  
- `add_group.group_inv_of_inv`  
- `add_group.group0_2_L6`  
- `add_group.group0_2_L2`  
- `mult_monoid.unit_is_neutral`

**by** `auto`

Properties that require two elements of a ring.

**Lemma** (in `ring0`) `Ring_ZF_1_L4`:

**Assumes**: \( A1: a \in R \) \( b \in R \)

**Shows**:
- \( a+b \in R\)
- \( a-b \in R\)
- \( a\cdot b \in R\)
- \( a+b = b+a\)

**Using**:
- `assms Ring_ZF_1_L1(3) Ring_ZF_1_L3`
- `add_group_monoid.group0_1_L1`
- `mult_monoid.group0_1_L1`
- `unfolding IsCommutative_def`

**by** `auto`

Cancellation of an element on both sides of equality. This is a property of groups, written in the (additive) notation we use for the additive operation in rings.

**Lemma** (in `ring0`) `ring_cancel_add`:

**Assumes**: \( A1: a \in R \) \( b \in R \) \( A2: a + b = a \)

**Shows**: \( b = 0\)

**Using**:
- `assms add_group.group0_2_L7`

**by** `simp`

Any element of a ring multiplied by zero is zero.

**Lemma** (in `ring0`) `Ring_ZF_1_L6`:

**Assumes**: \( A1: x \in R \)

**Shows**: \( 0 \cdot x = 0 \) \( x \cdot 0 = 0\)

**Proof**:
- `let a = x\cdot1`
- `let b = x\cdot0`
- `let c = 1\cdot x`
let \( d = 0 \cdot x \)

from A1 have

\[ a + b = x(1 + 0) \quad c + d = (1 + 0) \cdot x \]

using Ring_ZF_1_L2 ring_oper_distr by auto

moreover have \( x(1 + 0) = a \cdot (1 + 0) \cdot x = c \)

using Ring_ZF_1_L2 Ring_ZF_1_L3 by auto

ultimately have \( a + b = a \) and T1: \( c + d = c \)

by auto

moreover from A1 have

\[ a \in R \quad b \in R \] and T2: \( c \in R \quad d \in R \)

using Ring_ZF_1_L2 Ring_ZF_1_L4 by auto

ultimately have \( b = 0 \) using ring_cancel_add

by blast

moreover from T2 T1 have \( d = 0 \) using ring_cancel_add

by blast

ultimately show \( x \cdot 0 = 0 \quad 0 \cdot x = 0 \) by auto

qed

Negative can be pulled out of a product.

lemma (in ring0) Ring_ZF_1_L7:

assumes A1: \( a \in R \quad b \in R \)

shows \( (-a) \cdot b = -(a \cdot b) \)

\( a \cdot (-b) = -(a \cdot b) \)

\( (-a) \cdot b = a \cdot (-b) \)

proof -

from A1 have I:

\[ a \cdot b \in R \quad (-a) \in R \quad ((-a) \cdot b) \in R \]

\( (-b) \in R \quad a \cdot (-b) \in R \)

using Ring_ZF_1_L3 Ring_ZF_1_L4 by auto

moreover have \( (-a) \cdot b + a \cdot b = 0 \)

and II: \( a \cdot (-b) + a \cdot b = 0 \)

proof -

from A1 I have

\( (-a) \cdot b + a \cdot b = ((-a) + a) \cdot b \)

\( a \cdot (-b) + a \cdot b = a \cdot ((-b) + b) \)

using ring_oper_distr by auto

moreover from A1 have

\( ((-a) + a) \cdot b = 0 \)

\( a \cdot ((-b) + b) = 0 \)

using add_group.group0_2_L6 Ring_ZF_1_L6 by auto

ultimately show

\( (-a) \cdot b + a \cdot b = 0 \)

\( a \cdot (-b) + a \cdot b = 0 \)

by auto

qed

ultimately show \( (-a) \cdot b = -(a \cdot b) \)

using add_group.group0_2_L9 by simp
moreover from I II show \( a \cdot (-b) = -(a \cdot b) \)
using add_group.group0_2_L9 by simp
ultimately show \( (-a) \cdot b = a \cdot (-b) \) by simp
qed

Minus times minus is plus.

lemma (in ring0) Ring_ZF_1_L7A: assumes \( a \in \mathbb{R} \) \( b \in \mathbb{R} \)
shows \( (-a) \cdot (-b) = a \cdot b \)
using assms Ring_ZF_1_L3 Ring_ZF_1_L7 Ring_ZF_1_L4
by simp

Subtraction is distributive with respect to multiplication.

lemma (in ring0) Ring_ZF_1_L8: assumes \( a \in \mathbb{R} \) \( b \in \mathbb{R} \) \( c \in \mathbb{R} \)
shows \( a \cdot (b-c) = a \cdot b - a \cdot c \)
\( (b-c) \cdot a = b \cdot a - c \cdot a \)
using assms Ring_ZF_1_L3 ring_oper_distr Ring_ZF_1_L7 Ring_ZF_1_L4
by auto

Other basic properties involving two elements of a ring.

lemma (in ring0) Ring_ZF_1_L9: assumes \( a \in \mathbb{R} \) \( b \in \mathbb{R} \)
shows \( (-b) - a = (-a) - b \)
\( -(a+b) = (-a) - b \)
\( (-a-b) = ((-a)+b) \)
\( a - (-b) = a + b \)
using assms Ring_ZF_1_L1(3) add_group.group0_4_L4 add_group.group_inv_of_inv
by auto

If the difference of two element is zero, then those elements are equal.

lemma (in ring0) Ring_ZF_1_L9A:
assumes \( A1: a \in \mathbb{R} \) \( b \in \mathbb{R} \) and \( A2: a-b = 0 \)
shows \( a=b \) using add_group.group0_2_L11A assms
by auto

Other basic properties involving three elements of a ring.

lemma (in ring0) Ring_ZF_1_L10:
assumes \( a \in \mathbb{R} \) \( b \in \mathbb{R} \) \( c \in \mathbb{R} \)
shows \( a + (b+c) = a + b + c \)
\( a - (b+c) = a - b - c \)
\( a - (b-c) = a - b + c \)
using assms Ring_ZF_1_L1(3) add_group.group_oper_assoc
add_group.group0_4_L4A by auto

Another property with three elements.

lemma (in ring0) Ring_ZF_1_L10A:
assumes \( A1: a \in \mathbb{R} \) \( b \in \mathbb{R} \) \( c \in \mathbb{R} \)
shows $a+(b-c) = a+b-c$
using assms Ring_ZF_1_L3 Ring_ZF_1_L10 by simp

Associativity of addition and multiplication.

**lemma (in ring0) Ring_ZF_1_L11:**
assumes $a \in R$ $b \in R$ $c \in R$
shows $a+b+c = a+(b+c)$
a·b·c = a·(b·c)
using assms add_group.group_oper_assoc mult_monoid.sum_associative by auto

An interpretation of what it means that a ring has no zero divisors.

**lemma (in ring0) Ring_ZF_1_L12:**
assumes HasNoZeroDivs(R,A,M)
and $a \in R$ $a \neq 0$
$b \in R$ $b \neq 0$
shows $a·b \neq 0$
using assms HasNoZeroDivs_def by auto

In rings with no zero divisors we can cancel nonzero factors.

**lemma (in ring0) Ring_ZF_1_L12A:**
assumes A1: HasNoZeroDivs(R,A,M) and A2: $a \in R$ $b \in R$ $c \in R$
and A3: $a·c = b·c$
and A4: $c \neq 0$
shows $a=b$
proof -
from A2 have T: $a·c \in R$ $a-b \in R$
using Ring_ZF_1_L4 by auto
with A1 A2 A3 have $a-b = 0$ $\lor$ $c=0$
using Ring_ZF_1_L3 Ring_ZF_1_L8 HasNoZeroDivs_def by simp
with A2 A4 have $a \in R$ $b \in R$ $a-b = 0$
by auto
then show $a=b$ by (rule Ring_ZF_1_L9A)
qed

In rings with no zero divisors if two elements are different, then after multiplied by a nonzero element they are still different.

**lemma (in ring0) Ring_ZF_1_L12B:**
assumes A1: HasNoZeroDivs(R,A,M) and A2: $a \in R$ $b \in R$
and A3: $0 \neq a$ $1 \neq b$
shows $a·c \neq b·c$
using A1 Ring_ZF_1_L12A by auto

In rings with no zero divisors multiplying a nonzero element by a noneone element changes the value.

**lemma (in ring0) Ring_ZF_1_L12C:**
assumes A1: HasNoZeroDivs(R,A,M) and A2: $a \in R$ $b \in R$ and A3: $0 \neq a$ $1 \neq b$

479
shows \( a \neq a \cdot b \)

proof -

\[
\begin{align*}
\{ & \text{ assume } a = a \cdot b \\
& \text{ with } A1 \ A2 \ \text{have } a = 0 \lor b-1 = 0 \\
& \text{ using RingZF_1_L3 \ RingZF_1_L2 \ RingZF_1_L8} \\
& \text{ RingZF_1_L3 \ RingZF_1_L2 \ RingZF_1_L4 \ HasNoZeroDivs_def} \\
& \text{ by simp} \\
& \text{ with } A2 \ A3 \ \text{have False} \\
& \text{ using RingZF_1_L2 \ RingZF_1_L9A by auto} \\
\} & \text{ then show } a \neq a \cdot b \text{ by auto}
\end{align*}
\]

qed

If a square is nonzero, then the element is nonzero.

\textbf{lemma (in ring0) RingZF_1_L13:}

\[
\begin{align*}
\text{assumes } a \in R & \text{ and } a^2 \neq 0 \\
\text{shows } a \neq 0 \\
\text{ using assms RingZF_1_L2 \ RingZF_1_L6 by auto}
\end{align*}
\]

Square of an element and its opposite are the same.

\textbf{lemma (in ring0) RingZF_1_L14:}

\[
\begin{align*}
\text{assumes } a \in R & \text{ shows } (-a)^2 = ((a)^2) \\
\text{ using assms RingZF_1_L7A by simp}
\end{align*}
\]

Adding zero to a set that is closed under addition results in a set that is also closed under addition. This is a property of groups.

\textbf{lemma (in ring0) RingZF_1_L15:}

\[
\begin{align*}
\text{assumes } H \subseteq R & \text{ and } H \ {\text{is closed under}} \ A \\
\text{shows } (H \cup \{0\}) & \ {\text{is closed under}} \ A \\
\text{ using assms add_group.group0_2_L17 by simp}
\end{align*}
\]

Adding zero to a set that is closed under multiplication results in a set that is also closed under multiplication.

\textbf{lemma (in ring0) RingZF_1_L16:}

\[
\begin{align*}
\text{assumes } A1: H \subseteq R & \text{ and } A2: H \ {\text{is closed under}} \ M \\
\text{shows } (H \cup \{0\}) & \ {\text{is closed under}} \ M \\
\text{ using assms RingZF_1_L2 \ RingZF_1_L6 \ IsOpClosed_def by auto}
\end{align*}
\]

The ring is trivial if \( 0 = 1 \).

\textbf{lemma (in ring0) RingZF_1_L17: shows } R = \{0\} \iff 0=1

proof

\[
\begin{align*}
\text{assume } R & = \{0\} \\
\text{ then show } 0=1 & \text{ using RingZF_1_L2 by blast}
\end{align*}
\]

next assume \( A1: 0 = 1 \)

\[
\begin{align*}
\text{then have } R & \subseteq \{0\} \\
& \text{ using RingZF_1_L3 \ RingZF_1_L6 by auto} \\
\text{moreover have } \{0\} & \subseteq R \text{ using RingZF_1_L2 by auto}
\end{align*}
\]

480
ultimately show $R = \{0\}$ by auto

qed

The sets $\{m \cdot x. x \in R\}$ and $\{-m \cdot x. x \in R\}$ are the same.

lemma (in ring0) Ring_ZF_1_L18: assumes A1: $m \in R$
shows $\{m \cdot x. x \in R\} = \{-m \cdot x. x \in R\}$
proof
{ fix a assume a $\in \{m \cdot x. x \in R\}$
  then obtain x where x$\in R$ and a = $m \cdot x$
  by auto
  with A1 have $(-x) \in R$ and a $= (-m) \cdot (-x)$
  using Ring_ZF_1_L3 Ring_ZF_1_L7A by auto
  then have a $\in \{(-m) \cdot x. x \in R\}$
  by auto
} then show $\{m \cdot x. x \in R\} \subseteq \{-m \cdot x. x \in R\}$
by auto

next
{ fix a assume a $\in \{(-m) \cdot x. x \in R\}$
  then obtain x where x$\in R$ and a = $(-m) \cdot x$
  by auto
  with A1 have $(-x) \in R$ and a $= m \cdot (-x)$
  using Ring_ZF_1_L3 Ring_ZF_1_L7 by auto
  then have a $\in \{m \cdot x. x \in R\}$ by auto
} then show $\{-m \cdot x. x \in R\} \subseteq \{m \cdot x. x \in R\}$
by auto

qed

40.2 Rearrangement lemmas

In happens quite often that we want to show a fact like $(a + b)c + d = (ac + d - e) + (bc + e)$ in rings. This is trivial in romantic math and probably there is a way to make it trivial in formalized math. However, I don’t know any other way than to tediously prove each such rearrangement when it is needed. This section collects facts of this type.

Rearrangements with two elements of a ring.

lemma (in ring0) Ring_ZF_2_L1: assumes a$\in R$ b$\in R$
shows $a + b \cdot a = (b+1) \cdot a$
using assms Ring_ZF_1_L2 ring_oper_distr Ring_ZF_1_L3 Ring_ZF_1_L4
by simp

Rearrangements with two elements and cancelling.

lemma (in ring0) Ring_ZF_2_L1A: assumes a$\in R$ b$\in R$
shows
a$-b+b = a$
a$+b-a = b$
(-a) + b+a = b$
(-a) + (b+a) = b
a+(b-a) = b
using assms add_group.inv_cancel_two add_group.group0_4_L6A
Ring_ZF_1_L1(3) by auto

In rings \( a - (b+1)c = (a-d-c)+(d-bc) \) and \( a+b+(c+d) = a+(b+c)+d \).

lemma (in ring0) Ring_ZF_2_L2:
assumes A1: \( a \in R \) \( b \in R \) \( c \in R \) \( d \in R \)
shows \( a-(b+1)c = (a-d-c)+(d-bc) \)
a+b+(c+d) = a+b+c+d
proof -
let B = b·c
from ringAssum assms have A: {is commutative on} R and a∈R B ∈ R c∈R d∈R
unfolding IsAring_def using Ring_ZF_1_L4 by auto
then have A B: \( A = A \langle A \langle a \cdot x, b \rangle, A \langle c \cdot x, d \rangle \rangle \)
using add_group.group0_4_L8(3) by auto
with A1 show \( a-(b+1)c = (a-d-c)+(d-bc) \)
a+b+(c+d) = a+b+c+d
using Ring_ZF_1_L10(1) by simp
qed

Rearrangement about adding linear functions.

lemma (in ring0) Ring_ZF_2_L3:
assumes A1: \( a \in R \) \( b \in R \) \( c \in R \) \( d \in R \) \( x \in R \)
shows \( (a \cdot x + b) + (c \cdot x + d) = (a+c) \cdot x + (b+d) \)
proof -
from A1 have A: {is commutative on} R
\( a \cdot x \in R \) \( b \in R \) \( c \cdot x \in R \) \( d \in R \)
using Ring_ZF_1_L1 Ring_ZF_1_L4 by auto
then have A A1: \( A \langle A \langle a \cdot x, b \rangle, A \langle c \cdot x, d \rangle \rangle = A \langle A \langle a \cdot x, c \cdot x \rangle, A \langle b, d \rangle \rangle \)
using add_group.group0_4_L8(3) by auto
with A1 show \( (a \cdot x + b) + (c \cdot x + d) = (a+c) \cdot x + (b+d) \)
using ring_oper_distr by simp
qed

Rearrangement with three elements

lemma (in ring0) Ring_ZF_2_L4:
assumes M: {is commutative on} R
and a∈R b∈R c∈R
shows a·(b·c) = a·c·b and a·b·c = a·c·b
using assms IsCommutative_def Ring_ZF_1_L11

482
by simp_all

Some other rearrangements with three elements.

lemma (in ring0) ring_rearr_3_elemA:
  assumes A1: M {is commutative on} R and
  A2: a∈R  b∈R  c∈R
  shows
    a·(a·c) - b·(-b·c) = (a·a + b·b)·c
    a·(-b·c) + b·(a·c) = 0
proof
  from A2 have T:
    b·c ∈ R  a·a ∈ R  b·b ∈ R
    b·(b·c) ∈ R  a·(b·c) ∈ R
    using Ring_ZF_1_L4 by auto
  with A2 show
    a·(a·c) - b·(-b·c) = (a·a + b·b)·c
    using Ring_ZF_1_L7  Ring_ZF_1_L3  Ring_ZF_1_L11
      ring_oper_distr  by simp
  from A2 T have
    a·(-b·c) + b·(a·c) = (-a·(b·c)) + b·a·c
    using Ring_ZF_1_L7  Ring_ZF_1_L11 by simp
  also from A1 A2 T have \ldots = 0
    using IsCommutative_def  Ring_ZF_1_L11  Ring_ZF_1_L3
      by simp
  finally show a·(-b·c) + b·(a·c) = 0
    by simp
qed

Some rearrangements with four elements. Properties of abelian groups.

lemma (in ring0) Ring_ZF_2_L5:
  assumes a∈R  b∈R  c∈R  d∈R
  shows
    a - b - c - d = a - d - b - c
    a + b + c - d = a - d + b + c
    a + b - c - d = a - c + (b - d)
    a + b + c + d = a + c + (b + d)
using assms  Ring_ZF_1_L1(3) add_group.rearr_ab_gr_4_elemB
  add_group.rearr_ab_gr_4_elemA  by auto

Two big rearrangements with six elements, useful for proving properties of
complex addition and multiplication.

lemma (in ring0) Ring_ZF_2_L6:
  assumes A1: a∈R  b∈R  c∈R  d∈R  e∈R  f∈R
  shows
    a·(c·e - d·f) - b·(c·f + d·e) =
    (a·c - b·d)·e - (a·d + b·c)·f
    a·(c·f + d·e) + b·(c·e - d·f) =
    (a·c - b·d)·f + (a·d + b·c)·e
    a·(c·e) - b·(d·f) = a·c - b·d + (a·e - b·f)

483
proof -
from \( A1 \) have \( T: \)
\[
\begin{align*}
  c \cdot e &\in R \quad d \cdot f \in R \\
  a \cdot c &\in R \quad b \cdot d \in R \\
  a \cdot d &\in R \quad b \cdot c \in R \\
  b \cdot f &\in R \\
  a \cdot c \cdot e &\in R \\
  a \cdot d \cdot f &\in R \\
  b \cdot c \cdot e &\in R \\
  b \cdot d \cdot f &\in R \\
  a \cdot e &\in R \\
  a \cdot d \cdot e &\in R \\
  a \cdot c \cdot f &\in R \\
  b \cdot d \cdot e &\in R \\
  a \cdot c \cdot f &\in R \\
  a \cdot c \cdot e &\in R \\
  a \cdot c \cdot f &\in R \\
  a \cdot d \cdot f &\in R \\
  a \cdot c \cdot f &\in R \\
  a \cdot c \cdot e &\in R \\
  d \cdot c &\in R \\
  b \cdot f &\in R \\
  a \cdot c \cdot e &\in R \\
  a \cdot c \cdot f &\in R \\
  b \cdot d \cdot e &\in R \\
  a \cdot e &\in R \\
  d \cdot c &\in R \\
  b \cdot f &\in R \\
  a \cdot c \cdot e &\in R \\
  a \cdot e &\in R \\
  d \cdot c &\in R \\
  b \cdot f &\in R \\
  a \cdot c \cdot e &\in R \\
  a \cdot e &\in R \\
  d \cdot c &\in R \\
  b \cdot f &\in R
\end{align*}
\]
using \( \text{Ring}_\text{ZF}_1 \text{L14} \) by \( \text{auto} \)
with \( A1 \) show \( a \cdot (c \cdot e - d \cdot f) - b \cdot (c \cdot f + d \cdot e) = \\
(a \cdot c - b \cdot d) \cdot e - (a \cdot d + b \cdot c) \cdot f \)
using \( \text{Ring}_\text{ZF}_1 \text{L18} \text{ ring} \text{ oper} \text{ distr} \text{ Ring}_\text{ZF}_1 \text{L11} \text{ Ring}_\text{ZF}_1 \text{L10} \text{ Ring}_\text{Zf}_2 \text{L5} \) by \( \text{simp} \)
from \( A1 \text{ } T \) show \( a \cdot (c \cdot f + d \cdot e) + b \cdot (c \cdot e - d \cdot f) = \\
(a \cdot c - b \cdot d) \cdot f + (a \cdot d + b \cdot c) \cdot e \)
using \( \text{Ring}_\text{ZF}_1 \text{L18} \text{ ring} \text{ oper} \text{ distr} \text{ Ring}_\text{ZF}_1 \text{L11} \text{ Ring}_\text{ZF}_1 \text{L10} \text{A Ring}_\text{Zf}_2 \text{L5} \text{ Ring}_\text{ZF}_1 \text{L10} \) by \( \text{simp} \)
from \( A1 \text{ } T \) show \( a \cdot (c+e) - b \cdot (d+f) = a \cdot c - b \cdot d + (a \cdot e - b \cdot f) \\
a \cdot (d+f) + b \cdot (c+e) = a \cdot d + b \cdot c + (a \cdot f + b \cdot e) \)
using \( \text{ring} \text{ oper} \text{ distr} \text{ Ring}_\text{ZF}_1 \text{L10} \text{ Ring}_\text{Zf}_2 \text{L5} \) by \( \text{auto} \)
qed

end

41 Binomial theorem

theory \text{Ring} \text{Binomial}_\text{ZF} \text{ imports} \text{ Monoid}_\text{ZF}_1 \text{ Ring}_\text{ZF}

begin

This theory aims at formalizing sufficient background to be able to state and prove the binomial theorem.

484
41.1 Sums of multiplicities of powers of ring elements and binomial theorem

The binomial theorem asserts that for any two elements of a commutative ring the n-th power of the sum $x + y$ can be written as a sum of certain multiplicities of terms $x^{n-k}y^k$, where $k \in \{0..n\}$. In this section we setup the notation and prove basic properties of such multiplicities and powers of ring elements. We show the binomial theorem as an application.

The next locale (context) extends the ring0 locale with notation for powers, multiplicities and sums and products of finite lists of ring elements.

Locale ring3 = ring0 +
fixes listsum ($\sum_{70}$)
defines listsum_def [simp]: $\sum s \equiv \text{Fold}(A,0,s)$

fixes listprod ($\prod_{70}$)
defines listprod_def [simp]: $\prod s \equiv \text{Fold}(M,1,s)$

fixes nat_mult (infix ·)
defines nat_mult_def [simp]: $n \cdot x \equiv \sum \{\langle k, x \rangle . k \in n\}$

fixes pow
defines pow_def [simp]: $\text{pow}(n,x) \equiv \prod \{\langle k, x \rangle . k \in n\}$

A ring with addition forms a monoid, hence all propositions proven in the monoid1 locale (defined in the Monoid_ZF_1 theory) can be used in the ring3 locale, applied to the additive operation.

Sublocale ring3 < add_monoid: monoid1 R A ringa ringzero listsum nat_mult
using ringAssum
unfolding IsAring_def IsAgroup_def monoid1_def monoid0_def
by auto

A ring with multiplication forms a monoid, hence all propositions proven in the monoid1 locale (defined in the Monoid_ZF_1 theory) can be used in the ring3 locale, applied to the multiplicative operation.

Sublocale ring3 < mul_monoid: monoid1 R M ringm ringone listprod pow
using ringAssum
unfolding IsAring_def IsAgroup_def monoid1_def monoid0_def
by auto

$0 \cdot x = 0$ and $x^0 = 1$. It is a bit surprising that we do not need to assume that $x \in R$ (i.e. $x$ is an element of the ring). These properties are really proven in the Monoid_ZF_1 theory where there is no assumption that $x$ is an element of the monoid.

Lemma (in ring3) mult_pow_zero: shows $0 \cdot x = 0$ and $\text{pow}(0,x) = 1$
using add_monoid.nat_mult_zero mul_monoid.nat_mult_zero by simp_all
Natural multiple and power of a ring element is a ring element.

**Lemma** in ring3: mult_pow_type: assumes \( n \in \mathbb{N} \) \( x \in R \)
shows \( n \cdot x \in R \) and \( \text{pow}(n,x) \in R \)
using assms add_monoid.nat_mult_type mul_monoid.nat_mult_type
by simp_all

The usual properties of multiples and powers: \((n+1)x = nx + x\) and \(x^{n+1} = x^n x\). These are just versions of nat_mult_add_one from Monoid_ZF_1 written in the notation defined in the ring3 locale.

**Lemma** in ring3: nat_mult_pow_add_one: assumes \( n \in \mathbb{N} \) \( x \in R \)
shows \((n + 1) \cdot x = (n \cdot x) + x\) and \(\text{pow}(n + 1,x) = \text{pow}(n,x) \cdot x\)
using assms add_monoid.nat_mult_add_one mul_monoid.nat_mult_add_one
by simp_all

Associativity for the multiplication by natural number and the ring multiplication:

**Lemma** in ring3: nat_mult_assoc: assumes \( n \in \mathbb{N} \) \( x \in R \) \( y \in R \)
shows \( n \cdot x \cdot y = n \cdot (x \cdot y) \)
proof -
from assms(1,3) have \( n \in \mathbb{N} \) and \( 0 \cdot y = 0 \cdot (x \cdot y) \)
using mult_pow_zero(1) Ring_ZF_1_L6 by simp_all
moreover from assms(2,3) have \( \forall k \in \mathbb{N} . \ k \cdot y = k \cdot (x \cdot y) \)
using nat_mult_pow_add_one(1) mult_pow_type ring_oper_distr(2) Ring_ZF_1_L4(3)
by simp
ultimately show thesis by (rule ind_on_nat1)
qed

Addition of natural numbers is distributive with respect to natural multiple. This is essentially lemma nat_mult_add from Monoid_ZF_1.thy, just transferred to the ring3 locale.

**Lemma** in ring3: nat_add_mult_distrib: assumes \( n \in \mathbb{N} \) \( m \in \mathbb{N} \) \( x \in R \)
shows \((n + m) \cdot x = n \cdot x + m \cdot x\)
using assms add_monoid.nat_mult_add by simp

Associativity for the multiplication by natural number and the ring multiplication extended to three elements of the ring:

**Lemma** in ring3: nat_mult_assoc1: assumes \( n \in \mathbb{N} \) \( x \in R \) \( y \in R \) \( z \in R \)
shows \( n \cdot x \cdot y \cdot z = n \cdot (x \cdot y \cdot z) \)
using assms Ring_ZF_1_L4(3) nat_mult_assoc by simp

When we multiply an expression whose value belongs to a ring by a ring element and we get an expression whose value belongs to a ring.

**Lemma** in ring3: mult_elem_ring_type:
assumes \( n \in \mathbb{N} \) \( x \in R \) and \( \forall k \in \mathbb{N} . \ q(k) \in R \)
shows \( \forall k \in \mathbb{N} . \ q(k) \cdot x \in R \) and \((\sum \langle \langle k,q(k) \cdot x \rangle. \ k \in \mathbb{N} \rangle) \in R \)
The sum of expressions whose values belong to a ring is an expression whose value belongs to a ring.

**lemma (in ring3) sum_expr_ring_type:**
assumes \( n \in \text{nat} \) \( \forall k \in n. \ q(k) \in R \) \( \forall k \in n. \ p(k) \in R \)
shows \( \forall k \in n. \ q(k)+p(k) \in R \) and \( \left( \sum \{ (k,q(k)+p(k)). k \in n \} \right) \in R \)

**Combining mult_elem_ring_type and sum_expr_ring_type we obtain that a (kind of) linear combination of expressions whose values belong to a ring belongs to the ring.**

**lemma (in ring3) lin_comb_expr_ring_type:**
assumes \( n \in \text{nat} \) \( x \in R \) \( y \in R \) \( \forall k \in n. \ q(k) \in R \) \( \forall k \in n. \ p(k) \in R \)
shows \( \forall k \in n. \ q(k) \cdot x+p(k) \cdot y \in R \) and \( \left( \sum \{ (k,q(k) \cdot x+p(k) \cdot y)). k \in n \} \right) \in R \)

**A ring3 version of seq_sum_pull_one_elem from Monoid_ZF_1:**

**lemma (in ring3) rng_seq_sum_pull_one_elem:**
assumes \( j \in \text{nat} \) \( \forall k \in j + 1. \ q(k) \in R \)
shows \( \left( \sum \{ (k,q(k)). k \in j + 1 \} \right) = q(0)+\left( \sum \{ (k,q(k \#+ 1)). k \in j \} \right) \)

**Distributive laws for finite sums in a ring:** \( \left( \sum_{k=0}^{n-1} q(k) \right) \cdot x = \sum_{k=0}^{n-1} q(k) \cdot x \)
and \( x \cdot \left( \sum_{k=0}^{n-1} q(k) \right) = \sum_{k=0}^{n-1} x \cdot q(k) \).

**theorem (in ring3) fin_sum_distrib:**
assumes \( x \in R \) \( n \in \text{nat} \) \( \forall k \in n. \ q(k) \in R \)
shows \( \left( \sum \{ (k,q(k)). k \in n \} \right) \cdot x = \sum \{ (k,q(k) \cdot x). k \in n \} \)
\( x \cdot \left( \sum \{ (k,q(k)). k \in n \} \right) = \sum \{ (k,x \cdot q(k)). k \in n \} \)

**proof -**

| 487 |
{ fix } j assume j∈n and
  I: (∑(k,q(k)). k∈j})·x = (∑{k,q(k)·x}. k∈j})
from assms(2) <j∈n> have j∈nat using elem_nat_is_nat(2)
  by simp
moreover from assms(2,3) <j∈n> have II: ∀k∈j #+ 1. q(k) ∈ R
  using mem_add_one_subset by blast
ultimately have
    (∑{k,q(k)}. k∈j #+ 1}) = (∑{k,q(k)}. k∈j}) + q(j)
using add_monoid.seq_sum_pull_one_elem(2) by simp
hence
    (∑{k,q(k)}. k∈j #+ 1})·x = ((∑{k,q(k)}. k∈j}) + q(j))·x
by simp
moreover from assms(1) <j∈n> have II
    (∑{k,q(k)}. k∈j}) ∈ R q(j) ∈ R and x∈R
  using add_monoid.sum_in_mono by simp_all
ultimately have
    (∑{k,q(k)}. k∈j #+ 1})·x = (∑{k,q(k)·x}. k∈j}) + q(j)·x
with I have
    (∑{k,q(k)}. k∈j #+ 1})·x = (∑{k,q(k)·x}. k∈j}) + q(j)·x
by simp
moreover from assms(1) II have ∀k∈j #+ 1. q(k)·x ∈ R
  using Ring_ZF_1_L4(3) by simp
with <j∈nat> have
    (∑{k,q(k)·x}. k∈j #+ 1}) = (∑{k,q(k)·x}. k∈j}) + q(j)·x
using add_monoid.seq_sum_pull_one_elem(2) by simp
ultimately have
    (∑{k,q(k)}. k∈j #+ 1})·x = (∑{k,q(k)·x}. k∈j #+ 1})
by simp
} thus thesis by simp
qed
ultimately show (∑{k,q(k)}. k∈n})·x = ∑{k,q(k)·x}. k∈n
by (rule fin_nat_ind1)
from assms(1,2) have n∈nat and
  x·(∑{k,q(k)}. k∈0}) = ∑{k,x·q(k)}. k∈0
using add_monoid.sum_empty Ring_ZF_1_L6(2) by simp_all
moreover have
  ∀j∈n. x·(∑{k,q(k)}. k∈j}) = (∑{k,x·q(k)}. k∈j})
→ x·(∑{k,q(k)}. k∈j #+ 1}) = ∑{k,x·q(k)}. k∈j #+ 1}
proof -
{ fix } j assume j∈n and
  I: x·(∑{k,q(k)}. k∈j}) = (∑{k,x·q(k)}. k∈j})
from assms(2) <j∈n> have j∈nat using elem_nat_is_nat(2)
  by simp
moreover from assms(2,3) <j∈n> have II: ∀k∈j #+ 1. q(k) ∈ R
  using mem_add_one_subset by blast
ultimately have
    (∑{k,q(k)}. k∈j #+ 1}) = (∑{k,q(k)}. k∈j}) + q(j)
488
using add_monoid.seq_sum_pull_one_elem(2) by simp

hence
\[ x \cdot \left( \sum_{k \in j \#+ 1} \{ k, q(k) \} \right) = x \cdot \left( \left( \sum_{k \in j} \{ k, q(k) \} \right) + q(j) \right) \]
by simp

moreover from assms(1) \langle j \in \mathbb{N} \rangle II have
\[ (\sum_{k \in j} \{ k, q(k) \}) \in R \quad q(j) \in R \quad \text{and} \quad x \in R \]
using add_monoid.sum_in_mono by simp_all

ultimately have
\[ x \cdot \left( \sum_{k \in j \#+ 1} \{ k, q(k) \} \right) = \left( \sum_{k \in j} \{ k, x \cdot q(k) \} \right) + x \cdot q(j) \]
using ring_oper_distr(1) by simp

with I have
\[ x \cdot \left( \sum_{k \in j \#+ 1} \{ k, q(k) \} \right) = \left( \sum_{k \in j \#+ 1} \{ k, x \cdot q(k) \} \right) + x \cdot q(j) \]
by simp

moreover from assms(1) II have \forall k \in j \#+ 1. x \cdot q(k) \in R

using add_monoid.seq_sum_pull_one_elem(2) by simp

ultimately have
\[ x \cdot \left( \sum_{k \in j \#+ 1} \{ k, q(k) \} \right) = \left( \sum_{k \in j \#+ 1} \{ k, x \cdot q(k) \} \right) + x \cdot q(j) \]
by simp

thus thesis by simp
qed

ultimately show \[ x \cdot \left( \sum_{k \in \mathbb{N}} \{ k, q(k) \} \right) = \sum_{k \in \mathbb{N}} \{ k, x \cdot q(k) \} \]
by (rule fin_nat_ind1)

qed

In rings we have \[ \sum_{k=0}^{n-1} q(k) + p(k) = (\sum_{k=0}^{n-1} p(k)) + (\sum_{k=0}^{n-1} q(k)) \]. This is the same as theorem \texttt{sum_comm_distrib} in \texttt{Monoid.ZF.1.thy}, except that we do not need the assumption about commutativity of the operation as addition in rings is always commutative.

lemma (in ring3) \texttt{sum_ring_distrib}:
assumes \( n \in \mathbb{N} \) and \( \forall k \in \mathbb{N}. p(k) \in R \equiv q(k) \in R \)
shows \( \sum_{k \in \mathbb{N}} \{ k, p(k) + q(k) \} = \sum_{k \in \mathbb{N}} \{ k, p(k) \} + \sum_{k \in \mathbb{N}} \{ k, q(k) \} \)
using assms Ring.ZF.1_L1(3) add_monoid.sum_comm_distrib by simp

To shorten the notation in the proof of the binomial theorem we give a name to the binomial term \( \binom{n}{k} x^{n-k} y^k \).

definition (in ring3) \texttt{BT} where
\[ \texttt{BT}(n, k, x, y) \equiv \text{Binom}(n, k) \cdot \text{pow}(n \#- k, x) \cdot \text{pow}(k, y) \]

If \( n, k \) are natural numbers and \( x, y \) are ring elements then the binomial term is an element of the ring.

lemma (in ring3) \texttt{bt_type}:
assumes \( n \in \mathbb{N} \) and \( k \in \mathbb{N} \) and \( x \in R \equiv y \in R \)
shows \( \texttt{BT}(n, k, x, y) \in R \)
using assms mult_pow_type binom_in_nat Ring.ZF.1_L4(3)
The binomial term is 1 when the \( n = 0 \) and \( k = 0 \). Somehow we do not need the assumption that \( x, y \) are ring elements.

**Lemma (in ring3) bt_at_zero:** shows \( BT(0,0,x,y) = 1 \)
using binom_zero_zero mult_pow_zero(2) add_monoid.nat_mult_one
Ring_ZF_1_L2(2) Ring_ZF_1_L3(5)

unfolding BT_def by simp

The binomial term is \( x^n \) when \( k = 0 \).

**Lemma (in ring3) bt_at_zero1:** assumes \( n \in \text{nat} \ x \in \text{R} \)
shows \( BT(n,0,x,y) = pow(n,x) \)
unfolding BT_def using assms mult_pow_zero(2) binom_left_boundary
mult_pow_type(2) add_monoid.nat_mult_one Ring_ZF_1_L3(5)
by simp

When \( k = 0 \) multiplying the binomial term by \( x \) is the same as adding one to \( n \).

**Lemma (in ring3) bt_at_zero2:** assumes \( n \in \text{nat} \ y \in \text{R} \)
shows \( BT(n,n,x,y) = pow(n,y) \)
unfolding BT_def using assms binom_right_boundary mult_pow_zero(2)
add_monoid.nat_mult_one Ring_ZF_1_L2(2) mult_pow_type(2) Ring_ZF_1_L3(6)
by simp

The binomial term is \( y^n \) when \( k = n \).

**Lemma (in ring3) bt_at_right:** assumes \( n \in \text{nat} \ y \in \text{R} \)
shows \( BT(n,n,x,y) = pow(n,y) \)
unfolding BT_def using assms mult_pow_zero(2) add_monoid.nat_mult_one Ring_ZF_1_L3(5)
by simp

When \( k = n \) multiplying the binomial term by \( x \) is the same as adding one to \( n \).

**Lemma (in ring3) bt_at_right1:** assumes \( n \in \text{nat} \ y \in \text{R} \)
shows \( BT(n,n,x,y) = pow(n,y) \)
unfolding BT_def using assms binom_right_boundary mult_pow_zero(2)
add_monoid.nat_mult_one Ring_ZF_1_L2(2) mult_pow_type(2) Ring_ZF_1_L3(6)
by simp

A key identity for binomial terms needed for the proof of the binomial theorem:

**Lemma (in ring3) bt_rec_identity:**
assumes \( M \text{ is commutative on} \ R \ j \in \text{nat} \ k \in j \ x \in \text{R} \ y \in \text{R} \)
shows \( BT(j,k #+ 1,x,y) \cdot x + BT(j,k,x,y) \cdot y = BT(j #+ 1,k #+ 1,x,y) \)
proof -
from assms(2,3,4) have \( k \in \text{nat} \ Binom(j,k #+ 1) \in \text{nat} \)
and \( Binom(j,k) \in \text{nat} \ Binom(j #+ 1,k #+ 1) \in \text{nat} \)
and I: \( pow(j #- (k #+ 1),x) \in \text{R} \) and II: \( pow(j #- k,x) \in \text{R} \)
using elem_nat_is_nat(2) binom_in_nat mult_pow_type(2)
by simp_all

490
with assms(5) have III: \( \text{pow}(k #+ 1,y) \in \mathbb{R} \)
using mult_pow_type(2) by blast
let \( L = BT(j,k #+ 1,x,y) \cdot x + BT(j,k,x,y) \cdot y \)
let \( p = \text{pow}(j #- k,x) \cdot \text{pow}(k #+ 1,y) \)
from assms(2,4) \( k \in \mathbb{N} \) III have \( p \in \mathbb{R} \)
using mult_pow_type(2) Ring_ZF_1_L4(3) by simp
from assms(2,3,4,5) have \( BT(j,k,x,y) \cdot y = \text{Binom}(j,k) \cdot p \)
using elem_nat_is_nat(2) binom_in_nat mult_pow_type(2)
  nat_mult_assoc1 Ring_ZF_1_L11(2) nat_mult_pow_add_one(2)
unfolding BT_def by simp
moreover have \( BT(j,k #+ 1,x,y) \cdot x = \text{Binom}(j,k #+ 1) \cdot p \)
proof -
  from assms(1,4) have \( \text{Binom}(j,k #+ 1) \in \mathbb{N} \) III have \( x = \text{Binom}(j,k #+ 1) \cdot \text{pow}(j #- (k #+ 1) #+ 1,x) \cdot \text{pow}(k #+ 1,y) \)
  using nat_mult_assoc1 Ring_ZF_2_L4(2) nat_mult_pow_add_one(2) by simp
with assms(2,3) have \( \text{pow}(j #- (k #+ 1) #+ 1,x) = \text{pow}(j #- k,x) \)
  using nat_subtr_simpl0 by simp
ultimately show thesis unfolding BT_def by simp
qed
ultimately have \( L = \text{Binom}(j,k #+ 1) \cdot p + \text{Binom}(j,k) \cdot p \)
by simp
with assms(2,3) have \( \text{Binom}(j,k #+ 1) \in \mathbb{N} \) <Binom(j,k #+ 1) #+ 1> \( p \in \mathbb{R} \)
proof -
  from assms(3,4) have \( \sum_{k=0}^{n} \left\langle k, \text{Binom}(n,k) \cdot \text{pow}(n #- k,x) \cdot \text{pow}(k,y) \right\rangle = 1 \)
  using bt_at_zero Ring_ZF_1_L2(2) add_monoid.seq_sum_singleton by simp
491
then show thesis using mult_pow_zero(2) by simp

qed

moreover have \( \forall j \in \mathbb{N}. \)

\[
pow(j, x+y) = (\sum\{(k, BT(j, k, x, y)) \cdot k \in j\}#1)\]  
\( \rightarrow \)

\[
pow(j #+ 1, x+y) = (\sum\{(k, BT(j #+ 1, k, x, y)) \cdot k \in j #+ 1\}#1)
\]

proof -

{ fix j

let \( s_0 = \sum\{(k, BT(j, k, x, y)) \cdot k \in j #+ 1\} \)

let \( s_1 = \sum\{(k, BT(j, k, x, y) \cdot x) \cdot k \in j #+ 1\} \)

let \( s_2 = \sum\{(k, BT(j, k, x, y) \cdot y) \cdot k \in j #+ 1\} \)

let \( s_3 = \sum\{(k, BT(j, k #+ 1, x, y)) \cdot k \in j\} \)

let \( s_4 = \sum\{(k, BT(j, k, x, y) \cdot x) \cdot k \in j\} \)

let \( s_5 = \sum\{(k, BT(j, k #+ 1, x, y) \cdot x + BT(j, k, x, y)) \cdot k \in j\} \)

let \( s_6 = \sum\{(k, BT(j #+ 1, k, x, y)) \cdot k \in j #+ 1\} \)

let \( s_7 = \sum\{(k, BT(j #+ 1, k, x, y)) \cdot k \in j #+ 1\} \)

let \( s_8 = \sum\{(k, BT(j #+ 1, k, x, y) \cdot x + BT(j, k, x, y)) \cdot k \in j\} \)

assume \( j \in \mathbb{N} \) and \( pow(j, x+y) = s_0 \)

then have \( j #+ 1 \in \mathbb{N} \) and \( j #+ 1 #+ 1 \in \mathbb{N} \) by simp_all

have I: \( \forall k \in j #+ 1. BT(j, k, x, y) \in R \) and

II: \( \forall k \in j #+ 1. BT(j, k, x, y) \cdot x \in R \) and

III: \( \forall k \in j #+ 1. BT(j, k, x, y) \cdot y \in R \) and

IV: \( \forall k \in j. BT(j, k #+ 1, x, y) \in R \) and

V: \( \forall k \in j. BT(j, k, x, y) \cdot x \in R \) and

VI: \( \forall k \in j #+ 1. BT(j #+ 1, k, x, y) \in R \) and

VII: \( \forall k \in j #+ 1 #+ 1. BT(j #+ 1, k, x, y) \in R \) and

proof -

from assms(3,4) \( \langle j \in \mathbb{N} \rangle \) show \( \forall k \in j #+ 1. BT(j, k, x, y) \in R \) using elem_nat_is_nat(2) bt_type by blast

with assms(3,4) \( \langle j \in \mathbb{N} \rangle \) show

\( \forall k \in j #+ 1. BT(j, k, x, y) \cdot x \in R \) and

\( \forall k \in j #+ 1. BT(j, k, x, y) \cdot y \in R \) and

\( \forall k \in j. BT(j, k, x, y) \cdot y \in R \) using Ring_ZF_1_L4(3) by simp_all

from assms(3,4) \( \langle j \in \mathbb{N} \rangle \) have \( \forall k \in j. BT(j, k #+ 1, x, y) \in R \) using elem_nat_is_nat(2) bt_type by simp

with \( \langle j \in \mathbb{N} \rangle \) assms(3) show \( \forall k \in j. BT(j, k #+ 1, x, y) \cdot x \in R \) using mult_elem_ring_type(1) by simp

from assms(3,4) \( \langle j \#+ 1 \in \mathbb{N} \rangle \) show

\( \forall k \in j \#+ 1. BT(j \#+ 1, k, x, y) \in R \) using elem_nat_is_nat(2) bt_type by blast

from assms(3,4) \( \langle j \#+ 1 \#+ 1 \in \mathbb{N} \rangle \) show

\( \forall k \in j \#+ 1 \#+ 1. BT(j \#+ 1, k, x, y) \in R \) using elem_nat_is_nat(2) bt_type by blast

qed

have \( pow(j #+ 1, x+y) = s_0 \cdot x + s_0 \cdot y \) proof -

from assms(3,4) \( \langle j \in \mathbb{N} \rangle \) have

\( pow(j #+ 1, x+y) = pow(j, x+y) \cdot x + pow(j, x+y) \cdot y \)
using Ring_ZF_1_L4(1) mult_pow_type nat_mult_pow_add_one(2)

ring_oper_distr(1) by simp
with ‹pow(j,x+y) = s0› show thesis by simp
qed
also have s0·x + s0·y = s1 + s2
proof -
from assms(3) ‹j #+ 1 ∈ nat› I have s0·x = s1
by (rule fin_sum_distrib)
moreover from assms(4) ‹j #+ 1 ∈ nat› I have s0·y = s2 by (rule fin_sum_distrib)
ultimately show thesis by simp
qed
also have s1 + s2 = (BT(j,0,x,y)·x + s3) + (s4 + BT(j,j,x,y)·y)
proof -
from ‹j ∈ nat› II have s1 = BT(j,0,x,y)·x + s3
using rng_seq_sum_pull_one_elem(1) by simp
moreover from ‹j ∈ nat› III have s2 = s4 + BT(j,j,x,y)·y
using rng_seq_sum_pull_one_elem(2) by simp
ultimately show thesis by simp
qed
also from assms(3,4) IV V ‹j ∈ nat› have
(BT(j,0,x,y)·x + s3) + (s4 + BT(j,j,x,y)·y) =
BT(j,0,x,y)·x + (s3 + s4) + BT(j,j,x,y)·y
using bt_type Ring_ZF_1_L4(3) add_monoid.sum_in_mono Ring_ZF_2_L2(3)
by simp
also have BT(j,0,x,y)·x + (s3 + s4) + BT(j,j,x,y)·y =
BT(j,0,x,y)·x + s5 + BT(j,j,x,y)·y
proof -
from ‹j ∈ nat› IV V have s3 + s4 = s5
using sum_ring_distrib by simp
thus thesis by simp
qed
also from assms(1,3,4) ‹j ∈ nat› have
BT(j,0,x,y)·x + s5 + BT(j,j,x,y)·y =
BT(j,0,x,y)·x + s6 + BT(j,j,x,y)·y
using bt_rec_identity by simp
also have BT(j,0,x,y)·x + s6 + BT(j,j,x,y)·y = s7 + BT(j,j,x,y)·y
proof -
from ‹j ∈ nat› VI have s7 = BT(j #+ 1,0,x,y) + s6
by (rule rng_seq_sum_pull_one_elem)
with assms(3) ‹j ∈ nat› show thesis
using bt_at_zero2 by simp
qed
also have s7 + BT(j,j,x,y)·y = s8
proof -
from ‹j #+ 1 ∈ nat› VII have

493
\[
s_8 = s_7 + BT(j + 1,j + 1,x,y)
\]
by (rule rng_seq_sum_pull_one_elem)
with assms(4) \( \langle j \in \text{nat} \rangle \) show thesis
using bt_at_right1 by simp
qed

finally have pow(j + 1,x+y) = s_8 by simp
} thus thesis by simp
qed
ultimately have pow(n,x+y) = \[
\sum\{BT(n,k,x,y). k \in n + 1\}
\]
by (rule ind_on_nat1)
then show thesis unfolding BT_def by simp
qed

end

42 More on rings

theory Ring_ZF_1 imports Ring_ZF Group_ZF_3

begin

This theory is devoted to the part of ring theory specific the construction of real numbers in the Real_ZF_x series of theories. The goal is to show that classes of almost homomorphisms form a ring.

42.1 The ring of classes of almost homomorphisms

Almost homomorphisms do not form a ring as the regular homomorphisms do because the lifted group operation is not distributive with respect to composition – we have \( s \circ (r \cdot q) \neq s \circ r \cdot s \circ q \) in general. However, we do have \( s \circ (r \cdot q) \approx s \circ r \cdot s \circ q \) in the sense of the equivalence relation defined by the group of finite range functions (that is a normal subgroup of almost homomorphisms, if the group is abelian). This allows to define a natural ring structure on the classes of almost homomorphisms.

The next lemma provides a formula useful for proving that two sides of the distributive law equation for almost homomorphisms are almost equal.

lemma (in group1) Ring_ZF_1_1_L1:
assumes A1: \( s \in \text{AH} \) \( r \in \text{AH} \) \( q \in \text{AH} \) and A2: \( n \in G \)
shows \[
((s \circ (r \cdot q))(n))^{-1} = \delta(s,(r(n),q(n)))
\]
proof -
from groupAssum isAbelian A1 have T1:
\( r \cdot q \in \text{AH} \) \( s \circ r \in \text{AH} \) \( s \circ q \in \text{AH} \) \( (s \circ r) \cdot (s \circ q) \in \text{AH} \)
\( r \circ s \in \text{AH} \) \( q \circ s \in \text{AH} \) \( (r \circ s) \cdot (q \circ s) \in \text{AH} \)
using Group_ZF_3_2_L15 Group_ZF_3_4_T1 by auto

494
from A1 A2 have T2: r(n) ∈ G q(n) ∈ G s(n) ∈ G
s(r(n)) ∈ G s(q(n)) ∈ G δ(s,(r(n),q(n))) ∈ G
s(r(n))s(q(n)) ∈ G r(s(n)) ∈ G q(s(n)) ∈ G
r(s(n))q(s(n)) ∈ G
using AlmostHoms_def apply_funtype Group_ZF_3_2_L4B
group0_2_L1 monoid0.group0_1_L1 by auto
with T1 A1 A2 isAbelian show
((s◦(r·q))(n))·(((s◦r)·(s◦q))(n))⁻¹ = δ(s,(r(n),q(n)))
((r·q)·s)(n) = ((r·s)·(q·s))(n)
using Group_ZF_3_2_L12 Group_ZF_3_4_L2 Group_ZF_3_4_L1 group0_4_L6A by auto
qed

The sides of the distributive law equations for almost homomorphisms are almost equal.

lemma (in group1) Ring_ZF_1_1_L2:
  assumes A1: s∈AH r∈AH q∈AH
  shows s◦(r·q) ∼ = (s◦r)·(s◦q)
(r·q)·s = (r·s)·(q·s)
proof -
  from A1 have ∀ n∈G. ⟨r(n),q(n)⟩ ∈ G×G
  using AlmostHoms_def apply_funtype by auto
  moreover from A1 have {δ(s,x). x ∈ G×G} ∈ Fin(G)
  using AlmostHoms_def by simp
  ultimately have {δ(s,(r(n),q(n))). n∈G} ∈ Fin(G)
  by (rule Finite1_L6B)
  with A1 have
  {((s(r·q))(n))·(((s◦r)·(s◦q))(n))⁻¹. n ∈ G} ∈ Fin(G)
  using Ring_ZF_1_1_L1 by simp
  moreover from groupAssum isAbelian A1 A1 have
  s◦(r·q) ∈ AH (s◦r)·(s◦q) ∈ AH
  using Group_ZF_3_2_L15 Group_ZF_3_4_T1 by auto
  ultimately show s◦(r·q) ∼ = (s◦r)·(s◦q)
  using Group_ZF_3_4_L12 by simp
  from groupAssum isAbelian A1 have
  (r·q)·s : G→G (r·s)·(q·s) : G→G
  using Group_ZF_3_2_L15 Group_ZF_3_4_T1 AlmostHoms_def by auto
  moreover from A1 have
  ∀ n∈G. ((r·q)·s)(n) = ((r·s)·(q·s))(n)
  using Ring_ZF_1_1_L1 by simp
  ultimately show (r·q)·s = (r·s)·(q·s)
  using fun_extension_iff by simp
qed

The essential condition to show the distributivity for the operations defined on classes of almost homomorphisms.

lemma (in group1) Ring_ZF_1_1_L3:
assumes A1: R = QuotientGroupRel(AH,Op1,FR)
and A2: a ∈ AH//R b ∈ AH//R c ∈ AH//R
shows M(a,A(b,c)) = A(M(a,b),M(a,c)) ∧
M(A(b,c),a) = A(M(b,a),M(c,a))

proof
from A2 obtain s q r where D1: s ∈ AH r ∈ AH q ∈ AH
a = R{s} b = R{q} c = R{r}
using quotient_def by auto
from A1 have T1:equiv(AH,R) using Group_ZF_3_3_L3 by simp
with A1 A3 D1 groupAssum isAbelian have
M(a,A(b,c)) = R{s ◦ (q·r)}
using Group_ZF_3_3_L4 EquivClass_1_L10 Group_ZF_3_2_L15 Group_ZF_3_4_L13A by simp
also have R{(s·(q·r))} = R{(s·q)·(s·r)}
proof -
from T1 D1 have equiv(AH,R) s·(q·r) ∼ (s·q)·(s·r)
using Ring_ZF_1_1_L2 by auto
with A1 show thesis using equiv_class_eq by simp
finally show M(a,A(b,c)) = A(M(b,a),M(c,a)) by simp
from A1 A3 T1 D1 groupAssum isAbelian show
M(A(b,c),a) = A(M(b,a),M(c,a))
using Group_ZF_3_3_L4 EquivClass_1_L10 Group_ZF_3_4_L13A Group_ZF_3_2_L15 Ring_ZF_1_1_L2 Group_ZF_3_4_T1 by simp
qed

The projection of the first group operation on almost homomorphisms is distributive with respect to the second group operation.

lemma (in group1) Ring_ZF_1_1_L4:
assumes A1: R = QuotientGroupRel(AH,Op1,FR)
shows IsDistributive(AH//R,A,M)
proof -
from A1 A2 have ∀a∈(AH//R).∀b∈(AH//R).∀c∈(AH//R).
M(a,A(b,c)) = A(M(a,b),M(a,c)) ∧
M(A(b,c),a) = A(M(b,a),M(c,a))
using Ring_ZF_1_1_L3 by simp
then show thesis using IsDistributive_def by simp
qed

The classes of almost homomorphisms form a ring.

theorem (in group1) Ring_ZF_1_1_T1:
assumes \( R = \text{QuotientGroupRel}(A^H, O^1, FR) \)
and \( A = \text{ProjFun2}(A^H, R, O^1) \)
\( M = \text{ProjFun2}(A^H, R, O^2) \)
shows \( \text{IsAring}(A^H/R, A, M) \)
using \( \text{assms QuotientGroupOp_def Group_ZF_3_3_T1 Group_ZF_3_4_T2} \)
\( \text{Ring_ZF}_1_1_L4 \) \( \text{IsAring_def} \) by simp

end

43 Ordered rings

theory OrderedRing_ZF imports Ring_ZF OrderedGroup_ZF_1

begin

In this theory file we consider ordered rings.

43.1 Definition and notation

This section defines ordered rings and sets up appropriate notation.

We define ordered ring as a commutative ring with linear order that is
preserved by translations and such that the set of nonnegative elements is
closed under multiplication. Note that this definition does not guarantee
that there are no zero divisors in the ring.

definition
\( \text{IsAnOrdRing}(R,A,M,r) \equiv \)
( \( \text{IsAring}(R,A,M) \) \land \( M \text{ is commutative on } R \) \land
\( r \subseteq R \times R \) \land \( \text{IsLinOrder}(R,r) \) \land
(\( \forall a \ b. \ \forall c \in R. \ (a, b) \in r \rightarrow \{A(a,c), A(b,c)\} \in r \) \land
(\( \text{Nonnegative}(R,A,r) \text{ is closed under } M \}))

The next context (locale) defines notation used for ordered rings. We do
that by extending the notation defined in the \text{ring0} locale and adding some
assumptions to make sure we are talking about ordered rings in this context.

locale ring1 = ring0 +

assumes \( \text{mult_commut: } M \text{ is commutative on } R \)

fixes \( r \)

assumes \( \text{ordincl: } r \subseteq R \times R \)

assumes \( \text{linord: } \text{IsLinOrder}(R,r) \)

fixes \( \text{lesseq (infix \leq 68)} \)
defines \( \text{lesseq_def [simp]: } a \leq b \equiv (a, b) \in r \)
fixes sless (infix < 68)
defines sless_def [simp]: a < b ≡ a ≤ b ∧ a ≠ b

assumes ordgroup: ∀a b. ∀c∈R. a ≤ b → a + c ≤ b + c

assumes pos_mult_closed: Nonnegative(R,A,r) {is closed under} M

fixes abs (| _ |)
defines abs_def [simp]: |a| ≡ AbsoluteValue(R,A,r)(a)

fixes positiveset (R+)
defines positiveset_def [simp]: R+ ≡ PositiveSet(R,A,r)

The next lemma assures us that we are talking about ordered rings in the ring1 context.

lemma (in ring1) OrdRing_ZF_1_L1: shows IsAnOrdRing(R,A,M,r)
  using ring0_def ringAssum mult_commut ordincl linord ordgroup
  pos_mult_closed IsAnOrdRing_def by simp

We can use theorems proven in the ring1 context whenever we talk about an ordered ring.

lemma OrdRing_ZF_1_L2: assumes IsAnOrdRing(R,A,M,r) shows ring1(R,A,M,r)
  using assms IsAnOrdRing_def ring1_axioms.intro ring0_def ring1_def by simp

In the ring1 context a ≤ b implies that a, b are elements of the ring.

lemma (in ring1) OrdRing_ZF_1_L3: assumes a ≤ b shows a∈R b∈R
  using assms ordincl by auto

Ordered ring is an ordered group, hence we can use theorems proven in the group3 context.

lemma (in ring1) OrdRing_ZF_1_L4: shows IsAnOrdGroup(R,A,r)
  r {is total on} R
  A {is commutative on} R
  group3(R,A,r)
proof -
{ fix a b g assume A1: g∈R and A2: a≤b
  with ordgroup have a+g ≤ b+g
    by simp
  moreover from ringAssum A1 A2 have
    a+g = g*a b+g = g*b
    using OrdRing_ZF_1_L3 IsAring_def IsCommutative_def by auto
  ultimately have
    a+g ≤ b+g g+a ≤ g+b
    by auto
  }
hence
\[ \forall g \in R. \ \forall a \ b. \ a \leq b \implies a + g \leq b + g \ \land \ g + a \leq g + b \]
by simp
with ringAssum ordincl linord show
IsAnOrdGroup(R,A,r,group3(R,A,r),r \{is total on\} R, A \{is commutative on\} R)
using IsAring_def Order_ZF_1_L2 IsAnOrdGroup_def group3_def IsLinOrder_def
by auto
qed

The order relation in rings is transitive.

lemma (in ring1) ring_ord_transitive: assumes A1: a \leq b \ b \leq c
shows a \leq c
proof -
  from A1 have
group3(R,A,r) \langle a,b \rangle \in r \ \langle b,c \rangle \in r
using OrdRing_ZF_1_L4 by auto
then have \langle a,c \rangle \in r by (rule group3.Group_order_transitive)
then show a \leq c by simp
qed

Transitivity for the strict order: if a < b and b \leq c, then a < c. Property of ordered groups.

lemma (in ring1) ring_strict_ord_trans: assumes A1: a<b and A2: b \leq c
shows a<c
proof -
  from A1 A2 have
group3(R,A,r) \langle a,b \rangle \in r \ \langle b,c \rangle \in r
using OrdRing_ZF_1_L4 by auto
then have \langle a,c \rangle \in r \land a \neq b \land b \neq c by (rule group3.OrderedGroup_ZF_1_L4A)
then show a<c by simp
qed

Another version of transitivity for the strict order: if a \leq b and b < c, then a < c. Property of ordered groups.

lemma (in ring1) ring_strict_ord_transit: assumes A1: a\leq b and A2: b<c
shows a<c
proof -
  from A1 A2 have
group3(R,A,r) \langle a,b \rangle \in r \ \langle b,c \rangle \in r
using OrdRing_ZF_1_L4 by auto
then have \langle a,c \rangle \in r \land a \neq c by (rule group3.group_strict_ord_transit)

The next lemma shows what happens when one element of an ordered ring is not greater or equal than another.

**lemma** (in ring1) OrdRing_ZF_1_L4A: assumes A1: a ∈ R b ∈ R and A2: ¬ (a ≤ b) shows b ≤ a (¬a) ≤ (¬b) a ≠ b

**proof** -
from A1 A2 have I: group3(R,A,r)
   r {is total on} R
   a ∈ R b ∈ R (a, b) ∉ r
   using OrdRing_ZF_1_L4 by auto
then have ⟨b,a⟩ ∈ r by (rule group3.OrderedGroup_ZF_1_L8)
   then show b ≤ a by simp
from I have ⟨GroupInv(R,A)(a),GroupInv(R,A)(b)⟩ ∈ r
   by (rule group3.OrderedGroup_ZF_1_L8)
   then show (¬a) ≤ (¬b) by simp
from I show a ≠ b by (rule group3.OrderedGroup_ZF_1_L8)
qed

A special case of OrdRing_ZF_1_L4A when one of the constants is 0. This is useful for many proofs by cases.

**corollary** (in ring1) ord_ring_split2: assumes A1: a ∈ R shows a ≤ 0 ∨ (0 ≤ a ∧ a ≠ 0)

**proof** -
   { from A1 have I: a ∈ R 0 ∈ R
     using Ring_ZF_1_L2 by auto
     moreover assume A2: ¬ (a ≤ 0)
     ultimately have 0 ≤ a by (rule OrdRing_ZF_1_L4A)
     moreover from I A2 have a ≠ 0 by (rule OrdRing_ZF_1_L4A)
     ultimately have 0 ≤ a ∧ a ≠ 0 by simp
     then show thesis by auto
   }
   qed

Taking minus on both sides reverses an inequality.

**lemma** (in ring1) OrdRing_ZF_1_L4B: assumes a ≤ b shows (¬b) ≤ (¬a)

**proof** -
using assms OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_1_L5 by simp

The next lemma just expands the condition that requires the set of non-negative elements to be closed with respect to multiplication. These are properties of totally ordered groups.

**lemma** (in ring1) OrdRing_ZF_1_L5:

assumes 0 ≤ a  0 ≤ b
shows 0 ≤ a · b

500
Double nonnegative is nonnegative.

**Lemma (in ring1) OrdRing_ZF_1_L5A:** assumes $0 \leq a$
shows $0 \leq 2 \cdot a$
using assms OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_1_L5G OrdRing_ZF_1_L3 Ring_ZF_1_L3 by simp

A sufficient (somewhat redundant) condition for a structure to be an ordered ring. It says that a commutative ring that is a totally ordered group with respect to the additive operation such that set of nonnegative elements is closed under multiplication, is an ordered ring.

**Lemma OrdRing_ZF_1_L6:**
assumes IsAring(R,A,M)
M {is commutative on} R
Nonnegative(R,A,r) {is closed under} M
IsAnOrdGroup(R,A,r)
r {is total on} R
shows IsAnOrdRing(R,A,M,r)
using assms IsAnOrdGroup_def Order_ZF_1_L3 IsAnOrdRing_def by simp

$a \leq b$ iff $a - b \leq 0$. This is a fact from OrderedGroup.thy, where it is stated in multiplicative notation.

**Lemma (in ring1) OrdRing_ZF_1_L7:**
assumes $a \in R$  $b \in R$
shows $a \leq b \iff a - b \leq 0$
using assms OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_1_L9 by simp

Negative times positive is negative.

**Lemma (in ring1) OrdRing_ZF_1_L8:**
assumes $A1$: $a \leq 0$  and $A2$: $0 \leq b$
shows $a \cdot b \leq 0$
proof -
from $A1$ $A2$ have $T1$: $a \in R$  $b \in R$  $a \cdot b \in R$
  using OrdRing_ZF_1_L3 Ring_ZF_1_L4 by auto
from $A1$ $A2$ have $0 \leq (-a) \cdot b$
  using OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_1_L5A OrdRing_ZF_1_L5 by simp
with $T1$ show $a \cdot b \leq 0$
  using Ring_ZF_1_L7 OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_1_L5AA by simp
qed

We can multiply both sides of an inequality by a nonnegative ring element. This property is sometimes (not here) used to define ordered rings.
lemma (in ring1) OrdRing_ZF_1_L9:
assumes A1: a ≤ b and A2: 0 ≤ c
shows
a·c ≤ b·c
c·a ≤ c·b
proof -
from A1 A2 have T1:
apR bR cR a·c ∈ R b·c ∈ R
using OrdRing_ZF_1_L3 Ring_ZF_1_L4 by auto
with A1 A2 have (a-b)·c ≤ 0
using OrdRing_ZF_1_L7 OrdRing_ZF_1_L8 by simp
with T1 show a·c ≤ b·c
using Ring_ZF_1_L8 OrdRing_ZF_1_L7 by simp
with mult_commut T1 show c·a ≤ c·b
using IsCommutative_def by simp
qed

A special case of OrdRing_ZF_1_L9: we can multiply an inequality by a positive ring element.

lemma (in ring1) OrdRing_ZF_1_L9A:
assumes A1: a ≤ b and A2: c ∈ R +
shows
a·c ≤ b·c
c·a ≤ c·b
proof -
from A2 have 0 ≤ c using PositiveSet_def
by simp
with A1 show a·c ≤ b·c c·a ≤ c·b
using OrdRing_ZF_1_L9 by auto
qed

A square is nonnegative.

lemma (in ring1) OrdRing_ZF_1_L10:
assumes A1: a ∈ R shows 0 ≤ (a^2)
proof -
{ assume 0≤a
then have 0≤(a^2) using OrdRing_ZF_1_L5 by simp
} moreover
{ assume ¬(0≤a)
with A1 have 0≤((-a)^2)
using OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_1_L8A OrdRing_ZF_1_L5 by simp
with A1 have 0≤(a^2) using Ring_ZF_1_L14 by simp }
ultimately show thesis by blast
qed

1 is nonnegative.

corollary (in ring1) ordring_one_is_nonneg: shows 0 ≤ 1
proof -
have $0 \leq (1^2)$ using \texttt{Ring\_ZF\_1\_L2 OrdRing\_ZF\_1\_L10}
by \texttt{simp}
then show $0 \leq 1$ using \texttt{Ring\_ZF\_1\_L2 Ring\_ZF\_1\_L3}
by \texttt{simp}
qed

In nontrivial rings one is positive.

\textbf{lemma (in ring1) ordring_one_is_pos}: assumes $0 \neq 1$
shows $1 \in R$+
using assms \texttt{Ring\_ZF\_1\_L2 ordring_one_is_nonneg PositiveSet_def}
by auto

Nonnegative is not negative. Property of ordered groups.

\textbf{lemma (in ring1) OrdRing\_ZF\_1\_L11}: assumes $0 \leq a$
shows $\neg (a \leq 0 \land a \neq 0)$
using assms \texttt{OrdRing\_ZF\_1\_L4 group3.OrderedGroup\_ZF\_1\_L5AB}
by simp

A negative element cannot be a square.

\textbf{lemma (in ring1) OrdRing\_ZF\_1\_L12}: assumes $a \leq 0 \ a \neq 0$
shows $\neg (\exists b \in R. \ a = (b^2))$
proof -
\{ assume $\exists b \in R. \ a = (b^2)$
with A1 have False using \texttt{OrdRing\_ZF\_1\_L10 OrdRing\_ZF\_1\_L11}
by auto
\} then show thesis by auto
qed

If $a \leq b$, then $0 \leq b - a$.

\textbf{lemma (in ring1) OrdRing\_ZF\_1\_L13}: assumes $a \leq b$
shows $0 \leq b-a$
using assms \texttt{OrdRing\_ZF\_1\_L4 group3.OrderedGroup\_ZF\_1\_L9D}
by simp

If $a < b$, then $0 < b - a$.

\textbf{lemma (in ring1) OrdRing\_ZF\_1\_L14}: assumes $a \leq b \ a \neq b$
s show $0 \leq b-a \ 0 \neq b-a$
\texttt{b-a} $\in R$+
using assms \texttt{OrdRing\_ZF\_1\_L4 group3.OrderedGroup\_ZF\_1\_L9E}
by auto

If the difference is nonnegative, then $a \leq b$.

\textbf{lemma (in ring1) OrdRing\_ZF\_1\_L15}: assumes $a \in R \ b \in R$ and $0 \leq b-a$
shows $a \leq b$
using assms \texttt{OrdRing\_ZF\_1\_L4 group3.OrderedGroup\_ZF\_1\_L9F}
A nonnegative number is does not decrease when multiplied by a number greater or equal 1.

lemma (in ring1) OrdRing_ZF_1_L16:
  assumes A1: 0≤a and A2: 1≤b
  shows a≤a·b
proof -
  from A1 A2 have T: a∈R b∈R ab ∈ R using OrdRing_ZF_1_L3 Ring_ZF_1_L4 by auto
  from A1 A2 have 0 ≤ a·(b-1) using OrdRing_ZF_1_L13 OrdRing_ZF_1_L5 by simp
  with T show a≤a·b using Ring_ZF_1_L8 Ring_ZF_1_L2 Ring_ZF_1_L3 OrdRing_ZF_1_L15 by simp
qed

We can multiply the right hand side of an inequality between nonnegative ring elements by an element greater or equal 1.

lemma (in ring1) OrdRing_ZF_1_L17:
  assumes A1: 0≤a and A2: a≤b and A3: 1≤c
  shows a≤b·c
proof -
  from A1 A2 have 0≤b by (rule ring_ord_transitive)
  with A3 have b≤b·c using OrdRing_ZF_1_L16 by simp
  with A2 show a≤b·c by (rule ring_ord_transitive)
qed

Strict order is preserved by translations.

lemma (in ring1) ring_strict_ord_trans_inv:
  assumes a<b and c∈R
  shows a+c < b+c
  c+a < c+b
  using assms OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_1_L12B by simp

We can put an element on the other side of a strict inequality, changing its sign.

lemma (in ring1) OrdRing_ZF_1_L18:
  assumes a∈R b∈R and a-b < c
  shows a < c+b
  using assms OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_1_L12B by simp

We can add the sides of two inequalities, the first of them strict, and we get a strict inequality. Property of ordered groups.
lemma (in ring1) OrdRing_ZF_1_L19:
  assumes a<b and c≤d
  shows a+c < b+d
  using assms OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_1_L12C
  by simp

We can add the sides of two inequalities, the second of them strict and we
get a strict inequality. Property of ordered groups.

lemma (in ring1) OrdRing_ZF_1_L20:
  assumes a≤b and c<d
  shows a+c < b+d
  using assms OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_1_L12D
  by simp

43.2 Absolute value for ordered rings

Absolute value is defined for ordered groups as a function that is the identity
on the nonnegative set and the negative of the element (the inverse in the
multiplicative notation) on the rest. In this section we consider properties
of absolute value related to multiplication in ordered rings.

Absolute value of a product is the product of absolute values: the case when
both elements of the ring are nonnegative.

lemma (in ring1) OrdRing_ZF_2_L1:
  assumes 0≤a 0≤b
  shows |a·b| = |a|·|b|
  using assms OrdRing_ZF_1_L5 OrdRing_ZF_1_L4
    group3.OrderedGroup_ZF_1_L2 group3.OrderedGroup_ZF_3_L2
  by simp

The absolute value of an element and its negative are the same.

lemma (in ring1) OrdRing_ZF_2_L2: assumes a∈R
  shows |-a| = |a|
  using assms OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_3_L7A by simp

The next lemma states that |a·(-b)| = |(-a)·b| = |(-a)·(-b)| = |a·b|.

lemma (in ring1) OrdRing_ZF_2_L3:
  assumes a∈R  b∈R
  shows |(-a)·b| = |a·b|
    |a·(-b)| = |a·b|
    |(-a)·(-b)| = |a·b|
  using assms Ring_ZF_1_L4 Ring_ZF_1_L7 Ring_ZF_1_L7A
    OrdRing_ZF_2_L2 by auto

This lemma allows to prove theorems for the case of positive and negative
elements of the ring separately.

505
lemma (in ring1) OrdRing_ZF_2_L4: assumes a∈R and 0≠a
  shows 0 ≤ -a
  using assms OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_1_L8A
  by auto

Absolute value of a product is the product of absolute values.

lemma (in ring1) OrdRing_ZF_2_L5: assumes A1: a∈R b∈R
  shows |a·b| = |a|·|b|
proof -
  { assume A2: 0≤a have |a·b| = |a|·|b|
    proof -
      { assume 0≤b
        with A2 have |a|·b = |a|·|b|
          using OrdRing_ZF_2_L1 by simp }
    moreover
      { assume ¬(0≤b)
        with A1 A2 have |a|·(-b) = |a|·|b|
          using OrdRing_ZF_2_L4 OrdRing_ZF_2_L1 by simp
        with A1 have |a|·b = |a|·|b|
          using OrdRing_ZF_2_L2 OrdRing_ZF_2_L3 by simp }
    ultimately show thesis by blast
    qed }
  moreover
  { assume ¬(0≤a)
    with A1 have A3: 0 ≤ (-a)
      using OrdRing_ZF_2_L4 by simp
    have |a|·b = |a|·|b|
      proof -
        { assume 0≤b
          with A1 A3 have |(‐a)·b| = |a|·|b|
            using OrdRing_ZF_2_L1 by simp
          with A1 have |a|·b = |a|·|b|
            using OrdRing_ZF_2_L2 OrdRing_ZF_2_L3 by simp }
        moreover
          { assume ¬(0≤b)
            with A1 A3 have |(‐a)·(‐b)| = |a|·|b|
              using OrdRing_ZF_2_L4 OrdRing_ZF_2_L1 by simp
            with A1 have |a|·b = |a|·|b|
              using OrdRing_ZF_2_L2 OrdRing_ZF_2_L3 by simp }
          ultimately show thesis by blast
          qed }
    ultimately show thesis by blast
    qed }

Triangle inequality. Property of linearly ordered abelian groups.

lemma (in ring1) ord_ring_triangle_ineq: assumes a∈R b∈R
  shows |a+b| ≤ |a|+|b|
  using assms OrdRing_ZF_1_L4 group3.OrdGroup_triangle_ineq
If \( a \leq c \) and \( b \leq c \), then \( a + b \leq 2 \cdot c \).

**lemma (in ring1) OrdRing_ZF_2_L6:**

\[ \text{assumes } a \leq c \quad b \leq c \quad \text{shows } a + b \leq 2 \cdot c \]

\[ \text{using assms OrdRing_ZF_1_L4 \ group3.OrderedGroup_ZF_1_L5B} \]

\[ \text{OrdRing_ZF_1_L3 \ Ring_ZF_1_L3 by simp} \]

### 43.3 Positivity in ordered rings

This section is about properties of the set of positive elements \( R_+ \).

The set of positive elements is closed under ring addition. This is a property of ordered groups, we just reference a theorem from OrderedGroup_ZF theory in the proof.

**lemma (in ring1) OrdRing_ZF_3_L1:**

\[ \text{shows } R_+ \text{ closed under} \ A \]

\[ \text{using OrdRing_ZF_1_L4 \ group3.OrderedGroup_ZF_1_L13} \]

\[ \text{by simp} \]

Every element of a ring can be either in the positive set, equal to zero or its opposite (the additive inverse) is in the positive set. This is a property of ordered groups, we just reference a theorem from OrderedGroup_ZF theory.

**lemma (in ring1) OrdRing_ZF_3_L2:**

\[ \text{assumes } a \in R \]

\[ \text{shows } \text{Exactly_1_of_3_holds (} a = 0 \quad a \in R_+ \quad (-a) \in R_+ \text{)} \]

\[ \text{using assms OrdRing_ZF_1_L4 \ group3.OrdGroup_decomp} \]

\[ \text{by simp} \]

If a ring element \( a \neq 0 \), and it is not positive, then \(-a\) is positive.

**lemma (in ring1) OrdRing_ZF_3_L2A:**

\[ \text{assumes } a \in R \quad a \neq 0 \quad a \notin R_+ \]

\[ \text{shows } (-a) \in R_+ \]

\[ \text{using assms OrdRing_ZF_1_L4 \ group3.OrdGroup_cases} \]

\[ \text{by simp} \]

\( R_+ \) is closed under multiplication if the ring has no zero divisors.

**lemma (in ring1) OrdRing_ZF_3_L3:**

\[ \text{shows } (R_+ \text{ closed under} \ M) \iff \text{HasNoZeroDivs(R,A,M)} \]

**proof**

\[ \text{assume A1: HasNoZeroDivs(R,A,M)} \]

\[ \{ \text{fix } a \quad b \quad \text{assume } a \in R_+ \quad b \in R_+ \]

\[ \text{then have } 0 \leq a \quad a \neq 0 \quad 0 \leq b \quad b \neq 0 \]

\[ \text{using PositiveSet_def by auto} \]

\[ \text{with A1 have } a \cdot b \in R_+ \]

\[ \text{using OrdRing_ZF_1_L5 \ Ring_ZF_1_L2 \ OrdRing_ZF_1_L3 \ Ring_ZF_1_L12} \]

\[ \text{OrdRing_ZF_1_L4 \ group3.OrderedGroup_ZF_1_L12} \]

\[ \text{by simp} \]

\[ \} \text{ then show } R_+ \text{ closed under} \ M \text{ using IsOpClosed_def} \]

\[ \text{by simp} \]
next assume \( A2: R, \{\text{is closed under}\} M \)
\[
\{ \text{fix } a \ b \ \text{assume } A3: a \in R \ b \in R \ \text{and } a \neq 0 \ b \neq 0 \\
\text{with } A2 \ \text{have } |a-b| \in R_+ \\
\text{using } \text{OrdRing}_1 \text{ZF}_1 \text{L4 group3.OrderedGroup}_3 \text{ZF}_3 \text{L12 IsOpClosed_def} \\
\text{OrdRing}_2 \text{ZF}_2 \text{L5 by simp} \\
\text{with } A3 \ \text{have } a \cdot b \neq 0 \\
\text{using } \text{PositiveSet_def Ring}_1 \text{L4} \\
\text{OrdRing}_1 \text{ZF}_1 \text{L4 group3.OrderedGroup}_3 \text{ZF}_3 \text{L2A by auto} \\
\} \text{ then show } \text{HasNoZeroDivs}(R,A,M) \text{ using HasNoZeroDivs_def by auto} \\
\]
qed

Another (in addition to \( \text{OrdRing}_1 \text{ZF}_1 \text{L6} \) sufficient condition that defines order in an ordered ring starting from the positive set.

\text{theorem (in ring0) ring_ord_by_positive_set:}
\text{assumes}
A1: \( M \{\text{is commutative on}\} R \) \text{ and}
A2: \( P \subseteq R \ P \{\text{is closed under}\} A \ 0 \not\in P \) \text{ and}
A3: \( \forall a\in R. \ a \neq 0 \longrightarrow (a \in P) \ \text{Xor} ((-a) \in P) \) \text{ and}
A4: \( P \{\text{is closed under}\} M \) \text{ and}
A5: \( r = \text{OrderFromPosSet}(R,A,P) \)
\text{shows}
IsAnOrdGroup(R,A,r)
IsAnOrdRing(R,A,M,r)
r \{\text{is total on}\} R
PositiveSet(R,A,r) = P
Nonnegative(R,A,r) = P \cup \{0\}
HasNoZeroDivs(R,A,M)
\text{proof -}
from A2 A3 A5 show
I: \( \text{IsAnOrdGroup}(R,A,r) \ r \{\text{is total on}\} R \) \text{ and}
II: \( \text{PositiveSet}(R,A,r) = P \) \text{ and}
III: \( \text{Nonnegative}(R,A,r) = P \cup \{0\} \)
using \( \text{Ring}_0 \text{ZF}_1 \text{L1} \) \text{group0.Group_ord_by_positive_set by auto}
from A2 A4 III have \( \text{Nonnegative}(R,A,r) \) \{\text{is closed under}\} M
using \( \text{Ring}_1 \text{ZF}_1 \text{L16} \) by simp
with ringAssum A1 I show \( \text{IsAnOrdRing}(R,A,M,r) \)
using \( \text{OrdRing}_1 \text{ZF}_1 \text{L6} \) by simp
with A4 II show \( \text{HasNoZeroDivs}(R,A,M) \)
using \( \text{OrdRing}_1 \text{ZF}_1 \text{L2 ring1.OrdRing}_1 \text{ZF}_3 \text{L3} \) by auto
qed

Nontrivial ordered rings are infinite. More precisely we assume that the neutral element of the additive operation is not equal to the multiplicative neutral element and show that the the set of positive elements of the ring is not a finite subset of the ring and the ring is not a finite subset of itself.
theorem (in ring1) ord_ring_infinite: assumes $0 \neq 1$
shows $\mathbb{R}_+ \notin \text{Fin}(\mathbb{R})$
$\mathbb{R} \notin \text{Fin}(\mathbb{R})$
using assms ring_1_L17 OrdRing_1_L4 group3.Linord_group_infinite
by auto

If every element of a nontrivial ordered ring can be dominated by an element from $B$, then we $B$ is not bounded and not finite.

lemma (in ring1) OrdRing_3_L4: assumes $0 \neq 1$ and $\forall a \in \mathbb{R}. \exists b \in B. a \leq b$
shows $\neg \text{IsBoundedAbove}(B, r)$
$B \notin \text{Fin}(\mathbb{R})$
using assms ring_1_L17 OrdRing_1_L4 group3.OrderedGroup_2_L2A
by auto

If $m$ is greater or equal the multiplicative unit, then the set $\{m \cdot n : n \in \mathbb{R}\}$ is infinite (unless the ring is trivial).

lemma (in ring1) OrdRing_3_L5: assumes $A1: 0 \neq 1$ and $A2: 1 \leq m$
shows $\{m \cdot x. x \in \mathbb{R}\} \notin \text{Fin}(\mathbb{R})$
$\{m \cdot x. x \in \mathbb{R}\} \notin \text{Fin}(\mathbb{R})$
$\{(-m) \cdot x. x \in \mathbb{R}\} \notin \text{Fin}(\mathbb{R})$

proof -
  from $A2$ have $T: m \in \mathbb{R}$ using OrdRing_1_L3 by simp
  from $A2$ have $0 \leq 1$  $1 \leq m$
    using ordring_one_is_nonneg by auto
  then have $I: 0 \leq m$ by (rule ring_ord_transitive)
  let $B = \{m \cdot x. x \in \mathbb{R}\}$
  fix a assume $A3: a \in \mathbb{R}$
  then have $a \leq 0 \lor (0 \leq a \land a \neq 0)$
    using ord_ring_split2 by simp
  moreover
  { assume $A4: a \leq 0$
    from $A1$ have $m \cdot 1 \in B$ using ordring_one_is_pos
      by auto
    with $T$ have $m \in B$ using ring_1_L3 by simp
    moreover from $A4$ I have $a \leq m$ by (rule ring_ord_transitive)
    ultimately have $\exists b \in B. a \leq b$ by blast }
  moreover
  { assume $A4: 0 \leq a \land a \neq 0$
    with $A3$ have $m \cdot a \in B$ using PositiveSet_def
      by auto
    moreover
    from $A2$ $A4$ have $1 \cdot a \leq m \cdot a$ using OrdRing_1_L9
      by simp
    with $A3$ have $a \leq m \cdot a$ using ring_1_L3
      by simp
  }
ultimately have $\exists b \in B. a \leq b$ by auto
}
ultimately have $\exists b \in B. a \leq b$ by auto
} then have $\forall a \in R. \exists b \in B. a \leq b$
by simp
with A1 show $B \notin \text{Fin}(R)$ using OrdRing_ZF_3_L4
by simp
moreover have $B \subseteq \{m \cdot x. x \in R\}$
using PositiveSet_def by auto
ultimately show $\{m \cdot x. x \in R\} \notin \text{Fin}(R)$ using Fin_subset
by auto
with T show $\{(-m) \cdot x. x \in R\} \notin \text{Fin}(R)$ using Ring_ZF_1_L18
by simp
qed

If $m$ is less or equal than the negative of multiplicative unit, then the set $\{m \cdot n: n \in R\}$ is infinite (unless the ring is trivial).

lemma (in ring1) OrdRing_ZF_3_L6: assumes A1: $0 \neq 1$ and A2: $m \leq -1$
shows $\{m \cdot x. x \in R\} \notin \text{Fin}(R)$
proof -
from A2 have $(-(-1)) \leq -m$
using OrdRing_ZF_1_L4B by simp
with A1 have $\{(-m) \cdot x. x \in R\} \notin \text{Fin}(R)$
using Ring_ZF_1_L2 Ring_ZF_1_L3 OrdRing_ZF_3_L5
by simp
with A2 show $\{m \cdot x. x \in R\} \notin \text{Fin}(R)$
using OrdRing_ZF_1_L3 Ring_ZF_1_L18 by simp
qed

All elements greater or equal than an element of $R_+$ belong to $R_+$. Property
of ordered groups.

lemma (in ring1) OrdRing_ZF_3_L7: assumes A1: $a \in R_+$ and A2: $a \leq b$
shows $b \in R_+$
proof -
from A1 A2 have $\text{group3}(R,A,r)$
$\text{a} \in \text{PositiveSet}(R,A,r)$
$(a,b) \in r$
using OrdRing_ZF_1_L4 by auto
then have $b \in \text{PositiveSet}(R,A,r)$
by (rule group3.OrderedGroup_ZF_1_L19)
then show $b \in R_+$ by simp
qed

A special case of OrdRing_ZF_3_L7: a ring element greater or equal than 1 is
positive.

corollary (in ring1) OrdRing_ZF_3_L8: assumes A1: $0 \neq 1$ and A2: $1 \leq a$
shows $a \in R_+$
proof -
from A1 A2 have 1 ∈ R+ 1 ≤ a
using ordring_one_is_pos by auto
then show a ∈ R+ by (rule OrdRing_ZF_3_L7)
qed

Adding a positive element to a strictly increases a. Property of ordered groups.

lemma (in ring1) OrdRing_ZF_3_L9: assumes A1: a ∈ R b ∈ R+
such that a ≤ a+b a ≠ a+b
using assms OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_1_L22
by auto

A special case of OrdRing_ZF_3_L9: in nontrivial rings adding one to a increases a.

corollary (in ring1) OrdRing_ZF_3_L10: assumes A1: 0 ≠ 1 and A2: a ∈ R
such that a ≤ a+1 a ≠ a+1
using assms ordring_one_is_pos OrdRing_ZF_3_L9
by auto

If a is not greater than b, then it is strictly less than b + 1.

lemma (in ring1) OrdRing_ZF_3_L11: assumes A1: 0 ≠ 1 and A2: a ≤ b
such that a ≤ GreaterOf(r,1,a)
workings with A2 show a < b+1 by (rule ring_strict_ord_transit)
qed

For any ring element a the greater of a and 1 is a positive element that is greater or equal than m. If we add 1 to it we get a positive element that is strictly greater than m. This holds in nontrivial rings.

lemma (in ring1) OrdRing_ZF_3_L12: assumes A1: 0 ≠ 1 and A2: a ∈ R
such that a ≤ GreaterOf(r,1,a)
GreaterOf(r,1,a) ∈ R+
GreaterOf(r,1,a) + 1 ∈ R+
a ≤ GreaterOf(r,1,a) + 1 a ≠ GreaterOf(r,1,a) + 1
proof -
from linord have r {is total on} R using IsLinOrder_def
by simp
moreover from A2 have 1 ∈ R a ∈ R
using Ring_ZF_1_L2 by auto
ultimately have
1 ≤ GreaterOf(r,1,a) and
I: a ≤ GreaterOf(r,1,a)
using Order_ZF_3_L2 by auto
with A1 show
a ≤ GreaterOf(r,1,a) and
We can multiply strict inequality by a positive element.

A sufficient condition for an element to be in the set of positive ring elements.

If a ring has no zero divisors, the square of a nonzero element is positive.

In rings with no zero divisors we can (strictly) increase a positive element by multiplying it by an element that is greater than 1.
If the right hand side of an inequality is positive we can multiply it by a number that is greater than one.

**Lemma (in ring1) OrdRing_ZF_3_L17:**

- **Assumptions:**
  - A1: HasNoZeroDivs(R,A,M)
  - A2: \( b \in R^+ \)
  - A3: \( a \leq b \)
  - A4: \( 1 < c \)

- **Shows:** \( a < b \cdot c \)

**Proof:**

- From A1 A2 A4 have \( b < b \cdot c \) using OrdRing_ZF_3_L16 by auto
- With A3 show \( a < b \cdot c \) by (rule ring_strict_ord_transit)

**QED**

We can multiply a right hand side of an inequality between positive numbers by a number that is greater than one.

**Lemma (in ring1) OrdRing_ZF_3_L18:**

- **Assumptions:**
  - A1: HasNoZeroDivs(R,A,M)
  - A2: \( a \in R^+ \)
  - A3: \( a \leq b \)
  - A4: \( 1 < c \)

- **Shows:** \( a < b \cdot c \)

**Proof:**

- From A2 A3 have \( b \in R^+ \) using OrdRing_ZF_3_L7 by blast
- With A1 A3 A4 show \( a < b \cdot c \) using OrdRing_ZF_3_L17 by simp

**QED**

In ordered rings with no zero divisors if at least one of \( a, b \) is not zero, then \( 0 < a^2 + b^2 \), in particular \( a^2 + b^2 \neq 0 \).

**Lemma (in ring1) OrdRing_ZF_3_L19:**

- **Assumptions:**
  - A1: HasNoZeroDivs(R,A,M)
  - A2: \( a \in R \) \( b \in R \)
  - A3: \( a \neq 0 \lor b \neq 0 \)

- **Shows:** \( 0 < a^2 + b^2 \)

**Proof:**

- \{ assume \( a \neq 0 \) with A1 A2 have \( 0 \leq a^2 \) \( a^2 \neq 0 \) using OrdRing_ZF_3_L15 by auto
  - then have \( 0 < a^2 \) by auto
  - moreover from A2 have \( 0 \leq b^2 \) using OrdRing_ZF_1_L10 by simp
  - ultimately have \( 0 + 0 < a^2 + b^2 \) using OrdRing_ZF_1_L19 by simp
  - then have \( 0 < a^2 + b^2 \) using Ring_ZF_1_L2 Ring_ZF_1_L3 by simp \}

- moreover
  \{ assume \( A4: a = 0 \)
  - then have \( a^2 + b^2 = 0 + b^2 \) \}

513
using Ring_ZF_1_L2 Ring_ZF_1_L6 by simp
also from A2 have "... = b^2"
  using Ring_ZF_1_L4 Ring_ZF_1_L3 by simp
finally have a^2 + b^2 = b^2 by simp
moreover from A3 A4 have b ≠ 0 by simp
with A1 A2 have 0 ≤ b^2 and b^2 ≠ 0
  using OrdRing_ZF_3_L15 by auto
hence 0 < b^2 by auto
ultimately have 0 < a^2 + b^2 by simp }
ultimately show 0 < a^2 + b^2
  by auto
qed

end

44 Cardinal numbers
theory Cardinal_ZF imports ZF.CardinalArith func1
begin

This theory file deals with results on cardinal numbers (cardinals). Cardinals are a generalization of the natural numbers, used to measure the cardinality (size) of sets. Contributed by Daniel de la Concepcion.

44.1 Some new ideas on cardinals

All the results of this section are done without assuming the Axiom of Choice. With the Axiom of Choice in play, the proofs become easier and some of the assumptions may be dropped.

Since General Topology Theory is closely related to Set Theory, it is very interesting to make use of all the possibilities of Set Theory to try to classify homeomorphic topological spaces. These ideas are generally used to prove that two topological spaces are not homeomorphic.

There exist cardinals which are the successor of another cardinal, but; as happens with ordinals, there are cardinals which are limit cardinal.

definition

LimitC(i) ≡ Card(i) ∧ 0<i ∧ (∀y. (y<i ∧ Card(y)) → csucc(y)<i)

Simple fact used a couple of times in proofs.

lemma nat_less_infty: assumes n∈nat and InfCard(X) shows n<X
proof -
  from assms have n∈nat and nat≤X using lt_def InfCard_def by auto
  then show n<X using lt_trans2 by blast
There are three types of cardinals, the zero one, the successors of other cardinals and the limit cardinals.

**lemma** Card_cases_disj:
assumes Card(i)
shows i=0 | (∃ j. Card(j) ∧ i=csucc(j)) | LimitC(i)

**proof**
from assms have D: Ord(i) using Card_is_Ord by auto
{
  assume F: i≠0
  assume Contr: ~LimitC(i)
  from F D have 0<i using Ord_0_lt by auto
  with Contr assms have ∃ y. y < i ∧ Card(y) ∧ ¬ csucc(y) < i
    using LimitC_def by blast
  then obtain y where y < i ∧ Card(y) ∧ ¬ csucc(y) < i by blast
  with D have y < i i≤csucc(y) and O: Card(y)
    using not_lt_imp_le lt_Ord Card_csucc Card_is_Ord by auto
  with assms have csucc(y)≤i≤csucc(y) using csucc_le by auto
  then have i=csucc(y) using le_anti_sym by auto
  with O have ∃ j. Card(j) ∧ i=csucc(j) by auto
} thus thesis by auto
qed

Given an ordinal bounded by a cardinal in ordinal order, we can change to the order of sets.

**lemma** le_imp_lesspoll:
assumes Card(Q)
shows A ≤ Q =⇒ A ≲ Q

**proof**
- assume A ≤ Q
  then have A<Q∨A=Q using le_iff by auto
  then have A=Q∨A< Q using eqpoll_refl by auto
  with assms have A=Q∨A< Q using lt_Card_imp_lesspoll by auto
  then show A<Q using lesspoll_def eqpoll_imp_lepoll by auto
qed

There are two types of infinite cardinals, the natural numbers and those that have at least one infinite strictly smaller cardinal.

**lemma** InfCard_cases_disj:
assumes InfCard(Q)
shows Q=nat ∨ (∃ j. csucc(j)≤Q ∧ InfCard(j))

**proof**
{
  assume ∀ j. ¬ csucc(j) ≤ Q ∨ ¬ InfCard(j)
  then have D: ¬ csucc(nat) ≤ Q using InfCard_nat by auto
  with D assms have ¬(csucc(nat) ≤ Q) using le_imp_lesspoll InfCard_is_Card
}


by auto
with assms have \( Q < \text{csucc(nat)} \)
  using not_le_iff_lt Card_is_Ord Card_csucc Card_is_Ord
  Card_is_Ord InfCard_is_Card Card_nat by auto
with assms have \( Q \leq \text{nat} \) using Card_lt_csucc_iff InfCard_is_Card Card_nat
  by auto
with assms have \( Q = \text{nat} \) using InfCard_def le_anti_sym by auto
}
thus thesis by auto
qed

A more readable version of standard Isabelle/ZF \texttt{Ord_linear_lt}

\begin{lemma}{Ord_linear_lt_IML}
  \begin{assumptions}{Ord(i) Ord(j)}
  shows \( i < j \lor i = j \lor j < i \)
  \end{assumptions}
  using \text{lt_def} \text{Ord_linear} \text{disjE} by simp
\end{lemma}

A set is injective and not bijective to the successor of a cardinal if and only if it is injective and possibly bijective to the cardinal.

\begin{lemma}{Card_less_csucc_eq_le}
  \begin{assumptions}{Card(m)}
  shows \( A \prec \text{csucc(m)} \iff A \preccurlyeq m \)
  \end{assumptions}
  proof
  have \( S: \text{Ord(\text{csucc(m)})} \) using \text{Card_csucc} \text{Card_is_Ord} \text{assms} by auto
  {\begin{assumptions}{A \prec \text{csucc(m)}}
    with \( S \) have \( |A| \approx A \) using \text{lesspoll_imp_eqpoll} by auto
    also from \( A \) have \( \ldots < \text{csucc(m)} \) by auto
    finally have \( |A| < \text{csucc(m)} \) by auto
    then have \( |A| \preccurlyeq \text{csucc(m)} - (|A| \approx \text{csucc(m)}) \) using \text{lesspoll_def} by auto
    with \( S \) have \( |A| \preccurlyeq \text{csucc(m)} \) \( |A| \neq \text{csucc(m)} \) using \text{lepoll_cardinal_le} by auto
    then have \( |A| \leq \text{csucc(m)} \) \( |A| \neq \text{csucc(m)} \) using \text{Card_def} \text{Card_cardinal}
    by auto
    then have \( I: -(\text{csucc(m)} \prec |A|) \) \( |A| \neq \text{csucc(m)} \) using \text{le_imp_not_lt} by auto
    from \( S \) have \( \text{csucc(m)} \prec |A| \lor |A| = \text{csucc(m)} \lor |A| < \text{csucc(m)} \)
      using \text{Card_cardinal} \text{Card_is_Ord} \text{Ord_linear_lt_IML} by auto
    with \( I \) have \( |A| < \text{csucc(m)} \) by simp
    with \( \text{assms} \) have \( |A| \leq m \) using \text{Card_lt_csucc_iff} \text{Card_cardinal}
      by auto
    then have \( |A| = m \lor |A| < m \) using \text{le_iff} by auto
    then have \( |A| = m \lor |A| < m \) using \text{eqpoll_refl} by auto
    then have \( T: |A| \leq m \) using \text{lt_card_imp_lesspoll} \text{assms} by auto
    then have \( T: |A| \leq m \) using \text{lesspoll_def} \text{eqpoll_imp_lepoll} by auto
    from \( A \) \( S \) have \( A \approx |A| \) using \text{lesspoll_imp_eqpoll} \text{eqpoll_sym} by auto
    also from \( T \) have \( \ldots \leq m \) by auto
    finally show \( A \preccurlyeq m \) by simp
  }\end{assumptions}
\end{proof}
If the successor of a cardinal is infinite, so is the original cardinal.

\textbf{lemma \texttt{csucc\_inf\_imp\_inf}:}
\begin{itemize}
  \item \texttt{assumes \texttt{Card(j)} and \texttt{InfCard(csucc(j))}}
  \item \texttt{shows \texttt{InfCard(j)}}
\end{itemize}
\begin{proof}
\begin{itemize}
  \item \texttt{assume f:Finite (j)}
  \item \texttt{then obtain n where n nat \approx n using \texttt{Finite\_def by auto}}
  \item \texttt{with assms(1) have TT: j=n n nat using cardinal\_cong nat into Card Card\_def by auto}
  \item \texttt{then have Q: succ(j) \in nat using nat\_succI by auto}
  \item \texttt{with f TT have T: Finite(succ(j)) Card(succ(j)) using nat\_into\_Card nat\_succI by auto}
  \item \texttt{from T(2) have Card(succ(j)) \land j < succ(j) using Card\_is\_Ord by auto}
  \item \texttt{moreover from this have Ord(succ(j)) using Card\_is\_Ord by auto}
  \item \texttt{moreover}
  \begin{itemize}
    \item \texttt{fix x}
    \item \texttt{assume A: x < succ(j)}
    \begin{itemize}
      \item \texttt{assume Card(x) \land j < x with A have False using lt\_trans1 by auto}
    \end{itemize}
    \item \texttt{hence -(Card(x) \land j < x) by auto}
  \end{itemize}
  \item \texttt{ultimately have (\mu L. Card(L) \land j < L) = succ(j) by (rule Least\_equality)}
  \item \texttt{then have csucc(j) = succ(j) using csucc\_def by auto}
  \item \texttt{with Q have csucc(j) \in nat by auto}
  \item \texttt{then have csucc(j) < nat using lt\_def Card\_nat Card\_is\_Ord by auto}
  \item \texttt{with assms(2) have False using InfCard\_def lt\_trans2 by auto}
  \end{itemize}
  \item \texttt{then have -(Finite (j)) by auto}
  \item \texttt{with assms(1) show thesis using Inf\_Card\_is\_InfCard by auto}
\end{itemize}
\end{proof}

Since all the cardinals previous to \texttt{nat} are finite, it cannot be a successor cardinal; hence it is a \texttt{Limit\_C} cardinal.

\textbf{corollary \texttt{Limit\_C\_nat}:}
\begin{itemize}
  \item \texttt{shows Limit\_C(nat)}
\end{itemize}
\begin{proof}
\end{proof}
44.2 Main result on cardinals (without the Axiom of Choice)

If two sets are strictly injective to an infinite cardinal, then so is its union. For the case of successor cardinal, this theorem is done in the Isabelle library in a more general setting; but that theorem is of not use in the case where \( \text{LimitC}(Q) \) and it also makes use of the Axiom of Choice. The mentioned theorem is in the theory file Cardinal_AC.thy

Note that if \( Q \) is finite and different from 1, let’s assume \( Q = n \), then the union of \( A \) and \( B \) is not bounded by \( Q \). Counterexample: two disjoint sets of \( n-1 \) elements each have a union of \( 2n-2 \) elements which are more than \( n \).

Note also that if \( Q = 1 \) then \( A \) and \( B \) must be empty and the union is then empty too; and \( Q \) cannot be 0 because no set is injective and not bijective to 0.

The proof is divided in two parts, first the case when both sets \( A \) and \( B \) are finite; and second, the part when at least one of them is infinite. In the first part, it is used the fact that a finite union of finite sets is finite. In the second part it is used the linear order on cardinals (ordinals). This proof can not be generalized to a setting with an infinite union easily.

```lean
lemma less_less_imp_un_less:
  assumes A≺Q and B≺Q and InfCard(Q)
  shows A ∪ B≺Q
proof-
assume Finite (A) ∧ Finite(B)
then have Finite(A ∪ B) using Finite_Un by auto
then obtain n where R: A ∪ B ≈ n n ∈ nat using Finite_def
by auto
then have |A ∪ B| < nat using lt_def cardinal_cong
    nat_into_Card Card_def Card_nat Card_is_Ord by auto
with assms(3) have T: |A ∪ B| < Q using InfCard_def lt_trans2 by auto
from R have Ord(n) A ∪ B ≲ n n ≤ n using nat_into_Card Card_is_Ord eqpoll_imp_lepoll
by auto
then have A ∪ B ≈ |A ∪ B| using lepoll_Ord_imp_eqpoll eqpoll_sym by auto
also from T assms(3) have ...≺ Q using lt_Card_imp_lesspoll InfCard_is_Card
by auto
finally have A ∪ B ≺ Q by simp
moreover
-
assume -(Finite (A) ∧ Finite(B))
hence A: ~Finite (A) ∨ ~Finite(B) by auto
from assms have B: |A| ≈ A |B| ≈ B using lesspoll_imp_eqpoll lesspoll_imp_eqpoll
-InfCard_is_Card Card_is_Ord by auto
from B(1) have Aeq: ∀ x. (|A| ≈ x) −→ (A ≈ x)
    using eqpoll_sym eqpoll_trans by blast
from B(2) have Beq: ∀ x. (|B| ≈ x) −→ (B ≈ x)
    using eqpoll_sym eqpoll_trans by blast
with A Aeq have ~Finite(|A|) ∨ ~Finite(|B|) using Finite_def
by auto
then have D: InfCard(|A|) ∨ InfCard(|B|)
    using Inf_Card_is_InfCard Inf_Card_is_InfCard Card_cardinal by blast
{
    assume AS: |A| < |B|
    {
        assume ~InfCard(|A|)
        with D have InfCard(|B|) by auto
    }
moreover
{
    assume InfCard(|A|)
    then have nat ≤ |A| using InfCard_def by auto
    with AS have nat < |B| using lt_trans1 by auto
    then have nat ≤ |B| using leI by auto
    then have InfCard(|B|) using InfCard_def Card_cardinal by auto
}
ultimately have INFB: InfCard(|B|) by auto
then have 2< |B| using nat_less_infty by simp
then have AG: 2 ≤ |B| using lt_Card_imp_lesspoll Card_cardinal lesspoll_def
by auto
from B(2) have |B| ≈ B by simp
also from assms(2) have ...≺ Q by auto

519
finally have $|B| \prec Q$ by simp
from B(1) have Card($|B|$) $\preccurlyeq |A|$ using eqpoll_sym Card_cardinal eqpoll_imp_lepoll

by auto
with AS have $A \prec |B|$ using lt_Card_imp_lesspoll lesspoll_trans1 by auto
then have I1: $A \preccurlyeq |B|$ using lesspoll_def by auto
from B(2) have I2: $B \preccurlyeq |B|$ using eqpoll_sym eqpoll_imp_lepoll by auto

have $A \cup B \preccurlyeq A + B$ using Un_lepoll_sum by auto
also from I1 I2 have $\ldots \preccurlyeq |B| + |B|$ using sum_lepoll_mono by auto
also from AG have $\ldots \preccurlyeq |B| \cdot |B|$ using sum_lepoll_prod by auto
also from assms(3) INF have $\ldots \approx |B|$ using InfCard_square_eqpoll by auto
finally have $A \cup B \preccurlyeq |B|$ by simp
also from TTT have $\ldots \prec Q$ by auto
finally have $A \cup B \prec Q$ by simp

moreover
{
assume AS: $|B| < |A|
{
assume ~InfCard(|B|)
with D have InfCard(|A|) by auto
}
moreover
{
assume InfCard(|B|)
then have nat$\leq |B|$ using InfCard_def by auto
with AS have nat$< |A|$ using lt_trans1 by auto
then have nat$\leq |A|$ using leI by auto
then have InfCard(|A|) using InfCard_def Card_cardinal by auto
}
ultimately have INF: InfCard(|A|) by auto
then have $2^{|A|}$ using nat_lesseq_infty by simp
then have AG: $2^{|A|}$ using lt_Card_imp_lesspoll Card_cardinal lesspoll_def by auto
from B(1) have $|A| = A$ by simp
also from assms(1) have $\ldots \prec Q$ by auto
finally have TTT: $|A| \prec Q$ by simp
from B(2) have Card(|A|) $B \preccurlyeq |B|$ using eqpoll_sym Card_cardinal eqpoll_imp_lepoll

by auto
with AS have $B \prec |A|$ using lt_Card_imp_lesspoll lesspoll_trans1 by auto
then have I1: $B \preccurlyeq |A|$ using lesspoll_def by auto
from B(1) have I2: $A \preccurlyeq |A|$ using eqpoll_sym eqpoll_imp_lepoll by auto
have $A \cup B \preccurlyeq A + B$ using Un_lepoll_sum by auto
also from I1 I2 have $\ldots \preccurlyeq |A| + |A|$ using sum_lepoll_mono by auto

520
also from AG have $\ldots \leq |A| \cdot |A|$ using sum_lepoll_prod by auto
also from INFB assms(3) have $\ldots \approx |A|$ using InfCard_square_eqpoll
by auto
finally have $A \cup B \leq |A|$ by simp
also from TTT have $\ldots \prec Q$ by auto
finally have $A \cup B \prec Q$ by simp
} moreover
{
assume AS: $|A|=|B|
with D have INFB: InfCard(|A|) by auto
then have $2 \prec |A|$ using nat_less_infty by simp
then have AG: $2 \leq |A|$ using lt_Card_imp_lesspoll Card_cardinal using
lesspoll_def
by auto
from B(1) have $|A|\approx A$ by simp
also from assms(1) have $\ldots \prec Q$ by auto
finally have TTT: $|A|\prec Q$ by simp
from AS B have I1: $A \leq |A|$ and I2: $B \leq |A|$ using eqpoll_refl eqpoll_imp_lepoll
eqpoll_sym by auto
have $A \cup B \leq A+B$ using Un_lepoll_sum by auto
also from I1 I2 have $\ldots \leq |A| + |A|$ using sum_lepoll_mono by auto
also from AG have $\ldots \leq |A| \cdot |A|$ using sum_lepoll_prod by auto
also from assms(3) INFB have $\ldots \approx |A|$ using InfCard_square_eqpoll
by auto
finally have $A \cup B \leq |A|$ by simp
also from TTT have $\ldots \prec Q$ by auto
finally have $A \cup B \prec Q$ by simp
}
ultimately have $A \cup B \prec Q$ using Ord_linear_lt_IML Card_cardinal Card_is_Ord
by auto
}
ultimately show $A \cup B \prec Q$ by auto
qed

44.3 Choice axioms

We want to prove some theorems assuming that some version of the Axiom
of Choice holds. To avoid introducing it as an axiom we will define an
appropriate predicate and put that in the assumptions of the theorems.
That way technically we stay inside ZF.

The first predicate we define states that the axiom of $Q$-choice holds for
subsets of $K$ if we can find a choice function for every family of subsets of
$K$ whose (that family’s) cardinality does not exceed $Q$.

definition
AxiomCardinalChoice {{the axiom of}_Q {choice holds for subsets}_} where
{the axiom of} $Q$ {choice holds for subsets}$K \equiv \text{Card}(Q) \wedge (\forall M N. (M$
\[ \leq Q \land (\forall t \in M. N_t \neq 0 \land N_t \subseteq K) \rightarrow (\exists f \in \Pi(M, \lambda t. N_t) \land (\forall t \in M. f_t \in N_t)) \]

Next we define a general form of \( Q \) choice where we don’t require a collection of files to be included in a file.

definition
AxiomCardinalChoiceGen {{the axiom of}_Q {choice holds}} where
{the axiom of} Q {choice holds} \equiv Card(Q) \land (\forall M N. (M \leq Q \land (\forall t \in M. N_t \neq 0)) \rightarrow (\exists f \in \Pi(M, \lambda t. N_t) \land (\forall t \in M. f_t \in N_t)))

The axiom of finite choice always holds.

theorem finite_choice:
assumes \( n \in \text{nat} \)
shows {the axiom of} \( n \) {choice holds}
proof -
note assms(1)
moreover
{ 
  fix M N assume \( M \leq 0 \land (\forall t \in M. N_t \neq 0) \)
  then have \( N = 0 \) using lepoll_0_is_0 by auto
  then have \( \{ (t,0). t \in M \} : \Pi(M, \lambda t. N_t) \) unfolding Pi_def domain_def function_def
  Sigma_def by auto
moreover from \( \langle M = 0 \rangle \) have \( (\forall t \in M. \{ (t,0). t \in M \}) \in N_t \) by auto
ultimately have \( (\exists f : \Pi(M, \lambda t. N_t) \land (\forall t \in M. f_t \in N_t)) \) by auto
}
then have \( (\forall M N. (M \leq 0 \land (\forall t \in M. N_t \neq 0)) \rightarrow (\exists f : \Pi(M, \lambda t. N_t) \land (\forall t \in M. f_t \in N_t))) \)
by auto
then have {the axiom of} \( 0 \) {choice holds} using AxiomCardinalChoiceGen_def
by auto
moreover {
  fix x
  assume as: \( x \in \text{nat} \) {the axiom of} \( x \) {choice holds}
  { 
    fix M N assume ass: \( M \leq \text{succ}(x) \land (\forall t \in M. N_t \neq 0) \)
    { 
      assume \( M \leq x \)
      from as(2) ass(2) have 
      \( (M \leq x \land (\forall t \in M. N_t \neq 0)) \rightarrow (\exists f \in \Pi(M, \lambda t. N_t) \land (\forall t \in M. f_t \in N_t)) \) 
      unfolding AxiomCardinalChoiceGen_def by auto
      with \( \langle M \leq x \rangle \) ass(2) have \( (\exists f \in \Pi(M, \lambda t. N_t) \land (\forall t \in M. f_t \in N_t)) \) 
      by auto
    }
  }
moreover 
  { 
    assume \( M \approx \text{succ}(x) \)
  }
}
then obtain \( f \) where \( f : f \in \text{bij}(\text{succ}(x), M) \) using \( \text{eqpoll_sym} \) \( \text{eqpoll_def} \) by \( \text{blast} \)

moreover
have \( x \in \text{succ}(x) \) unfolding \( \text{succ_def} \) by auto
ultimately have \( \text{restrict}(f, \text{succ}(x) - \{x\}) \in \text{bij}(\text{succ}(x) - \{x\}, M - \{fx\}) \) using \( \text{bij_restrict_rem} \) by auto

moreover
have \( x \not\in x \) unfolding \( \text{succ_def} \) by \( \text{auto} \)
ultimately have \( \text{restrict}(f, x) \in \text{bij}(x - \{x\}, M - \{fx\}) \) by \( \text{auto} \)
then have \( f : f \in \Pi(M - \{fx\}, \lambda t. N t) \) unfolding \( \text{AxiomCardinalChoiceGen_def} \) by \( \text{auto} \)
then obtain \( g \) where \( g : g \in \Pi(M - \{fx\}, \lambda t. N t) \) \( \forall t \in M - \{fx\}. g t \in N t \) by \( \text{auto} \)
from \( f \) have \( ff : fx \in M \) using \( \text{bij_def} \) \( \text{inj_def} \) \( \text{apply_funtype} \) by \( \text{auto} \)
with \( \text{ass(2)} \) have \( N(fx) \neq 0 \) by \( \text{auto} \)
then obtain \( y \) where \( y : y \in N(fx) \) by \( \text{auto} \)
from \( g(1) \) have \( gg : g \subseteq \Sigma(M - \{fx\}, ()(N)) \) unfolding \( \text{Sigma_def} \) by \( \text{auto} \)
with \( y \) \( ff \) have \( g \cup \{\langle fx, y \rangle\} \subseteq \Sigma(M, ()(N)) \) unfolding \( \text{Sigma_def} \) by \( \text{auto} \)
moreover
from \( g(1) \) have \( \text{dom} : M - \{fx\} \subseteq \text{domain}(g) \) unfolding \( \text{Pi_def} \) by auto
then have \( M \subseteq \text{domain}(g \cup \{\langle fx, y \rangle\}) \) unfolding \( \text{domain_def} \) by auto
moreover
from \( gg \) \( g(1) \) have \( \text{noe} : - (\exists t. \langle fx, t \rangle \in g) \) and \( \text{function}(g) \)
unfolding \( \text{domain_def} \) \( \text{Pi_def} \) \( \text{Sigma_def} \) by auto
with \( \text{dom} \) have \( fg : \text{function}(g \cup \{\langle fx, y \rangle\}) \) unfolding \( \text{function_def} \) by \( \text{blast} \)
ultimately have \( PP : g \cup \{\langle fx, y \rangle\} \in \Pi(M, \lambda t. N t) \) unfolding \( \text{Pi_def} \) by \( \text{auto} \)
moreover
have \( \langle fx, y \rangle \in g \cup \{\langle fx, y \rangle\} \) by auto
from this \( fg \) have \( \langle g \cup \{\langle fx, y \rangle\}\rangle(fx) = y \) by \( \text{rule function_apply_equality} \)
with \( y \) have \( \langle g \cup \{\langle fx, y \rangle\}\rangle(fx) \in N(fx) \) by auto
moreover
\{ 
fix \( t \) assume \( A : t \in M - \{fx\} \)
with \( g(1) \) have \( \langle t, gt \rangle \in g \) using \( \text{applyPair} \) by auto
then have \( \langle t, gt \rangle \in (g \cup \{\langle fx, y \rangle\}) \) by auto
then have \( (g \cup \{\langle fx, y \rangle\})(t) = gt \) using \( \text{apply_equality} \) \( PP \) by auto
with \( A \) have \( (g \cup \{\langle fx, y \rangle\})(t) \in Nt \) using \( \text{g(2)} \) by auto
\}
ultimately have $\forall t \in M. (g \cup \{fx, y\})t \in Nt$ by auto

with PP have $\exists g. g \in \Pi(M, \lambda t. Nt) \land (\forall t \in M. gt \in Nt)$ by auto

ultimately have $\exists g. g \in \Pi(M, \lambda t. Nt) \land (\forall t \in M. gt \in Nt)$ using as(1) ass(1)

lepoll_succ_disj by auto

ultimately have $\forall N. M \subseteq \text{succ}(x) \land (\forall t \in M. Nt \neq 0) \longrightarrow (\exists g. g \in \Pi(M, \lambda t. Nt))$

by auto

then have {the axiom of} succ(x) {choice holds}

using AxiomCardinalChoiceGen_def nat_into_Card as(1) nat_succI by auto

ultimately show thesis by (rule nat_induct)

qed

The axiom of choice holds if and only if the AxiomCardinalChoice holds for every couple of a cardinal $Q$ and a set $K$.

lemma choice_subset_imp_choice:
    shows {the axiom of} $Q$ {choice holds} $\iff \forall K. {the axiom of} Q$ {choice holds for subsets} $K$

unfolding AxiomCardinalChoice_def AxiomCardinalChoiceGen_def by blast

A choice axiom for greater cardinality implies one for smaller cardinality

lemma greater_choice_imp_smaller_choice:
    assumes $Q \lesssim Q_1 \quad \text{Card}(Q)$
    shows {the axiom of} $Q_1$ {choice holds} $\longrightarrow {\{the axiom of} Q \ {choice holds\}}$ using assms

AxiomCardinalChoiceGen_def lepoll_trans by auto

If we have a surjective function from a set which is injective to a set of ordinals, then we can find an injection which goes the other way.

lemma surj_fun_inv:
    assumes $f \in \text{surj}(A, B) \quad A \subseteq \text{Ord}(Q)$
    shows $B \subseteq A$

proof-
    let $g = \{\langle m, j \rangle. j \in A \land f(j) = m \}. m \in B$

    have $g : B \rightarrow \text{range}(g)$ using lam_is_fun_range by simp

    then have fun: $g : B \rightarrow g(B)$ using range_image_domain by simp

    from assms(2, 3) have $OA: \forall j \in A. \text{Ord}(j)$ using lt_def Ord_in_Ord by auto

    { fix $x$
      assume $x \in g(B)$

      then have $x \in \text{range}(g)$ and $\exists y \in B. \langle y, x \rangle \in g$ by auto

      then obtain $y$ where $T: x = (\mu j. j \in A \land f(j) = y)$ and $y \in B$ by auto

      with assms(1) $OA$ obtain $z$ where $P: z \in A \land f(z) = y \quad \text{Ord}(z)$ unfolding surj_def

524
by auto
with T have \( x \in A \land f(x) = y \) using LeastI by simp
hence \( x \in A \) by simp
\}
then have \( g(B) \subseteq A \) by auto
with fun have \( \text{fun2: } g:B \rightarrow A \) using fun_weaken_type by auto
then have \( g \in \text{inj}(B,A) \)
proof -
\{
  fix \( w \ x \)
  assume AS: \( gw = gx \) \( w \in B \ x \in B \)
  from assms(1) OA AS(2,3) obtain \( wz \ xz \) where
    \( \text{P1: } wz \in A \land f(wz) = w \ \text{Ord}(wz) \) and
    \( \text{P2: } xz \in A \land f(xz) = x \ \text{Ord}(xz) \)
    unfolding surj_def by blast
  from P1 have \( (\mu j. j \in A \land fj = w) \in A \land f(\mu j. j \in A \land fj = w) = w \)
    by (rule LeastI)
  moreover from P2 have \( (\mu j. j \in A \land fj = x) \in A \land f(\mu j. j \in A \land fj = x) = x \)
    by (rule LeastI)
  ultimately have \( R: \) \( f(\mu j. j \in A \land fj = w) = w \)
  by auto
  from AS have \( (\mu j. j \in A \land f(j) = w) \in A \land f(\mu j. j \in A \land f(j) = w) = w \)
    by auto
  hence \( f(\mu j. j \in A \land f(j) = w) = f(\mu j. j \in A \land f(j) = x) \) by auto
  with \( R(1) \) have \( w = f(\mu j. j \in A \land fj = x) \) by auto
  with \( R(2) \) have \( w = x \) by auto
\}
  hence \( \forall w \in B. \forall x \in B. \ g(w) = g(x) \rightarrow w = x \)
  by auto
  with fun2 show \( g \in \text{inj}(B,A) \) unfolding inj_def by auto
qed
then show thesis unfolding lepoll_def by auto
qed
The difference with the previous result is that in this one \( A \) is not a subset of an ordinal, it is only injective with one.

**Theorem surj_fun_inv_2:**
assumes \( f: \text{surj}(A,B) \ A \sqsubseteq Q \ \text{Ord}(Q) \)
shows \( B \subseteq A \)
proof-
from assms(2) obtain \( h \) where \( \text{h_def: } h \in \text{inj}(A,Q) \) using lepoll_def by auto
then have \( \text{bij: } h \in \text{bij}(A,\text{range}(h)) \) using inj_bij_range by auto
then obtain \( h1 \) where \( \text{h1_def: } h1 \in \text{bij}(\text{range}(h),A) \) using bij_converse_bij by auto
then have \( h1 \in \text{surj}(\text{range}(h),A) \) using bij_def by auto
with assms(1) have \( (f \circ h1) \subseteq \text{surj}(\text{range}(h),B) \) using comp_surj by auto
moreover
\{

fix \( x \)
assume \( p: x \in \text{range}(h) \)
from bij have \( h \in \text{surj}(A, \text{range}(h)) \) using bij_def by auto
with \( p \) obtain \( q \) where \( q \in A \) and \( h(q) = x \) using surj_def by auto
then have \( x \in \text{Q} \) using h_def inj_def by auto

begin

This theory file deals with normal subgroup test and some finite group theory. Then we define group homomorphisms and prove that the set of endomorphisms forms a ring with unity and we also prove the first isomorphism theorem.

45.1 Conjugation of subgroups

First we show some properties of conjugation

The conjugate of a subgroup is a subgroup.

**theorem** (in group0) **conj_group_is_group**:  
assumes \( \text{IsAsubgroup}(H, P) \) \( g \in G \)  
shows \( \text{IsAsubgroup}(\{g \cdot (h \cdot g^{-1}) \cdot h \mid h \in H\}, P) \)
**proof**-  
have \( \text{sub}: \text{H} \subseteq G \) using assms(1) group0_3_L2 by auto  
from assms(2) have \( g^{-1} \in G \) using inverse_in_group by auto
where \( \{g \cdot (h \cdot g^{-1}) \cdot h \mid h \in H\} \) by auto  
from \( h(1) \) have \( h^{-1} \in H \) using group0_3_T3A assms(1) by auto
from \( h(1) \) sub have \( h \in G \) by auto
then have \( h^{-1} \in G \) using inverse_in_group by auto
with \( \langle g^{-1} \in G >\) have \( (h^{-1}) \cdot (g^{-1}) \in G \) using group_op_closed by auto
from \( h(2) \) have \( r^{-1} = (g \cdot (h \cdot g^{-1}))^{-1} \) by auto moreover
from \( h \in G \) \( <g^{-1} \in G >\) have \( s: \langle g^{-1} \rangle \in G \) using group_op_closed by blast
ultimately have \( r^{-1} = (h \cdot (g^{-1}))^{-1} \cdot (g^{-1}) \) using group_inv_of_two assms(2)
by auto

end

45 Groups 4

theory Group_ZF_4 imports Group_ZF_1 Group_ZF_2 Finite_ZF Cardinal_ZF

begin

45.1 Conjugation of subgroups

First we show some properties of conjugation

The conjugate of a subgroup is a subgroup.
moreover
  from \( h \in G \) \((g^{-1})^{-1}g\) have \((h \cdot (g^{-1}))^{-1} = (g^{-1})^{-1}h^{-1}\) using group_inv_of_two
by auto
  moreover have \((g^{-1})^{-1}g\) using group_inv_of_inv assms(2) by auto
  ultimately have \(r^{-1} = (g \cdot (h^{-1})) \cdot (g^{-1})\) by auto
with \( h^{-1} \in G \) \(g^{-1} \in G\) have \(r^{-1} = g \cdot (h^{-1}) \cdot (g)^{-1}\) using group_oper_assoc
assms(2) by auto
  moreover from \( s \) assms(2) \( h \in G\) using group_op_closed by auto
  moreover have \( h^{-1} \in H \) ultimately have \(r^{-1} \in \{g \cdot h \cdot g^{-1}\}, h \in H\) \(r \in G\) by auto
}
then have \( \forall r \in \{g \cdot h \cdot g^{-1}\}, h \in H\). \(r^{-1} \in \{g \cdot h \cdot g^{-1}\}, h \in H\) and \(\{g \cdot h \cdot g^{-1}\}\)
\(h \in H\) \(\subseteq G\) by auto moreover
\{ 
  fix \( s \) \( t \) assume \( s \in \{g \cdot h \cdot g^{-1}\}, h \in H\) and \( t \in \{g \cdot h \cdot g^{-1}\}, h \in H\)
then obtain \( h_1 \) \( t_1 \) where \( h_1 \cdot s = g \cdot (h_1) \cdot (g^{-1})\) and \( t_1 \cdot t = g \cdot (t_1) \cdot (g^{-1})\)
by auto
from \( h(1) \) have \( h \in G\) using sub by auto
then have \( g \cdot h \cdot g^{-1}\) using group_op_closed assms(2) by auto
then have \((g \cdot h) \cdot g^{-1}\) using inverse_in_group by auto
from \( h(1) \) have \( h_1 \in G\) using sub by auto
with \( (g^{-1}) \cdot G\) have \( h_1 \cdot (g^{-1}) \in G\) using group_op_closed by auto
from \( h(2) \) \( h(2) \) have \( s \cdot t = g \cdot (h_1) \cdot (g^{-1})\) by auto
moreover from \( h(2) \) \( h(2) \) have \( h_1 \cdot h_1 \cdot g \cdot h_1 \cdot g^{-1} \) using group0_2_L2 by auto
then have \( h_1 \cdot h_1 \cdot g \cdot h_1 \cdot g^{-1} \) using group0_2_L6 assms(2) by auto
then have \( g \cdot (h_1) \cdot g \cdot h_1 \cdot g^{-1} \) using group_oper_assoc
assms(2) inverse_in_group by auto
with \( g \cdot h \cdot g^{-1}\) \( h \in G\) have \( g \cdot (h_1) \cdot g \cdot h_1 \cdot g^{-1} \) using group_oper_assoc
assms(2) inverse_in_group group_op_closed by auto
with \( g \cdot h \cdot g^{-1}\) \( h \in G\) have \( g \cdot (h_1) \cdot g \cdot h_1 \cdot g^{-1} \) using group_oper_assoc
assms(2) inverse_in_group group_op_closed by auto
then have \( g \cdot (h_1) \cdot g \cdot h_1 \cdot g^{-1} \) = \( g \cdot (h_1) \cdot g \cdot h_1 \cdot g^{-1} \) by auto
with \( g \cdot h \cdot g^{-1}\) \( h \in G\) have \( g \cdot (h_1) \cdot g \cdot h_1 \cdot g^{-1} \) using group_oper_assoc
inverses_in_group assms(2) group_op_closed by auto
with \( g \cdot h \cdot g^{-1}\) \( h \in G\) have \((h_1) \cdot g \cdot h_1 \cdot g^{-1}\) = \( g \cdot (h_1) \cdot g \cdot h_1 \cdot g^{-1}\) using group_oper_assoc
inverse_in_group assms(2) group_op_closed by auto
ultimately have \( s \cdot t = g \cdot (h_1) \cdot (g^{-1})\) by auto moreover
from \( h(1) \) \( h(1) \) have \( h_1 \cdot h_1 \cdot h_1 \cdot h_1 \) using assms(1) group0_3_L6 by auto
ultimately have \( s \cdot t \in \{g \cdot h \cdot g^{-1}\}, h \in H\) by auto
} 
then have \( \{g \cdot (h \cdot g^{-1})\}, h \in H\) \(\) is closed under \(P\) unfolding IsOpClosed_def
by auto moreover
from assms(1) have \( 1 \in H\) using group0_3_L5 by auto
then have \( g \cdot (1 \cdot g^{-1}) \in \{g \cdot (h \cdot g^{-1})\}, h \in H\) by auto
then have \( \{g \cdot (h \cdot g^{-1})\}, h \in H\} \neq \emptyset\) by auto ultimately
527
show thesis using group0_3_T3 by auto
qed

Every set is equipollent with its conjugates.

theorem (in group0) conj_set_is_eqpoll:
  assumes H⊆G g∈G
  shows H={g.(h.g⁻¹). h∈H}
proof-
  have fun:{⟨h,g.(h.g⁻¹)⟩. h∈H}:H→{g.(h.g⁻¹). h∈H} unfolding Pi_def function_def
  domain_def by auto
  { fix h1 h2 assume h1∈H. h2∈H.| h1={h.g.(h.g⁻¹)}. h∈H} h2
  with fun have g.(h1.g⁻¹)=g.(h2.g⁻¹) h1.g⁻¹∈G h2.g⁻¹∈G using apply_equality
  assms(1)
  group_op_closed inverse_in_group assms(2) by auto
  then have h1.g⁻¹=h2.g⁻¹ using group0_2_L19(2) assms(2) by auto
  with ⟨h1∈H⟩ ⟨h2∈G⟩ have h1=h2 using group0_2_L19(1) inverse_in_group
  assms(2) by auto }
  then have ∀h1∈H. ∀h2∈H. {⟨h,g.(h.g⁻¹)⟩. h∈H} h1={⟨h,g.(h.g⁻¹)⟩. h∈H} h2
  → h1=h2 by auto
  with fun have {⟨h,g.(h.g⁻¹)⟩. h∈H} inj(H,{g.(h.g⁻¹). h∈H}) unfolding
  inj_def by auto moreover
  { fix ggh assume ggh∈{g.(h.g⁻¹). h∈H}
    then obtain h where h∈H ggh=g.(h.g⁻¹) by auto
    then have ⟨h,ggh⟩∈{⟨h,g.(h.g⁻¹)⟩. h∈H} by auto
    then have ⟨h,g.g(g⁻¹)⟩. h∈H g=ggh using apply_equality fun by auto
    with ⟨h∈H⟩ have ∃h∈H. {⟨h,g.(h.g⁻¹)⟩. h∈H} h=ggh by auto
  }
  with fun have {⟨h,g.(h.g⁻¹)⟩. h∈H} surj(H,{g.(h.g⁻¹) h∈H}) unfolding
  surj_def by auto
  ultimately have {⟨h,g.(h.g⁻¹)⟩. h∈H} bij(H,{g.(h.g⁻¹). h∈H}) unfolding
  bij_def by auto
  then show thesis unfolding eqpoll_def by auto
qed

Every normal subgroup contains its conjugate subgroups.

theorem (in group0) norm_group_cont_conj:
  assumes IsAnormalSubgroup(G,P,H) g∈G
  shows {g.(h.g⁻¹). h∈H}⊆H
proof-
  { fix r assume r∈{g.(h.g⁻¹). h∈H}
    then obtain h where h:r=g.(h.g⁻¹) h∈H by auto moreover
    from h(2) have h∈G using group0_3_L2 assms(1) unfolding IsAnormalSubgroup_def
    by auto moreover
    from assms(2) have g⁻¹∈G using inverse_in_group by auto
    ultimately have r=g.h.g⁻¹ h∈H using group_op_assoc assms(2) by auto

528
then have \( r \in H \) using assms unfolding IsAnormalSubgroup_def by auto

then show \( \{g \cdot (h \cdot g^{-1}) \cdot h \} \subseteq H \) by auto

qed

If a subgroup contains all its conjugate subgroups, then it is normal.

**theorem** (in group0) cont_conj_is_normal:
  assumes IsAsubgroup(H,P) \( \forall g \in G. \{g \cdot (h \cdot g^{-1}) \cdot h \} \subseteq H \)
  shows IsAnormalSubgroup(G,P,H)
proof-
  { fix h g assume h \in H g \in G
    with assms(2) have \( g \cdot (h \cdot g^{-1}) \in H \) by auto
    moreover from \( g \in G \) have \( h \in G \) \in G \in G \in G \in G \) using group0_3_L2 assms(1)
    inverse_in_group by auto
    ultimately have \( g \cdot (h \cdot g^{-1}) \in H \) using group_oper_assoc by auto
  }
  then show thesis using assms(1) unfolding IsAnormalSubgroup_def by auto

qed

If a group has only one subgroup of a given order, then this subgroup is normal.

**corollary** (in group0) only_one_equipoll_sub:
  assumes IsAsubgroup(H,P) \( \forall M. \) IsAsubgroup(M,P) \( \land H \approx M \longrightarrow M=H \)
  shows IsAnormalSubgroup(G,P,H)
proof-
  { fix g assume g: \in G
    with assms(1) have IsAsubgroup(\{g \cdot (h \cdot g^{-1}) \cdot h \},P) using conj_group_is_group by auto
    moreover from assms(1) g have \( H=\{g \cdot (h \cdot g^{-1}) \cdot h \} \) using conj_set_is_eqpoll group0_3_L2 by auto
    ultimately have \( g \cdot (h \cdot g^{-1}) \cdot h \subseteq H \) by auto
    then have \( g \cdot (h \cdot g^{-1}) \cdot h \subseteq H \) by auto
  }
  then show thesis using cont_conj_is_normal assms(1) by auto

qed

The trivial subgroup is then a normal subgroup.

**corollary** (in group0) trivial_normal_subgroup:
  shows IsAnormalSubgroup(G,P,\{1\})
proof-
  have \( \{1\} \subseteq G \) using group0_2_L2 by auto
  moreover have \( \{1\} \neq \emptyset \) by auto
  moreover
  { fix a b assume a\in\{1\} b\in\{1\}
    then have a=1b=1 by auto

529
then have \( P(a,b)=1\cdot1 \) by auto
then have \( P(a,b)=1 \) using group0_2_L2 by auto
then have \( P(a,b)\in\{1\} \) by auto

\(
\text{then have } \{1\}\{\text{is closed under}\} P \text{ unfolding IsOpClosed_def by auto }
\)
moreover
\(
\text{fix } a \text{ assume } a\in\{1\}
\text{ then have } a=1 \text{ by auto }
\text{ then have } a^{-1}=1^{-1} \text{ by auto }
\text{ then have } a^{-1}=1 \text{ using group_inv_of_one by auto }
\text{ then have } a^{-1}\in\{1\} \text{ by auto }
\)
then have \( \forall a\in\{1\}. a^{-1}\in\{1\} \) by auto ultimately
have \( \text{IsAsubgroup(}\{1\},P) \) using group0_3_T3 by auto
moreover
\(
\text{fix } M \text{ assume } M:\text{IsAsubgroup(M,P) } \{1\}\approx M
\text{ then have one: } 1\in M M\approx\{1\} \text{ using eqpoll_sym group0_3_L5 by auto }
\text{ then obtain } f \text{ where } f\in\text{bij}(M,\{1\}) \text{ unfolding eqpoll_def by auto }
\text{ then have inj:f\in\text{inj}(M,\{1\}) \text{ unfolding bij_def by auto }
\text{ then have fun:f:M\rightarrow\{1\} \text{ unfolding inj_def by auto }
\}
\)
fix b assume b:b\in M b\neq 1
with \( <1\in M > \) have \( f b\neq f 1 \text{ using inj unfolding inj_def by auto }
moreover from fun b(1) have \( f b\in\{1\} \) by (rule apply_type)
moreover from fun one(1) have \( f 1\in\{1\} \) by (rule apply_type)
ultimately have False by auto

\}
with \( <1\in M > \) have \( M\approx\{1\} \) by auto

ultimately show thesis using only_one_eqpoll_sub by auto qed

The whole group is normal as a subgroup

\textbf{lemma (in group0)} whole_normal_subgroup:
shows \( \text{IsANormalSubgroup(G,P,G)} \)
\textbf{proof-}
\text{have } G\subseteq G \text{ by auto moreover }
\text{have } \forall x\in G. \ x^{-1}\in G \text{ using inverse_in_group by auto moreover }
\text{have } G\neq 0 \text{ using group0_2_L2 by auto moreover }
\text{have } G\{\text{is closed under}\} P \text{ using group_op_closed }
\text{ unfolding IsOpClosed_def by auto ultimately }
\text{have } \text{IsAsubgroup(G,P) using group0_3_T3 by auto moreover }
\}
fix n g assume ng:n\in G g\in G
then have \( P\langle P\langle g, n\rangle, \text{GroupInv}(G, P) \ g\rangle \in G \)
using group_op_closed inverse_in_group by auto

ultimately show thesis unfolding IsAnormalSubgroup_def by auto
qed

45.2 Simple groups

In this subsection we study the groups that build the rest of the groups: the
simple groups.

Since the whole group and the trivial subgroup are always normal, it is
natural to define simplicity of groups in the following way:
definition
IsSimple (\{\_\_\}, \{is a simple group\} 89)
where [G,f] \{is a simple group\} ≡ IsAgroup(G,f) \land (\forall M. IsAnormalSubgroup(G,f,M)
\rightarrow M=G\lor M=\{TheNeutralElement(G,f)\})

From the definition follows that if a group has no subgroups, then it is
simple.
corollary (in group0) noSubgroup_imp_simple:
assumes \( \forall H. \text{IsAsubgroup}(H,P) \rightarrow H=G\lor H=\{1\} \)
shows [G,P] \{is a simple group\}
proof-
\begin{align*}
& \text{have IsAgroup}(G,P) \text{ using groupAssum by auto moreover} \\
& \quad \text{fix } M \text{ assume IsAnormalSubgroup}(G,P,M) \\
& \quad \text{then have IsAsubgroup}(M,P) \text{ unfolding IsAnormalSubgroup_def by auto} \\
& \quad \quad \text{with assms have } M=G\lor M=\{1\} \text{ by auto} \\
& \quad \text{ultimately show thesis unfolding IsSimple_def by auto} \\
& \quad \text{qed}
\end{align*}

We add a context for an abelian group
locale abelian_group = group0 +
\begin{align*}
& \text{assumes isAbelian: } P \{\text{is commutative on} \} G
\end{align*}

Since every subgroup is normal in abelian groups, it follows that commuta-
tive simple groups do not have subgroups.
corollary (in abelian_group) abelian_simple_noSubgroups:
\begin{align*}
& \text{assumes } [G,P] \{\text{is a simple group}\} \\
& \text{shows } \forall H. \text{IsAsubgroup}(H,P) \rightarrow H=G\lor H=\{1\}
\end{align*}
proof-
\begin{align*}
& \text{fix } H \text{ assume } A: \text{IsAsubgroup}(H,P) H \neq \{1\} \\
& \quad \text{then have IsAnormalSubgroup}(G,P,H) \text{ using } Group_ZF_2_4_L6(1) \text{ groupAssum isAbelian} \\
& \quad \quad \text{by auto} \\
& \quad \quad \text{with assms(1) A have } H=G \text{ unfolding IsSimple_def by auto} \\
& \quad \text{then show thesis by auto} \\
& \quad \text{qed}
\end{align*}
45.3 Finite groups

This subsection deals with finite groups and their structure.

The subgroup of a finite group is finite.

**Lemma (in group0) finite_subgroup:**

**Assumes**

Finite(G) IsAsubgroup(H,P)

**Shows**

Finite(H)

**Using**

group0_3_L2 subset_Finite assms by force

The space of cosets is also finite. In particular, quotient groups.

**Lemma (in group0) finite_cosets:**

**Assumes**

Finite(G) IsAsubgroup(H,P)

**Defines**

r ≡ QuotientGroupRel(G,P,H)

**Shows**

Finite(G//r)

**Proof**

- have fun:{(g,r{g}). g∈G}:G→(G//r) unfolding Pi_def function_def domain_def by auto

  { fix C assume C:C∈G//r
    have equiv:equiv(G,r) using Group_ZF_2_4_L3 assms(2) unfolding r_def by auto
    then have refl1(G,r) unfolding equiv_def by auto
    with C have C≠0 using EquivClass_1_L5 by auto
    then obtain c where c:c∈C by auto
    with C have r{c}=C using EquivClass_1_L2 equiv by auto
    with c C have (c,C)∈{(g,r{g}). g∈G} using EquivClass_1_L1 equiv by auto
    then have ∃c∈G. {(g,r{g}). g∈G}c=C by auto
  }

  with fun have surj:{(g,r{g}). g∈G}∈surj(G,G//r) unfolding surj_def by auto

  from assms(1) obtain n where n∈nat G≈n unfolding Finite_def by auto
  then have G:G≤n Ord(n) using eqpoll_imp_lepoll by auto
  then have G//r≤n using surj_fun_inv_2 surj by auto
  with G(1) have G//r≤n using lepoll_trans by blast

All the cosets are equipollent.

**Lemma (in group0) cosets_equipoll:**

**Assumes**

IsAsubgroup(H,P) g1∈Gg2∈G

**Defines**

r ≡ QuotientGroupRel(G,P,H)

**Shows**

r{g1} ≈ r{g2}

**Proof**

have equiv:equiv(G,r) using Group_ZF_2_4_L3 assms(1) unfolding r_def by auto
from assms(3,2) have \( GG: (g_1^{-1}) \cdot g_2 \in G \) using inverse_in_group group_op_closed by auto 
then have bij: \( \text{RightTranslation}(G, P, (g_1^{-1}) \cdot g_2) \in \text{bij}(G, G) \) using trans_bij(1) by auto 
have \( r(g_2) \subseteq G // r \) unfolding quotient_def by auto 
then have sub2: \( r(g_2) \subseteq G \) using EquivClass_1_L1 equiv assms(3) by blast 
have \( r(g_1) \subseteq G // r \) unfolding quotient_def by auto 
then have sub: \( r(g_1) \subseteq G \) using EquivClass_1_L1 equiv assms(2) by blast 
with bij have restrict(\( \text{RightTranslation}(G, P, (g_1^{-1}) \cdot g_2) \), \( r(g_1) \)) \( \subseteq \text{bij}(r(g_1), \text{RightTranslation}(G, P, (g_1^{-1}) \cdot g_2)) \) using restrict_bij unfolding bij_def by auto 
then have \( r(g_1) \approx \text{RightTranslation}(G, P, (g_1^{-1}) \cdot g_2)(r(g_1)) \) unfolding eqpoll_def by auto 
with \( GG \) sub have \( A_0: r(g_1) = \{ \text{RightTranslation}(G, P, (g_1^{-1}) \cdot g_2) t. t \in r(g_1) \} \) using func_image_def group0_5_L1(1) by force 
\begin{align*}
\{ \text{fix t assume } & t \in r(g_1) \} \\
\text{then obtain } q & \text{ where } q: t = \text{RightTranslation}(G, P, (g_1^{-1}) \cdot g_2) q \quad q \in r(g_1) \\
& \text{by auto} \\
\text{then have } & (g_1, q) \in r q \in G \text{ using image_iff sub by auto} \\
\text{then have } & g_1 \cdot (q^{-1}) \in H \quad q^{-1} \in G \text{ using inverse_in_group unfolding r_def QuotientGroupRel_def by auto} \\
& \text{from } G \quad \text{q sub have } t: t = q \cdot ((g_1^{-1}) \cdot g_2) \text{ using group0_5_L2(1) by auto} \\
& \text{then have } g_2 \cdot t^{-1} = g_2 \cdot (q \cdot ((g_1^{-1}) \cdot g_2))^{-1} \text{ by auto} \\
& \text{with } \langle q \in G \rangle \text{ \( G \) have } g_2 \cdot t^{-1} = g_2 \cdot ((g_1^{-1}) \cdot g_2)^{-1} \cdot q^{-1} \text{ using group_inv_of_two by auto} \\
& \text{by auto} \\
& \text{then have } g_2 \cdot t^{-1} = g_2 \cdot ((g_2^{-1}) \cdot (g_1^{-1}) \cdot q^{-1}) \text{ using group_inv_of_two inverse_in_group assms(2)} \\
& \text{assms(3) by auto} \\
& \text{then have } g_2 \cdot t^{-1} = g_2 \cdot ((g_2^{-1}) \cdot g_1 \cdot q^{-1}) \text{ using group_inv_of_inv assms(2)} \\
\text{by auto moreover} \\
& \text{have } (g_2^{-1}) \cdot g_1 \in G \text{ using assms(2) inverse_in_group assms(3) group_op_closed by auto} \\
& \text{by auto} \\
& \text{with assms(3) } q^{-1} \in G \text{ have } g_2 \cdot ((g_2^{-1}) \cdot g_1 \cdot q^{-1}) = g_2 \cdot ((g_2^{-1}) \cdot g_1) \cdot q^{-1} \text{ using group_oper_assoc by auto} \\
& \text{moreover have } g_2 \cdot ((g_2^{-1}) \cdot g_1) = g_2 \cdot (g_2^{-1}) \cdot g_1 \text{ using assms(2) inverse_in_group assms(3)} \\
& \text{group_oper_assoc by auto} \\
& \text{then have } g_2 \cdot ((g_2^{-1}) \cdot g_1) = g_1 \text{ using group0_2_L6 assms(3) group0_2_L2} \\
\text{assms(2) by auto ultimately} \\
& \text{have } g_2 \cdot t^{-1} = g_1 \cdot q^{-1} \text{ by auto} \\
& \text{with } \langle g_1 \cdot q^{-1} \in H \rangle \text{ have } g_2 \cdot t^{-1} \in H \text{ by auto moreover} \\
& \text{from } t \langle q \in G \rangle \langle q \in G \rangle \text{ have } t \in G \text{ using inverse_in_group assms(2) group_op_closed by auto} \\
& \text{ultimately have } (g_2, t) \in r \text{ unfolding QuotientGroupRel_def r_def using assms(3) by auto} \\
& \text{then have } t \in r(g_2) \text{ using image_iff assms(4) by auto} \\
\end{align*}
then have A1: \( \{ \text{RightTranslation}(G, P, (g_1^{-1}) \cdot g_2) t. t \in r(g_1) \} \subseteq r(g_2) \) by auto
\[
\text{fix t assume } t \in r(g2) \\
\text{then have } (g2, t) \in r \text{ using } \text{sub2 image_iFF by auto} \\
\text{then have } H : g2 : t^{-1} \in H \text{ unfolding } \text{QuotientGroupRel_def r_def by auto} \\
\text{then have } G : g2 : t^{-1} \in G \text{ using } \text{group0_3_L2 assms(1) by auto} \\
\text{then have } g1 : (g1^{-1}, (g2 : t^{-1})) = g1 : g1^{-1} : (g2 : t^{-1}) \text{ using } \text{group_oper_assoc} \\
\text{assms(2) inverse_in_group by auto} \\
\text{with } G \text{ have } g1 : (g1^{-1}, (g2 : t^{-1})) = g2 : t^{-1} \text{ using } \text{group0_2_L6 assms(2) group0_2_L2} \\
\text{by auto} \\
\text{with } H \text{ have } HH : g1 : (g1^{-1}, (g2 : t^{-1})) \in H \text{ by auto} \\
\text{from } \langle t \in G \rangle \text{ have } GGG : t : g2^{-1} \in G \text{ using } \text{inverse_in_group assms(3) group_op_closed} \\
\text{by auto} \\
\text{from } \langle t \in G \rangle \text{ have } (t : g2^{-1})^{-1} = g2^{-1} : t^{-1} \text{ using } \text{group_inv_of_two inverse_in_group} \\
\text{assms(3) by auto} \\
\text{also have } \ldots = g2 : t^{-1} \text{ using } \text{group_inv_of_inv assms(3) by auto} \\
\text{finally have } (t : g2^{-1})^{-1} = g2 : t^{-1} \text{ by auto} \\
\text{then have } g1^{-1}, (t : g2^{-1})^{-1} = g1^{-1}, (g2 : t^{-1}) \text{ by auto} \\
\text{then have } ((t : g2^{-1})^{-1} g1^{-1} = g1^{-1}, (g2 : t^{-1})) \text{ using } \text{group_inv_of_two GGG} \\
\text{assms(2) by auto} \\
\text{then have } HHH : g1, ((t : g2^{-1})^{-1} g1^{-1}) \in H \text{ using } \text{HH by auto} \\
\text{from } \langle t \in G \rangle \text{ have } (t : g2^{-1}) : g1 \in G \text{ using } \text{assms(2) inverse_in_group assms(3)} \\
\text{group_op_closed by auto} \\
\text{with } HHH \text{ have } (g1, (t : g2^{-1}) : g1) \in r \text{ using } \text{assms(2) unfolding } \text{r_def QuotientGroupRel_def} \\
\text{by auto} \\
\text{then have } r_g1 : t : g2^{-1}, g1 \in r(g1) \text{ using } \text{image_iFF by auto} \\
\text{from } \text{assms(3) have } g2^{-1} : G \text{ using } \text{inverse_in_group by auto} \\
\text{from } \langle t \in G \rangle \text{ have } t : g2^{-1}, g1 : ((g1^{-1}) : g2) = t : (g2^{-1} : g1 : ((g1^{-1}) : g2)) \text{ using } \text{group_oper_assoc} \\
\text{inverse_in_group assms(3) assms(2) by auto} \\
\text{also from } \langle t \in G \rangle \text{ have } \ldots = t : ((g2^{-1} : g1) : ((g1^{-1} : g2))) \text{ using } \text{group_oper_assoc} \\
\text{assms(3) by auto} \\
\text{also from } G : g2^{-1} : G \text{ have } \ldots = t : ((g2^{-1} : (g1^{-1} : g2))) \text{ using } \text{group_oper_assoc} \\
\text{assms(2) by auto} \\
\text{also have } \ldots = t : (g2^{-1} : (g1^{-1} : g2)) \text{ using } \text{group_oper_assoc assms(2)} \\
\text{inverse_in_group assms(3) by auto} \\
\text{also from } \langle t \in G \rangle \text{ have } \ldots = t \text{ using } \text{group0_2_L6 assms(3) group0_2_L6} \\
\text{assms(2) group0_2_L2 assms(3) by auto} \\
\text{finally have } t : g2^{-1}, g1 : ((g1^{-1}) : g2) = t \text{ by auto} \\
\text{with } \langle t : g2^{-1} \rangle : g1 \in G \text{ have } \text{RightTranslation}(G, P, ((g1^{-1}) : g2))(t : g2^{-1}, g1) = t \\
\text{using } \text{group0_5_L2(1) by auto} \\
\text{then have } t \in \{ \text{RightTranslation}(G, P, ((g1^{-1}) : g2)) t \in r(g1) \} \text{ using } \text{rg1} \\
\text{by force} \\
\text{then have } r(g2) \subseteq \{ \text{RightTranslation}(G, P, ((g1^{-1}) : g2)) t \in r(g1) \} \text{ by blast} \\
\text{with } A1 \text{ have } r(g2) = \{ \text{RightTranslation}(G, P, ((g1^{-1}) : g2)) t \in r(g1) \} \text{ by auto} \\
\text{with } A0 \text{ show thesis by auto} \\
\text{qed} 
\]

The order of a subgroup multiplied by the order of the space of cosets is the...
order of the group. We only prove the theorem for finite groups.

**Theorem (in group0) Lagrange:**

- **Assumes:** \( \text{Finite}(G) \) \( \text{IsAsubgroup}(H,P) \)
- **Defines:** \( r \equiv \text{QuotientGroupRel}(G,P,H) \)
- **Shows:** \(|G| = |H| \#* |G//r|\)

**Proof**

- Have \( \text{equiv}(G,r) \) using \( \text{Group}_2\_4\_L3 \) \( \text{assms}(2) \) unfolding \( r \_\text{def} \)
  - Have \( r\{1\} \subseteq G \) unfolding \( r \_\text{def} \) \( \text{QuotientGroupRel}_\text{def} \) by auto
  - Have \( \forall aa \in G. \ aa \in H \iff (aa,1) \in r \) using \( \text{Group}_2\_4\_LS\_C \) unfolding \( r \_\text{def} \)
    - Then have \( \forall aa \in G. \ aa \in H \iff (1,aa) \in r \) using equiv unfolding \( \text{sym}\_\text{def} \) \( \text{equiv}\_\text{def} \) by auto
    - Then have \( \forall aa \in G. \ aa \in H \iff aa \in r\{1\} \) using \( \text{image}\_\text{iff} \) by auto
      - With \( <r\{1\}> \) have \( H:H=r\{1\} \) using \( \text{group0}_3\_L2 \) \( \text{assms}(2) \) by blast
        - Fix \( c \) assume \( c \in (G//r) \)
          - Then obtain \( g \) where \( g \in G \ c=r\{g\} \) unfolding \( \text{quotient}\_\text{def} \) by auto
            - Then have \( c \equiv r\{1\} \) using \( \text{cosets}\_\text{equipoll} \) \( \text{assms}(2) \) \( \text{group0}_2\_L2 \) unfolding \( r \_\text{def} \) by auto
              - Then have \( |c|=|H| \) using \( H \) \( \text{cardinal}\_\text{cong} \) by auto
            - Then have \( \forall c \in (G//r). \ |c|=|H| \) by auto moreover
              - Have \( \text{Finite}(G//r) \) using \( \text{assms} \) \( \text{finite}\_\text{cosets} \) by auto moreover
              - Have \( \bigcup (G//r)=G \) using \( \text{Union}\_\text{quotient} \) \( \text{Group}_2\_4\_L3 \) \( \text{assms}(2,3) \) by auto moreover
                - From \( \langle \bigcup (G//r)=G \rangle \) have \( \text{Finite} \langle \bigcup (G//r) \rangle \) using \( \text{assms}(1) \) by auto moreover
                  - Have \( \forall c1 \in (G//r). \ \forall c2 \in (G//r). \ c1 \neq c2 \implies c1 \cap c2 = 0 \) using \( \text{quotient}\_\text{disj} \)
                    - Equiv by blast ultimately
                        - Show thesis using \( \text{card}\_\text{partition} \) by auto
            qed

45.4 Subgroups generated by sets

In this section we study the minimal subgroup containing a set

Since \( G \) is always a group containing the set, we may take the intersection of all subgroups bigger than the set; and hence the result is the subgroup we searched.

**Definition (in group0)**

- **SubgroupGenerated (\( _\_ G \) 80)
  - Where \( X \subseteq G \implies \langle X \rangle_G \equiv \bigcap \{H \in \text{Pow}(G). \ X \subseteq H \land \text{IsAsubgroup}(H,P)\}\)

Every generated subgroup is a subgroup

**Theorem (in group0) subgroupGen_is_subgroup:**

- Assumes \( X \subseteq G \)
shows $\text{IsAsubgroup}(\langle X \rangle_G, P)$
proof-
  have $\text{restrict}(P, G \times G) = P$ using group_oper_fun restrict_idem unfolding Pi_def by auto
  then have $\text{IsAsubgroup}(G, P)$ unfolding IsAsubgroup_def using groupAssum by auto
  with assms have $G \in \{ H \in \text{Pow}(G). X \subseteq H \land \text{IsAsubgroup}(H, P) \}$ by auto
  then have $\{ H \in \text{Pow}(G). X \subseteq H \land \text{IsAsubgroup}(H, P) \} \neq 0$ by auto
  then show thesis using subgroup_inter SubgroupGenerated_def assms by auto
qed

The generated subgroup contains the original set

**Theorem** (in group0) subgroupGen_contains_set:
  assumes $X \subseteq G$
  shows $X \subseteq \langle X \rangle_G$
proof-
  have $\text{restrict}(P, G \times G) = P$ using group_oper_fun restrict_idem unfolding Pi_def by auto
  then have $\text{IsAsubgroup}(G, P)$ unfolding IsAsubgroup_def using groupAssum by auto
  with assms have $G \in \{ H \in \text{Pow}(G). X \subseteq H \land \text{IsAsubgroup}(H, P) \}$ by auto
  then have $\{ H \in \text{Pow}(G). X \subseteq H \land \text{IsAsubgroup}(H, P) \} \neq 0$ by auto
  then show thesis using subgroup_inter SubgroupGenerated_def assms by auto
qed

Given a subgroup that contains a set, the generated subgroup from that set is smaller than this subgroup

**Theorem** (in group0) subgroupGen_minimal:
  assumes $\text{IsAsubgroup}(H, P) \land X \subseteq H$
  shows $\langle X \rangle_G \subseteq H$
proof-
  from assms have $\text{sub}: X \subseteq G$ using group0_3_L2 by auto
  from assms have $H \in \{ H \in \text{Pow}(G). X \subseteq H \land \text{IsAsubgroup}(H, P) \}$ using group0_3_L2 by auto
  then show thesis using sub subgroup_inter SubgroupGenerated_def assms by auto
qed

46 Groups 5

theory Group_ZF_5 imports Group_ZF_4 Ring_ZF Semigroup_ZF

In this theory we study group homomorphisms.
46.1 Homomorphisms

A homomorphism is a function between groups that preserves the group operations.

In general we may have a homomorphism not only between groups, but also between various algebraic structures with one operation like magmas, semigroups, quasigroups, loops and monoids. In all cases the homomorphism is defined by using the morphism property. In the multiplicative notation we will write that \( f \) has a morphism property if \( f(x \cdot_G y) = f(x) \cdot_H f(y) \) for all \( x, y \in G \). Below we write this definition in raw set theory notation and use the expression \texttt{IsMorphism} instead of the possible, but longer \texttt{HasMorphismProperty}.

**Definition**

\[
\text{IsMorphism}(G, P, F, f) \equiv \forall g_1 \in G. \forall g_2 \in G. f(P(g_1, g_2)) = F(f(g_1), f(g_2))
\]

A function \( f : G \to H \) between algebraic structures \((G, \cdot_G)\) and \((H, \cdot_H)\) with one operation (each) is a homomorphism if it has the morphism property.

**Definition**

\[
\text{Homomor}(f, G, P, H, F) \equiv f : G \to H \land \text{IsMorphism}(G, P, F, f)
\]

Now a lemma about the definition:

**Lemma** \texttt{homomor_eq}:

\[
\text{assumes } \text{Homomor}(f, G, P, H, F) \quad \text{g} \in G \quad \text{g} \in G
\]

\[
\text{shows } f(P(g_1, g_2)) = F(f(g_1), f(g_2))
\]

\[
\text{using } \text{assms } \text{unfolding } \text{Homomor_def } \text{IsMorphism_def by auto}
\]

An endomorphism is a homomorphism from a group to the same group. In case the group is abelian, it has a nice structure.

**Definition**

\[
\text{End}(G, P) \equiv \{ f \in G \to G. \quad \text{Homomor}(f, G, P, G, P) \}
\]

The defining property of an endomorphism written in notation used in \texttt{group0} context:

**Lemma** \texttt{endomor_eq}:

\[
\text{assumes } f \in \text{End}(G, P) \quad g_1 \in G \quad g_2 \in G
\]

\[
\text{shows } f(g_1 \cdot g_2) = f(g_1) \cdot f(g_2)
\]

\[
\text{using } \text{assms } \text{unfolding } \text{End_def by auto}
\]

A function that maps a group \( G \) into itself and satisfies \( f(g_1 \cdot g_2) = f(g_1) \cdot f(g_2) \) is an endomorphism.

**Lemma** \texttt{eq_endomor}:

\[
\text{assumes } f : G \to G \land \quad \forall g_1 \in G. \quad \forall g_2 \in G. \quad f(g_1 \cdot g_2) = f(g_1) \cdot f(g_2)
\]

\[
\text{shows } f \in \text{End}(G, P)
\]

\[
\text{using } \text{assms } \text{unfolding } \text{End_def } \text{Homomor_def } \text{IsMorphism_def by simp}
\]

The set of endomorphisms forms a submonoid of the monoid of function from a set to that set under composition.
lemma (in group0) end_composition:
  assumes f1∈End(G,P) f2∈End(G,P)
  shows Composition(G){f1,f2} ∈ End(G,P)
proof-
  from assms have fun: f1:G→G f2:G→G unfolding End_def by auto
  then have f1 O f2:G→G using comp_fun by auto
  from assms fun(2) have
      ∀g1∈G. ∀g2∈G. (f1 O f2)(g1·g2) = ((f1 O f2)(g1))·((f1 O f2)(g2))
          using group_op_closed comp_fun_apply endomor_eq apply_type
              by simp
  with fun <f1 O f2:G→G> show thesis using eq_endomor func_ZF_5_L2
              by simp
qed

We will use some binary operations that are naturally defined on the function space $G \rightarrow G$, but we consider them restricted to the endomorphisms of $G$. To shorten the notation in such case we define an abbreviation $\text{InEnd}(F,G,P)$ which restricts a binary operation $F$ to the set of endomorphisms of $G$.

abbreviation $\text{InEnd}(_ \{in End\} [_-,_]$)
  where $\text{InEnd}(F,G,P) \equiv \text{restrict}(F,\text{End}(G,P)×\text{End}(G,P))$

Endomorphisms of a group form a monoid with composition as the binary operation, with the identity map as the neutral element.

theorem (in group0) end_comp_monoid:
  shows IsAmonoid(End(G,P),InEnd(Composition(G),G,P))
  and $\text{TheNeutralElement}(End(G,P),\text{InEnd}(\text{Composition}(G),G,P)) = \text{id}(G)$
proof -
  let $C_0 = \text{InEnd}(\text{Composition}(G),G,P)$
  have fun: id(G):G→G unfolding id_def by auto
    { fix g h assume g∈G \& h∈G
      then have id(G)(g·h)=(id(G)g)·(id(G)h)
          using group_op_closed by simp
    }
  with groupAssum fun have id(G) ∈ End(G,P) using eq_endomor by simp
  moreover have $A0$: id(G)=TheNeutralElement(G → G, Composition(G))
  using Group_ZF_2_5_L2(2) by auto
  ultimately have A1: TheNeutralElement(G → G, Composition(G)) ∈ End(G,P)
  by auto
  moreover have $A2$: End(G,P) ⊆ G→G unfolding End_def by blast
  moreover have $A3$: End(G,P) {is closed under} Composition(G)
    using end_composition unfolding IsOpClosed_def by blast
  ultimately show IsAmonoid(End(G,P),C_0)
    using monoid0.group0_1_T1 Group_ZF_2_5_L2(1) unfolding monoid0_def
    by blast
  have IsAmonoid(G→G,Composition(G)) using Group_ZF_2_5_L2(1) by auto
  with A0 A1 A2 A3 show TheNeutralElement(End(G,P),C_0) = id(G)
    using group0_1_L6 by auto
The set of endomorphisms is closed under pointwise addition (derived from the group operation). This is so because the group is abelian.

**Theorem (in abelian_group) end_pointwise_addition:**

Assumes $f \in \text{End}(G,P)$, $g \in \text{End}(G,P)$, $F = P$ (lifted to function space over $G$)

Shows $F(f,g) \in \text{End}(G,P)$

**Proof:**

From assumptions (1, 2) have fun: $f: G \rightarrow G$, $g: G \rightarrow G$ unfolding $\text{End}_{\text{def}}$ by simp_all

With assumptions (3) have fun2: $F(f,g): G \rightarrow G$

Using monoid0.Group_ZF_2_1_L0, group0_2_L1 by simp

Fix $g_1$, $g_2$ assume $g_1 \in G$, $g_2 \in G$

With isAbelian assumptions have

$(F(f,g))(g_1 \cdot g_2) = (F(f,g))(g_1) \cdot (F(f,g))(g_2)$

Using Group_ZF_2_1_L3, group_op_closed endomor_eq

Apply type group0_4_L8(3), Group_ZF_2_1_L3 by simp

With fun2 show thesis using eq_endomor by simp

Qed

The value of a product of endomorphisms on a group element is the product of values.

**Lemma (in abelian_group) end_pointwise_add_val:**

Assumes $f \in \text{End}(G,P)$, $g \in \text{End}(G,P)$, $x \in G$, $P$ (lifted to function space over $G$)

Shows $\text{InEnd}(F,G,P)(f,g)(x) = f(x) \cdot g(x)$

Using assumptions group_oper_fun, monoid.group0_1_L3B, func_ZF_1_L4 unfolding $\text{End}_{\text{def}}$ by simp

Qed

The inverse of an abelian group is an endomorphism.

**Lemma (in abelian_group) end_inverse_group:**

Shows $\text{GroupInv}(G,P) \in \text{End}(G,P)$

Using inverse_in_group, group_inv_of_two, isAbelian, IsCommutative_def

group0_2_T2, groupAssum, Homomor_def

unfolding $\text{End}_{\text{def}}$, $\text{IsMorphism}_{\text{def}}$ by simp

Qed

The set of homomorphisms of an abelian group is an abelian subgroup of the group of functions from a set to a group, under pointwise addition.

**Theorem (in abelian_group) end_addition_group:**

Assumes $F = P$ (lifted to function space over $G$)

Shows IsAgroup(End(G,P), InEnd(F,G,P)) and

InEnd(F,G,P) (is commutative on) End(G,P)

**Proof:**

Have $\text{End}(G,P) \neq \emptyset$ using end_comp_monoid(1), monoid0.group0_1_L3A unfolding $\text{monoid0}_{\text{def}}$ by auto

Moreover have $\text{End}(G,P) \subseteq G \rightarrow G$ unfolding $\text{End}_{\text{def}}$ by auto

Moreover from isAbelian assumptions have $\text{End}(G,P)$ (is closed under) $G$

Qed
unfolding \textit{IsOpClosed\_def} using \textit{end\_pointwise\_addition} by auto

moreover from group\_assum \textit{assms(1)} have
\[ \forall f \in \text{End}(G,P). \text{GroupInv}(G \to G,F)(f) \in \text{End}(G,P) \]
using monoid\_0.group\_0\_1\_L1 end\_composition\(1)\) end\_inverse\_group
func\_ZF\_5\_L2 group\_0\_2\_T2 Group\_ZF\_2\_1\_L6
unfolding monoid\_0\_def End\_def by force

ultimately show \text{IsAgroup}(\text{End}(G,P),\text{InEnd}(F,G,P))
using \textit{assms(1)} group\_0.group\_0\_3\_T3 Group\_ZF\_2\_1\_T2
unfolding IsAsubgroup\_def group\_0\_def by blast

from \textit{assms(1)} isAbelian show
\text{InEnd}(F,G,P) \{ is commutative on \} End(G,P)
using Group\_ZF\_2\_1\_L7 unfolding End\_def IsCommutative\_def by auto

\textbf{qed}

Endomorphisms form a subgroup of the space of functions that map the
group to itself.

\textbf{lemma (in abelian\_group) end\_addition\_subgroup:}
shows \text{IsAsubgroup}(\text{End}(G,P),P \{\text{lifted to function space over} \} G)
using end\_addition\_group unfolding IsAsubgroup\_def by simp

The neutral element of the group of endomorphisms of a group is the con-
stant function with value equal to the neutral element of the group.

\textbf{lemma (in abelian\_group) end\_add\_neut\_elem:}
assumes \(F = P \{\text{lifted to function space over} \} G\)
shows \(\text{The Neutral Element}(\text{End}(G,P),\text{InEnd}(F,G,P)) = \text{ConstantFunction}(G,1)\)
using \textit{assms} end\_addition\_subgroup lift\_group\_subgr\_neut by simp

For the endomorphisms of a group \(G\) the group operation lifted to the func-
tion space over \(G\) is distributive with respect to the composition operation.

\textbf{lemma (in abelian\_group) distributive\_comp\_pointwise:}
assumes \(F = P \{\text{lifted to function space over} \} G\)
shows \(\text{IsDistributive}(\text{End}(G,P),\text{InEnd}(F,G,P),\text{InEnd}(\text{Composition}(G),G,P))\)

\textbf{proof -}
\begin{itemize}
  \item \(C_G = \text{Composition}(G)\)
  \item \(C_E = \text{InEnd}(C_G,G,P)\)
  \item \(F_E = \text{InEnd}(F,G,P)\)
  \begin{itemize}
    \item fix \(b\ c\ d\) assume \(A_S: b \in \text{End}(G,P)\ c \in \text{End}(G,P)\ d \in \text{End}(G,P)\)
    \begin{itemize}
      \item with \textit{assms(1)} have \(I_1: C_G\langle b, F\langle c, d\rangle \rangle = b \circ (F(c,d))\)
        using monoid.Group\_ZF\_2\_1\_L0 func\_ZF\_5\_L2 unfolding End\_def by auto
      \item with \textit{assms} have \(I_2: F(C_G\langle b,c\rangle,C_G\langle b,d\rangle) = F(b \circ c,b \circ d)\)
        unfolding End\_def using func\_ZF\_5\_L2 by auto
    \end{itemize}
    \end{itemize}
  \end{itemize}
\end{itemize}

\textbf{qed}
with assms(1) AS(2, 3) have \((b \circ (F(c,d)))(g) = b((F(c,d))(g))\) 
using comp_fun_apply monoid.Group_ZF_2_1_L0 unfolding End_def by force 
with groupAssum assms(1) AS \(g \in G\) have 
\((b \circ (F(c,d)))(g) = (F(b \circ c, b \circ d))(g)\) 
using Group_ZF_2_1_L3 unfolding homomor_eq func_eq 
unfolding End_def by auto 
\} hence \(\forall g \in G. \ (b \circ (F(c,d)))(g) = (F(b \circ c, b \circ d))(g)\) by simp 
with compfun comp2fun ig1 ig2 have 
\(C_G(b,F(c,d)) = F(C_G(b,c),C_G(b,d))\) 
using func_eq by simp 
moreover from AS(2, 3) have \(F(c,d) = F_E(c, d)\) 
using restrict by simp 
moreover from AS have \(C_G(b,c) = C_E(b,c)\) and \(C_G(b,d) = C_E(b,d)\) 
using restrict by auto 
moreover from assms AS have \(C_G(b,F(c,d)) = E(c, F(c,d))\) 
using end_pointwise_addition by simp 
moreover from AS have \(F(C_G(b,c),C_G(b,d)) = F_E(C_G(b,c),C_G(b,d))\) 
using end_composition by simp 
ultimately have eq1: \(C_E(b,F_E(c,d)) = F_E(C_E(b,c),C_E(b,d))\) 
by simp 
from assms(1) AS have 
\(\text{compfun: } (F(c,d)) \circ 0 : G \rightarrow G \ F(c \circ 0, b \circ 0) : G \rightarrow G\) 
using monoid.Group_ZF_2_1_L0 comp_fun unfolding End_def by auto 
\} fix \(g\) assume \(g \in G\) 
with AS(1) have \(bg: b(g) \in G\) unfolding End_def using apply_type 
by auto 
from \(\langle g \in G, AS(1) \rangle\) have \(((F(c,d)) \circ 0) \circ bg = (F(c,d))(bg)\) 
using comp_fun_apply unfolding End_def by force 
also from assms(1) AS(2, 3) bg have \(\ldots = (c(b(g))) \cdot (d(b(g)))\) 
using Group_ZF_2_1_L3 unfolding End_def by auto 
also from \(\langle g \in G, AS(1) \rangle\) have \(\ldots = ((c \circ 0)(bg)) \cdot ((d \circ 0)(bg))\) 
using comp_fun_apply unfolding End_def by auto 
also from assms(1) \(\langle g \in G, AS(1) \rangle\) have \(\ldots = (F(c \circ 0, d \circ 0)) g\) 
using comp_fun Group_ZF_2_1_L3 unfolding End_def by auto 
finally have \(((F(c,d)) \circ 0) \circ bg = (F(c \circ 0, d \circ 0))(g)\) by simp 
\} hence \(\forall g \in G. \ ((F(c,d)) \circ 0) \circ bg = (F(c \circ 0, d \circ 0))(g)\) by simp 
with compfun have \(\text{eq1: } (F(c,d)) \circ 0 = F(c \circ 0, d \circ 0)\) 
using func_eq by blast 
with assms(1) AS have \(C_G(F(c,d),b) = F(C_G(c,b),C_G(d,b))\) 
using monoid.Group_ZF_2_1_L0 funcZF_5_L2 unfolding End_def by simp 
moreover from AS(2, 3) have \(F(c, d) = F_E(c, d)\) 
using restrict by simp 
moreover from AS have \(C_G(c,b) = C_E(c,b)\) and \(C_G(d,b) = C_E(d,b)\) 
using restrict by auto 
moreover from assms AS have \(C_G(F(c,d),b) = C_E(F(c,d),b)\)
using end_pointwise_addition by auto

moreover from AS have \( F(C_G(c,b), C_G(d,b)) = F_E(C_G(c,b), C_G(d,b)) \)
using end_composition by auto

ultimately have \( C_E(F_E(c,d), b) = F_E(C_E(c,b), C_E(d,b)) \)
by auto

} then show thesis unfolding IsDistributive_def by auto

qed

The endomorphisms of an abelian group is in fact a ring with the previous operations.

theorem (in abelian_group) end_is_ring:
  assumes F = P (lifted to function space over) G
  shows
    IsAring(End(G,P), InEnd(P {lifted to function space over} G,G,P), InEnd(Composition(G),G,P))
using assms end_addition_group end_comp_monoid(1) distributive_comp_pointwise
unfolding IsAring_def by auto

The theorems proven in the ring0 context are valid in the abelian_group context as applied to the endomorphisms of \( G \).

sublocale abelian_group < endo_ring: ring0
  End(G,P)
  InEnd(P {lifted to function space over} G,G,P)
  InEnd(Composition(G),G,P)
  \( \lambda x \ b. \ InEnd(P {lifted to function space over} G,G,P)(x,b) \)
  \( \lambda x. \ GroupInv(End(G, P), InEnd(P {lifted to function space over} G,G,P))(x) \)

  \( \lambda x \ b. \ InEnd(P {lifted to function space over} G,G,P)(x, GroupInv(End(G, P), InEnd(P {lifted to function space over} G,G,P))(b)) \)
  \( \lambda x. \ InEnd(Composition(G),G,P)(x, b) \)
  TheNeutralElement(End(G, P), InEnd(P {lifted to function space over} G,G,P))(x, b)
  \( \lambda x. \ InEnd(Composition(G),G,P)(x, x) \)
  TheNeutralElement(End(G, P), InEnd(Composition(G),G,P))
  \( \lambda x. \ InEnd(Composition(G),G,P)(x, x) \)
  \( \lambda x. \ InEnd(Composition(G),G,P)(x, x) \)
using end_is_ring unfolding ring0_def by blast

46.2 First isomorphism theorem

Now we will prove that any homomorphism \( f : G \to H \) defines a bijective homomorphism between \( G/H \) and \( f(G) \).

A group homomorphism sends the neutral element to the neutral element.

lemma image_neutral:
assumes `IsAgroup(G,P)` `IsAgroup(H,F)` `Homomor(f,G,P,H,F)`

shows `f(TheNeutralElement(G,P)) = TheNeutralElement(H,F)`

proof -

let `e_G = TheNeutralElement(G,P)`
let `e_H = TheNeutralElement(H,F)`
from `assms(3)` have `ff: f:G→H`
  unfolding `Homomor_def` by simp
have `g: e_G = P(e_G,e_G)` `e_G ∈ G`
  using `assms(1)` `group0.group0_2_L2` unfolding `group0_def` by simp_all
with `assms` have `f(e_G) = F(f(e_G),f(e_G))`
  unfolding `Homomor_def` `IsMorphism_def` by force
moreover from `ff g(2)` have `h: f(e_G) ∈ H`
  using `apply_type` by simp
with `assms(2)` have `im: f(e_G) ∈ H`
  using `group0.group0_2_L2` unfolding `group0_def` by simp
ultimately have `F(f(e_G),e_H) = F(f(e_G),f(e_G))`
  by simp
with `assms(2)` have `LeftTranslation(H,F,f(e_G))(e_H) = LeftTranslation(H,F,f(e_G))(f(e_G))`
  using `group0.group0_5_L2(2)` `group0.group0_2_L2` unfolding `group0_def`
  by simp
moreover from `assms(2)` have `LeftTranslation(H,F,f(e_G))(e_H) ∈ inj(H,H)`
  using `group0.trans_bij(2)` unfolding `group0_def` `bij_def` by simp
ultimately show thesis using `h` `assms(2)` `group0.group0_2_L2`
  unfolding `inj_def` `group0_def` by force
qed

If `f : G → H` is a homomorphism, then it commutes with the inverse

lemma `image_inv`:

assumes `IsAgroup(G,P)` `IsAgroup(H,F)` `Homomor(f,G,P,H,F)` `g ∈ G`

shows `f(GroupInv(G,P)(g)) = GroupInv(H,F)(f(g))`

proof -

from `assms(3)` have `ff: f:G→H`
  unfolding `Homomor_def` by simp
with `assms(4)` have `im: f(g) ∈ H`
  using `apply_type` by simp
from `assms(1,4)` have `inv: GroupInv(G,P)(g) ∈ G`
  using `group0.inverse_in_group` unfolding `group0_def` by simp
with `ff` have `inv2: f(GroupInv(G,P)g) ∈ H`
  using `apply_type` by simp
from `assms(1,4)` have `f(TheNeutralElement(G,P)) = f(P(g,GroupInv(G,P)(g)))`
  using `group0.group0_2_L6` unfolding `group0_def` by simp
also from `assms` have `... = F(f(g),f(GroupInv(G,P)(g)))`
  unfolding `Homomor_def` `IsMorphism_def` by simp
finally have `f(TheNeutralElement(G,P)) = F(f(g),f(GroupInv(G,P)(g)))`
  by simp
with `assms` `inv2` show thesis

543
The preimage of a subgroup is a subgroup

**Theorem preimage_sub:**

**Assumes**
- IsAgroup(G,P)
- IsAgroup(H,F)
- Homomor(f,G,P,H,F)
- IsAsubgroup(K,F)

**Shows**
- IsAsubgroup(f(K),P)

**Proof:**
- From assumptions (3) have ff: f:G→H
- Unfolding Homomor_def by simp
- From assumptions (2) have Hgr: group0(H,F)
- Unfolding group0_def by simp
- From assumptions (1) have Ggr: group0(G,P)
- Unfolding group0_def by simp
- Moreover
- From assumptions ff Ggr Hgr have TheNeutralElement(G,P) ∈ f(K)
- Using image_neutral group0.group0_3_L5 func1_1_L15 group0.group0_2_L2
- By simp
- Hence f(K) ≠ 0 by blast
- Moreover from ff have f(K) ⊆ G using func1_1_L3 by simp
- Moreover from assumptions ff Ggr Hgr have f(K) {is closed under} P
- Using func1_1_L15 group0.group0_3_L6 group0.group_op_closed func1_1_L15
- Unfolding IsOpClosed_def Homomor_def IsMorphism_def by simp
- Moreover from assumptions ff Ggr Hgr have
  - ∀x∈f(K). GroupInv(G, P)(x) ∈ f(K)
  - Using group0.group0_3_T3A image_inv func1_1_L15
  - group0.inverse_in_group by simp
- Ultimately show thesis by (rule group0.group0_3_T3)

**QED**

The preimage of a normal subgroup is normal

**Theorem preimage_normal_subgroup:**

**Assumes**
- IsAgroup(G,P)
- IsAgroup(H,F)
- Homomor(f,G,P,H,F)
- IsAnormalSubgroup(H,F,K)

**Shows**
- IsAnormalSubgroup(G,P,f(K))

**Proof:**
- From assumptions (3) have ff: f:G→H
- Unfolding Homomor_def by simp
- From assumptions (2) have Hgr: group0(H,F)
- Unfolding group0_def by simp
- With assumptions (4) have K⊂H using group0.group0_3_L2
- Unfolding IsAnormalSubgroup_def by simp
- From assumptions (1) have Ggr: group0(G,P)
- Unfolding group0_def by simp
- Moreover from assumptions have IsAsubgroup(f(K),P)
- Using preimage_sub unfolding IsAnormalSubgroup_def by simp
- Moreover
  - { fix g assume gG: g∈G
  -   { fix h assume h ∈ {P|g,P(h, GroupInv(G, P)(g))}. h ∈ f(K) }
  -   then obtain k where
  -     k: h = P|g,P(k, GroupInv(G, P)(g)) k ∈ f(K) }
by auto
from k(1) have f(h) = f(P(k, GroupInv(G, P)(g))) by simp
moreover from ff k(2) have k ∈ G using vimage_iff
  unfolding Pi_def by blast
ultimately have f: f(h) = F(f(g), F(f(k), GroupInv(H, F)(f(g)))) by simp
from assms(1) ff Ggr g ∈ G using group0.group_op_closed
  group0.inverse_in_group image_inv homomor_eq by auto
ultimately have f: f(h) = F(F(f(g), f(k)), GroupInv(H, F)(f(g))) ∈ K
using func1_1_L15 apply_type unfolding IsAnormalSubgroup_def by auto
moreover from f(k) ∈ K and K ⊆ H have f(g) ∈ H
  using func_imagedef sub apply_type unfolding by auto
moreover from f(k) ∈ K and f(g) ∈ H have f(h) ∈ K
  by blast
hence f(f(g), f(k), GroupInv(H, F)(f(g))) ∈ K by auto
ultimately have f(h) ∈ K by blast
with ff h ∈ f−1(K) using func1_1_L15 by simp
{P⟨g, P⟨h, GroupInv(G, P)(g)⟩⟩. h ∈ f−1(K)} ⊆ f−1(K)
by simp
ultimately show thesis using group0.cont_conj_is_normal by simp
qed

The kernel of an homomorphism is a normal subgroup.
corollary kernel_normal_sub:
  assumes IsAgroup(G, P) IsAgroup(H, F) Homomor(f, G, P, H, F)
  shows IsAnormalSubgroup(G, P, f−1(TheNeutralElement(H, F)))
using assms preimage_normal_subgroup group0.trivial_normal_subgroup
  unfolding group0_def by auto

The image of a subgroup is a subgroup
theorem image_subgroup:
  assumes IsAgroup(G, P) IsAgroup(H, F)
    Homomor(f, G, P, H, F) f: G → H IsAsubgroup(K, P)
  shows IsAsubgroup(fK, F)
proof -
from assms(1, 5) have sub: K ⊆ G using group0.group0_3_L2
  unfolding group0_def by simp
from assms(2) have group0(H, F) unfolding group0_def by simp
moreover from assms(4) have f(K) ⊆ H
  using func_imagedef sub apply_type by auto
moreover
from assms(1, 4, 5) sub have f(TheNeutralElement(G, P)) ∈ f(K)
  using group0.group0_3_L5 func_imagedef unfolding group0_def by auto
hence f(K) ≠ 0 by blast
moreover
{ fix \( x \in f(K) \)
with \( \text{assms}(4) \) sub obtain \( q \) where \( q \in K \) \( x = f(q) \)
using func_imagedef by auto
with \( \text{assms}(1-4) \) sub have \( \text{GroupInv}(H,F)(x) = f(\text{GroupInv}(G,P)q) \)
using image_inv by auto
with \( \text{assms}(1,4,5) \) \( q(1) \) sub have \( \text{GroupInv}(H,F)(x) \in f(K) \)
using group0.group0_3_T3A func_imagedef unfolding group0_def by auto
} hence \( \forall x \in f(K) \). \( \text{GroupInv}(H,F)(x) \in f(K) \) by auto
moreover
{ fix \( x \), \( y \)
assume \( x \in f(K) \) \( y \in f(K) \)
with \( \text{assms}(4) \) sub obtain \( q_x \), \( q_y \) where
\( q_x \in K \) \( x = f(q_x) \) \( q_y \in K \) \( y = f(q_y) \)
using func_imagedef by auto
with \( \text{assms}(1-3) \) sub have \( F(x,y) = f(P(q_x,q_y)) \)
using homomor_eq by force
moreover from \( \text{assms}(1,5) \) \( q(1,3) \) have \( P(q_x,q_y) \in K \)
using group0.group0_3_L6 unfolding group0_def by simp
ultimately have \( F(x,y) \in f(K) \)
using \( \text{assms}(4) \) sub func_imagedef by auto
} then have \( f(K) \) \{is closed under\} \( F \) unfolding IsOpClosed_def
by simp
ultimately show thesis using group0.group0_3_T3 by simp
qed

The image of a group under a homomorphism is a subgroup of the target group.

corollary image_group:
assumes \( \text{IsAgroup}(G,P) \) \( \text{IsAgroup}(H,F) \) \( \text{Homomor}(f,G,P,H,F) \)
shows \( \text{IsAsubgroup}(f(G),F) \)
proof-
from \( \text{assms}(1) \) have \( \text{restrict}(P,G \times G) = P \)
using group0.group_oper_fun restrict_domain unfolding group0_def by blast
with \( \text{assms} \) show thesis using image_subgroup
unfolding Homomor_def IsAsubgroup_def by simp
qed

Now we are able to prove the first isomorphism theorem. This theorem states that any group homomorphism \( f : G \rightarrow H \) gives an isomorphism between a quotient group of \( G \) and a subgroup of \( H \).

theorem isomorphism_first_theorem:
assumes \( \text{IsAgroup}(G,P) \) \( \text{IsAgroup}(H,F) \) \( \text{Homomor}(f,G,P,H,F) \)
defines \( r \equiv \text{QuotientGroupRel}(G,P,f-(\text{TheNeutralElement}(H,F))) \) and \( P \equiv \text{QuotientGroupOp}(G,P,f-(\text{TheNeutralElement}(H,F))) \)
supposes \( \exists f. \text{Homomor}(f,G//r,P,f(G),\text{restrict}(F,(f(G)) \times (f(G)))) \land f \in \text{bij}(G//r,f(G)) \)
proof-
let \( f = \{ (r(g),f(g)). g \in G \} \)

546
from assms(3) have ff: f:G→H
  unfolding Homomor_def by simp
from assms(1-5) have equiv(G,r)
  using group0.Group_ZF_2_4_L3 kernel_normal_sub
  unfolding group0_def IsAnormalSubgroup_def by simp
from assms(4) ff have \( f \in \text{Pow}((G//r) \times f(G)) \)
  unfolding quotient_def using func_imagedef by auto
moreover have \( (G//r) \subseteq \text{domain}(f) \)
  unfolding domain_def quotient_def by auto
moreover
\{ 
  fix \( x,y,t \) assume \( A: (x,y) \in f(x,t) \in f \)
  then obtain \( g_y, g_r \) where \( (x,y) \in \{r\{g_y\}, f(g_y)\} \)
  and \( g_r \in G \) \( g_y \in G \) by auto
  hence \( B: r\{g_y\}=r\{g_r\} \) \( y=f(g_y) \) \( t=f(g_r) \) by auto
  from ff \( \langle g_y, g_r \rangle \in B(2,3) \) have \( y \in H \)
  using apply_type by simp_all
  with \( \langle \text{equiv}(G,r) \rangle \)
  \( \langle g_r, g_y \rangle = r\{g_r\} \) have \( \{g_y, g_r\} \in r \)
  using same_image_equiv by simp
with assms(4) ff have
  \( f\langle P\{a_1, a_2\}\rangle = \text{TheNeutralElement}(H,F) \)
  unfolding QuotientGroupRel_def using func1_1_L15 by simp
with assms(1-4) B(2,3) \( \langle g_y, g_r \rangle \in B(2,3) \)
  \( \langle g_r, g_y \rangle \in B(2,3) \) have \( y=t \)
  using image_inv group0.inverse_in_group group0.group0_2_L11A
  unfolding group0_def Homomor_def IsMorphism_def by auto
\} hence \( \forall x,y. (x,y) \in f \longrightarrow (\forall z. (x,z) \in f \longrightarrow y=z) \) by auto
ultimately have \( \text{ff}_\text{fun}: f:G//r \to f(G) \)
  unfolding Pi_def function_def by auto
\{ 
  fix \( a_1, a_2 \) assume \( A_1: a_1 \in G//r \) \( a_2 \in G//r \)
  then obtain \( g_1, g_2 \) where \( g_1 \in G \) \( g_2 \in G \) and \( a_1 = r\{g_1\} \) \( a_2 = r\{g_2\} \)
  unfolding quotient_def by auto
  with assms \( \langle \text{equiv}(G,r) \rangle \) have \( \langle P\{a_1, a_2\}\rangle f\langle P\{g_1, g_2\}\rangle \in f \)
  using Group_ZF_2_4_L5A kernel_normal_sub group0.Group_ZF_2_2_L2
  group0.group_op_closed
  unfolding QuotientGroup0p_def group0_def by auto
  with \( \text{ff}_\text{fun} \) have \( \text{eq}: f\langle P\{a_1, a_2\}\rangle = f\langle P\{g_1, g_2\}\rangle \)
  using apply_equality
  by simp
from \( \langle g_1, g_2 \in G \rangle \) \( \langle a_1, f(g_1) \rangle \in f \) and \( \langle a_2, f(g_2) \rangle \in f \) by auto
with assms(1,2,3) ff_fun \( \langle g_1, g_2 \rangle \in G \) \( \langle a_1, a_2 \rangle \in G \)
  eq have \( f\langle f\langle a_1\rangle, f\langle a_2\rangle\rangle = f\langle P\{a_1, a_2\}\rangle \)
  using apply_equality unfolding Homomor_def IsMorphism_def by simp
moreover from \( A \) \( \text{ff}_\text{fun} \) have \( f\langle a_1 \rangle \in f(G) \) \( f\langle a_2 \rangle \in f(G) \)
  using apply_type by auto
ultimately have \( \text{restrict}(F, f(G) \times f(G))\langle f\langle a_1\rangle, f\langle a_2\rangle\rangle = f\langle P\{a_1, a_2\}\rangle \)
  by simp
\} hence \( r: \forall a_1 \in G//r. \forall a_2 \in G//r. \text{restrict}(F, f(G) \times f(G))\langle f\langle a_1\rangle, f\langle a_2\rangle\rangle = f\langle P\{a_1, a_2\}\rangle \)
  by simp
with \( \text{ff}_\text{fun} \) have \( \text{HOM}: \text{Homomor}(j, G//r, P, f(G), \text{restrict}(F, (f(G)) \times (f(G)))) \)
unfolding Homomor_def IsMorphism_def by simp
from assms have G: IsAgroup(G//r,P) and
    H: IsAgroup(f(G), restrict(F,f(G)×f(G)))
    using Group_ZF_2_4_T1 kernel_normal_sub image_group
    unfolding IsAsubgroup_def by simp_all
{ fix b₁ b₂ assume AS: f(b₁) = f(b₂) b₁∈G//r b₂∈G//r
  from G AS(3) have invb2: GroupInv(G//r,P)(b₂)∈G//r
    using group0.inverse_in_group
    unfolding group0_def by simp
  with G AS(2) have I: P⟨b₁,GroupInv(G//r,P)(b₂)⟩∈G//r
    using group0.group_op_closed
    unfolding group0_def by auto
  then obtain g where g∈G and gg: P⟨b₁,GroupInv(G//r,P)(b₂)⟩=r{g}
    unfolding quotient_def by auto
  from ‹g∈G› have ⟨r{g},f(g)⟩ ∈ f by blast
  with ff_fun gg have E: f(P⟨b₁,GroupInv(G//r,P)(b₂)⟩) = f(g)
    using apply_equality by simp
  from ff_fun AS(2,3) have fff: f(b₁) ∈ f(G) f(b₂) ∈ f(G)
    using apply_type by simp_all
  from fff(1) pp have EE: F(P⟨f(b₁),GroupInv(G//r,P)(b₂)⟩) =
    restrict(F,f(G)×f(G))⟨f(b₁),⟨GroupInv(G//r,P)(b₂)⟩⟩
    by auto
  from ff have f(G) ⊆ H using func1_1_L6(2) by simp
  with fff have f(b₁)∈H f(b₂)∈H by auto
  with assms(1-4) G H HOM ff_fun AS(1,3) fff(2) EE have
    TheNeutralElement(H,F) =
    restrict(F,f(G)×f(G))⟨f(b₁),⟨GroupInv(G//r,P)(b₂)⟩⟩
    using group0.group0_2_L6(1) restrict image_inv group0.group0_3_T1
    image_group
    unfolding group0_def by simp
  also from G H HOM AS(2,3) E have ... = f(g)
    using group0.inverse_in_group unfolding group0_def IsMorphism_def
    Homomor_def
  finally have TheNeutralElement(H,F) = f(g) by simp
  with ff ⟨g∈G⟩ have g∉{TheNeutralElement(H,F)} using func1_1_L15
    by simp
  with assms ⟨g∈G⟩ have P⟨b₁,GroupInv(G//r,P)(b₂)⟩ = TheNeutralElement(G//r,P)
    using group0.Group_ZF_2_4_L5E kernel_normal_sub unfolding group0_def
    by simp
  with AS(2,3) G have b₁=b₂ using group0.group0_2_L11A unfolding group0_def
    by auto
} with ff_fun have ‹f ∈ inj(G//r,f(G))› unfolding inj_def by blast

548
moreover
{ fix m assume m ∈ f(G)
  with ff obtain g where g∈G = f(g) using func_imagedef by auto
  hence ⟨r(g),m⟩ ∈ f by blast
  with ff_fun have f(r(g)) = m using apply_equality by auto
  with g∈G have f(r(A)) = m unfolding quotient_def by auto
}
ultimately have f ∈ bij(G//r,fG)
using ff_fun by blast
with HOM show thesis by blast
qed

The inverse of a bijective homomorphism is an homomorphism. Meaning
that in the previous result, the homomorphism we found is an isomorphism.

theorem bij_homomor:
  assumes f∈bij(G,H) IsAgroup(G,P) Homomor(f,G,P,H,F)
  shows Homomor(converse(f),H,F,G,P)
proof -
{ fix h1 h2 assume h1∈H h2∈H
  with assms(1) obtain g1 g2 where
    g1: g1∈G f(g1)=h1 and g2: g2∈G f(g2)=h2
  unfolding bij_def surj_def by blast
  with assms(2,3) have converse(f)(f(P⟨g1,g2⟩)) = converse(f)(F⟨h1,h2⟩)
  using homomor_eq by simp
  with assms(1,2) g1 g2 have P(converse(f)(h1), converse(f)(h2)) = converse(f)(F⟨h1,h2⟩)
  using left_inverse group0.group_op_closed unfolding group0_def bij_def
  by auto
}
{ with assms(1) show thesis using bij_converse_bij bij_is_fun
  unfolding Homomor_def IsMorphism_def by simp
qed

A very important homomorphism is given by taking every element to its
class in a group quotient. Recall that λx∈X.p(x) is an alternative notation
for function defined as a set of pairs, see lemma lambda_fun_alt in theory
func1.thy.

lemma (in group0) quotient_map:
  assumes IsAnormalSubgroup(G,P,H)
  defines r ≡ QuotientGroupRel(G,P,H) and q ≡ λx∈G. QuotientGroupRel(G,P,H){x}
  shows Homomor(q,G,P,G//r,QuotientGroupOp(G,P,H))
using groupAssum assms group_op_closed lam_funtype lamE EquivClass_1_L10

Group_ZF_2_4_L3 Group_ZF_2_4_L5A Group_ZF_2_4_T1
unfolding IsAnormalSubgroup_def QuotientGroupOp_def Homomor_def IsMorphism_def
by simp

In the context of group0, we may use all results of semigr0.
This section defines the concept of a ring ideal, and defines some basic concepts and types, finishing with the theorem that shows that the quotient of the additive group by the ideal is actually a full ring.

47.1 Ideals

In ring theory ideals are special subsets of a ring that play a similar role as normal subgroups in the group theory. An ideal is a subgroup of the additive group of the ring, which is closed by left and right multiplication by any ring element.

definition (in ring) Ideal (_＜R) where
  I ＜R ≡ (∀x∈I. ∀y∈R. y·x∈I ∧ x·y∈I) ∧ IsAsubgroup(I,A)

To write less during proofs, we will write I to denote the set of ideals of the ring R.

abbreviation (in ring) ideals (I) where
  I ≡ {J∈Pow(R). J ＜R}

The first examples of ideals are the whole ring and the zero ring:

lemma (in ring) ring_self_ideal:
  shows R ＜R
  using add_group.group_self_subgroup Ring_ZF_1_L4(3)

The singleton containing zero is an ideal.

lemma (in ring) zero_ideal:
  shows {0} ＜R unfolding Ideal_def
  using Ring_ZF_1_L6 add_group.unit_singl_subgr by auto

An ideal is a subset of the ring.

lemma (in ring) ideal_dest_subset:
  assumes I ＜R

550
shows \( I \subseteq \mathbb{R} \) using assms add_group.group0_3_L2
unfolding Ideal_def by auto

Ideals are closed with respect to the ring addition.

lemma (in ring0) ideal_dest_sum:
  assumes \( I \triangleleft \mathbb{R} \ x \in I \ y \in I \)
  shows \( x+y \in I \) using assms add_group.group0_3_L6
  unfolding Ideal_def by auto

Ideals are closed with respect to the ring multiplication.

lemma (in ring0) ideal_dest_mult:
  assumes \( I \triangleleft \mathbb{R} \ x \in I \ y \in \mathbb{R} \)
  shows \( x \cdot y \in I \) \( y \cdot x \in I \) using assms unfolding Ideal_def by auto

Ideals are closed with respect to taking the opposite in the ring.

lemma (in ring0) ideal_dest_minus:
  assumes \( I \triangleleft \mathbb{R} \ x \in I \)
  shows \(-x) \in I \)
  using assms add_group.group0_3_T3A unfolding Ideal_def by auto

Every ideals contains zero.

lemma (in ring0) ideal_dest_zero:
  assumes \( I \triangleleft \mathbb{R} \)
  shows \( 0 \in I \)
  using assms add_group.group0_3_L5 unfolding Ideal_def by auto

The simplest way to obtain an ideal from others is the intersection, since
the intersection of arbitrary collection of ideals is an ideal.

theorem (in ring0) intersection_ideals:
  assumes \( \forall \ J \in J . \ (J \triangleleft \mathbb{R}) \ J \neq 0 \)
  shows \( (\bigcap J) \triangleleft \mathbb{R} \)
  using assms ideal_dest_mult add_group.subgroup_inter
  unfolding Ideal_def by auto

In particular, intersection of two ideals is an ideal.

corollary (in ring0) inter_two_ideals: assumes \( I \triangleleft \mathbb{R} \ J \triangleleft \mathbb{R} \)
  shows \( (I \cap J) \triangleleft \mathbb{R} \)
proof -
  let \( J = \{I, J\} \)
  from assms have \( \forall \ J \in J . \ (J \triangleleft \mathbb{R}) \) and \( J \neq 0 \) by simp_all
  then have \( (\bigcap J) \triangleleft \mathbb{R} \)
    using intersection_ideals by blast
  thus thesis by simp
qed

From any set, we may construct the minimal ideal containing that set

definition (in ring0) generatedIdeal \((\_)_\mathbb{R}\):
  where \( X \subseteq \mathbb{R} \implies (X)_\mathbb{R} \equiv \bigcap \{ I \in I . \ X \subseteq I \} \)
The ideal generated by a set is an ideal

corollary (in ring0) generated_ideal_is_ideal:
  assumes XÎR shows ⟨X⟩I îR
proof -
  let J = {IÎI. X Î I}
  have ∀J ∈ J. (J îR) by auto
  with assms have (∩J) îR using ring_self_ideal intersection_ideals
  by blast
  with assms show thesis using generatedIdeal_def by simp
qed

The ideal generated by a set is contained in any ideal containing the set.

corollary (in ring0) generated_ideal_small:
  assumes XÎI I îR
  shows ⟨X⟩I î I
proof -
  from assms have IÎ{JÎPow(R). J îR ∧ X Î J}
  using ideal_dest_subset by auto
  then have ∩{JÎPow(R). J îR ∧ X Î J} î I by auto
  moreover from assms have X î R using ideal_dest_subset by auto
  ultimately show ⟨X⟩I î I using generatedIdeal_def by auto
qed

The ideal generated by a set contains the set.

corollary (in ring0) generated_ideal_contains_set:
  assumes XÎR shows X î ⟨X⟩I
  using assms ring_self_ideal generatedIdeal_def by auto

To be able to show properties of an ideal generated by a set, we have the following induction result

lemma (in ring0) induction_generated_ideal:
  assumes
  X≠0
  XÎR
  ∀yÎR. ∀zÎR. P(q) −−> P(y·q·z)
  ∀yÎR. ∀zÎR. P(y) ∧ P(z) −−> P(y+z)
  ∀xÎX. P(x)
  shows ∀yÎ⟨X⟩I. P(y)
proof -
  let J = {mÎ⟨X⟩I. P(m)}
  from assms(2,5) have XÎJ
  using generated_ideal_contains_set by auto
  from assms(2) have JÎR
  using generated_ideal_is_ideal ideal_dest_subset by auto
  moreover
  { fix y z assume yÎR zÎJ
    then have yÎR 1ÎR zÎ⟨X⟩I P(z)
    using Ring_ZF_1_L2(2) by simp_all
  }
with assms(3) have P(y·z·1) and P(1·z·y) by simp_all
with \( J \subseteq R \) \( y \in R \) \( z \in J \) have P(y·z) and P(z·y)
using Ring_ZF_1_L4(3) Ring_ZF_1_L3(5,6) by auto
with assms(2) \( x \in \langle X \rangle \) \( y \in R \) have y·z·J \( z·y \in J \)
using ideal_dest_mult generated_ideal_is_ideal by auto
}\ hence \( \forall x \in J. \forall y \in R. y \cdot x \in J \land x \cdot y \in J \) by auto
moreover have IsASubgroup(J,A)
proof
- from assms(1) \( X \subseteq J \) \( J \subseteq R \)
  have J \( \neq \emptyset \) and J \( \subseteq R \)
  by auto
moreover
{ fix x y assume as: \( x \in J \) \( y \in J \)
  with assms(2,4) have P(x+y)
    using ideal_dest_subset generated_ideal_is_ideal by blast
  with assms(2) \( x \in J \) \( y \in J \) have x+y \( \in J \)
  using generated_ideal_is_ideal ideal_dest_sum by auto
}\ then have J \{is closed under\} A
unfolding IsOpClosed_def by auto
moreover
{ fix x assume \( x \in J \)
  with \( J \subseteq R \) have \( x \in \langle X \rangle \) \( x \in R \) P(x)
  by auto
  with assms(3)
  have P((-1)·x·1)
    using Ring_ZF_1_L2(2) Ring_ZF_1_L3(1)
    by simp
  with assms(2) \( x \in \langle X \rangle \) \( x \in R \) have \((-x)\)\( \in J \)
  using Ring_ZF_1_L3(1,5,6) Ring_ZF_1_L7(1) Ring_ZF_1_L2(2)
  generated_ideal_is_ideal ideal_dest_minus by auto
}\ hence \( \forall x \in J. (-x) \in J \) by simp
ultimately show IsASubgroup(J,A)
  by (rule add_group.group0_3_T3)
qed
ultimately have J \( \triangleleft \) R
unfolding Ideal_def by auto
with \( \langle X \rangle \subseteq J \) show thesis using generated_ideal_small by auto
qed

An ideal is very particular with the elements it may contain. If it contains the neutral element of multiplication then it is in fact the whole ring and not a proper subset.

**Theorem (in ring0)** ideal_with_one:
assumes \( I \triangleleft R \) \( 1 \in I \) shows \( I = R \)
using assms ideal_dest_subset ideal_dest_mult(2) Ring_ZF_1_L3(5)
by force

The only ideal containing an invertible element is the whole ring.

**Theorem (in ring0)** ideal_with_unit:
assumes \( I \triangleleft R \) \( x \in I \) \( \exists y \in R. y \cdot x = 1 \lor x \cdot y = 1 \)
s shows \( I = R \)
The previous result drives us to define what a maximal ideal would be: an ideal such that any bigger ideal is the whole ring:

**definition** (in ring0) maximalIdeal (_\_ R) where

\[ I \triangleleft R \equiv I \triangleleft R \land I \neq R \land (\forall J \in I. I \subseteq J \land J \neq R \rightarrow I = J) \]

Before delving into maximal ideals, let’s define some operation on ideals that are useful when formulating some proofs. The product ideal of ideals \( I, J \) is the smallest ideal containing all products of elements from \( I \) and \( J \):

**definition** (in ring0) productIdeal (infix \( \cdot \)) where

\[ I \triangleleft R \Rightarrow J \triangleleft R \Rightarrow I \cdot I J \equiv \langle M(I \times J) \rangle \]

The sum ideal of ideals is the smallest ideal containing both \( I \) and \( J \):

**definition** (in ring0) sumIdeal (infix \( + \)) where

\[ I \triangleleft R \Rightarrow J \triangleleft R \Rightarrow I + I J \equiv \langle I \cup J \rangle \]

Sometimes we may need to sum an arbitrary number of ideals, and not just two.

**definition** (in ring0) sumArbitraryIdeals (\( \oplus \)) where

\[ J \subseteq I \Rightarrow \oplus I J \equiv \langle \bigcup J \rangle \]

Each component of the sum of ideals is contained in the sum.

**lemma** (in ring0) comp_in_sum_ideals:

assumes \( I \triangleleft R \) and \( J \triangleleft R \)

shows \( I \subseteq I + I J \) and \( J \subseteq I + I J \) and \( I \cup J \subseteq I + I J \)

**proof** -

- from assms have \( I \cup J \subseteq R \) using ideal_dest_subset
  - by auto
- with assms show \( I \subseteq I + I J \) \( J \subseteq I + I J \) \( I \cup J \subseteq I + I J \)
  - using generated_ideal_contains_set sumIdeal_def
  - by auto

**qed**

Every element in the arbitrary sum of ideals is generated by only a finite subset of those ideals.

**lemma** (in ring0) sum_ideals_finite_sum:

assumes \( J \subseteq I \) \( s \in (\oplus I J) \)

shows \( \exists T \in \text{FinPow}(J). s \in (\oplus I T) \)

**proof** -

- { assume \( \bigcup J = 0 \)
  - then have \( J \subseteq \{0\} \) by auto
  - with assms(2) have thesis
    - using subset_Finite nat_into_Finite
    - unfolding FinPow_def by blast
  }

moreover
\{ let P = ⋁t. ∃T∈FinPow(J). t∈(⊕I T) \\
assume ∪J≠0 
moreover from assms(1) have ∪J⊆R by auto 
moreover 
\{ fix y z q assume P(q) y∈R z∈R q∈(∪J)_I 
then obtain T where T∈FinPow(J) and q ∈ ⊕I T 
by auto 
from assms(1) <T∈FinPow(J) > have ∪T ⊆ R T⊆I 
unfolding FinPow_def by auto 
with <q∈⊕I T> · y∈R> <z∈R> have y·q·z ∈ ⊕I T 
using generated_ideal_is_ideal sumArbitraryIdeals_def 
unfolding Ideal_def by auto 
with <T∈FinPow(J) > have P(y·q·z) by auto 
\} hence ∀y∈R. ∀z∈R. ∀q∈(∪J)_I. P(q) → P(y·q·z) 
by auto 
moreover 
\{ fix y z assume P(y) P(z) 
then obtain T_y T_z where T: T_y∈FinPow(J) y ∈ ⊕I T_y 
 T_z∈FinPow(J) z ∈ ⊕I T_z by auto 
from T(1,3) have A: T_y∪T_z ∈ FinPow(J) 
unfolding FinPow_def using Finite_Un by auto 
with assms(1) have a: T_y∪T_z ∈ FinPow(J) 
sub: ∪(T_y∪T_z) ⊆ R 
unfolding FinPow_def by auto 
then have ∪(T_y∪T_z) ⊆ (∪(T_y∪T_z))_I 
using generated_ideal_contains_set by simp 
hence ∪T_y ⊆ (∪(T_y∪T_z))_I T_z ⊆ (∪(T_y∪T_z))_I by auto 
with sub have (∪T_y)_I ⊆ (∪(T_y∪T_z))_I T_z ⊆ (∪(T_y∪T_z))_I 
using generated_ideal_small generated_ideal_is_ideal 
by auto 
moreover from assms(1) T(1,3) have T_y⊆I T_z⊆I 
unfolding FinPow_def by auto 
moreover note T(2,4) 
ultimately have y ∈ (∪(T_y∪T_z))_I z ∈ (∪(T_y∪T_z))_I 
using sumArbitraryIdeals_def sumArbitraryIdeals_def 
by auto 
with <∪(T_y∪T_z) ⊆ R> have y+z ∈ (∪(T_y∪T_z))_I 
using generated_ideal_is_ideal ideal_dest_sum by auto 
moreover 
from <T_y⊆I> <T_z⊆I> have T_y∪T_z ⊆ I by auto 
then have (⊕I (T_y∪T_z)) = (∪(T_y∪T_z))_I 
using sumArbitraryIdeals_def by auto 
ultimately have y+z ∈ (⊕I (T_y∪T_z)) by simp 
with A have P(y+z) by auto 
\} hence ∀y∈R. ∀z∈R. P(y) ∧ P(z) → P(y+z) 
by auto 
moreover 
\{ fix t assume t∈∪J 
then obtain J where t∈J J∈J by auto 
\}
then have \( \{J\} \in \text{FinPow}(J) \) unfolding \( \text{FinPow_def} \)
using \( \text{eqpoll_imp_Finite_iff} \) \( \text{nat_into_Finite} \)
by auto

moreover from \( \text{assms(1)} \) \( J \notin J \)
using \( \text{sumArbitraryIdeals_def} \)
by auto

with \( \text{assms(1)} \) \( t \in J \), \( J \in J \)
have \( t \in \bigoplus I \) \( \{J\} \)
by \( \text{force} \)

ultimately have \( P(t) \)
by auto

hence \( \forall t \in \bigcup J. P(t) \)
by auto

ultimately have \( \forall t \in \bigcup J. P(t) \)
by \( \text{rule induction_generated_ideal} \)

with \( \text{assms} \)
have \( \text{thesis} \)
using \( \text{sumArbitraryIdeals_def} \)
by auto

qed

By definition of product of ideals and of an ideal itself, it follows that the product of ideals is an ideal contained in the intersection

\textbf{theorem (in ring0) product_in_intersection:}

assumes \( I \triangleleft R, J \triangleleft R \)
shows \( I \cdot J \subseteq I \cap J \) and \( (I \cdot J) \triangleleft R \) and \( M(I \times J) \subseteq I \cdot J \)

\textbf{proof -}

have \( M(I \times J) \subseteq I \cap J \)

by \( \text{auto} \)

with \( \text{assms} \)
show \( \text{thesis} \)

using \( \text{sumArbitraryIdeals_def} \)
by \( \text{auto} \)

qed

We will show now that the sum of ideals is no more that the sum of the ideal elements.

\textbf{lemma (in ring0) sum_elements:}

556
assumes $I \triangleleft R \quad J \triangleleft R \quad x \in I \quad y \in J$
shows $x + y \in I + J$

proof -
from asssms(1,2) have $I \cup J \subseteq R$ using ideal_dest_subset by auto

with asssms(3,4) have $x \in (I \cup J)_I \quad y \in (I \cup J)_J$
using generated_ideal_contains_set by auto

with asssms(1,2) $<I \cup J \subseteq R>$ show thesis
using generated_ideal_is_ideal ideal_dest_subset ideal_dest_sum sumIdeal_def by auto

qed

For two ideals the set containing all sums of their elements is also an ideal.

lemma (in ring0) sum_elements_is_ideal:
assumes $I \triangleleft R \quad J \triangleleft R$
shows $(A(I \times J)) \triangleleft R$

proof -
from asssms have $i j: I \times J \subseteq R \times R$ using ideal_dest_subset by auto

have $\text{Aim}: A(I \times J) \subseteq R$ using add_group.group_oper_fun func1_1_L6(2) by auto
moreover { fix $x \quad y$
assume $x \in R \quad y \in A(I \times J)$
from $i j \quad \text{obtain} \quad y_i \quad y_j$
where $y: y = y_i + y_j \quad y_i \in I \quad y_j \in J$
using add_group.group_oper_fun func_imagedef by auto
from $x \in R \quad y \quad i \quad j$ have $x \cdot y = (x \cdot y_i) + (x \cdot y_j)$
using ring_oper_distr add_group.group_op_closed by auto
moreover from asssms $\text{y}(2,3)$ have $x \cdot y_i \in I \quad y_i \in I \quad x \cdot y_j \in J \quad y_j \in J$
using ideal_dest_mult by auto
ultimately have $x \cdot y \in A(I \times J) \quad y \cdot x \in A(I \times J)$
using $i j \quad \text{add_group.group_oper_fun func_imagedef by auto}$
} hence $\forall x \in A(I \times J). \quad \forall y \in R. \quad y \cdot x \in A(I \times J) \land x \cdot y \in A(I \times J)$
by auto

moreover have IsAsubgroup($A(I \times J), A$)

proof -
from asssms $i j$ have $0 + 0 \in A (I \times J)$
using add_group.group_oper_fun ideal_dest_zero func_imagedef by auto
with $\text{Aim}$ have $A(I \times J) \neq 0$ and $A(I \times J) \subseteq R$
by auto

moreover { fix $x \quad y$
assume $x y: x \in A(I \times J) \quad y \in A(I \times J)$
with $i j$ obtain $x_i \quad x_j \quad y_i \quad y_j$
where $x_i \in I \quad x_j \in I \quad x_i \cdot x_j \quad y_i \in J \quad y_j \in J \quad y_i + y_j$
using add_group.group_oper_fun func_imagedef by auto
from $x \in A(I \times J) \quad A(I \times J) \subseteq R \quad \text{have} \quad x \in R$
by auto
from asssms $<x_i \in I> \quad <x_j \in J> \quad <y_i \in J> \quad <y_j \in J>$

557
The set of all sums of elements of two ideals is their sum ideal i.e. the ideal generated by their union.

**corollary** (in ring0) sum_ideals_is_sum_elements:
assumes I <R J <R
shows (A(I × J)) = I + I J

**proof**
from assms have ij: J ⊆ R using ideal_dest_subset
by auto
then have ij_prd: I × J ⊆ R × R by auto
with assms show A(I × J) ⊆ I + J
using add_group.add_group0_3_T3
by simp

ultimately have x+y ∈ A(I × J)
using ij add_group.add_group0_3_T3
by auto

moreover have x+y = (x+y_i) + (x_j + y_j)
using Ring_ZF_1_L10(1) by simp

ultimately have (-x) ∈ A(I × J)
using add_group.add_group0_3_T3
by auto

ultimately show thesis using add_group.add_group0_3_T3
by simp

ultimately show (A(I × J)) <R unfolding Ideal_def by auto

qed
using add_group.group0_3_L5 add_group.group_oper_fun func_imagedef unfolding Ideal_def by auto 
} hence I ⊆ A(I × J) by auto moreover 
{ fix x assume x: x∈J 
   with ij(2) have x=0+x using Ring_ZF_1_L3(4) by auto 
   with assms(1) ij_prd J have x ∈ A(I × J) 
      using add_group.group0_3_L5 add_group.group_oper_fun func_imagedef unfolding Ideal_def by auto 
} hence J ⊆ A(I × J) by auto 
ultimately have I∪J ⊆ A(I × J) by auto 
with assms show I+J ⊆ A(I × J) 
using generated_ideal_small sum_elements_is_ideal sumIdeal_def by auto 
qed 

The sum ideal of two ideals is indeed an ideal.

corollary (in ring0) sum_ideals_is_ideal:
assumes I ◂ R J ◂ R shows (I+J) ◂ R using assms sum_ideals_is_sum_elements 
sum_elements_is_ideal ideal_dest_subset by auto 

The operation of taking the sum of ideals is commutative.

corollary (in ring0) sum_ideals_commute:
assumes I ◂ R J ◂ R shows (I + J) = (J + I) 
proof -
  have I ∪ J = J ∪ I by auto 
  with assms show thesis using sumIdeal_def by auto 
qed 

Now that we know what the product of ideals is, we are able to define what a prime ideal is:

definition (in ring0) primeIdeal (_ ◂ p R) where 

Any maximal ideal is a prime ideal.

theorem (in ring0) maximal_is_prime:
assumes Q ◂ m R shows Q ◂ p R 
proof -
  have Q∈I using assms ideal_dest_subset unfolding maximalIdeal_def by auto 
  unfolding unfolding UN_lemma unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfolding unfold

559
Definition (there is no proof yet).

For that we define the following concept of a prime ring. Note that in case of non-commutative rings, the zero divisor concept is too constrictive.

\[
\begin{align*}
\text{IsAring}(R, A, M) &\quad \equiv \\
&\quad (\forall x \in R. \forall y \in R. (\forall z \in R. M(x, z)y = \text{TheNeutralElement}(R, A)) \implies \\
x = \text{TheNeutralElement}(R, A) \lor y = \text{TheNeutralElement}(R, A))
\end{align*}
\]

In case of non-commutative rings, the zero divisor concept is too constrictive. For that we define the following concept of a prime ring. Note that in case that our ring is commutative, this is equivalent to having no zero divisors (there is no of that proof yet).

definition primeRing (\{is a prime ring\}) where
IsAring(R, A, M) \implies [R, A, M]{is a prime ring} \equiv \\
(\forall x \in R. \forall y \in R. (\forall z \in R. M(x, z)y = \text{TheNeutralElement}(R, A)) \implies \\
x = \text{TheNeutralElement}(R, A) \lor y = \text{TheNeutralElement}(R, A))
Prime rings appear when the zero ideal is prime.

lemma (in ring0) prime_ring_zero_prime_ideal:
  assumes \([R,A,M\} \text{ is a prime ring} \ R \neq \{0\}\)
  shows \(\{0\} \triangleleft \ p\ R\)
proof -
{  
  fix I J assume ij: \(I \in \mathcal{I} \ J \in \mathcal{I} \ I, J \subseteq \{0\}\)
  from ij(1,2) have \(I \times J \subseteq R \times R\) by auto
  {  
    assume \(\sim(I \subseteq \{0\}) \ \sim(J \subseteq \{0\})\)
    then obtain \(x_i, x_j\) where \(x_i \neq 0 \ x_j \neq 0 \ x_i \in I \ x_j \in J\)
    by auto
  from ij(1,2) \(x_i \in I \ x_j \in J\)
  {  
    fix u assume u\(\in R\)
    with \(I \subseteq \{0\}\) \(x_i \in I \ x_j \in J\)
    have \(x_i \in R \ x_j \in R\) by auto
  more over \{  
    fix s t q assume \(s \in R \ t \in R \ q \in \langle \{y\} \rangle\)
    from \(s \in R \ t \in R\) have \(s \times t \subseteq R \times R\)
    using generated_ideal_is_ideal ideal_dest_subset by auto
    let \(P = \lambda q. (\forall z \in Y. q \cdot z = 0)\)
    let \(Q = \lambda q. (\forall z \in R. x \cdot z \cdot q = 0)\)
    have \(\forall y \in Y. Q(y)\)
  proof -
    from \(y \in R\) have \(\{y\} \neq 0\) and \(\{y\} \subseteq R\) by auto
    moreover
    {  
      fix s t q assume yzq: \(s \in R \ t \in R \ q \in \langle \{y\} \rangle\)
      from \(s \in R \ t \in R\) yzq(3) have \(q \in R\)
      using generated_ideal_is_ideal ideal_dest_subset by auto
    qed
  qed
If the trivial ideal \(\{0\}\) is a prime ideal then the ring is a prime ring.

lemma (in ring0) zero_prime_ideal_prime_ring:
  assumes \(\{0\} \triangleleft \ p\ R\)
  shows \([R,A,M\} \text{ is a prime ring}\)
proof -
{  
  fix x y z assume \(x \in R \ y \in R \ \forall z \in R. x \cdot z \cdot y = 0\)
  let \(X = \langle \{x\} \rangle\)
  let \(Y = \langle \{y\} \rangle\)
  from \(x \in R \ y \in R\) have \(X \times Y \subseteq R \times R\)
  using generated_ideal_is_ideal ideal_dest_subset by auto
  let \(P = \lambda q. (\forall z \in Y. q \cdot z \cdot y = 0)\)
  let \(Q = \lambda q. (\forall z \in R. x \cdot z \cdot q = 0)\)
  have \(\forall y \in Y. Q(y)\)
  proof -
    from \(y \in R\) have \(\{y\} \neq 0\) and \(\{y\} \subseteq R\) by auto
  moreover
    {  
      fix s t q assume yzq: \(s \in R \ t \in R \ q \in \langle \{y\} \rangle\)
      from \(s \in R \ t \in R\) yzq(3) have \(q \in R\)
      using generated_ideal_is_ideal ideal_dest_subset by auto
    }
561
\{\text{fix } u \text{ assume } u \in \mathbb{R} \\
\text{from } \text{yzq}(1,2) \quad \langle x \in \mathbb{R} \rangle \quad \langle q \in \mathbb{R} \rangle \quad \langle u \in \mathbb{R} \rangle \\
\text{have } x \cdot (s \cdot q) = (x \cdot u) \cdot q \cdot t \\\n\text{using } \text{RingZF1L11}(2) \quad \text{RingZF1L4}(3) \text{ by auto} \\
\text{with } \langle u \in \mathbb{R} \rangle \quad \text{yzq}(1,2,4) \text{ have } x \cdot (s \cdot q) = 0 \\
\text{using } \text{RingZF1L4}(3) \quad \text{RingZF1L6}(1) \text{ by simp} \}
\text{hence } \forall z \in \mathbb{R}. \quad x \cdot z \cdot (s \cdot q) = 0 \text{ by simp} \\
\text{moreover } \forall z \in \mathbb{R}. \quad x \cdot z \cdot q = 0 \quad \langle \forall y \in \mathbb{R}. \quad x \cdot y = 0 \rangle \\
\text{using } \text{RingZF1L2}(2) \quad \text{RingZF1L3}(5) \text{ by force} \\
\text{from } \forall y \in \mathbb{R}. \quad y \cdot Y \subseteq \mathbb{R} \\
\text{using } \text{generatedIdealIsIdealIdealDestSubset} \text{ by auto} \\
\text{have } \forall y \in \mathbb{X}. \quad P(y) \\
\text{proof} \\
\text{from } \langle x \in \mathbb{R} \rangle \text{ have } \{x\} \neq 0 \quad \langle x \in \mathbb{R} \rangle \text{ by auto} \\
\text{moreover } \\
\{\text{fix } q_1, q_2, q_3 \\
\text{assume } q: q_1 \in \mathbb{R} \quad q_2 \in \mathbb{R} \quad q_3 \in \{x\} \quad \forall k \in \mathbb{Y}. \quad q_1 \cdot k = 0 \\
\text{from } \langle x \in \mathbb{R} \rangle \quad \langle q_1 \in \{x\} \rangle \text{ have } q_3 \in \mathbb{R} \\
\text{using } \text{generatedIdealIsIdealIdealDestSubset} \text{ by auto} \\
\text{with } \langle y \in \{x\} \rangle \quad \langle q \in \mathbb{R} \rangle \quad \text{yzq}(1,2,4) \text{ have } \forall z \in \{y\} \text{, } q_1 \cdot q_2 \cdot z \cdot z = 0 \\
\text{using } \text{idealDestMult}(2) \quad \text{RingZF1L4}(3) \quad \text{RingZF1L1L1}(2) \\
\text{RingZF1L6}(2) \text{ by auto} \}
\text{hence } \forall y \in \mathbb{R}. \quad \forall z \in \mathbb{R}. \quad q = 0 \quad \langle \forall y \in \mathbb{X}. \quad y = 0 \rangle \\
\text{by auto} \\
\text{moreover } \forall y \in \mathbb{R}. \quad \forall z \in \mathbb{R}. \quad (\forall z \in \{y\}) \quad y \cdot z = 0 \quad \langle \forall z \in \{y\} \rangle \\
\text{by auto} \\
\text{moreover } \forall y \in \mathbb{Y}. \quad x \cdot y = 0 \\
\text{by auto} \\
\text{ultimately show thesis by } \langle \text{rule induction_generated_ideal} \rangle \\
\text{qed} \\
\text{from } \langle x \cdot y \subseteq \mathbb{R} \rangle \quad \langle y \in \mathbb{X}. \quad P(y) \rangle \text{ have } M(x \cdot y) \subseteq \{0\} \\
\text{using } \text{mult_monoid.monoidOperFunFuncImagedef} \text{ by auto} \\
\text{with } \langle x \cdot y \subseteq \mathbb{R} \rangle \quad \langle y \in \mathbb{R} \rangle \text{ have } X \cdot Y \subseteq \{0\} \\
\text{using } \text{generatedIdealSmallZeroIdealProductIdealDef}
proof -
{ fix x y assume x∈R y∈R have X ⊆ {0} ∨ Y ⊆ {0}
  using generated_ideal_is_ideal ideal_dest_subset
  unfolding primeIdeal_def by auto
  with \langle x∈R, y∈R \rangle have x=0 ∨ y=0
  using generated_ideal_contains_set by auto
} hence ∀x∈R. ∀y∈R. (∀z∈R. x·z·y = 0) --- x=0 ∨ y=0
by auto
with ringAssum show thesis using primeRing_def by auto
qed

We can actually use this definition of a prime ring as a condition to check
for prime ideals.

theorem (in ring0) equivalent_prime_ideal:
  assumes Pq,R
  shows ∀x∈R. ∀y∈R. (∀z∈R. x·z·y∈P) --- x∈P ∨ y∈P
proof -
{ fix x y assume x∈R y∈R ∀z∈R. x·z·y∉P
  let X = \{x\}
  let Y = \{y\}
  from \langle y∈R \rangle have \{y\}≠0 and \{y\}⊆R by auto
  moreover have ∀y_q∈R.
    ∀z∈R. ∀q∈\{y\} \langle ∀t∈\{x\}, ∀u∈R. t·u·q∈P \langle ∀t∈\{x\}, ∀u∈R. t·u·(y_q)∈P
  proof -
  { fix y_q z t u
    assume y_q∈R z∈R q∈Y ∀t∈X. ∀u∈R. t·u·q∈P t∈X u∈R
    from \langle x∈R, y∈R, q∈Y \rangle \langle t∈X \rangle have q∈R t∈R
    using generated_ideal_is_ideal ideal_dest_subset by auto
    from \langle y∈R \rangle q∈R· have u·y_q∈R using Ring_ZF_1_L4(3) by auto
    with asss \langle x∈R \rangle \langle t∈X \rangle have t·u·q∈P \langle t∈X \rangle
    have (t·(u·y_q))·z ∈ P
    using ideal_dest_mult(1) unfolding primeIdeal_def by auto
    with \langle y_q∈R \rangle \langle z∈R \rangle \langle q∈R \rangle \langle t∈R \rangle have t·u·(y_q)·z ∈ P
    using Ring_ZF_1_L4(3) Ring_ZF_1_L11(2) by auto
  } thus thesis by simp
qed

moreover have ∀y∈R. ∀z∈R.
  (∀t∈\{x\}, ∀u∈R. t·u·y ∈ P) ∧ (∀t∈\{x\}, ∀u∈R. t·u·z ∈ P) --- (∀t∈\{x\}, ∀u∈R. t·u·(y+z) ∈ P)
proof -
{ fix y z t u assume ass: y_q∈R z∈R t∈X u∈R
  ∀t∈\{x\}, ∀u∈R. t·u·y_q∈P ∀t∈\{x\}, ∀u∈R. t·u·z∈P
  from \langle x∈R \rangle \langle t∈X \rangle have t∈R
  using ideal_dest_subset generated_ideal_is_ideal by auto
  from asss ass(3,4,5,6) have (t·u·y_q)+(t·u·z) ∈ P
  using ideal_dest_sum unfolding primeIdeal_def by simp
  with \langle t∈R \rangle \langle y∈R \rangle \langle z∈R \rangle \langle u∈R \rangle have t·u·(y_q+z) ∈ P
  using Ring_ZF_1_L4(3) ring_oper_distr(1) by simp

563
moreover have \( \forall y_a \in \{x\} \). \( \forall u \in R. \ y_a \cdot u \cdot y \in P \)

proof -
from \( \langle x \in R \rangle \) have \( \langle x \rangle \neq 0 \) and \( \langle x \rangle \subseteq R \) by auto
moreover have \( \forall y_a \in R. \forall z \in R. \forall q \cdot x \)
\( (\forall u \in R. \ q \cdot u \cdot y \in P) \rightleftarrows (\forall u \in R. \ y_a \cdot z \cdot u \cdot y \in P) \)

proof -
\{ fix \( y, z \) assumes \( y \in R. \) \( z \in R \)
from \( \langle x \in R \rangle \) \( \langle q \in X \rangle \) have \( q \in R \)
using generated_ideal_is_ideal ideal_dest_subset by auto
from assms \( \langle x \in R \rangle \) \( \langle q \in X \rangle \) \( \langle r \in R \rangle \) \( \langle u \in R \rangle \)
have \( y_a \cdot q \cdot z \cdot u \cdot y \in P \)
using Ring_ZF_1_L4(3) ideal_dest_mult(2)
unfolding primeIdeal_def by simp
\}
thus thesis by simp

qed

moreover have \( \forall y_a \in R. \forall z \in R. \)
\( (\forall u \in R. \ y_a \cdot u \cdot y \in P) \land (\forall u \in R. \ z \cdot u \cdot y \in P) \rightleftarrows (\forall u \in R. \ (y_a + z) \cdot u \cdot y \in P) \)

proof -
\{ fix \( y, z \) assumes \( y \in R. \) \( z \in R \)
\( \forall u \in R. \ y_a \cdot u \cdot y \in P \land \forall u \in R. \ z \cdot u \cdot y \in P \)
with assms \( \langle y \rangle \) \( \langle z \rangle \) \( \langle u \rangle \) \( \langle r \rangle \) \( \langle q \rangle \)
have \( ((y_a + z) \cdot u \cdot y) \in P \)
using ideal_dest_sum ring_oper_distr(2) Ring_ZF_1_L4(3)
unfolding primeIdeal_def by simp
\}
thus thesis by simp

qed

moreover from \( \langle \forall z \in R. \ x \cdot z \cdot y \in P \rangle \) have \( \forall x \in \{x\} \).
\( \forall u \in R. \ x \cdot u \cdot y \in P \)

ultimately show thesis by (rule induction_generated_ideal)

qed

hence \( \forall x_0 \in \{y\} \).
\( \forall t \in \{\langle x \rangle \} \).
\( \forall u \in R. \ t \cdot u \cdot x_0 \in P \) by auto
ultimately have \( R : \forall q \in Y. \forall t \in X. \forall u \in R. \ t \cdot u \cdot q \in P \)
by (rule induction_generated_ideal)
from \( \langle x \in R \rangle \) \( \langle y \in R \rangle \) have subprd: \( X \times Y \subseteq R \times R \)
using ideal_dest_subset generated_ideal_is_ideal by auto
\{ fix \( t \) assumes \( t \in M(X \times Y) \)
with subprd obtain \( t_x \) \( t_y \) where
\( t_x \in X \) \( t_y \in Y \)
and \( t = t_x \cdot t_y \)
using func_imagedef mult_monoid.monoid_oper_fun by auto
with \( R : t_x \in X \) \( t_y \in Y \) have \( t_x \cdot 1 \cdot t_y \in P \)
using Ring_ZF_1_L2(2) by auto
moreover from \( \langle x \in R \rangle \) \( \langle t_x \in X \rangle \) have \( t_x \in R \)
\}

564
ultimately have \( t \in P \) using \( \text{RingZF1L3(5)} \) \( t \) by auto

\[
\text{hence } M(X \times Y) \subseteq P \text{ by auto}
\]

with assms \( \langle x \in R \rangle \langle y \in R \rangle \) have \( X \subseteq P \lor Y \subseteq P \)

unfolding primeIdeal_def

using generated_ideal_small productIdeal_def

generated_ideal_is_ideal ideal_dest_subset by auto

with \( \langle x \in R \rangle \langle y \in R \rangle \langle y \notin P \rangle \) have \( x \in P \)

by blast

thus thesis unfolding primeIdeal_def by auto

qed

The next theorem provides a sufficient condition for a proper ideal \( P \) to be a prime ideal: if for all \( x, y \in R \) it holds that for all \( z \in R \) \( x z y \in P \) only when \( x \in P \) or \( y \in P \) then \( P \) is a prime ideal.

**Theorem (in ring0) equivalent_prime_ideal_2:**

assumes \( \forall x \in R. \forall y \in R. (\forall z \in R. x z y \in P) \rightarrow x \in P \lor y \in P \)

shows \( P \subseteq P \neq R \)

**Proof:**

\[
\begin{align*}
\text{fix } I & \times J \times x_n \\
\text{assume } I & \subseteq R \text{ J} \subseteq R \text{ I} = J \subseteq P \text{ x} \in J \text{ x} \notin P \text{ x} \in I \\
\text{from } I & \subseteq R \text{ J} \subseteq R \text{ have } I \times J \subseteq R \times R \text{ by auto} \\
\text{fix } z & \text{ assume } z \in R \\
\text{with } \langle I \subseteq R \rangle \langle J \subseteq R \rangle \langle I = J \subseteq P \rangle \langle I \times J \subseteq R \times R \rangle \text{ have } (x_n, z) \in M(I \times J) \\
\text{using ideal_dest_mult(1)} \text{ func_imagedef mult_monoid.monoid_oper_fun} \\
\text{by auto} \\
\text{with } \langle I \subseteq R \rangle \langle J \subseteq R \rangle \langle I = J \subseteq P \rangle \text{ have } x_n z x \in P \\
\text{using generated_ideal_contains_set func1_1_L6(2)} \\
\text{mult_monoid.monoid_oper_fun productIdeal_def by force} \\
\text{by blast} \\
\text{with assms(1) } \langle I \subseteq R \rangle \langle J \subseteq R \rangle \langle x \in J \rangle \langle y \notin P \rangle \langle x_n \in I \rangle \text{ have } x_n \in P \\
\text{by blast} \\
\text{with assms(2,3) show thesis unfolding primeIdeal_def} \\
\text{by auto} \\
\text{qed}
\end{align*}
\]

**47.2 Ring quotient**

Similar to groups, rings can be quotiented by normal additive subgroups; but to keep the structure of the multiplicative monoid we need extra structure in the normal subgroup. This extra structure is given by the ideal.

Any ideal is a normal subgroup.

**Lemma (in ring0) ideal_normal_add_subgroup:**

assumes \( I \subseteq R \)

shows IsAnormalSubgroup(R, A, I)

using ringAssum assms GroupZF2_4L6(1)
unfolding \text{IsAring_def Ideal_def} by \text{auto}

Each ring \( R \) is a group with respect to its addition operation. By the lemma \text{ideal_normal_add_subgroup} above an ideal \( I \subseteq R \) is a normal subgroup of that group. Therefore we can define the quotient of the ring \( R \) by the ideal \( I \) using the notion of quotient of a group by its normal subgroup, see section \text{Normal subgroups and quotient groups in GroupZF_2 theory}.

definition (in ring0) QuotientBy where
\( \langle I \triangleleft R \rightarrow \text{QuotientBy}(I) \equiv R//\text{QuotientGroupRel}(R,A,I) \rangle \)

Any ideal gives rise to an equivalence relation

corollary (in ring0) ideal_equiv_rel:
\( \text{assumes } I \triangleleft R \text{ shows } \text{equiv}(R,\text{QuotientGroupRel}(R,A,I)) \)
using \text{assms add_group.GroupZF_2_4_L3 unfolding Ideal_def by auto}

Any quotient by an ideal is an abelian group.

lemma (in ring0) quotientBy_add_group:
\( \text{assumes } I \triangleleft R \text{ shows } \text{IsAgroup}(\text{QuotientBy}(I), \text{QuotientGroupOp}(R,A,I)) \) and
\( \text{QuotientGroupOp}(R,A,I) \text{ is commutative on } \text{QuotientBy}(I) \)
using \text{assms ringAssum ideal_normal_add_subgroup GroupZF_2_4_T1 GroupZF_2_4_L6(2) QuotientBy_def QuotientBy_def Ideal_def unfolding IsAring_def by auto}

Since every ideal is a normal subgroup of the additive group of the ring it is quite obvious that that addition is congruent with respect to the quotient group relation. The next lemma shows something a little bit less obvious: that the multiplicative ring operation is also congruent with the quotient relation and gives rise to a monoid in the quotient.

lemma (in ring0) quotientBy_mul_monoid:
\( \text{assumes } I \triangleleft R \text{ shows } \text{Congruent2}(\text{QuotientGroupRel}(R,A,I),M) \) and
\( \text{IsAmonoid}(\text{QuotientBy}(I),\text{ProjFun2}(R,\text{QuotientGroupRel}(R,A,I),M)) \)
proof -
\( \text{let } r = \text{QuotientGroupRel}(R,A,I) \)
\( \{ \text{fix } x \ y \ s \ t \text{ assume } (x,y) \in r \text{ and } (s,t) \in r \text{ then have } \text{yyst: } x \in R \text{ y} \in R \text{ s} \in R \text{ t} \in R \text{ x-y} \in I \text{ s-t} \in I \text{ unfolding QuotientGroupRel_def by auto} \}
\text{from } \langle x \in R \rangle \langle y \in R \rangle \langle s \in R \rangle \langle t \in R \rangle \text{ have }
\langle x-s)\rangle(y-t) = (((x-s)+((-y\cdot s)) \cdot (y-s))\rangle(-y-t)
\text{using RingZF_1_L3(1,3,7) RingZF_1_L4(3,4) by simp}
\text{with } \langle x \in R \rangle \langle y \in R \rangle \langle s \in R \rangle \langle t \in R \rangle \text{ have }
\langle x-s\rangle(y-t) = (((x-s)\langle(y-s)) \cdot ((y-s)-y-t))
\text{using RingZF_1_L3(1) RingZF_1_L4(1,2,3) RingZF_1_L10(1) by simp} \)
with \( x \in \mathbb{R} \), \( y \in \mathbb{R} \), \( s \in \mathbb{R} \), \( t \in \mathbb{R} \) have
\[
(x \cdot s) - (y \cdot t) = ((x - y) \cdot s) + (y \cdot (s - t))
\]
using Ring_ZF_1_L8 by simp
with assms xyst have \( \langle M(x,s), M(y,t) \rangle \in r \)
using ideal_dest_sum ideal_dest_mult(1,2) Ring_ZF_1_L4(3)
unfolding QuotientGroupRel_def by auto
} then show Congruent2(r,M) unfolding Congruent2_def by simp
with assms show IsAmonoid(QuotientBy(I), ProjFun2(R,r, M))
using add_group.Group_ZF_2_4_L3 mult_monoid.Group_ZF_2_2_T1 QuotientBy_def
unfolding Ideal_def by auto
qed

Each ideal defines an equivalence relation on the ring with which both addition and multiplication are congruent. The next couple of definitions set up notation for the operations that result from projecting the ring addition and multiplication on the quotient space. We will write \( x + I \) \( y \) to denote the result of the quotient operation (with respect to an ideal \( I \)) on classes \( x \) and \( y \)

definition (in ring0) ideal_radd (_{+_}_) where
\[
x{+I} y \equiv \text{QuotientGroupOp}(R, A, I)(x,y)
\]
Similarly \( x \cdot I \) \( y \) is the value of the projection of the ring’s multiplication on the quotient space defined by the an ideal \( I \), which as we know is a normal subgroup of the ring with addition.

definition (in ring0) ideal_rmult (_{\cdot}_) where
\[
x{\cdot I} y \equiv \text{ProjFun2}(R, \text{QuotientGroupRel}(R,A,I), M)(x,y)
\]
The value of the projection of taking the negative in the ring on the quotient space defined by an ideal \( I \) will be denoted \(-I\).

definition (in ring0) ideal_rmin ({-_}_) where
\[
{-I} y \equiv \text{GroupInv}(\text{QuotientBy}(I), \text{QuotientGroupOp}(R, A, I))(y)
\]
Subtraction in the quotient space is defined by the \(+I\) and \(-I\) operations in the obvious way.

definition (in ring0) ideal_rsub (_{-_}_) where
\[
x{-I} y \equiv x{+I}({-I} y)
\]
The class of the zero of the ring with respect to the equivalence relation defined by an ideal \( I \) will be denoted \( 0_I \).

definition (in ring0) ideal_rzero (0 _) where
\[
0_I \equiv \text{QuotientGroupRel}(R, A, I)\{0\}
\]
Similarly the class of the neutral element of multiplication in the ring with respect to the equivalence relation defined by an ideal \( I \) will be denoted \( 1_I \).

definition (in ring0) ideal_rone (1 _) where

567
\[ 1_I \equiv \text{QuotientGroupRel}(R,A,I)\{1\} \]

The class of the sum of two units of the ring will be denoted 2_I.

definition (in ring0) ideal_rtwo (\_\_\_) where
\[ 2_I \equiv \text{QuotientGroupRel}(R,A,I)\{2\} \]

The value of the projection of the ring multiplication onto the the quotient space defined by an ideal I on a pair of the same classes \( \langle x,x \rangle \) is denoted \( x^{2_I} \).

definition (in ring0) ideal_rsqr (\_\_\_) where
\[ x^{2_I} \equiv \text{ProjFun2}(R, \text{QuotientGroupRel}(R,A,I), M)(x,x) \]

The class of the additive neutral element of the ring (i.e. 0) with respect to the equivalence relation defined by an ideal is the neutral of the projected addition.

lemma (in ring0) neutral_quotient:
assumes I:R
shows QuotientGroupRel(R,A,I)\{0\} = TheNeutralElement(QuotientBy(I),QuotientGroupOp(R,A,I))
using ringAssum assms Group_ZF_2_4_L5B ideal_normal_add_subgroup QuotientBy_def unfolding IsAring_def by auto

Similarly, the class of the multiplicative neutral element of the ring (i.e. 1) with respect to the equivalence relation defined by an ideal is the neutral of the projected multiplication.

lemma (in ring0) one_quotient:
assumes I:R
defines r \equiv QuotientGroupRel(R,A,I)
shows r\{1\} = TheNeutralElement(QuotientBy(I),ProjFun2(R,r,M))
using ringAssum assms Group_ZF_2_2_L1 ideal_equiv_rel quotientBy_mul_monoid QuotientBy_def unfolding IsAring_def by auto

The class of 2 (i.e. 1 + 1) is the same as the value of the addition projected on the quotient space on the pair of classes of 1.

lemma (in ring0) two_quotient:
assumes I:R
defines r \equiv QuotientGroupRel(R,A,I)
shows r\{2\} = QuotientGroupOp(R,A,I)\{r\{1\},r\{1\}\}
using ringAssum assms EquivClass_1_L10 ideal_equiv_rel Ring_ZF_1_L10 ideal_normal_add_subgroup QuotientBy_def unfolding IsAring_def QuotientGroupOp_def by simp

The class of a square of an element of the ring is the same as the result of the projected multiplication on the pair of classes of the element.

lemma (in ring0) sqrt_quotient:
assumes $I \triangleleft R \forall x \in R$
defines $r \equiv \text{QuotientGroupRel}(R, A, I)$
shows $r\{x^2\} = \text{ProjFun2}(R, r, M)(r\{x\}, r\{x\})$
using assms $\text{EquivClass}_1.L10$ $\text{ideal_equiv_rel}$ $\text{quotientBy_mul_monoid}(1)$
by auto

The projection of the ring addition is distributive with respect to the projection of the ring multiplication.

lemma (in $\text{ring0}$) quotientBy_distributive:
assumes $I \triangleleft R$
defines $r \equiv \text{QuotientGroupRel}(R, A, I)$
shows $\text{IsDistributive}(\text{QuotientBy}(I), \text{QuotientGroupOp}(R, A, I), \text{ProjFun2}(R, r, M))$
using ringAssum assms $\text{QuotientBy}_\text{def}$ $\text{ring_oper_distr}(1)$ $\text{Ring_ZF}_1.L4(1,3)$
$\text{quotientBy_mul_monoid}(1)$ $\text{EquivClass}_1.L10$ $\text{ideal_equiv_rel}$ $\text{Group_ZF}_2.4.L5A$
$\text{ideal_normal_add_subgroup}$ $\text{ring_oper_distr}(2)$
unfolding quotient_def $\text{QuotientGroupOp}_\text{def}$ $\text{IsAring}_\text{def}$ $\text{IsDistributive}_\text{def}$
by auto

The quotient group is a ring with the quotient multiplication.

theorem (in $\text{ring0}$) quotientBy_is_ring:
assumes $I \triangleleft R$
defines $r \equiv \text{QuotientGroupRel}(R, A, I)$
shows $\text{IsAring}(\text{QuotientBy}(I), \text{QuotientGroupOp}(R, A, I), \text{ProjFun2}(R, r, M))$
using assms quotientBy_distributive $\text{quotientBy_mul_monoid}(2)$ $\text{quotientBy_add_group}$
unfolding $\text{IsAring}_\text{def}$
by auto

An important property satisfied by many important rings is being Noetherian: every ideal is finitely generated.

definition (in $\text{ring0}$) isFinGen (\_(is finitely generated)) where
$I \triangleleft R \rightarrow I \{\text{is finitely generated}\} \equiv \exists S \in \text{FinPow}(R). I = \langle S \rangle$

For Noetherian rings the arbitrary sum can be reduced to the sum of a finite subset of the initial set of ideals

theorem (in $\text{ring0}$) sum_ideals_noetherian:
assumes $\forall I \in \mathcal{I}. \{\text{is finitely generated}\} \ J \subseteq \mathcal{I}$
shows $\exists T \in \text{FinPow}(\mathcal{J}). \langle \oplus_\mathcal{I} \cap \mathcal{J} \rangle = \langle \oplus_\mathcal{T} \rangle$
proof -
from assms (2) have $\bigcup \mathcal{J} \subseteq R$ using ideal_dest_subset by auto
then have $(\bigcup \mathcal{J}) \triangleleft R$ using generated_ideal_is_ideal by simp
with assms (2) have $(\oplus_\mathcal{I} \cap \mathcal{J}) \triangleleft R$ using sumArbitraryIdeals_def by simp
with assms (1) have $(\oplus_\mathcal{I} \cap \mathcal{J}) \{\text{is finitely generated}\}$
using ideal_dest_subset by auto
with \((\oplus_I J) \triangleq R\) obtain \(S\) where \(S \in \text{FinPow}(R)\) \((\oplus_I J) = \langle S \rangle_I\)
using isFinGen_def by auto
let \(N = \lambda s \in S. \{ J \in \text{FinPow}(J). s \in (\oplus_I J) \}\)
from \(S \in \text{FinPow}(R)\) have Finite(S) unfolding FinPow_def by auto
then have \((\forall t \in S. N(t) \neq 0) \rightarrow (\exists f. f \in (\prod t \in S. N(t)) \land (\forall t \in S. f(t) \in N(t)))\)
using eqpoll_iff finite_choice AxiomCardinalChoiceGen_def unfolding Finite_def by blast
moreover
\{ fix \(t\) assume \(t \in S\)
then have \(N(t) = \{ J \in \text{FinPow}(J). t \in (\oplus_I J) \}\)
using lamE by auto
moreover
from \(S \in \text{FinPow}(R)\) \((\oplus_I J) = \langle S \rangle_I\) \(t \in S\) have \(t \in (\oplus_I J)\)
using generated_ideal_contains_set unfolding FinPow_def by auto
with assms(2) have \(\exists T \in \text{FinPow}(J). t \in (\oplus_I T)\)
using sum_ideals_finite_sum by simp
ultimately have \(N(t) \neq 0\) using assms(2) sum_ideals_finite_sum by auto
\}
ultimately obtain \(f\) where
\(f: f \in (\prod t \in S. N(t)) \forall t \in S. f(t) \in N(t)\)
by auto
\{ fix \(y\) assume \(y \in f(S)\)
with image_iff obtain \(x \in S\) where \(x \in S\) and \(\langle x, y \rangle \in f\) by auto
with \(f(1)\) have \(y \in N(x)\) unfolding Pi_def by simp
\}
moreover
from \(f(1)\) have \(f_Fun: f : S \rightarrow (\bigcup \{ N(t) \cdot t \in S \})\)
unfolding Pi_def Sigma_def by blast
with \(\text{Finite}(S)\) have \(\text{Finite}(f(S))\)
using Finite1_L6A Finite_Fin_iff Fin_into_Finite by blast
ultimately have \(\text{Finite}(\bigcup \{ f(S) \})\) using Finite_Union by auto
with \(f_Fun\) \(f(2)\) have \(B : (\bigcup \{ f(S) \}) \in \text{FinPow}(J)\)
using func_imagedef lamE unfolding FinPow_def by auto
then have \((\bigcup \{ f(S) \}) \subseteq J\) unfolding FinPow_def by auto
with assms(2) have \((\bigcup \{ f(S) \}) \subseteq I\) by auto
hence \(\text{sub}: (\bigcup \{ f(S) \}) \subseteq R\) by auto
\{ fix \(t\) assume \(t \in S\)
with \(f(2)\) have \(f(t) \in \text{FinPow}(J)\) \(t \in (\oplus_I (f(t)))\)
using lamE by auto
from assms(2) \(f(t) \in \text{FinPow}(J)\) have \(f(t) \subseteq I\)
unfolding FinPow_def by auto
from \(f_Fun\) \((t \in S)\) have \(f(t) \subseteq \bigcup (\bigcup \{ f(S) \})\) using func_imagedef by auto
This section studies the ideals of quotient rings, and defines ring homomorphisms.

### 48.1 Ring homomorphisms

Morphisms in general are structure preserving functions between algebraic structures. In this section we study ring homomorphisms.

A ring homomorphism is a function between rings which has the morphism property with respect to both addition and multiplication operation, and maps one (the neutral element of multiplication) in the first ring to one in the second ring.

**definition**

\[
\text{ringHomomor}(f,R,A,M,S,U,V) \equiv f : R \to S \land \text{IsMorphism}(R,A,U,f) \land \text{IsMorphism}(R,M,V,f) \\
\land f(\text{TheNeutralElement}(R,M)) = \text{TheNeutralElement}(S,V)
\]
The next locale defines notation which we will use in this theory. We assume that we have two rings, one (which we will call the origin ring) defined by the triple \((R, A, M)\) and the second one (which we will call the target ring) by the triple \((S, U, V)\), and a homomorphism \(f : R \to S\).

locale ring_homo = 
  fixes \(R\ A\ M\ S\ U\ V\ f\)
  assumes origin: IsAring\((R, A, M)\)
  and target: IsAring\((S, U, V)\)
  and homomorphism: ringHomomor\((f, R, A, M, S, U, V)\)

  fixes ringa (infixl \(+\) 90)
  defines ringa_def [simp]: \(x +_R y \equiv A\langle x, y \rangle\)

  fixes ringminus \((-\) 89)
  defines ringminus_def [simp]: \((-_R x) \equiv \text{GroupInv}(R, A)(x)\)

  fixes ringsub (infixl \(-\) 90)
  defines ringsub_def [simp]: \(x -_R y \equiv x +_R (-_R y)\)

  fixes ringm (infixl \(\cdot\) 95)
  defines ringm_def [simp]: \(x \cdot_R y \equiv M\langle x, y \rangle\)

  fixes ringzero (0_R)
  defines ringzero_def [simp]: \(0_R \equiv \text{TheNeutralElement}(R, A)\)

  fixes ringone (1_R)
  defines ringone_def [simp]: \(1_R \equiv \text{TheNeutralElement}(R, M)\)

  fixes ringtwo (2_R)
  defines ringtwo_def [simp]: \(2_R \equiv 1_R +_R 1_R\)

  fixes ringsq (_\(\cdot\) 96) 97)
  defines ringsq_def [simp]: \(x^2_R \equiv x \cdot_R x\)

  fixes ringas (infixl \(+_S\) 90)
  defines ringas_def [simp]: \(x +_S b \equiv U(x, b)\)

  fixes ringminuss \((-_S\) 89)
  defines ringminuss_def [simp]: \((-_S x) \equiv \text{GroupInv}(S, U)(x)\)

  fixes ringsubs (infixl \(-_S\) 90)
  defines ringsubs_def [simp]: \(x -_S b \equiv x +_S (-_S b)\)

  fixes ringms (infixl \(\cdot_S\) 95)
  defines ringms_def [simp]: \(x \cdot_S b \equiv V( x, b)\)

  fixes ringzeros (0_S)
  defines ringzeros_def [simp]: \(0_S \equiv \text{TheNeutralElement}(S, U)\)
We will write \( I \triangleleft R \) to denote that \( I \) is an ideal of the ring \( R \). Note that in this notation the \( R \) part by itself has no meaning, only the whole \( \triangleleft R \) serves as postfix operator.

**abbreviation** (in ring_homo) ideal_origin (_\( \triangleleft \)R_0)  
where \( I \triangleleft R_0 \equiv \text{ring0.Ideal}(R,A,M,I) \)

\( I \triangleleft R_0 \) means that \( I \) is an ideal of \( S \).

**abbreviation** (in ring_homo) ideal_target (_\( \triangleleft \)R_t)  
where \( I \triangleleft R_t \equiv \text{ring0.Ideal}(S,U,V,I) \)

\( I \triangleleft R_0 \) means that \( I \) is a prime ideal of \( R \).

**abbreviation** (in ring_homo) prime_ideal_origin (_\( \triangleleft \)pR_0)  
where \( I \triangleleft pR_0 \equiv \text{ring0.primeIdeal}(R,A,M,I) \)

We will write \( I \triangleleft pR_0 \) to denote that \( I \) is a prime ideal of the ring \( S \).

**abbreviation** (in ring_homo) prime_ideal_target (_\( \triangleleft \)pR_t)  
where \( I \triangleleft pR_t \equiv \text{ring0.primeIdeal}(S,U,V,I) \)

\( \ker \) denotes the kernel of \( f \) (which is assumed to be a homomorphism between \( R \) and \( S \)).

**abbreviation** (in ring_homo) kernel (ker 90) where  
\( \ker \equiv f^{-\{0_S\}} \)

The theorems proven in the ring0 context are valid in the ring_homo context when applied to the ring \( R \).

**sublocale** ring_homo < origin_ring:ring0  
using origin unfolding ring0_def by auto

The theorems proven in the ring0 context are valid in the ring_homo context when applied to the ring \( S \).

**sublocale** ring_homo < target_ring:ring0 S U V ringas ringminuss ringsubs ringms ringzeros ringones ringtwos ringsqs  
using target unfolding ring0_def by auto

A ring homomorphism is a homomorphism both with respect to addition and multiplication.

**lemma** ringHomHom: assumes ringHomomor(f,R,A,M,S,U,V)
shows $\text{Homomorf}(f,R,A,S,U)$ and $\text{Homomorf}(f,R,M,S,V)$
using assms unfolding $\text{ringHomomor_def}$ $\text{Homomor_def}$ by simp_all

Since in the ring_homo locale $f$ is a ring homomorphism it implies that $f$ is a function from $R$ to $S$.

lemma (in ring_homo) f_is_fun: shows $f:R \rightarrow S$
using homomorphism unfolding $\text{ringHomomor_def}$ by simp

In the ring_homo context $A$ is the addition in the first (source) ring $M$ is the multiplication there and $U,V$ are the addition and multiplication resp. in the second (target) ring. The next lemma states the all these are binary operations, a trivial, but frequently used fact.

lemma (in ring_homo) AMUV_are_ops: shows $A:R \times R \rightarrow R$ $M:R \times R \rightarrow R$ $U:S \times S \rightarrow S$ $V:S \times S \rightarrow S$
using origin target unfolding $\text{IsAring_def}$ $\text{IsAgroup_def}$ $\text{IsAmonoid_def}$ $\text{IsAssociative_def}$ by auto

The kernel is a subset of $R$ on which the value of $f$ is zero (of the target ring)

lemma (in ring_homo) kernel_def_alt: shows $\ker = \{ r \in R . f(r) = 0_S \}$
using f_is_fun func1_1_L14 by simp

the homomorphism $f$ maps each element of the kernel to zero of the target ring.

lemma (in ring_homo) image_kernel: assumes $x \in \ker$
shows $f(x) = 0_S$
using assms func1_1_L15 f_is_fun by simp

As a ring homomorphism $f$ preserves multiplication.

lemma (in ring_homo) homomor_dest_mult: assumes $x \in R$ $y \in R$
shows $f(x \cdot R y) = (f(x)) \cdot S (f(y))$
using assms origin target homomorphism unfolding $\text{ringHomomor_def}$ $\text{IsMorphism_def}$ by simp

As a ring homomorphism $f$ preserves addition.

lemma (in ring_homo) homomor_dest_add: assumes $x \in R$ $y \in R$
shows $f(x + R y) = (f(x)) + S (f(y))$
using homomor_eq origin target homomorphism assms unfolding $\text{IsAring_def}$ $\text{ringHomomor_def}$ $\text{IsMorphism_def}$ by simp

For $x \in R$ the value of $f$ is in $S$. 574
lemma (in ring_homo) homomor_val: assumes \( x \in R \)
shows \( f(x) \in S \)
using homomorphism assms apply_funtype unfolding ringHomomor_def
by blast

A ring homomorphism preserves taking negative of an element.

lemma (in ring_homo) homomor_dest_minus: 
asumes \( x \in R \)
shows \( f(-_R x) = -_S f(x) \)
using origin target homomorphism assms ringHomHom image_inv
unfolding IsAring_def
by auto

A ring homomorphism preserves substraction.

lemma (in ring_homo) homomor_dest_subs: 
asumes \( x \in R \) \( y \in R \)
shows \( f(x -_R y) = f(x) -_S f(y) \)
using assms homomor_dest_add homomor_dest_minus
using origin_ring.Ring_ZF_1_L3(1) by auto

A ring homomorphism maps zero to zero.

lemma (in ring_homo) homomor_dest_zero: 
shows \( f(0_R) = 0_S \)
using origin target homomorphism ringHomHom(1) image_neutral
unfolding IsAring_def by auto

The kernel of a homomorphism is never empty.

lemma (in ring_homo) kernel_non_empty: shows \( 0_R \in \ker \) and \( \ker \neq 0 \)
using homomor_dest_zero origin_ring.Ring_ZF_1_L2(1)
func1_1_L15 f_is_fun by auto

The image of the kernel by \( f \) is the singleton \( \{0_R\} \).

corollary (in ring_homo) image_kernel_2: shows \( f(\ker) = \{0_S\} \)
proof -
have \( f: R \to S \) \( \ker \subseteq R \) \( \ker \neq 0 \) \( \forall x \in \ker. \ f(x) = 0_S \)
using f_is_fun kernel_def_alt kernel_non_empty by auto
then show \( f(\ker) = \{0_S\} \) using image_constant_singleton by simp
qed

The inverse image of an ideal (in the target ring) is a normal subgroup of the addition group and an ideal in the origin ring. The kernel of the homomorphism is a subset of the inverse of image of every ideal.

lemma (in ring_homo) preimage_ideal: 
asumes \( J \triangleleft R \)
shows \( \IsAnormalSubgroup(R, A, f^{-}(J)) \)
Kernel of the homomorphism in an ideal.

lemma (in ring_homo) kernel_ideal: shows ker ⊆ f(J)
  using target_ring.Ideal_def by auto

The inverse image of a prime ideal by a homomorphism is not the whole ring. Proof by contradiction.

lemma (in ring_homo) vimage_prime_ideal_not_all:
  assumes J:p R t
  shows f(J) ≠ R
proof -
  { assume R = f(J)
    then have R = {x ∈ R. f(x) ∈ J} using f_is_fun func1_1_L15
      by simp
    then have 1_R ∈ {x ∈ R. f(x) ∈ J} using origin_ring.Ring_ZF_1_L2(2)
      by simp
    with origin_ringRINGASUM target_ring.RINGASUM homomorphism assms
    have False using target_ring.Ideal_with_one
      unfolding target_ring.primeIdeal_def ringHomomor_def by auto
  } thus f(J) ≠ R by blast
Even more, if the target ring of the homomorphism is commutative and the ideal is prime then its preimage is also. Note that this is not true in general.

**lemma** (in ring_homo) **preimage_prime_ideal_comm**:
assumes J:q[R] V {is commutative on} S
shows (f-(J)):q[R]
**proof** -

have ∀x∈R. ∀y∈R. (∀z∈R. x·Rz·Ry ∈ f-(J)) → x∈f-(J) ∨ y∈f-(J)
**proof** -

{ fix x y assume x∈R y∈R and as: ∀z∈R. x·Rz·Ry ∈ f-(J)
and y /∈ f-(J)
{ fix s assume s∈S
with assms(2) {<x> R <γ> R} have 
(f(x))·S·S(f(y)) = s·S·S(f(y))
using f_is_fun apply_funtype target_ring.Ring_ZF_1_L11(2)
unfolding IsCommutative_def by auto
with {<x> R <γ> R} have (fx)·S·S(fy) = s·S(f(x·Ry))
using homomor_dest_mult by simp
with assms(1) {<x> ∈ S} {<x> ∈ R} as have (fx)·S·S(fy) ∈ J
using origin_ring.Ring_ZF_1_L2(2) origin_ring.Ring_ZF_1_L3(5)
func1_1_L15 f_is_fun target_ring.ideal_dest_mult(2)
unfolding target_ring.primeIdeal_def by auto
}
} hence ∀z∈S. (fx)·S·S (fy) ∈ J by simp
with assms(1) {<x> R <γ> R} have f(x)∈ J ∨ f(y)∈J
using target_ring.equivalent_prime_ideal f_is_fun apply_funtype
by simp
with {<x> R <γ> R} {<y> /∈ f-(J)} have x∈f-(J)
using func1_1_L15 f_is_fun by simp
}

thus thesis by blast
**qed**

moreover from assms have (f-J):q[R]
using preimage_ideal
unfolding target_ring.primeIdeal_def by auto
moreover from assms(1) have f-(J) ≠ R using vimage_prime_ideal_not_all
by simp
ultimately show (f-(J)):q[R]
by (rule origin_ring.equivalent_prime_ideal_2)
**qed**

We can replace the assumption that the target ring of the homomorphism is commutative with the assumption that homomorphism is surjective in **preimage_prime_ideal_comm** above and we can show the same assertion that the preimage of a prime ideal prime.

**lemma** (in ring_homo) **preimage_prime_ideal_surj**:
assumes J:q[R] f ∈ surj(R,S)
shows (f-(J)):q[R]
**proof** -
have $\forall x \in \mathbb{R}. \forall y \in \mathbb{R}. (\forall z \in \mathbb{R}. x \cdot R_z \cdot R y \in f^{-}(J)) \longrightarrow x \in f^{-}(J) \lor y \in f^{-}(J)$

proof -

{ fix x y assume $x \in \mathbb{R}$ $y \in \mathbb{R}$ and as: $\forall z \in \mathbb{R}. x \cdot R_z \cdot R y \in f^{-}(J)$
  and $y \notin f^{-}(J)$
  { fix $s$ assume $s \in S$
    with assms(2) obtain $t$ where $s=f(t)$ $t \in \mathbb{R}$
      unfolding surj_def by auto
    with $<x \in \mathbb{R}>$ $<y \in \mathbb{R}>$ as have $(f(x)) \cdot_S z \cdot_S (f(y)) \in J$
      using homomor_dest_mult origin_ring.Ring_ZF_1_L4(3)
      func1_1_L15 f_is_fun by simp
  } hence $\forall z \in S$. $(f(x)) \cdot_S z \cdot_S (f(y)) \in J$ by simp
  with target_ring.equivalent_prime_ideal assms(1) $<x \in \mathbb{R}>$ $<y \in \mathbb{R}>$
  have $f(x) \in J \lor f(y) \in J$ using f_is_fun apply_funtype
    by auto
  with $<x \in \mathbb{R}>$ $<y \in \mathbb{R}>$ $y \notin f^{-}(J)$ have $x \in f^{-}(J)$
    using func1_1_L15 f_is_fun by auto
  } thus thesis by blast
qed

moreover from assms have $(f^{-}(J)) \triangleleft_o \mathbb{R}_o$
  using preimage_ideal
  unfolding target_ring.primeIdeal_def by auto
moreover from assms(1) have $f^{-}(J) \neq \mathbb{R}$ using vimage_prime_ideal_not_all
  by simp
ultimately show $(f^{-}(J)) \triangleleft_p \mathbb{R}_o$
  by (rule origin_ring.equivalent_prime_ideal_2)
qed

48.2 Quotient ring with quotient map

The notion of a quotient ring (a.k.a factor ring, difference ring or residue class) is analogous to the notion of quotient group from the group theory.

The next locale ring2 extends the ring0 locale (defined in the Ring_ZF theory) with the assumption that some fixed set $I$ is an ideal. It also defines some notation related to quotient rings, in particular we define the function (projection) $f_I$ that maps each element $r$ of the ring $R$ to its class $r_I$ of $r$ where $r_I$ is the quotient group relation defined by $I$ as a (normal) subgroup of $R$ with addition.

locale ring2 = ring0 +
  fixes $I$
  assumes idealAssum: $I \triangleleft \mathbb{R}$
fixes quot $(R_I)$
defines quot_def [simp]: $R_I \equiv \text{QuotientBy}(I)$
fixes qrel $(r_I)$
defines qrel_def [simp]: $r_I \equiv \text{QuotientGroupRel}(R,A,I)$

578
fixes qfun \((f_I)\)
defines qfun_def [simp]: \(f_I \equiv \lambda r \in R. r_I\{r\}\)

fixes qadd \((A_I)\)
defines qadd_def [simp]: \(A_I \equiv \text{QuotientGroupOp}(R, A, I)\)

fixes qmul \((M_I)\)
defines qmul_def [simp]: \(M_I \equiv \text{ProjFun2}(R, qrel, M)\)

The expression \(J \triangleleft R_I\) will mean that \(J\) is an ideal of the quotient ring \(R_I\) (with the quotient addition and multiplication).

abbreviation (in ring2) qideal (_ \(\triangleleft R_I\)) where
\(J \triangleleft R_I \equiv \text{ring0}.\text{Ideal}(R_I, A_I, M_I, J)\)

In the ring2 The expression \(J \triangleleft p_R_I\) means that \(J\) is a prime ideal of the quotient ring \(R_I\).

abbreviation (in ring2) qprimeIdeal (_ \(\triangleleft p_R_I\)) where
\(J \triangleleft p_R_I \equiv \text{ring0}.\text{primeIdeal}(R_I, A_I, M_I, J)\)

Theorems proven in the ring0 context can be applied to the quotient ring in the ring2 context.

sublocale ring2 < quotient_ring: ring0 quot qadd qmul
\(\lambda x \ y. \text{ideal_radd}(x, I, y) \ \lambda y. \text{ideal_rmin}(I, y)\)
\(\lambda x \ y. \text{ideal_rsub}(x, I, y) \ \lambda x. \text{ideal_rmult}(x, I, y)\)
\(0_I, 1_I, 2_I, \lambda x. (x^2)\)
using quotientBy_is_ring idealAssum neutral_quotient
one_quotient two_quotient
unfolding ring0_def ideal_radd_def ideal_rmin_def
ideal_rsub_def ideal_rmult_def ideal_rzero_def
ideal_rone_def ideal_rtwo_def ideal_rsqr_def by auto

The quotient map is a homomorphism of rings. This is probably one of the
most sophisticated facts in IsarMathlib that Isabelle’s simp method proves
from 10 facts and 5 definitions.

theorem (in ring2) quotient_fun_homomor:
shows \(\text{ringHomomor}(f_I, R, A, M, R_I, A_I, M_I)\)
using ringAssum idealAssum ideal_normal_add_subgroup add_group.quotient_map

\text{RingZF\_1\_L4(3) EquivClass\_1\_L10 RingZF\_1\_L2(2) GroupZF\_2\_2\_L1} \text{Ideal_equiv_rel quotientBy_mul_monoid(1) QuotientBy_def}
unfolding IsAring_def Homomor_def IsMorphism_def ringHomomor_def
by simp

The quotient map is surjective

lemma (in ring2) quot_fun:
shows \(f_I \in \text{surj}(R, R_I)\)
using lam_funtype idealAssum QuotientBy_def
unfolding quotient_def surj_def by auto

The theorems proven in the ring_homo context are valid in the ring_homo context when applied to the quotient ring as the second (target) ring and the quotient map as the ring homomorphism.

sublocale ring2 < quot_homomorphism: ring_homo R A M quot qadd qmul qfun 
  λx y. ideal_radd(x,I,y) λy. ideal_rmin(I,y) λx y. ideal_rsub(x,I,y) λx y. ideal_rmult(x,I,y) 0 I 1 I 2 I λx. (x^2)
using ringAssum quotient_ring.ringAssum quotient_fun_homomor unfolding ring_homo_def by simp_all

The ideal we divide by is the kernel of the quotient map.

lemma (in ring2) quotient_kernel:
  shows quot_homomorphism.kernel = I
proof
  from idealAssum show quot_homomorphism.kernel ⊆ I 
  using add_group.Group_ZF_2_4_L5E ideal_normal_add_subgroup neutral_quotient QuotientBy_def 
  quot_homomorphism.image_kernel func1_1_L15 
  unfolding ideal_rzero_def QuotientBy_def by auto
  { fix t assume t∈I 
    then have t∈R using ideal_dest_subset idealAssum by auto 
    with idealAssum t∈f_I-{0_I} have t∈f_I using add_group.Group_ZF_2_4_L5E ideal_normal_add_subgroup 
    neutral_quotient QuotientBy_def func1_1_L15 surj_is_fun 
    unfolding ideal_rzero_def by auto 
  } thus I ⊆ quot_homomorphism.kernel by blast
qed

The theorems proven in the ring0 context are valid in the ring 2 context when applied to the quotient ring.

sublocale ring2 < quotient_ring: ring0 quot qadd qmul 
  λx y. ideal_radd(x,I,y) λy. ideal_rmin(I,y) λx y. ideal_rsub(x,I,y) λx y. ideal_rmult(x,I,y) 0 I 1 I 2 I λx. (x^2)
using idealAssum quotientBy_is_ring neutral_quotient one_quotient two_quotient unfolding ring0_def by simp_all

If an ideal I is a subset of the kernel of the homomorphism then the image of the ideal generated by I ∪ J, where J is another ideal, is the same as the image of J. Note that J+I notation means the ideal generated by the union of ideals J and I, see the definitions of sumIdeal and generatedIdeal in the Ring_ZF_2 theory, and also corollary sum_ideals_is_sum_elements for an alternative definition.

theorem (in ring_homo) kernel_empty_image:
assumes $J \trianglelefteq R$, $I \subseteq \ker I \trianglelefteq R$

shows $f(J+I) = f(J)$, $f(I+J) = f(J)$

proof
from assms(1,3) have $J+I \subseteq R$
  using origin_ring.sum_ideals_is_ideal
  origin_ring.ideal_dest_subset by auto
show $f(J+I) = f(J)$
proof
{ fix $t$ assume $t \in f(J+I)$
  with $J+I \subseteq R$ obtain $q$ where $q \in J+I$, $t = f(q)$
  using func_imagedef f_is_fun by auto
  with assms(1,3) $q \in J+I$ have $q \in A(J \times I)$, $J \times I \subseteq R \times R$
  using origin_ring.sum_ideals_is_sum_elements
  assms(1,3) origin_ring.ideal_dest_subset by auto
  from origin_ring.add_group.groupAssum $J \times I \subseteq R \times R$
  have $A(J \times I) = \{A(p). p \in J \times I\}$
  unfolding IsAgroup_def IsAmonoid_def IsAssociative_def
  using func_imagedef by auto
  with $q \in A(J \times I)$ obtain $q_i$, $q_j$ where $q_j \in J$, $q_i \in I$,
  $q=q_j + R q_i$ by auto
  with assms have $f(q) = (f(q_j)) +_R 0_S$
  using homomor_dest_add origin_ring.ideal_dest_subset image_kernel
  by blast
  moreover from assms(1) $q_j \in J$ have $f(q_j) \in S$
  using origin_ring.ideal_dest_subset f_is_fun apply_funtype by blast
  ultimately have $f(q) = f(q_j)$ using target_ring.Ring_ZF_1_L3(3)
  by auto
  with assms $t = f(q_j)$, $q_j \in J$ have $t \in f(J)$
  using func_imagedef f_is_fun origin_ring.ideal_dest_subset by auto
} thus $t \in f(J)$ by blast

{ fix $t$ assume $t \in f(J)$
  with assms(1) obtain $q$ where $q \in J$, $t = f(q)$
  using func_imagedef f_is_fun origin_ring.ideal_dest_subset by auto
  from $q \in J$ have $q \in J \trianglelefteq I$ by auto
  moreover from assms(1,3) have $J \trianglelefteq I \trianglelefteq R$
  using origin_ring.ideal_dest_subset by auto
  ultimately have $q \in (J \trianglelefteq I)$, using origin_ring.generated_ideal_contains_set
  by auto
  with assms(1,3) $J \trianglelefteq R$, $t = f(q)$, $t \in f(J+I)$
  using origin_ring.sumIdeal_def f_is_fun func_imagedef by auto
} thus $f(J) \subseteq f(J+I)$ by auto
qed
with assms(1,3) show $f(I+J) = f(J)$
using origin_ring.sum_ideals_commute by auto
qed
48.3 Quotient ideals

If we have an ideal \( J \) in a ring \( R \), and another ideal \( I \) contained in \( J \), then we can form the quotient ideal \( J/I \) whose elements are of the form \( a + I \) where \( a \) is an element of \( J \).

The preimage of an ideal is an ideal, so it applies to the quotient map; but the preimage ideal contains the quotient ideal.

**lemma (in ring2) ideal_quot_preimage:**
- assumes \( J \triangleleft R \)
- shows \((f_j{-}(J)) \triangleleft R I \subseteq f_j{-}(J)\)

**proof** -
- from assms quot_homomorphism.preimage_ideal show \((f_j{-}(J)) \triangleleft R \) by simp
  
  \{ fix \( x \) assume \( x \in I \) with idealAssum have \( x \in R \) using ideal_dest_subset by auto
    from assms \( \langle x \in I \rangle \) have \( f_j(x) \in J \) using quotient_kernel quot_homomorphism.image_kernel
    quotient_ring.ideal_dest_zero by simp
    with \( \langle x \in R \rangle \) have \( x \in f_j{-}(J) \) using quot_homomorphism.f_is_fun func1_1_L15
    by simp
  \}
  thus \( I \subseteq f_j{-}(J) \) by auto
  qed

Since the map is surjective, the image is also an ideal.

**lemma (in ring_homo) image_ideal_surj:**
- assumes \( J \triangleleft R \), \( f \in \text{surj}(R,S) \)
- shows \((f(J)) \triangleleft R t \)

**proof** -
- from assms homomorphism target_ring.ringAssum origin_ring.ringAssum
  have IsAsubgroup(f(J),U) using ringHomHom(1) image_subgroup f_is_fun
  unfolding IsAring_def origin_ring.Ideal_def by blast
  moreover
  \{ fix \( x \) \( y \) assume \( x \in f(J) \) \( y \in S \)
    from assms(1) \( \langle x \in f(J) \rangle \) obtain \( j \) where \( x = f(j) \) \( j \in J \)
    using func_imagedef f_is_fun origin_ring.ideal_dest_subset by auto
    from assms \( \langle y \in S \rangle \) \( \langle j \in J \rangle \) have \( j \in R \) and \( y \in f(R) \)
    using origin_ring.ideal_dest_subset surj_range_image_domain by auto
    with assms(1) obtain \( s \) where \( y = f(s) \) \( s \in R \)
      using func_imagedef origin_ring.ideal_dest_subset f_is_fun by auto
    with assms(1) \( \langle j \in R \rangle \) \( \langle x = f(j) \rangle \) \( \langle x \in f(J) \rangle \) have
      \( V(x,y) = f(M(j,s)) \) and \( V(y,x) = f(M(s,j)) \)
      using homomor_dest_mult origin_ring.ideal_dest_subset by auto
  \}
  qed

582
with assms(1) \(<j \in J>, \langle s \in R \rangle >\) have \((x, s y) \in f(J)\) and \((y, s x) \in f(J)\) using origin_ring.ideal_dest_mult_func_imagedef f_is_fun
origin_ring.ideal_dest_subset by auto
\}

hence \(\forall x \in f(J). \ \forall y \in S. \ (y, s x) \in f(J) \land (x, s y) \in f(J)\)
by auto
ultimately show thesis unfolding target_ring.Ideal_def by simp
qed

If the homomorphism is a surjection and given two ideals in the target ring the inverse image of their product ideal is the sum ideal of the product ideal of their inverse images and the kernel of the homomorphism.

corollary (in ring_homo) prime_ideal_quot:
assumes \(J \triangleleft R, \ K \triangleleft R, \ f \in \text{surj}(R, S)\)
says \(f-(\text{target_ring.productIdeal}(J, K)) = \text{origin_ring.sumIdeal}(\text{origin_ring.productIdeal}((f-(J)), (f-(K))), \ker)\)
proof
let \(P = \text{origin_ring.sumIdeal}(\text{origin_ring.productIdeal}((f-(J)), (f-(K))), f\{-0_S\})\)
let \(Q = \text{target_ring.productIdeal}(J, K)\)
from assms(1,2) have ideals: \((f-(J)) \triangleleft R, (f-(K)) \triangleleft R\)
using preimage_ideal by auto
then have idealJK: \(((f-(J)) \cdot (f-(K))) \triangleleft R\)
using origin_ring.product_in_intersection(2) by auto
then have \(P \triangleleft R\) and \(P \subseteq R\)
using kernel_ideal origin_ring.sum_ideals_is_ideal
origin_ring.ideal_dest_subset by simp_all
with assms(3) have \((f(P)) \triangleleft R\) using image_ideal_surj by simp
let \(X = ((f-(J)) \cdot (f-(K))) \cup (f\{-0_S\})\)
from assms(3) idealJK have \(X \subseteq R\)
using func1_1_L6A surj_is_fun origin_ring.ideal_dest_subset by blast
with idealJK have \(X \subseteq P\)
using kernel_ideal origin_ring.sum_ideal_def
origin_ring.generated_ideal_contains_set by simp
\{
fix \(t\) assume \(t \in V(J \times K)\)
multiple from assms have \(J \times K \subseteq S \times S\)
using target_ring.ideal_dest_subset by blast
ultimately obtain \(j, k\) where \(j \in J, k \in K\)
using func_imagedef AMUV_are_ops(4) by auto
from assms(1) \(<j \in J>, \ j \in S\)
using target_ring.ideal_dest_subset by blast
with assms(3) obtain \(j_0\) where \(j_0 \in R, f(j_0) = j\)
unfolding surj_def by auto
with assms(2,3) \(<k \in K>, \ k \in S\)
obtain \(k_0\) where \(k_0 \in R, f(k_0) = k\)
using target_ring.ideal_dest_subset unfolding surj_def by blast
with \(t = V(j, k)\) \(<f(j_0) = j >\) \(<j_0 \in R, t = f(M(j_0, k_0))\)
using homomor_dest_mult by simp
\}
from assms(3) have \((f-(J)) \times (f-(K)) \subseteq R \times R\)
  using func1_1_L6A f_is_fun by blast
moreover from assms(3) \(j\in R\) \(\langle f(j_0) = j \rangle \langle j \in J \rangle \langle k \in K \rangle \langle k_0 \in R \rangle \langle f(k_0) = k \rangle\)
  have \(\langle j_0, k_0 \rangle \in (f-J) \times (f-K)\)
  using func1_1_L15 unfolding surj_def by auto
ultimately have \(M(j_0, k_0) \in M((f-(J)) \times (f-(K)))\)
  using AMUV_are_ops func_imagedef by auto
with ideals \(\langle X \subseteq P \rangle \langle h \rangle\) have \(M(j_0, k_0) \in P\)
  using origin_ring.product_in_intersection(3)
  by blast
with \(\langle P \subseteq R \rangle \langle t = f(M(j_0, k_0)) \rangle \langle h \rangle\)
  have \(t \in (f(P))\)
  using f_is_fun func1_1_L15D by simp
} hence \(V(J \times K) \subseteq (f(P))\) by blast
with assms(1,2) \(\langle f(P) \rangle \langle h \rangle\)
  have \(\langle \exists P \subseteq (f(P)) \langle f(t) = f(s) \rangle \rangle\)
  unfolding vimage_def by blast
from \(\langle X \subseteq P \rangle \langle P \subseteq R \rangle\) have \(P_\text{ideal}: P \in \{N \subseteq P \ | \ f-(0_S) \subseteq N\}\)
  by auto
{ fix \(t\) assume \(\langle t \rangle\)
  then have \(f(t) \in (f(P))\) and \(t \in R\)
  using func1_1_L15 f_is_fun by simp_all
  with \(P_\text{ideal}\) obtain \(s\) where \(f(t) = f(s)\) \(s \in P\)
  using func_imagedef f_is_fun by auto
from \(P_\text{ideal}\) \(\langle s \subseteq P \rangle\)
  have \(s \in R\) by blast
from \(\langle t \in R \rangle \langle s \in R \rangle \langle \langle t \rangle = f(s) \rangle \langle h \rangle\)
  have \(f(t-Rs) = 0_S\)
  using f_is_fun apply_funtype target_ring.Ring_ZF_1_L3(7)
  homomor_dest_subs by simp
with \(\langle t \in R \rangle \langle s \in R \rangle \langle X \subseteq P \rangle\)
  have \(t-Rs \in P\)
  using origin_ring.Ring_ZF_1_L4(2) func1_1_L15 f_is_fun
  by auto
with \(P_\text{ideal}\)
  \(\langle s \in P \rangle\)
  have \(s + R(t-Rs) \in P\)
  using origin_ring.ideal_dest_sum by auto
with \(\langle t \in R \rangle \langle s \in R \rangle\)
  have \(t \in P\)
  using origin_ring.Ring_ZF_2_L1A(5) by auto
} with \(R\) show \(f-(Q) \subseteq P\)
  by auto
{ fix \(t\) assume as: \(t \in M((f-(J)) \times (f-(K)))\)
  have \((f-(J)) \times (f-(K)) \subseteq R \times R\)
  using func1_1_L15 f_is_fun by auto
  with as obtain \(t_j, t_k\) where
    \(\langle jk \rangle\) \(\langle t_j = f(t_j) \rangle \langle t_k \in f-(K) \rangle\)
    using AMUV_are_ops(2) func_imagedef IsAssociative_def
    by auto
  from as have \(t \in R\)
    using AMUV_are_ops(2) func1_1_L6(2) by blast
  from \(j k(2,3)\) have \(t_j \in R\) \(f(t_j) \in J\) and \(t_k \in R\) \(f(t_k) \in K\)
    using func1_1_L15 f_is_fun by simp_all
584
from jk(1) \langle t_j \in \mathbb{R} \rangle \langle t_k \in \mathbb{R} \rangle have \ ft: f(t) = ((f(t_j)) \cdot S(f(t_k)))
using homomor_dest_mult by simp
from \langle f(t_j) \in J \rangle \langle f(t_k) \in K \rangle have \ \langle f(t_j), f(t_k) \rangle \in J \times K
by simp
moreover from \text{assms} have \ V:S \times S \rightarrow S \text{ and } J \times K \subseteq S \times S
using AMUV_are_ops(4) target_ring.ideal_dest_subset
by auto
ultimately have \ V(f(t_j), f(t_k)) \in V(J \times K)
using func_imagedef by force
with \text{assms} ft \ \langle t \in \mathbb{R} \rangle have t \in f^{-}(Q)
using target_ring.product_in_intersection(3) func1_1_L15 f_is_fun

by auto
\}

hence \ M(f^{-}(J) \times f^{-}(K)) \subseteq f^{-}(Q)
by auto
moreover from \text{assms}(1,2) have \ id: (f^{-}(Q)) \prec R_0
using preimage_ideal target_ring.product_in_intersection(2)
by simp
ultimately have \ (f^{-} - J) \cdot_I (f^{-} - K) \subseteq f^{-}(Q)
using ideals origin_ring.generated_ideal_small origin_ring.productIdeal_def
by auto
with \text{assms}(1,2) have \ X \subseteq f^{-}(Q)
using target_ring.product_in_intersection(2) target_ring.ideal_dest_subset

by auto
with \text{idealJK id show}
((f^{-}(J)) \cdot_I (f^{-}(K))) +_I (f^{-}(0_S)) \subseteq f^{-}(Q)
using origin_ring.generated_ideal_small kernelIdeal origin_ring.sumIdeal_def
by simp
qed

If the homomorphism is surjective then the product ideal of ideals \ J, K in the target ring is the image of the product ideal (in the source ring) of the inverse images of \ J, K.

corollary (in ring_homo) prime_ideal_quot_2:
assumes \ J \prec \mathbb{R} \ K \prec \mathbb{R} \ f \in \text{surj}(R, S)
shows \ target_ring.productIdeal(J, K) = f(origin_ring.productIdeal((f^{-}(J)), (f^{-}(K))))
proof -
from \text{assms} have \ f(f^{-}(target_ring.productIdeal(J, K)))
= f(((f^{-}(J)) \cdot_I (f^{-}(K))) +_I (\text{ker}))
using prime_ideal_quot by simp
with \text{assms} show thesis
using target_ring.product_in_intersection(2) target_ring.ideal_dest_subset surj_image_vimage
origin_ring.product_in_intersection(2) preimage_ideal
kernel_empty_image(1) target_ring.zero_ideal
by simp
If the homomorphism is surjective and an ideal in the source ring contains
the kernel, then the image of that ideal is a prime ideal in the target ring.

**lemma (in ring_homo) preimage_ideal_prime:**
assumes $J \triangleleft p R \subseteq J f : \text{surj}(R,S)$
shows $(f(J)) \triangleleft p R t$

**proof**

1. from assms(1) have $J \subseteq R J \neq R$
2. unfolding origin_ring.primeIdeal_def
3. using origin_ring.ideal_dest_subset by auto

4. from assms(1,3) have $(f(J)) \triangleleft p R t$
5. using image_ideal_surj unfolding origin_ring.primeIdeal_def by auto

moreover

{ assume $f(J) = S$

from $J \subseteq R J \neq R$ obtain $t$ where $t \in R - J$ by auto

with assms(3) have $f(t) \in S$

using apply_funtype f_is_fun by auto

with assms(3) $J \subseteq R$ $f(J) = S$ obtain $j$

where $j$: $j \in J f(t) = f(j)$

using func_imagedef f_is_fun by auto

from $j \in J$ $J \subseteq R$ have $j \in R$ by auto

with assms(1,2) $j(1)$ have $j_{R} (t_{-}R j) \in J$

unfolding origin_ring.primeIdeal_def

using origin_ring.ideal_dest_sum by auto

with $j \in R$ $t \in R - J$ have False using origin_ring.Ring_ZF_2_L1A(5)

by auto

} hence $f(J) \neq S$ by auto

moreover

{ fix $I$ $K$ assume as: $I \in \text{target_ring.ideals} K \in \text{target_ring.ideals}$

let $A = (f-(I)) \cap (f-(K))$

from as(1,2) have $A \subseteq f-(f(A))$

using origin_ring.product_in_intersection(2) preimage_ideal(2)

origin_ring.ideal_dest_subset func1_1_L9 f_is_fun by auto

with assms(3) as have $A \subseteq f-(f(J))$

using prime_ideal_quot_2 vimage_mono by force

moreover from assms(1) have $J \subseteq f-(f(J))$

using func1_1_L9 f_is_fun origin_ring.ideal_dest_subset

unfolding origin_ring.primeIdeal_def by auto

moreover

{ fix $t$ assume $t \in f-(f(J))$

then have $f(t) \in f(J) t \in R$ using func1_1_L15 f_is_fun

by auto

}
from assms(1) \( \langle f(t) \in f(J) \rangle \) obtain \( s \) where \( f(t) = f(s) \) \( s \in J \)
using func_imgagedef f_is_fun origin_ring.ideal_dest_subset
unfolding origin_ring.primeIdeal_def by auto
from assms(1) \( \langle s \in J \rangle \) have \( s \in \mathbb{R} \)
unfolding origin_ring.primeIdeal_def
using origin_ring.ideal_dest_subset by auto
with assms(2) \( \langle f(t) = f(s) \rangle \) \( \langle t \in \mathbb{R} \rangle \) have \( t - R s \in J \)
using target_ring.Ring_ZF_1_L3(7) apply_funtype f_is_fun
homomor_dest_subs origin_ring.Ring_ZF_1_L4(2) func1_1_L15
by auto
with assms(1) \( \langle s \in J \rangle \) have \( s \in \mathbb{R} \)
unfolding origin_ring.primeIdeal_def
using origin_ring.ideal_dest_subset by auto
ultimately have \( A \subseteq J \)
by auto
ultimately have \( I \subseteq J \)
by auto
with assms(1) as(1,2) have \( f(I) \subseteq J \lor (f(K)) \subseteq J \)
using preimage_ideal origin_ring.ideal_dest_subset
unfolding origin_ring.primeIdeal_def
by auto
hence \( f(f(I)) \subseteq f(J) \lor f(f(K)) \subseteq f(J) \)
by auto
with assms(3) as(1,2) have \( I \subseteq f(J) \lor K \subseteq f(J) \)
using surj_image_vimage by auto
ultimately show thesis unfolding target_ring.primeIdeal_def by auto
qed

The ideals of the quotient ring are in bijection with the ideals of the original
ring that contain the ideal by which we made the quotient.

\[ \text{ideal_quot_bijection:} \]
\[ \text{assumes} \ f \in \text{surj}(\mathbb{R}, S) \]
\[ \text{defines} \ \text{idealFun} = \lambda J \in \text{target_ring.ideals}. \ f(J) \]
\[ \text{shows} \ \text{idealFun} \in \text{bij}(\text{target_ring.ideals}, \{ K \in \mathcal{I}. \ker \subseteq K \}) \]
\text{proof -}
let \( \mathcal{I}_t = \text{target_ring.ideals} \)
have \( \text{idealFun} : \mathcal{I}_t \rightarrow \{ f(J). \ J \in \mathcal{I}_t \} \)
unfolding idealFun_def using lam_funtype by simp
moreover
\{ fix \( t \) assume \( t \in \{ f(J). \ J \in \mathcal{I}_t \} \)
then obtain \( K \) where \( K \in \mathcal{I}_t \) \( f(K) = t \)
by auto
then have \( K \in \mathbb{R} \)
by simp
then have \( (f(K)) \subseteq f(K) \)
using preimage_ideal(2,3)
by simp_all
with \( f(K) = t \)
have \( \ker \subseteq t \in \mathbb{R} \)
by simp_all
with \( f(K) = t \)
have \( t \in \{ K \in \mathcal{I}. \ker \subseteq K \} \)
by blast
\}

587
hence \( \{f^{-1}(J). \quad J \in \mathcal{I}_t\} \subseteq \{K \in \mathcal{I}. \quad \ker K \subseteq K\} \) by blast

ultimately have \( I: \text{idealFun} : \mathcal{I}_t \rightarrow \{K \in \mathcal{I}. \quad \ker K \subseteq K\} \)

using \text{func1}_1\_L1B by auto

\{
  fix \( w \) \( x \) assume
  as: \( w \triangleleft R \) \( x \triangleleft R \) \( w \subseteq S \) \( x \subseteq S \) \text{idealFun}(w) = \text{idealFun}(x)
  then have \( f(f^{-1}(w)) = f(f^{-1}(x)) \) unfolding \text{idealFun_def} by simp
  with \text{assms(1)} as have \( w = x \) using \text{surj_image_vimage} by simp
\}

with \( I \) have \( \text{idealFun} \in \text{inj}(\mathcal{I}_t,\{K \in \mathcal{I}. \quad \ker K \subseteq K\}) \)

unfolding \text{inj_def} by auto

moreover

\{
  fix \( y \) assume \( y \triangleleft R \) \( y \subseteq \text{ker} \subseteq y \)
  from \( \langle y \subseteq R \rangle \) have \( y \subseteq f^{-1}(fy) \) using \text{func1}_1\_L9 \text{f_is_fun} by auto
  by auto

  moreover

  \{
    fix \( t \) assume \( t \in f^{-1}(f(y)) \)
    then have \( f(t) \in f(y) \) \( t \in R \) using \text{func1}_1\_L15 \text{f_is_fun} by auto
    from \( \langle f(t) \in f(y) \rangle \) \( \langle y \subseteq R \rangle \) obtain \( s \) where \( s \in y \) \( f(t) = f(s) \)
    using \text{func_imagedef f_is_fun} by auto
    from \( \langle s \subseteq y \rangle \) \( \langle y \subseteq R \rangle \) have \( s \in R \) by auto
    with \( \langle t \subseteq R \rangle \) \( \langle f(t) = f(s) \rangle \) \( \langle \ker \subseteq y \rangle \) have \( t - R s \in y \)
    using \text{target_ring.RingZF}\_1\_L3(7) \text{homomor_dest_subs}
    \text{origin_ring.RingZF}\_2\_L4(2) \text{func1}_1\_L15 \text{f_is_fun}
    by auto
    with \( \langle s \subseteq y \rangle \) \( \langle y \subseteq R \rangle \) \( \langle s \subseteq R \rangle \) \( \langle t \subseteq R \rangle \) have \( t \in y \)
    using \text{origin_ring.ideal_dest_sum} \text{origin_ring.RingZF}\_2\_L1A(5)
    by force
  \}

  ultimately have \( f^{-1}(f(y)) = y \) by blast

  moreover have \( f(y) \subseteq S \) using \text{func1}_1\_L6(2) \text{f_is_fun}
  unfolding \text{surj_def} by auto

  moreover from \text{assma(1)} \( \langle y \triangleleft R \rangle \) have \( (f(y)) \triangleleft R \), using \text{image_ideal_surj}
  by auto

  ultimately have \( \exists x \in \mathcal{I}_t. \quad \text{idealFun}(x) = y \)
  unfolding \text{idealFun_def} by auto

  \}

  with \( I \) have \( \text{idealFun} \in \text{surj}(\mathcal{I}_t,\{K \in \mathcal{I}. \quad \ker K \subseteq K\}) \)

  unfolding \text{surj_def} by auto

  ultimately show thesis unfolding \text{bij_def} by blast
\}

qed

Assume the homomorphism \( f \) is surjective and consider the function that maps an ideal \( J \) in the target ring to its inverse image \( f^{-1}(J) \) (in the source ring). Then the value of the converse of that function on any ideal containing the kernel of \( f \) is the image of that ideal under the homomorphism \( f \).

theorem (in ring_homo) quot_converse:
  defines \( F \equiv \lambda J \in \text{target_ring.ideals}. \quad f^{-1}(J) \)
assumes $J \triangleleft R \ker \subseteq J \ f \in \text{surj}(R,S)$
shows converse$(F)(J) = f(J)$

proof
let $g = \lambda J \in \{K \in I. \ker \subseteq K\}. f(J)$
let $I_t = \text{target\_ring.\_ideals}$
let $C_F = \text{converse}(F)$
from assms$(1,4)$ have $C_F \in \text{bij}(\{K \in I. \ker \subseteq K\}, I_t)$
  using bij_converse_bij ideal_quot_bijection
  by auto
then have $I: C_F: \{K \in I. \ker \subseteq K\} \rightarrow I_t$
  unfolding bij_def inj_def by auto
moreover from assms$(2,3)$ have $J: J \in \{K \in I. \ker \subseteq K\}$
  using origin_ring.ideal_dest_subset
  by auto
ultimately have ideal: $C_F(J) \in I_t$
  using apply_funtype
  by blast
moreover from assms$(1)$ have $F: I_t \rightarrow \{f-(J). J \in I_t\}$
  using lam_funtype
  by simp
with $I J$ have $F(C_F(J)) = J$
  using right_inverse_lemma
  by simp
ultimately have $g(J) = C_F(J)$
  by simp
with $J$ show thesis
  by simp
qed

Since the map is surjective, this bijection restricts to prime ideals on both sides.

corollary (in ring_homo) prime_ideal_quot_3:
assumes $K \triangleleft R \ f \in \text{surj}(R,S)$
shows $K \triangleleft p_R \leftrightarrow ((f-(K)) \triangleleft p_R)$

proof
\{ assume $K \triangleleft p_R$
  with assms$(2)$ show $(f-(K)) \triangleleft p_R$
    using preimage_prime_ideal_surj target_ring.ideal_dest_subset
    unfolding target_ring.primeIdeal_def
  by auto \}
\{ assume $(f-(K)) \triangleleft p_R$
  with assms$(1,2)$ have $K \triangleleft R$ and $K \neq S$
    using func1_1_L4 unfolding origin_ring.primeIdeal_def surj_def
  by auto
moreover
\{ fix $J P$ assume $jp: J \in \text{target\_ring.\_ideals}$
  $P \in \text{target\_ring.\_ideals}$
  $\text{target\_ring.\_productIdeal}(J, P) \subseteq K$
  from $jp(3)$ have $f-\text{((target\_ring.\_productIdeal}(J, P)) \subseteq f-(K)$
    by auto
  with assms$(2) \ j$p$(1,2)$ have

589
\[
A: ((f-\cdot(J)) \cdot (f-(P))) + I (ker) \subseteq f-(K)
\]
using prime_ideal_quot by auto
from \(\text{jp}(1,2)\) have
\((f-(J)) \cdot (f-(P)) \cup \ker \subseteq ((f-(J)) \cdot (f-(P))) + I (ker)\)
using preimage_ideal origin_ring.product_in_intersection(2)
kernel_ideal origin_ring.comp_in_sum_ideals(3) by simp
with \(A\) have \(((f-(J)) \cdot (f-(P))) \subseteq f-(K)\) by auto
with \(<(f-(K))>_{R} \text{jp}(1,2)\) have
\(f-(J) \subseteq f-(K) \lor f-(P) \subseteq f-(K)\)
using preimage_ideal origin_ring.ideal_dest_subset
unfolding origin_ring.primeIdeal_def by auto
then have \(f(f-(J)) \subseteq f(f-(K)) \lor f(f-(P)) \subseteq f(f-(K))\)
by blast
with \(\text{assms}\) \(\text{jp}(1,2)\) have \(J \subseteq K \lor P \subseteq K\)
using surj_image_vimage target_ring.ideal_dest_subset
by auto
}
ultimately show \(K \triangleleft_{p} R_{t}\)
unfolding target_ring.primeIdeal_def by auto
\}
qued

If the homomorphism is surjective then the function that maps ideals in the
target ring to their inverse images (in the source ring) is a bijection between
prime ideals in the target ring and the prime ideals containing the kernel in
the source ring.

corollary (in ring_homo) bij_prime_ideals:
defines \(F \equiv \lambda J \in \text{target_ring.ideals}. f-(J)\)
assumes \(f \in \text{surj}(R,S)\)
shows \(\text{restrict}(F,\{J \in \text{Pow}(S). J \triangleleft_{p} R_{t}\}) \in\)
\(\text{bij}(\{J \in \text{Pow}(S). J \triangleleft_{p} R_{t}\}, \{J \in \text{Pow}(R). \ker \subseteq J \land (J \triangleleft_{p} R)\})\)
proof -
let \(I_{t} = \text{target_ring.ideals}\)
from \(\text{assms}\) have \(I: F:I_{t} \rightarrow \{K \in I. \ker \subseteq K\}\)
using ideal_quot_bijection bij_is_fun by simp
have II: \(\{J \in \text{Pow}(S). J \triangleleft_{p} R_{t}\} \subseteq I_{t}\)
unfolding target_ring.primeIdeal_def by auto
have III: \(\{J \in \text{Pow}(S). J \triangleleft_{p} R_{t}\} \subseteq I_{t}\)
unfolding target_ring.primeIdeal_def by auto
with \(\text{assms}\) have \(\text{restrict}(F,\{J \in \text{Pow}(S). J \triangleleft_{p} R_{t}\}) \in\)
\(\text{bij}(\{J \in \text{Pow}(S). J \triangleleft_{p} R_{t}\}, F(J \in \text{Pow}(S). J \triangleleft_{p} R_{t}))\)
using restrict_bij ideal_quot_bijection unfolding bij_def
by auto
moreover have \(\{t \in \text{Pow}(R). \ker \subseteq t \land (t \triangleleft_{p} R)\} = F(J \in \text{Pow}(S). J \triangleleft_{p} R_{t})\)
proof
\{ fix \(t\) assume \(t \in F(J \in \text{Pow}(S). J \triangleleft_{p} R_{t})\)
with I III obtain \(q\) where \(q \in \text{Pow}(S)\) \(q \triangleleft_{p} R_{t}\) \(t=F(q)\)
using func_imagedef by auto
\}
590
from $q \in \text{Pow}(S)$ have $q \in I$
   unfolding target_ring.primeIdeal_def by auto
with I have $F(q) \in \{ t \in I. \ker \subseteq t \}$ using apply_funtype
   by blast
with assms $q \triangleright p \triangleleft t$
   have $t \triangleleft R \ker \subseteq t$
   using prime_ideal_quot_3 by simp_all
} then show $F\{ J \in \text{Pow}(S). J \triangleright p \triangleleft t \} \subseteq \{ t \in \text{Pow}(R). \ker \subseteq t \wedge (t \triangleright p) \}$
   using origin_ring.ideal_dest_subset by blast
\{ fix $t$ assume $t \in \{ t \in \text{Pow}(R). \ker \subseteq t \wedge (t \triangleright p) \}$
then have $t \in \text{Pow}(R) \ker \subseteq t \triangleleft p \triangleleft R$ by auto
then have tSet: $t \in \{ t \in I. \ker \subseteq t \}$
   unfolding origin_ring.primeIdeal_def by auto
with assms have cont: $\text{converse}(F)(t) \in I$
   using ideal_quot_bijection bij_converse_bij
   apply_funtype bij_is_fun by blast
moreover from assms tSet have $F(\text{converse}(F)(t)) = t$
   using ideal_quot_bijection right_inverse_bij by blast
ultimately have $f - (\text{converse}(F)(t)) = t$
   using assms(1) by simp
with assms II tSet $t \triangleright p \triangleleft R$ have $t \in F\{ J \in \text{Pow}(S). J \triangleright p \triangleleft R \}$
   using prime_ideal_quot_3 target_ring.ideal_dest_subset
   unfolding target_ring.primeIdeal_def by auto
\} thus $\{ t \in \text{Pow}(R). \ker \subseteq t \wedge (t \triangleright p) \} \subseteq F\{ J \in \text{Pow}(S). J \triangleright p \triangleleft R \}$
   by auto
qed
ultimately show thesis by auto
qed

end

49 Fields - introduction

theory Field_ZF imports Ring_ZF

begin

This theory covers basic facts about fields.

49.1 Definition and basic properties

In this section we define what is a field and list the basic properties of fields.

Field is a nottrivial commutative ring such that all non-zero elements have an inverse. We define the notion of being a field as a statement about three sets. The first set, denoted $K$ is the carrier of the field. The second set, denoted $A$
represents the additive operation on $K$ (recall that in ZF set theory functions are sets). The third set $M$ represents the multiplicative operation on $K$.

**definition**

\[
\text{IsAfield}(K, A, M) \equiv \\
(\text{IsAring}(K, A, M) \land (M \text{ is commutative on } K) \land \\
\text{TheNeutralElement}(K, A) \neq \text{TheNeutralElement}(K, M) \land \\
(\forall a \in K. a \neq \text{TheNeutralElement}(K, A) \rightarrow \\
(\exists b \in K. M(a, b) = \text{TheNeutralElement}(K, M)))
\]

The $\text{field0}$ context extends the $\text{ring0}$ context adding field-related assumptions and notation related to the multiplicative inverse.

**locale** $\text{field0} = \text{ring0 } K \ A \ M$

+ assumes $\text{mult\_commute}$: $M \text{ is commutative on } K$
+ assumes $\text{not\_triv}$: $0 \neq 1$
+ assumes $\text{inv\_exists}$: $\forall x \in K. x \neq 0 \rightarrow (\exists y \in K. x y = 1)$

+ fixes $\text{non\_zero}$ ($K_0$)
+ defines $\text{non\_zero\_def}[\text{simp}]$: $K_0 \equiv K - \{0\}$

+ fixes $\text{inv}$ ($^{-1}$ [96] 97)
+ defines $\text{inv\_def}[\text{simp}]$: $a^{-1} \equiv \text{GroupInv}(K_0, \text{restrict}(M, K_0 \times K_0))(a)$

The next lemma assures us that we are talking fields in the $\text{field0}$ context.

**lemma** (in $\text{field0}$) $\text{FieldZF}_1\_L1$: shows $\text{IsAfield}(K, A, M)$

using $\text{ringAssum}$ $\text{mult\_commute}$ $\text{not\_triv}$ $\text{inv\_exists}$ $\text{IsAfield\_def}$

by simp

We can use theorems proven in the $\text{field0}$ context whenever we talk about a field.

**lemma** $\text{field\_field0}$: assumes $\text{IsAfield}(K, A, M)$

shows $\text{field0}(K, A, M)$

using $\text{assms}$ $\text{IsAfield\_def}$ $\text{field0\_axioms\_intro}$ $\text{ring0\_def}$ $\text{field0\_def}$

by simp

Let’s have an explicit statement that the multiplication in fields is commutative.

**lemma** (in $\text{field0}$) $\text{field\_mult\_comm}$: assumes $a \in K \ b \in K$

shows $a b = b a$

using $\text{mult\_commute}$ $\text{assms}$ $\text{IsCommutative\_def}$

by simp

Fields do not have zero divisors.

**lemma** (in $\text{field0}$) $\text{field\_has\_no\_zero\_divs}$: shows $\text{HasNoZeroDivs}(K, A, M)$

proof -

\[
\begin{align*}
\begin{align*}
\text{fix } a & \ b \ \text{assume } A1: a \in K \ b \in K \text{ and } A2: a b = 0 \text{ and } A3: b \neq 0 \\
\text{from } \text{inv\_exists} \ A1 & \ A3 \ \text{obtain } c \ \text{where } I: c \in K \text{ and } II: b c = 1
\end{align*}
\end{align*}
\]

592
by auto
from A2 have a·b·c = 0·c by simp
with A1 I have a·(b·c) = 0
  using Ring_ZF_1_L11 Ring_ZF_1_L6 by simp
with A1 II have a=0 using Ring_ZF_1_L3 by simp }
then have ∀a∈K. ∀b∈K. a·b = 0 −→ a=0 ∨ b=0 by auto
then show thesis using HasNoZeroDivs_def by auto
qed

K₀ (the set of nonzero field elements is closed with respect to multiplication.

lemma (in field0) Field_ZF_1_L2: shows K₀ {is closed under} M
using Ring_ZF_1_L4 field_has_no_zero_divs Ring_ZF_1_L12 IsOpClosed_def by auto

Any nonzero element has a right inverse that is nonzero.

lemma (in field0) Field_ZF_1_L3: assumes A1: a∈K₀
shows ∃b∈K₀. a·b = 1
proof -
  from inv_exists A1 obtain b where b∈K and a·b = 1
  by auto
  with not_triv A1 show ∃b∈K₀. a·b = 1
    using Ring_ZF_1_L6 by auto
qed

If we remove zero, the field with multiplication becomes a group and we can
use all theorems proven in group0 context.

theorem (in field0) Field_ZF_1_L4: shows
  IsAgroup(K₀,restrict(M,K₀×K₀))
group0(K₀,restrict(M,K₀×K₀))
  1 = TheNeutralElement(K₀,restrict(M,K₀×K₀))
proof-
  let f = restrict(M,K₀×K₀)
  have M {is associative on} K
    K₀ ⊆ K K₀ {is closed under} M
    using Field_ZF_1_L1 IsAfield_def IsAring_def IsAgroup_def
    IsAmonoid_def Field_ZF_1_L2 by auto
  then have f {is associative on} K₀
    using func_ZF_4_L3 by simp
  moreover
  from not_triv have
    I: 1∈K₀ ∧ (∀a∈K₀. f(1,a) = a ∧ f(a,1) = a)
    using Ring_ZF_1_L2 Ring_ZF_1_L3 by auto
  then have ∃n∈K₀. ∀a∈K₀. f(n,a) = a ∧ f(a,n) = a
    by blast
  ultimately have II: IsAmonoid(K₀,f) using IsAmonoid_def
    by simp
  then have monoid0(K₀,f) using monoid0_def by simp

593
moreover note I
ultimately show $I = \text{TheNeutralElement}(K_0, f)$
by (rule monoid0.group0_1_L4)
then have $\forall a \in K_0. \exists b \in K_0. f(a, b) = \text{TheNeutralElement}(K_0, f)$
using Field_ZF_1_L3 by auto
with II show IsAgroup($K_0, f$) by (rule definition_of_group)
then show group0($K_0, f$) using group0_def by simp
qed

The inverse of a nonzero field element is nonzero.

lemma (in field0) Field_ZF_1_L5: assumes A1: $a \in K \ a \neq 0$
shows $a^{-1} \in K_0 \ (a^{-1})^2 \in K_0 \ a^{-1} \in K \ a^{-1} \neq 0$
proof -
from A1 have $a \in K_0$ by simp
then show $a^{-1} \in K_0$ using Field_ZF_1_L4 group0.inverse_in_group
by auto
then show $(a^{-1})^2 \in K_0 \ a^{-1} \in K \ a^{-1} \neq 0$
using Field_ZF_1_L2 IsOpClosed_def by auto
qed

The inverse is really the inverse.

lemma (in field0) Field_ZF_1_L6: assumes A1: $a \in K \ b \in K \ b \neq 0$
shows $a \cdot b \cdot b^{-1} = a \ a^{-1} \cdot b = a$
using assms Field_ZF_1_L5 Ring_ZF_1_L11 Field_ZF_1_L6 Ring_ZF_1_L3 by auto

A lemma with two field elements and cancelling.

lemma (in field0) Field_ZF_1_L7: assumes a\in K \ b\in K \ b\neq 0
shows a\cdot b\cdot b^{-1} = a \ a^{-1} \cdot b = a
using assms Field_ZF_1_L5 Ring_ZF_1_L11 Field_ZF_1_L6 Ring_ZF_1_L3 by auto

49.2 Equations and identities
This section deals with more specialized identities that are true in fields.
\[ \frac{a}{a^2} = \frac{1}{a} \, . \]

lemma (in field0) Field_ZF_2_L1: assumes A1: \( a \in K \) \( a \neq 0 \)
shows \( a \cdot (a^{-1})^2 = a^{-1} \)
proof -
  have \( a \cdot (a^{-1})^2 = a \cdot (a^{-1} \cdot a^{-1}) \) by simp
  also from A1 have \( \ldots = (a \cdot a^{-1}) \cdot a^{-1} \)
    using Field_ZF_1_L5 Ring_ZF_1_L11
    by simp
  also from A1 have \( \ldots = a^{-1} \)
    using Field_ZF_1_L6 Field_ZF_1_L5 Ring_ZF_1_L3
    by simp
  finally show \( a \cdot (a^{-1})^2 = a^{-1} \) by simp
qed

If we multiply two different numbers by a nonzero number, the results will
be different.

lemma (in field0) Field_ZF_2_L2:
  assumes \( a \in K \) \( b \in K \) \( c \in K \) \( a \neq b \) \( c \neq 0 \)
shows \( a \cdot c^{-1} \neq b \cdot c^{-1} \)
using assms field_has_no_zero_divs Field_ZF_1_L5 Ring_ZF_1_L12B
by simp

We can put a nonzero factor on the other side of non-identity (is this the
best way to call it?) changing it to the inverse.

lemma (in field0) Field_ZF_2_L3:
  assumes A1: \( a \in K \) \( b \in K \) \( b \neq 0 \)
  and A2: \( a \cdot b \neq c \)
shows \( a \neq c \cdot b^{-1} \)
proof -
  from A1 A2 have \( a \cdot b \cdot b^{-1} \neq c \cdot b^{-1} \)
    using Ring_ZF_1_L4 Field_ZF_2_L2 by simp
  with A1 show \( a \neq c \cdot b^{-1} \) using Field_ZF_1_L7
    by simp
qed

If if the inverse of \( b \) is different than \( a \), then the inverse of \( a \) is different than \( b \).

lemma (in field0) Field_ZF_2_L4:
  assumes \( a \in K \) \( a \neq 0 \) and \( b^{-1} \neq a \)
shows \( a^{-1} \neq b \)
using assms Field_ZF_1_L4 group0.group0_2_L11B
by simp

An identity with two field elements, one and an inverse.

lemma (in field0) Field_ZF_2_L5:
  assumes \( a \in K \) \( b \in K \) \( b \neq 0 \)
shows \( (1 + a \cdot b) \cdot b^{-1} = a + b^{-1} \)
using assms Ring_ZF_1_L4 Field_ZF_1_L5 Ring_ZF_1_L2 ring_oper_distr

595
An identity with three field elements, inverse and cancelling.

\textbf{lemma (in field0) Field_ZF_2_L6:} assumes \(A1: a \in K \quad b \in K \quad b \neq 0 \quad c \in K\)
shows \(a \cdot b \cdot (c \cdot b^{-1}) = a \cdot c\)

\textbf{proof -}
from \(A1\) have \(T: a \cdot b \in K \quad b^{-1} \in K\)
using Ring_ZF_1_L4 Field_ZF_1_L5 by auto
with \text{mult_commute} \(A1\) have \(a \cdot b \cdot (c \cdot b^{-1}) = a \cdot b \cdot (b^{-1} \cdot c)\)
using \text{IsCommutative_def} by simp
moreover from \(A1\ T\) have \(a \cdot b \in K \quad b^{-1} \in K \quad c \in K\)
by auto
then have \(a \cdot b \cdot b^{-1} \cdot c = a \cdot b \cdot (b^{-1} \cdot c)\)
by (rule Ring_ZF_1_L11)
ultimately have \(a \cdot b \cdot (c \cdot b^{-1}) = a \cdot b \cdot (b^{-1} \cdot c)\) by simp
with \(A1\) show \(a \cdot b \cdot (c \cdot b^{-1}) = a \cdot c\)
using Field_ZF_1_L7 by simp
\textbf{qed}

49.3 \( 1/0 = 0\)

In ZF if \(f : X \to Y\) and \(x \notin X\) we have \(f(x) = \emptyset\). Since \(\emptyset\) (the empty set) in ZF is the same as zero of natural numbers we can claim that \(1/0 = 0\) in certain sense. In this section we prove a theorem that makes makes it explicit.

The next locale extends the field0 locale to introduce notation for division operation.

\textbf{locale fielddd = field0 +}
fixes \text{division}
defines \text{division_def}[simp]: \(\text{division} \equiv \{p, \text{fst}(p) \cdot \text{snd}(p)^{-1}\}. p \in K \times K_0\)
fixes \text{fdiv (infixl} / 95)
defines \text{fdiv_def}[simp]: \(x/y \equiv \text{division}(x,y)\)

Division is a function on \(K \times K_0\) with values in \(K\).

\textbf{lemma (in fielddd) div_fun:} shows \(\text{division}: K \times K_0 \to K\)
\textbf{proof -}
have \(\forall p : K \times K_0. \text{fst}(p) \cdot \text{snd}(p)^{-1} \in K\)
\textbf{proof}
fix \(p\)
assume \(p \in K \times K_0\)
hence \(\text{fst}(p) \in K \quad \text{and} \quad \text{snd}(p) \in K_0\) by auto
then show \(\text{fst}(p) \cdot \text{snd}(p)^{-1} \in K\)
using Ring_ZF_1_L4 Field_ZF_1_L5 by auto
\textbf{qed}
then have \(\{p, \text{fst}(p) \cdot \text{snd}(p)^{-1}\}. p \in K \times K_0\): \(K \times K_0 \to K\)
by (rule ZF_fun_from_total)
thus thesis by simp
qed

So, really $1/0 = 0$. The essential lemma is apply_0 from standard Isabelle’s func.thy.

theorem (in fieldd) one_over_zero: shows $1/0 = 0$
proof
  have domain(division) = $K \times K_0$ using div_fun func1_1_L1
    by simp
  hence $(1,0) \notin$ domain(division) by auto
  then show thesis using apply_0 by simp
qed

end

50 Modules

theory Module_ZF imports Ring_ZF_3 Field_ZF

begin

A module is a generalization of the concept of a vector space in which scalars
do not form a field but a ring.

50.1 Definition and basic properties of modules

Let $R$ be a ring and $M$ be an abelian group. The most common definition
of a left $R$-module posits the existence of a scalar multiplication operation
$R \times M \rightarrow M$ satisfying certain four properties. Here we take a bit more
concise and abstract approach defining a module as a ring action on an
abelian group.

We know that endomorphisms of an abelian group $M$ form a ring with
pointwise addition as the additive operation and composition as the ring
multiplication. This assertion is a bit imprecise though as the domain of
pointwise addition is a binary operation on the space of functions $M \rightarrow M$
(i.e. its domain is $(M \rightarrow M) \times M \rightarrow M$) while we need the space of endo-
morphisms to be the domain of the ring addition and multiplication. There-
fore, to get the actual additive operation we need to restrict the pointwise
addition of functions $M \rightarrow M$ to the set of endomorphisms of $M$. Recall
from the Group_ZF_5 that the InEnd operator restricts an operation to the
set of endomorphisms and see the func_ZF theory for definitions of lifting an
operation on a set to a function space over that set.

definition EndAdd($M$,A) ≡ InEnd(A {lifted to function space over} $M,M,A$)
Similarly we define the multiplication in the ring of endomorphisms as the restriction of compositions to the endomorphisms of \( \mathcal{M} \). See the \texttt{func_ZF} theory for the definition of the \texttt{Composition} operator.

**Definition** \( \text{EndMult}(\mathcal{M}, A) \equiv \text{InEnd(Composition}(\mathcal{M}), \mathcal{M}, A) \)

We can now reformulate the theorem \texttt{end_is_ring} from the \texttt{Group_ZF} theory in terms of the addition and multiplication of endomorphisms defined above.

**Lemma** \texttt{(in abelian_group) end_is_ring1}:
- shows \texttt{IsAring(End(G,P),EndAdd(G,P),EndMult(G,P))}
- using \texttt{end_is_ring} unfolding \texttt{EndAdd_def} \texttt{EndMult_def} by simp

We define an action as a homomorphism into a space of endomorphisms (typically of some abelian group). In the definition below \( S \) is the set of scalars, \( A \) is the addition operation on this set, \( M \) is multiplication on the set, \( V \) is the group, \( A_V \) is the group operation, and \( H \) is the ring homomorphism that of the ring of scalars to the ring of endomorphisms of the group. On the right hand side of the definition \texttt{End(V,A_V)} is the set of endomorphisms. This definition is only ever used as part of the definition of a module and vector space, it’s just convenient to split it off to shorten the main definitions.

**Definition**
\[
\text{IsAction}(S,A,M,M,\mathcal{M},A,\mathcal{M},A,\mathcal{M},A,\mathcal{M},H) \equiv \text{ringHomomor(H,S,A,M,End(M,A),EndAdd(M,A,M),EndMult(M,A,M))}
\]

A module is a ring action on an abelian group.

**Definition** \texttt{IsLeftModule(S,A,M,M,\mathcal{M},A,\mathcal{M},A,\mathcal{M},A,\mathcal{M},H)}
- \texttt{IsAring(S,A,M)} \land \texttt{IsAgroup(M,M)} \land (\texttt{A_M} \{\text{is commutative on} \} \mathcal{M}) \land
- \texttt{IsAction(S,A,M,M,\mathcal{M},A,\mathcal{M},A,\mathcal{M},A,\mathcal{M},H)}

The next locale defines context (i.e. common assumptions and notation) when considering modules. We reuse notation from the \texttt{ring0} locale and add notation specific to modules. The addition and multiplication in the ring of scalars is denoted \( + \) and \( \cdot \), resp. The addition of module elements will be denoted \( +_V \). The multiplication (scaling) of scalars by module elements will be denoted \( \cdot_S \). \( \Theta \) is the zero module element, i.e. the neutral element of the abelian group of the module elements.

**Locale** \texttt{module0 = ring0 +}

fixes \( \mathcal{M} \), \( A_M \), \( H \)

assumes \texttt{mAbGr: IsAgroup(M,M) \land (A_M \{\text{is commutative on} \} M)}

assumes \texttt{mAction: IsAction(R,A,M,M,\mathcal{M},A,\mathcal{M},A,\mathcal{M},A,\mathcal{M},H)}

fixes zero_vec \( (\Theta) \)

defines \texttt{zero_vec_def [simp]: } \( \Theta \equiv \text{TheNeutralElement(M,M)} \)
fixes vAdd (infixl \text{+} 80)
defines vAdd_def [simp]: \( v_1 \text{+} v_2 \equiv M\langle v_1, v_2 \rangle \)

fixes scal (infix \text{\cdot} 90)
defines scal_def [simp]: \( s \text{\cdot} v \equiv (H(s))(v) \)

fixes negV (\text{\_})
defines negV_def [simp]: \(-v \equiv \text{GroupInv}(M,A_M)(v) \)

fixes vSub (infix \text{-} 80)
defines vSub_def [simp]: \( v_1 \text{-} v_2 \equiv v_1 \text{+} (-v_2) \)

We indeed talk about modules in the \text{module0} context.

lemma (in \text{module0}) module_in_module0: shows \( \text{IsLeftModule}(R,A,M,M,A_M,H) \)
  using ringAssum mAbGr mAction unfolding \text{IsLeftModule_def} by simp

Theorems proven in the \text{abelian_group} context are valid as applied to the \text{module0} context as applied to the abelian group of module elements.

lemma (in \text{module0}) abelian_group_valid_module0:
  shows abelian_group(M,A_M)
  using mAbGr group0_def abelian_group_def abelian_group_axioms_def by simp

Another way to state that theorems proven in the \text{abelian_group} context can be used in the \text{module0} context:

sublocale \text{module0} < mod_ab_gr: abelian_group \( M A_M \Theta vAdd negV \)
  using abelian_group_valid_module0 by auto

Theorems proven in the \text{ring_homo} context are valid in the \text{module0} context, as applied to ring \( R \) and the ring of endomorphisms of the group of module elements.

lemma (in \text{module0}) ring_homo_valid_module0:
  shows ring_homo(R,A,M,End(M,A_M),EndAdd(M,A_M),EndMult(M,A_M),H)
  using ringAssum mAction abelian_group_valid_module0 abelian_group.end_is_ring1 unfolding \text{IsAction_def ring_homo_def} by simp

Another way to make theorems proven in the \text{ring_homo} context available in the \text{module0} context:

sublocale \text{module0} < vec_act_homo: ring_homo R A M
  End(M,A_M) EndAdd(M,A_M) EndMult(M,A_M) H
  ringa ringminus ringsub ringm ringzero ringone ringtwo
In the ring of endomorphisms of the module the neutral element of the additive operation is the zero valued constant function. The neutral element of the multiplicative operation is the identity function.

```latex
lemma (in module0) add_mult_neut elems: shows
  TheNeutralElement(End(M, A_M), EndMult(M, A_M)) = id(M) and
  TheNeutralElement(End(M, A_M), EndAdd(M, A_M)) = ConstantFunction(M, Θ)
proof -
  show TheNeutralElement(End(M, A_M), EndMult(M, A_M)) = id(M)
    using mod_ab_gr.end_comp_monoid(2) unfolding EndMult_def
    by blast
  show TheNeutralElement(End(M, A_M), EndAdd(M, A_M)) = ConstantFunction(M, Θ)
    using mod_ab_gr.end_add_neut elem unfolding EndAdd_def by blast
qed
```

The value of the homomorphism defining the module is an endomorphism of the group of module elements and hence a function that maps the module into itself.

```latex
lemma (in module0) H_val_type: assumes r∈R shows
  H(r) ∈ End(M, A_M) and H(r):M→M
using mAction assms apply_funtype unfolding IsAction_def ringHomomor_def
End_def
by auto
```

In the module0 context the neutral element of addition of module elements is denoted Θ. Of course Θ is an element of the module.

```latex
lemma (in module0) zero_in_mod: shows Θ ∈ M
using mod_ab_gr.group0_2_L2 by simp
```

Θ is indeed the neutral element of addition of module elements.

```latex
lemma (in module0) zero_neutral: assumes x∈M
shows x +_V Θ = x and Θ +_V x = x
using assms mod_ab_gr.group0_2_L2 by simp_all
```
50.2 Module axioms

A more common definition of a module assumes that $R$ is a ring, $V$ is an abelian group and lists a couple of properties that the multiplications of scalars (elements of $R$) by the elements of the module $V$ should have. In this section we show that the definition of a module as a ring action on an abelian group $V$ implies these properties.

The scalar multiplication is distributive with respect to the module addition.

lemma (in module0) module_ax1: assumes $r \in R \ x \in M \ y \in M$
shows $r \cdot (x + y) = r \cdot x + r \cdot y$
using assms $\text{val_type}(1) \ mod_ab_gr.endomor_eq$ by simp

The scalar addition is distributive with respect to scalar multiplication.

lemma (in module0) module_ax2: assumes $r \in R \ s \in R \ x \in M$
shows $(r + s) \cdot x = r \cdot x + s \cdot x$
using assms val_type(1) mod_ab_gr.end_pointwise_add_val unfolding EndAdd_def by simp

Multiplication by scalars is associative with multiplication of scalars.

lemma (in module0) module_ax3: assumes $r \in R \ s \in R \ x \in M$
shows $(r \cdot s) \cdot x = r \cdot (s \cdot x)$
proof -
  let $e = \text{EndMult}(M,A_M)\langle H(r),H(s) \rangle$
  have $(r \cdot s) \cdot x = (H(r \cdot s))(x)$ by simp
  also have $(H(r \cdot s))(x) = e(x)$
  proof -
    from mAction assms(1,2) have $H(r \cdot s) = e$
    using vec_act_homo.homomor_dest_mult unfolding IsAction_def by blast
    then show thesis by (rule same_constr)
  qed
  also have $e(x) = r \cdot (s \cdot x)$
  proof -
    from assms(1,2) have $e(x) = (\text{Composition}(M)(H(r),H(s)))(x)$
    using $\text{val_type}(1)$ unfolding EndMult_def by simp
    also from assms have ... = $r \cdot (s \cdot x)$ using $\text{val_type}(2)$ func_ZF_5_L3 by simp
    finally show $e(x) = r \cdot (s \cdot x)$ by blast
  qed
  finally show thesis by simp
qed

Scaling a module element by one gives the same module element.

lemma (in module0) module_ax4: assumes $x \in M$
shows $1 \cdot x = x$
proof -
  let $n = \text{TheNeutralElement}(\text{End}(M,A_M),\text{EndMult}(M,A_M))$
  from mAction have $H(\text{TheNeutralElement}(R,H)) = n$
unfolding IsAction_def ringHomomor_def by blast
moreover have TheNeutralElement(R,M) = 1 by simp
ultimately have H(1) = n by blast
also have n = id(M) by (rule add_mult_neut elems)
finally have H(1) = id(M) by simp
with assms show 1·x = x by simp
qed

end

51 Vector spaces

theory VectorSpace_ZF imports Module_ZF

begin
Vector spaces have a long history of applications in mathematics and physics. To this collection of applications a new one has been added recently - Large Language Models. It turned out that representing words, phrases and documents as vectors in a high-dimensional vector space provides an effective way to capture semantic relationships and emulate contextual understanding. This theory has nothing to do with LLM’s however - it just defines vector space as a mathematical structure as it has been understood from at least the beginning of the XXth century.

51.1 Definition and basic properties of vector spaces

The canonical example of a vector space is \( \mathbb{R}^n \) - the set of \( n \)-tuples of real numbers. We can add them adding respective coordinates and scale them by multiplying all coordinates by the same number. In a more abstract approach we start with an abelian group (of vectors) and a field (of scalars) and define an operation of multiplying a vector by a scalar so that the distributive properties

\[
(\text{scalars } s, s_1, s_2 \text{ and vectors } v, v_1, v_2) \Rightarrow
s(v_1 + v_2) = sv_1 + sv_2 \text{ and } (s_1 + s_2)v = s_1v + s_2v
\]

are satisfied for any scalars \( s, s_1, s_2 \) and vectors \( v, v_1, v_2 \).

A vector space is a field action on an abelian group.

definition IsVectorSpace(S,A,M,V,A_V,H) ≡
IsAfield(S,A,M) ∧ IsAgroup(V,A_V) ∧ (A_V {is commutative on} V) ∧ IsAction(S,A,M,V,A_V,H)

The next locale defines context (i.e. common assumptions and notation) when considering vector spaces. We reuse notation from the field0 locale adding more similarly to the module0 locale.

locale vector_space0 = field0 +
fixes V A_V H
assumes mAbGr: IsAgroup(V,A_V) ∧ (A_V {is commutative on} V)
assumes mAction: IsAction(K,A,M,V,A\textsubscript{V},H)

fixes zero_vec (θ)
defines zero_vec_def [simp]: θ ≡ TheNeutralElement(V,A\textsubscript{V})

fixes vAdd (infixl \textsubscript{+} 80)
defines vAdd_def [simp]: v\textsubscript{1} \textsubscript{+} v\textsubscript{2} ≡ A\textsubscript{V}(v\textsubscript{1},v\textsubscript{2})

fixes scal (infix \cdot 90)
defines scal_def [simp]: s \cdot v ≡ (H(s))(v)

fixes negV (⁻)
defines negV_def [simp]: v⁻ ≡ GroupInv(V,A\textsubscript{V})(v)

fixes vSub (infix - 80)
defines vSub_def [simp]: v\textsubscript{1} - v\textsubscript{2} ≡ v\textsubscript{1} + V(⁻v\textsubscript{2})

We indeed talk about vector spaces in the vector_space0 context.

lemma (in vector_space0) V_vec_space: shows IsVectorSpace(K,A,M,V,A\textsubscript{V},H)
  using mAbGr mAction Field_ZF_1_L1 unfolding IsVectorSpace_def by simp

If a quintuple of sets forms a vector space then the assumptions of the vector_space0 hold for those sets.

lemma vec_spce_vec_spce_contxt: assumes IsVectorSpace(K,A,M,V,A\textsubscript{V},H)
  shows vector_space0(K, A, M, V, A\textsubscript{V}, H)
proof
  from assms show
    IsAring(K, A, M) M {is commutative on} K
    TheNeutralElement(K, A) \neq TheNeutralElement(K, M)
    \forall x\in K. x \neq TheNeutralElement(K, A) \rightarrow (\exists y\in K. M(x, y) = TheNeutralElement(K, M))
    IsAgroup(V, A\textsubscript{V}) \land A\textsubscript{V} {is commutative on} V
    IsAction(K, A, M, V, A\textsubscript{V}, H)
    unfolding IsAfield_def IsVectorSpace_def by simp_all
  qed

The assumptions of module0 context hold in the vector_space0 context.

lemma (in vector_space0) vec_spce_mod: shows module0(K, A, M, V, A\textsubscript{V}, H)
proof
  from mAbGr show IsAgroup(V,A\textsubscript{V}) \land (A\textsubscript{V} {is commutative on} V) by simp
  from mAction show IsAction(K,A,M,V,A\textsubscript{V},H) by simp
  qed

Propositions proven in the module0 context are valid in the vector_space0 context.

sublocale vector_space0 < vspce_mod: module0 K A M
51.2 Vector space axioms

In this section we show that the definition of a vector space as a field action on an abelian group implies the vector space axioms as listed on Wikipedia (March 2024). The first four axioms just state that vectors with addition form an abelian group. That is fine of course, but in such case the axioms for scalars being a field should be listed too, and they are not. The entry on modules is more consistent, it states that module elements form an abelian group, scalars form a ring and lists only four properties of multiplication of scalars by vectors as module axioms. The remaining four axioms are just restatement of module axioms and since vector spaces are modules we can prove them by referring to the module axioms proven in the module0 context.

Vector addition is associative.

lemma (in vector_space0) vec_spce_ax1: assumes u∈V v∈V w∈V
shows u +_{V} (v +_{V} w) = (u +_{V} v) +_{V} w
using assms vspce_mod.mod_ab_gr.group_oper_assoc by simp

Vector addition is commutative.

lemma (in vector_space0) vec_spce_ax2: assumes u∈V v∈V
shows u +_{V} v = v +_{V} u
using mAbGr assms unfolding IsCommutative_def by simp

The zero vector is a vector.

lemma (in vector_space0) vec_spce_ax3a: shows Θ ∈ V
using vspce_mod.zero_in_mod by simp

The zero vector is the neutral element of addition of vectors.

lemma (in vector_space0) vec_spce_ax3b: assumes v∈V shows v +_{V} Θ = v
using assms vspce_mod.zero_neutral by simp

The additive inverse of a vector is a vector.

lemma (in vector_space0) vec_spce_ax4a: assumes v∈V shows (−v) ∈ V
using assms vspce_mod.mod_ab_gr.inverse_in_group by simp

Sum of a vector and its additive inverse is the zero vector.

lemma (in vector_space0) vec_spce_ax4b: assumes v∈V
shows v +_{V} (−v) = Θ
using assms vspce_mod.mod_ab_gr.group0_2_L6 by simp

Scalar multiplication and field multiplication are "compatible" (as Wikipedia calls it).
lemma (in vector_space0) vec_spce_ax5: assumes \( x \in K \) \( y \in K \) \( v \in V \) shows \( x \cdot (y \cdot v) = (x \cdot y) \cdot v \)
using assms vspce_mod.module_ax3 by simp

Multiplying the identity element of the field by a vector gives the vector.

lemma (in vector_space0) vec_spce_ax6: assumes \( v \in V \) shows \( 1 \cdot v = v \)
using assms vspce_mod.module_ax4 by simp

Scalar multiplication is distributive with respect to vector addition.

lemma (in vector_space0) vec_spce_ax7: assumes \( x \in K \) \( u \in V \) \( v \in V \) shows \( x \cdot (u + v) = x \cdot u + x \cdot v \)
using assms vspce_mod.module_ax1 by simp

Scalar multiplication is distributive with respect to field addition.

lemma (in vector_space0) vec_spce_ax8: assumes \( x \in K \) \( y \in K \) \( v \in V \) shows \( (x + y) \cdot v = x \cdot v + y \cdot v \)
using assms vspce_mod.module_ax2 by simp

end

52 Ordered fields

theory OrderedField_ZF imports OrderedRing_ZF Field_ZF

begin

This theory covers basic facts about ordered fields.

52.1 Definition and basic properties

Here we define ordered fields and prove their basic properties.

Ordered field is a not trivial ordered ring such that all non-zero elements have
an inverse. We define the notion of being a ordered field as a statement about
four sets. The first set, denoted \( K \) is the carrier of the field. The second set,
denoted \( A \) represents the additive operation on \( K \) (recall that in ZF set theory
functions are sets). The third set \( M \) represents the multiplicative operation
on \( K \). The fourth set \( r \) is the order relation on \( K \).

definition
IsAnOrdField(K,A,M,r) ≡ (IsAnOrdRing(K,A,M,r) ∧
(M {is commutative on} K) ∧
TheNeutralElement(K,A) ≠ TheNeutralElement(K,M) ∧
(∀a∈K. a≠TheNeutralElement(K,A)→
(∃b∈K. M(a,b) = TheNeutralElement(K,M))))

The next context (locale) defines notation used for ordered fields. We do
that by extending the notation defined in the ring1 context that is used
for ordered rings and adding some assumptions to make sure we are talking about ordered fields in this context. We should rename the carrier from $R$ used in the \texttt{ring1} context to $K$, more appropriate for fields. Theoretically the Isar locale facility supports such renaming, but we experienced difficulties using some lemmas from \texttt{ring1} locale after renaming.

locale field1 = ring1 +

assumes mult_commute: M \{is commutative on\} R

assumes not_triv: $0 \neq 1$

assumes inv_exists: $\forall a \in R. a \neq 0 \rightarrow (\exists b \in R. a \cdot b = 1)$

fixes non_zero (R\{0\})

defines non_zero_def[simp]: $R_{\{0\}} \equiv R - \{0\}$

fixes inv (_\{-1\})

defines inv_def[simp]: $a^{-1} \equiv \text{GroupInv}(R_{\{0\}}, \text{restrict}(M, R_{\{0\}} \times R_{\{0\}}))(a)$

The next lemma assures us that we are talking fields in the \texttt{field1} context.

lemma (in field1) OrdField_ZF_1_L1: shows IsAnOrdField(R,A,M,r)

using OrdRing_ZF_1_L1 mult_commute not_triv inv_exists IsAnOrdField_def by simp

Ordered field is a field, of course.

lemma OrdField_ZF_1_L1A: assumes IsAnOrdField(K,A,M,r) shows IsAfield(K,A,M)

using assms IsAnOrdField_def IsAnOrdRing_def IsAfield_def by simp

Theorems proven in \texttt{field0} (about fields) context are valid in the \texttt{field1} context (about ordered fields).

lemma (in field1) OrdField_ZF_1_L1B: shows field0(R,A,M)

using OrdField_ZF_1_L1 OrdField_ZF_1_L1A field_field0 by simp

We can use theorems proven in the \texttt{field1} context whenever we talk about an ordered field.

lemma OrdField_ZF_1_L2: assumes IsAnOrdField(K,A,M,r) shows field1(K,A,M,r)

using assms IsAnOrdField_def OrdRing_ZF_1_L2 ring1_def IsAnOrdField_def field1_axioms_def field1_def by auto

In ordered rings the existence of a right inverse for all positive elements implies the existence of an inverse for all non zero elements.

lemma (in ring1) OrdField_ZF_1_L3:
assumes A1: \( \forall a \in R_+. \exists b \in R. \ a \cdot b = 1 \) and A2: \( c \in R \ c \neq 0 \)

shows \( \exists b \in R. \ c \cdot b = 1 \)

proof -

\{
  assume c \in R_+
  with A1 have \( \exists b \in R. \ c \cdot b = 1 \) by simp 
\}

moreover

\{
  assume c \notin R_+
  with A2 have \( (\neg c) \in R_+ \)
    using OrdRing_ZF_3_L2A by simp
  with A1 obtain b where b \in R and \( (\neg c) \cdot b = 1 \)
    by auto
  with A2 have \( (\neg b) \in R \ c \cdot (\neg b) = 1 \)
    using Ring_ZF_1_L3 Ring_ZF_1_L7 by auto
  then have \( \exists b \in R. \ c \cdot b = 1 \) by auto 
\}

ultimately show thesis by blast

qed

Ordered fields are easier to deal with, because it is sufficient to show the existence of an inverse for the set of positive elements.

lemma (in ring1) OrdField_ZF_1_L4:
assumes 0 \( \neq 1 \) and M {is commutative on} R
and \( \forall a \in R_+. \exists b \in R. \ a \cdot b = 1 \)
shows IsAnOrdField(R,A,M,r)
using assms OrdRing_ZF_1_L1 OrdField_ZF_1_L3 IsAnOrdField_def
by simp

The set of positive field elements is closed under multiplication.

lemma (in field1) OrdField_ZF_1_L5:
shows R_+ {is closed under} M
using OrdField_ZF_1_L1B field0.field_has_no_zero_divs OrdRing_ZF_3_L3
by simp

The set of positive field elements is closed under multiplication: the explicit version.

lemma (in field1) pos_mul_closed:
assumes A1: \( 0 < a \quad 0 < b \)
shows \( 0 < a \cdot b \)
proof -
from A1 have a \( \in R_+ \) and \( b \in R_+ \)
  using OrdRing_ZF_3_L14 by auto
then show \( 0 < a \cdot b \)
  using OrdField_ZF_1_L5 IsOpClosed_def PositiveSet_def
  by simp

qed

In fields square of a nonzero element is positive.

lemma (in field1) OrdField_ZF_1_L6:
assumes a\( \in R \quad a \neq 0 \)
shows \( a^2 \in R_+ \)
using assms OrdField_ZF_1_L1B field0.field_has_no_zero_divs
OrdRing_ZF_3_L15 by simp

The next lemma restates the fact Field_ZF that out notation for the field inverse means what it is supposed to mean.

lemma (in field1) OrdField_ZF_1_L7: assumes a∈R a≠0 shows a·(a⁻¹) = 1 (a⁻¹)·a = 1 using assms OrdField_ZF_1_L1B field0.Field_ZF_1_L6 by auto

A simple lemma about multiplication and cancelling of a positive field element.

lemma (in field1) OrdField_ZF_1_L7A: assumes A1: a∈R b∈R + shows a·b·b⁻¹ = a a⁻¹·b = a proof - from A1 have b∈R b≠0 using PositiveSet_def by auto with A1 show a·b·b⁻¹ = a and a⁻¹·b = a using OrdField_ZF_1_L1B field0.Field_ZF_1_L7 by auto qed

Some properties of the inverse of a positive element.

lemma (in field1) OrdField_ZF_1_L8: assumes A1: a∈R⁺ shows a⁻¹ ∈ R⁺ a·(a⁻¹) = 1 (a⁻¹)·a = 1 proof - from A1 have I: a∈R a≠0 using PositiveSet_def by auto with A1 have a·(a⁻¹)² ∈ R⁺ using OrdField_ZF_1_L1B field0.Field_ZF_1_L5 OrdField_ZF_1_L6 OrdField_ZF_1_L5 IsOpClosed_def by simp with I show a⁻¹ ∈ R⁺ using OrdField_ZF_1_L1B field0.Field_ZF_2_L1 by simp from I show a·(a⁻¹) = 1 (a⁻¹)·a = 1 using OrdField_ZF_1_L7 by auto qed

If a is smaller than b, then (b - a)⁻¹ is positive.

lemma (in field1) OrdField_ZF_1_L9: assumes a<b shows (b-a)⁻¹ ∈ R⁺ using assms OrdRing_ZF_1_L14 OrdField_ZF_1_L8 by simp

In ordered fields if at least one of a, b is not zero, then a² + b² > 0, in particular a² + b² ≠ 0 and exists the (multiplicative) inverse of a² + b².
lemma (in field1) OrdField_ZF_1_L10:
  assumes A1: a ∈ R b ∈ R and A2: a ≠ 0 ∨ b ≠ 0
  shows 0 < a^2 + b^2 and ∃ c ∈ R. (a^2 + b^2)·c = 1
proof -
  from A1 A2 show 0 < a^2 + b^2
    using OrdField_ZF_1_L1B field0.field_has_no_zero_divs
    OrdRing_ZF_3_L19 by simp
  then have
    (a^2 + b^2)^(-1) ∈ R and (a^2 + b^2)·(a^2 + b^2)^(-1) = 1
    using OrdRing_ZF_1_L3 PositiveSet_def OrdField_ZF_1_L8
    by auto
  then show ∃ c ∈ R. (a^2 + b^2)·c = 1 by auto
qed

52.2 Inequalities

In this section we develop tools to deal inequalities in fields.

We can multiply strict inequality by a positive element.

lemma (in field1) OrdField_ZF_2_L1:
  assumes a < b and c ∈ R
  shows a·c < b·c
using assms OrdField_ZF_1_L1B field0.field_has_no_zero_divs
  OrdRing_ZF_3_L13 by simp
A special case of OrdField_ZF_2_L1 when we multiply an inverse by an ele-

lemma (in field1) OrdField_ZF_2_L2:
  assumes A1: a ∈ R+ and A2: a^(-1) < b
  shows 1 < b·a
proof -
  from A1 A2 have (a^(-1))·a < b·a
    using OrdField_ZF_2_L1 by simp
  with A1 show 1 < b·a
    using OrdField_ZF_1_L8 by simp
qed

We can multiply an inequality by the inverse of a positive element.

lemma (in field1) OrdField_ZF_2_L3:
  assumes a ≤ b and c ∈ R+ shows a·c^(-1) ≤ b·c^(-1)
using assms OrdField_ZF_1_L8 OrdRing_ZF_1_L9A
  by simp
We can multiply a strict inequality by a positive element or its inverse.

lemma (in field1) OrdField_ZF_2_L4:
  assumes a < b and c ∈ R+
  shows
\begin{verbatim}
 a \cdot c < b \cdot c
 c \cdot a < c \cdot b
 a \cdot c^{-1} < b \cdot c^{-1}
 using assms OrdField_ZF_1_L1B field0.field_has_no_zero_divs
 OrdField_ZF_1_L8 OrdRing_ZF_3_L13 by auto

We can put a positive factor on the other side of an inequality, changing it to its inverse.

lemma (in field1) OrdField_ZF_2_L5:
  assumes A1: a \in R b \in R \+
  and A2: a \cdot b \leq c
  shows a \leq c \cdot b^{-1}
proof -
  from A1 A2 have a \cdot b^{-1} \leq c \cdot b^{-1}
    using OrdField_ZF_2_L3 by simp
  with A1 show a \leq c \cdot b^{-1} using OrdField_ZF_1_L7A
    by simp
qed

We can put a positive factor on the other side of an inequality, changing it to its inverse, version with a product initially on the right hand side.

lemma (in field1) OrdField_ZF_2_L5A:
  assumes A1: b \in R c \in R \+
  and A2: a \leq b \cdot c
  shows a \cdot c^{-1} \leq b
proof -
  from A1 A2 have a \cdot c^{-1} \leq b \cdot c^{-1}
    using OrdField_ZF_2_L3 by simp
  with A1 show a \cdot c^{-1} \leq b using OrdField_ZF_1_L7A
    by simp
qed

We can put a positive factor on the other side of a strict inequality, changing it to its inverse, version with a product initially on the left hand side.

lemma (in field1) OrdField_ZF_2_L6:
  assumes A1: a \in R b \in R \+
  and A2: a \cdot b < c
  shows a < c \cdot b^{-1}
proof -
  from A1 A2 have a \cdot b^{-1} < c \cdot b^{-1}
    using OrdField_ZF_2_L4 by simp
  with A1 show a < c \cdot b^{-1} using OrdField_ZF_1_L7A
    by simp
qed

We can put a positive factor on the other side of a strict inequality, changing it to its inverse, version with a product initially on the right hand side.

lemma (in field1) OrdField_ZF_2_L6A:
  assumes A1: b \in R c \in R \+
  and A2: a < b \cdot c
  shows a \cdot c^{-1} < b
proof -
\end{verbatim}
from A1 A2 have a·c⁻¹ < b·c⁻¹ using OrdField_ZF_2_L4 by simp
with A1 show a·c⁻¹ < b using OrdField_ZF_1_L7A by simp
qed

Sometimes we can reverse an inequality by taking inverse on both sides.

lemma (in field1) OrdField_ZF_2_L7:
  assumes A1: a∈R⁺ and A2: a⁻¹ ≤ b
  shows b⁻¹ ≤ a
proof -
  from A1 have a⁻¹ ∈ R⁺ using OrdField_ZF_1_L8 by simp
  with A2 have b ∈ R⁺ using OrdRing_ZF_3_L7 by blast
  then have T: b ∈ R⁺ b⁻¹ ∈ R⁺ using OrdField_ZF_1_L8 by auto
  with A1 A2 have b⁻¹·a⁻¹.a ≤ b⁻¹·b.a using OrdRing_ZF_1_L9A by simp
  moreover from A1 A2 T have
      b⁻¹ ∈ R a∈R a≠0 b∈R b≠0 using PositiveSet_def OrdRing_ZF_1_L3 by auto
      then have b⁻¹·a⁻¹.a = b⁻¹ and b⁻¹·b.a = a using OrdField_ZF_1_L1B field0.Field_ZF_1_L7 field0.Field_ZF_1_L6 Ring_ZF_1_L3 by auto
      ultimately show b⁻¹ ≤ a by simp
qed

Sometimes we can reverse a strict inequality by taking inverse on both sides.

lemma (in field1) OrdField_ZF_2_L8:
  assumes A1: a∈R⁺ and A2: a⁻¹ < b
  shows b⁻¹ < a
proof -
  from A1 A2 have a⁻¹ ∈ R⁺ a⁻¹ ≤ b using OrdField_ZF_1_L8 by auto
  then have b ∈ R⁺ using OrdRing_ZF_3_L7 by blast
  then have b∈R b≠0 using PositiveSet_def by auto
  with A2 have b⁻¹ ≠ a using OrdField_ZF_1_L1B field0.Field_ZF_1_L4 by simp
  with A1 A2 show b⁻¹ < a using OrdField_ZF_2_L7 by simp
qed

A technical lemma about solving a strict inequality with three field elements and inverse of a difference.
lemma (in field1) OrdField_ZF_2_L9:
  assumes A1: a<b and A2: (b-a)^{-1} < c
  shows 1 + a·c < b·c
proof -
  from A1 A2 have (b-a)^{-1} ∈ R_+ (b-a)^{-1} ≤ c
    using OrdField_ZF_1_L9 by auto
  then have T1: c ∈ R_+ using OrdRing_ZF_3_L7 by blast
  with A1 A2 have T2:
    a∈R b∈R c∈R c≠0 c^{-1} ∈ R
    using OrdRing_ZF_1_L3 OrdField_ZF_1_L8 PositiveSet_def
    by auto
  with T1 T2 have (c^{-1} + a)·c < b·c
    using Ring_ZF_2_L1A OrdField_ZF_2_L8 ring_strict_ord_trans_inv
    by simp
  with T1 T2 show 1 + a·c < b·c
    using ring_oper_distr OrdField_ZF_1_L8
    by simp
qed

52.3 Definition of real numbers

The only purpose of this section is to define what does it mean to be a model of real numbers.

We define model of real numbers as any quadruple of sets (K,A,M,r) such that (K,A,M,r) is an ordered field and the order relation r is complete, that is every set that is nonempty and bounded above in this relation has a supremum.

definition
  IsAmodelOfReals(K,A,M,r) ≡ IsAnOrdField(K,A,M,r) ∧ (r {is complete})
end

53 Integers - introduction

theory Int_ZF_1 imports OrderedGroup_ZF_1 Finite_ZF_1 ZF.Int Nat_ZF_1

begin

This theory file is an interface between the old-style Isabelle (ZF logic) material on integers and the IsarMathLib project. Here we redefine the meta-level operations on integers (addition and multiplication) to convert them to ZF-functions and show that integers form a commutative group with respect to addition and commutative monoid with respect to multiplication. Similarly, we redefine the order on integers as a relation, that is a subset of
We show that a subset of integers is bounded iff it is finite. As we are forced to use standard Isabelle notation with all these dollar signs, sharps etc. to denote “type coercions” (?) the notation is often ugly and difficult to read.

53.1 Addition and multiplication as ZF-functions.

In this section we provide definitions of addition and multiplication as subsets of \((\mathbb{Z} \times \mathbb{Z}) \times \mathbb{Z}\). We use the (higher order) relation defined in the standard \texttt{Int} theory to define a subset of \(\mathbb{Z} \times \mathbb{Z}\) that constitutes the ZF order relation corresponding to it. We define the set of positive integers using the notion of positive set from the \texttt{OrderedGroup_ZF} theory.

Definition of addition of integers as a binary operation on \(\mathbb{Z}\). Recall that in standard Isabelle/ZF \(\mathbb{Z}\) is the set of integers and the sum of integers is denoted by prepending \(\texttt{+}\) with a dollar sign.

\textbf{definition}
\[
\text{IntegerAddition} \equiv \{ (x,c) \in (\mathbb{Z} \times \mathbb{Z}) \times \mathbb{Z}. \text{fst}(x) \texttt{+} \text{snd}(x) = c \}
\]

Definition of multiplication of integers as a binary operation on \(\mathbb{Z}\). In standard Isabelle/ZF product of integers is denoted by prepending the dollar sign to \(\texttt{*}\).

\textbf{definition}
\[
\text{IntegerMultiplication} \equiv \{ (x,c) \in (\mathbb{Z} \times \mathbb{Z}) \times \mathbb{Z}. \text{fst}(x) \texttt{*} \text{snd}(x) = c \}
\]

Definition of natural order on integers as a relation on \(\mathbb{Z}\). In the standard Isabelle/ZF the inequality relation on integers is denoted \(\leq\) prepended with the dollar sign.

\textbf{definition}
\[
\text{IntegerOrder} \equiv \{ p \in \mathbb{Z} \times \mathbb{Z}. \text{fst}(p) \texttt{\leq} \text{snd}(p) \}
\]

This defines the set of positive integers.

\textbf{definition}
\[
\text{PositiveIntegers} \equiv \text{PositiveSet}(\mathbb{Z}, \text{IntegerAddition}, \text{IntegerOrder})
\]

IntegerAddition and IntegerMultiplication are functions on \(\mathbb{Z} \times \mathbb{Z}\).

\textbf{lemma} \texttt{Int_ZF_1_L1}:
\begin{itemize}
  \item \texttt{IntegerAddition} : \(\mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}\)
  \item \texttt{IntegerMultiplication} : \(\mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}\)
\end{itemize}

\textbf{proof}
\begin{itemize}
  \item have \[
  \{ (x,c) \in (\mathbb{Z} \times \mathbb{Z}) \times \mathbb{Z}. \text{fst}(x) \texttt{+} \text{snd}(x) = c \} \in \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}
  \]
  \item have \[
  \{ (x,c) \in (\mathbb{Z} \times \mathbb{Z}) \times \mathbb{Z}. \text{fst}(x) \texttt{*} \text{snd}(x) = c \} \in \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}
  \]
  \item using \texttt{func1_1_L1A} by \texttt{auto}
\end{itemize}

\textbf{then show} \texttt{IntegerAddition} : \(\mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}\)
\textbf{IntegerMultiplication} : int×int \rightarrow int \\
using IntegerAddition\_def IntegerMultiplication\_def by auto \\
qed

The next context (locale) defines notation used for integers. We define 0 to denote the neutral element of addition, 1 as the unit of the multiplicative monoid. We introduce notation \( m \leq n \) for integers and write \( m \ldots n \) to denote the integer interval with endpoints in \( m \) and \( n \). \( \text{abs}(m) \) means the absolute value of \( m \). This is a function defined in \texttt{OrderedGroup} that assigns \( x \) to itself if \( x \) is positive and assigns the opposite of \( x \) if \( x \leq 0 \). Unfortunately we cannot use the \(|\cdot|\) notation as in the \texttt{OrderedGroup} theory as this notation has been hogged by the standard Isabelle's \texttt{Int} theory. The notation \(-A\) where \( A \) is a subset of integers means the set \( \{-m : m \in A\} \). The symbol \( \text{maxf}(f,A) \) denotes the maximum of function \( f \) over the set \( A \). We also introduce a similar notation for the minimum.

locale int0 = \\
fixes \textit{ints} (\( Z \)) \\
defines \textit{ints\_def} [simp]: \( Z \equiv \text{int} \) \\
fixes \textit{ia} (infixl + 69) \\
defines \textit{ia\_def} [simp]: \( a+b \equiv \text{IntegerAddition}(\ a,b) \) \\
fixes \textit{iminus} (- _ 72) \\
defines \textit{rminus\_def} [simp]: \( -a \equiv \text{GroupInv}(Z,\text{IntegerAddition})(a) \) \\
fixes \textit{isub} (infixl - 69) \\
defines \textit{isub\_def} [simp]: \( a-b \equiv a+ (- b) \) \\
fixes \textit{imult} (infixl \cdot 70) \\
defines \textit{imult\_def} [simp]: \( a\cdot b \equiv \text{IntegerMultiplication}(\ a,b) \) \\
fixes \textit{setneg} (- _ 72) \\
defines \textit{setneg\_def} [simp]: \( -A \equiv \text{GroupInv}(Z,\text{IntegerAddition})(A) \) \\
fixes \textit{izero} (0) \\
defines \textit{izero\_def} [simp]: \( 0 \equiv \text{TheNeutralElement}(Z,\text{IntegerAddition}) \) \\
fixes \textit{ione} (1) \\
defines \textit{ione\_def} [simp]: \( 1 \equiv \text{TheNeutralElement}(Z,\text{IntegerMultiplication}) \) \\
fixes \textit{itwo} (2) \\
defines \textit{itwo\_def} [simp]: \( 2 \equiv 1+1 \) \\
fixes \textit{ithree} (3) \\
defines \textit{ithree\_def} [simp]: \( 3 \equiv 2+1 \) \\
fixes \textit{nonnegative} (\( Z^+ \))
defines nonnegative_def [simp]:
\[ Z^+ \equiv \text{Nonnegative}(\mathbb{Z},\text{IntegerAddition},\text{IntegerOrder}) \]

fixes positive \((Z_+)^\) defines positive_def [simp]:
\[ Z_+ \equiv \text{PositiveSet}(\mathbb{Z},\text{IntegerAddition},\text{IntegerOrder}) \]

fixes abs defines abs_def [simp]:
\[ \text{abs}(m) \equiv \text{AbsoluteValue}(\mathbb{Z},\text{IntegerAddition},\text{IntegerOrder})(m) \]

fixes lesseq (infix \(\leq\)) defines lesseq_def [simp]: \(m \leq n \equiv (m,n) \in \text{IntegerOrder}\)

fixes interval (infix .. 70) defines interval_def [simp]: \(m..n \equiv \text{Interval}(\text{IntegerOrder},m,n)\)

fixes maxf defines maxf_def [simp]: \(\text{maxf}(f,A) \equiv \text{Maximum}(\text{IntegerOrder},f(A))\)

fixes minf defines minf_def [simp]: \(\text{minf}(f,A) \equiv \text{Minimum}(\text{IntegerOrder},f(A))\)

Integer addition and multiplication are associative.

lemma (in int0) Int_ZF_1_L3: assumes \(x \in \mathbb{Z} \quad y \in \mathbb{Z} \quad z \in \mathbb{Z}\) shows \(x+y+z = x+(y+z) \quad x\cdot y \cdot z = x\cdot(y\cdot z)\)
using assms Int_ZF_1_L2 zadd_assoc zmult_assoc by auto
Integer addition and multiplication are commutative.

**Lemma (in int0) Int_ZF_1_L4:**
- Assumes \( x \in \mathbb{Z} \) and \( y \in \mathbb{Z} \).
- Shows \( x+y = y+x \) and \( x \cdot y = y \cdot x \).
- Using assms Int_ZF_1_L2, zadd_commute, zmult_commute.
- By auto.

Zero is neutral for addition and one for multiplication.

**Lemma (in int0) Int_ZF_1_L5:**
- Assumes \( A_1: x \in \mathbb{Z} \).
- Shows \( (\# 0) + x = x \land x + (\# 0) = x \).
- \( (\# 1) \cdot x = x \land x \cdot (\# 1) = x \).
- Proof -
  - From \( A_1 \) show \( (\# 0) + x = x \land x + (\# 0) = x \) using Int_ZF_1_L2, zadd_int0, Int_ZF_1_L4 by simp.
  - From \( A_1 \) have \( (\# 1) \cdot x = x \) using Int_ZF_1_L2, zmult_int1 by simp.
  - With \( A_1 \) show \( (\# 1) \cdot x = x \land x \cdot (\# 1) = x \) using Int_ZF_1_L4 by simp.
- QED.

Zero is neutral for addition and one for multiplication.

**Lemma (in int0) Int_ZF_1_L6:**
- Shows \( (\# 0) \in \mathbb{Z} \land (\forall x \in \mathbb{Z}. (\# 0) + x = x \land x + (\# 0) = x) \).
- \( (\# 1) \in \mathbb{Z} \land (\forall x \in \mathbb{Z}. (\# 1) \cdot x = x \land x \cdot (\# 1) = x) \).
- Using Int_ZF_1_L5 by auto.

Integers with addition and integers with multiplication form monoids.

**Theorem (in int0) Int_ZF_1_T1:**
- Shows IsAmonoid(\( \mathbb{Z} \), IntegerAddition).
- Shows IsAmonoid(\( \mathbb{Z} \), IntegerMultiplication).
- Proof -
  - Have \( \exists e \in \mathbb{Z}. \\forall x \in \mathbb{Z}. e + x = x \land x + e = x \).
  - Using Int_ZF_1_L6 by auto.
  - Then show IsAmonoid(\( \mathbb{Z} \), IntegerAddition).
  - Using IsAmonoid(\( \mathbb{Z} \), IntegerMultiplication).
  - Then show IsAmonoid_def, IsAssociative_def, Int_ZF_1_L1, Int_ZF_1_L3 by auto.
- QED.

Zero is the neutral element of the integers with addition and one is the neutral element of the integers with multiplication.

**Lemma (in int0) Int_ZF_1_L8:**
- Shows \( (\# 0) = 0 \) and \( (\# 1) = 1 \).
- Proof -
  - Have monoid0(Z, IntegerAddition).
  - Using Int_ZF_1_L1, monoid0_def by simp.

616
moreover have
$(\# \ 0) \in \mathbb{Z} \land
(\forall x \in \mathbb{Z}. \text{IntegerAddition}(\# \ 0, x) = x \land
\text{IntegerAddition}(x, \# \ 0) = x)$
using Int_ZF_1_L6 by auto
ultimately have $(\# \ 0) = \text{TheNeutralElement}(\mathbb{Z}, \text{IntegerAddition})$
by (rule monoid0.group0_1_L4)
then show $(\# \ 0) = 0$ by simp
have monoid0(int,IntegerMultiplication)
using Int_ZF_1_T1 monoid0_def by simp
moreover have $(\# \ 1) \in \text{int} \land
(\forall x \in \text{int}. \text{IntegerMultiplication}(\# \ 1, x) = x \land
\text{IntegerMultiplication}(x, \# \ 1) = x)$
using Int_ZF_1_L6 by auto
ultimately have $(\# \ 1) = \text{TheNeutralElement}(\text{int}, \text{IntegerMultiplication})$
by (rule monoid0.group0_1_L4)
then show $(\# \ 1) = 1$ by simp
qed

0 and 1, as defined in int0 context, are integers.

lemma (in int0) Int_ZF_1_L8A: shows 0 ∈ Z 1 ∈ Z
proof -
  have $(\# \ 0) \in \mathbb{Z} \ (\# \ 1) \in Z$ by auto
  then show 0 ∈ Z 1 ∈ Z using Int_ZF_1_L8 by auto
qed

Zero is not one.

lemma (in int0) int_zero_not_one: shows 0 ≠ 1
proof -
  have $(\# \ 0) \neq (\# \ 1)$ by simp
  then show 0 ≠ 1 using Int_ZF_1_L8 by simp
qed

The set of integers is not empty, of course.

lemma (in int0) int_not_empty: shows Z ≠ 0
  using Int_ZF_1_L8A by auto

The set of integers has more than just zero in it.

lemma (in int0) int_not_trivial: shows Z ≠ {0}
  using Int_ZF_1_L8A int_zero_not_one by blast

Each integer has an inverse (in the addition sense).

lemma (in int0) Int_ZF_1_L9: assumes A1: g ∈ Z
  shows $\exists b \in \mathbb{Z}. \ g+b = 0$
proof -
  from A1 have g+ $\_g = 0$
    using Int_ZF_1_L2 Int_ZF_1_L8 by simp

617
thus thesis by auto
qed

Integers with addition form an abelian group. This also shows that we can apply all theorems proven in the proof contexts (locales) that require the assumption that some pair of sets form a group like locale group0.

theorem Int_ZF_1_T2: shows IsAgroup(int,IntegerAddition)
  IntegerAddition {is commutative on} int
group0(int,IntegerAddition)
  using int0.Int_ZF_1_T1 int0.Int_ZF_1_L9 IsAgroup_def
group0_def int0.Int_ZF_1_L4 IsCommutative_def by auto

What is the additive group inverse in the group of integers?

lemma (in int0) Int_ZF_1_L9A: assumes A1: m ∈ int
  shows $-m = -m$
  proof -
    from A1 have m∈int $-m ∈ int IntegerAddition( m,\$-m) =
      TheNeutralElement(int,IntegerAddition)
      using zminus_type Int_ZF_1_L2 Int_ZF_1_L8 by auto
    then have $-m = GroupInv(int,IntegerAddition)(m)
    using Int_ZF_1_T2 group0.group0_2_L9 by blast
    then show thesis by simp
qed

Subtracting integers corresponds to adding the negative.

lemma (in int0) Int_ZF_1_L10: assumes A1: m∈Z n∈Z
  shows m-n = m $+ -n$
  using assms Int_ZF_1_T2 group0.inverse_in_group Int_ZF_1_L9A Int_ZF_1_L2
  by simp

Negative of zero is zero.

lemma (in int0) Int_ZF_1_L11: shows (-0) = 0
  using Int_ZF_1_T2 group0.group_inv_of_one by simp

A trivial calculation lemma that allows to subtract and add one.

lemma Int_ZF_1_L12:
  assumes m∈int shows m $-$ $\#1 $+ $\#1 = m
  using assms eq_zdiff_iff by auto

A trivial calculation lemma that allows to subtract and add one, version with ZF-operation.

lemma (in int0) Int_ZF_1_L13: assumes m∈Z
  shows (m $-$ $\#1) + 1 = m
  using assms Int_ZF_1_L8A Int_ZF_1_L2 Int_ZF_1_L8 Int_ZF_1_L12
  by simp

Adding or subtracting one changes integers.
lemma (in int0) Int_ZF_1_L14: assumes A1: \(m \in \mathbb{Z}\) shows 
\(m + 1 \neq m\) 
\(m - 1 \neq m\) 
proof - 
{ assume \(m + 1 = m\) 
  with A1 have 
    group0(Z,IntegerAddition) 
    \(m \in Z\) \(1 \in Z\) 
    IntegerAddition\(\langle m, 1 \rangle\) = m 
  using Int_ZF_1_L8A Int_ZF_1_L14 by auto 
  then have \(1 = \text{TheNeutralElement}(Z,\text{IntegerAddition})\) 
  by (rule group0.group0_2_L7) 
  then have False using int_zero_not_one by simp 
} then show I: \(m + 1 \neq m\) by auto 
{ from A1 have \(m - 1 + 1 = m\) 
  using Int_ZF_1_L8A Int_ZF_1_L14 by simp 
  moreover assume \(m - 1 = m\) 
  ultimately have \(m + 1 = m\) by simp 
  with I have False by simp 
} then show \(m - 1 \neq m\) by auto 
qed 

If the difference is zero, the integers are equal.

lemma (in int0) Int_ZF_1_L15: assumes A1: \(m \in \mathbb{Z}\) \(n \in \mathbb{Z}\) and A2: \(m-n = 0\) shows \(m=n\) 
proof - 
let \(G = \mathbb{Z}\) 
let \(f = \text{IntegerAddition}\) 
from A1 A2 have 
  group0(G, f) 
  \(m \in G\) \(n \in G\) 
  \(f\langle m, \text{GroupInv}(G, f)(n)\rangle = \text{TheNeutralElement}(G, f)\) 
  using Int_ZF_1_T2 by auto 
then show \(m=n\) by (rule group0.group0_2_L11A) 
qed 

53.2 Integers as an ordered group

In this section we define order on integers as a relation, that is a subset of \(\mathbb{Z} \times \mathbb{Z}\) and show that integers form an ordered group.

The next lemma interprets the order definition one way.

lemma (in int0) Int_ZF_2_L1: assumes A1: \(m \in \mathbb{Z}\) \(n \in \mathbb{Z}\) and A2: \(m \leq n\) shows \(m \leq n\) 
proof -
from A1 A2 have \( \langle m,n \rangle \in \{x \in \mathbb{Z} \times \mathbb{Z}. \ \text{fst}(x) \leq \text{snd}(x)\}\) by simp
then show thesis using IntegerOrder_def by simp
qed

The next lemma interprets the definition the other way.

lemma \( \text{(in int0)} \) Int_ZF_2_L1A: assumes A1: \( m \leq n \)
shows \( m \leq n \) \( m \in \mathbb{Z} \) \( n \in \mathbb{Z} \)
proof -
from A1 have \( \langle m,n \rangle \in \{p \in \mathbb{Z} \times \mathbb{Z}. \ \text{fst}(p) \leq \text{snd}(p)\}\)
using IntegerOrder_def by simp
thus \( m \leq n \) \( m \in \mathbb{Z} \) \( n \in \mathbb{Z} \) by auto
qed

Integer order is a relation on integers.

lemma Int_ZF_2_L1B: shows IntegerOrder \( \subseteq \) int \( \times \) int
proof
fix \( x \)
assume \( x \in \text{IntegerOrder} \)
then have \( x \in \{p \in \mathbb{Z} \times \mathbb{Z}. \ \text{fst}(p) \leq \text{snd}(p)\}\)
using IntegerOrder_def by simp
then show \( x \in \text{int} \times \text{int} \) by simp
qed

The way we define the notion of being bounded below, its sufficient for the relation to be on integers for all bounded below sets to be subsets of integers.

lemma \( \text{(in int0)} \) Int_ZF_2_L1C:
assumes A1: \( \text{IsBoundedBelow}(A,\text{IntegerOrder}) \)
shows \( A \subseteq \mathbb{Z} \)
proof -
from A1 have \( \text{IntegerOrder} \subseteq \mathbb{Z} \times \mathbb{Z} \)
IsBoundedBelow(A,IntegerOrder)
using Int_ZF_2_L1B by auto
then show \( A \subseteq \mathbb{Z} \) by (rule Order_ZF_3_L1B)
qed

The order on integers is reflexive.

lemma \( \text{(in int0)} \) int_ord_is_refl: shows \( \text{refl}(\mathbb{Z},\text{IntegerOrder}) \)
using Int_ZF_2_L1 zle_refl refl_def by auto

The essential condition to show antisymmetry of the order on integers.

lemma \( \text{(in int0)} \) Int_ZF_2_L3:
assumes A1: \( m \leq n \) \( n \leq m \)
shows \( m=n \)
proof -
from A1 have \( m \leq n \) \( n \leq m \) \( m \in \mathbb{Z} \) \( n \in \mathbb{Z} \)
using Int_ZF_2_L1A by auto
then show \( m=n \) using zle_anti_sym by auto
The order on integers is antisymmetric.

**lemma** (in int0) Int_ZF_2_L4: shows antisym(IntegerOrder)
**proof** -
  have \( \forall m\ n. \ m \leq n \land n \leq m \rightarrow m=n \)
    using Int_ZF_2_L3 by auto
  then show thesis using imp_conj antisym_def by simp
**qed**

The essential condition to show that the order on integers is transitive.

**lemma** Int_ZF_2_L5:
  assumes A1: \( \langle m,n \rangle \in \text{IntegerOrder} \langle n,k \rangle \in \text{IntegerOrder} \)
  shows \( \langle m,k \rangle \in \text{IntegerOrder} \)
**proof** -
  from A1 have T1: \( m \leq n \land n \leq k \) and T2: \( m \in \text{int} \ k \in \text{int} \)
    using int0.Int_ZF_2_L1A by auto
  from T1 have \( m \leq k \) by (rule zle_trans)
  with T2 show thesis using int0.Int_ZF_2_L1 by simp
**qed**

The order on integers is transitive. This version is stated in the int0 context using notation for integers.

**lemma** (in int0) Int_order_transitive:
  assumes A1: \( m \leq n \land n \leq k \)
  shows \( m \leq k \)
**proof** -
  from A1 have \( \langle m,n \rangle \in \text{IntegerOrder} \langle n,k \rangle \in \text{IntegerOrder} \)
    by auto
  then have \( \langle m,k \rangle \in \text{IntegerOrder} \) by (rule Int_ZF_2_L5)
  then show \( m \leq k \) by simp
**qed**

The order on integers is a partial order.

**lemma** Int_ZF_2_L7: shows IsPartOrder(int,IntegerOrder)
**proof** -
  have \( \forall m\ n\ k. \langle m, n \rangle \in \text{IntegerOrder} \land \langle n, k \rangle \in \text{IntegerOrder} \rightarrow \langle m, k \rangle \in \text{IntegerOrder} \)
    using Int_ZF_2_L5 by blast
  then show thesis by (rule Fol1_L2)
**qed**
The essential condition to show that the order on integers is preserved by translations.

**Lemma (in int0) int_ord_transl_inv:**
assumes A1: \( k \in \mathbb{Z} \) and A2: \( m \leq n \)
shows \( m+k \leq n+k \) \( \quad k+m \leq k+n \)

**Proof:**
- from A2 have \( m \leq n \) and \( m \in \mathbb{Z} \) \( n \in \mathbb{Z} \)
  using Int_ZF_2_L1A by auto
- using \( \text{zadd_right_cancel_zle} \text{ zadd_left_cancel_zle} \text{ Int_ZF_1_L2} \text{ Int_ZF_2_L1} \text{ Int_ZF_1_L2} \) by auto

**Qed**

Integers form a linearly ordered group. We can apply all theorems proven in group3 context to integers.

**Theorem (in int0) Int_ZF_2_T1:** shows
\( \text{IsAnOrdGroup}(\mathbb{Z}, \text{IntegerAddition}, \text{IntegerOrder}) \)
\( \text{IntegerOrder} \) \{is total on\} \( \mathbb{Z} \)
\( \text{group3}(\mathbb{Z}, \text{IntegerAddition}, \text{IntegerOrder}) \)
\( \text{IsLinOrder}(\mathbb{Z}, \text{IntegerOrder}) \)

**Proof:**
- have \( \forall k \in \mathbb{Z}. \forall m. m \leq n \rightarrow m+k \leq n+k \land k+m \leq k+n \)
  using \( \text{Int_ZF_2_L1A} \) by simp
- then show T1: \( \text{IsAnOrdGroup}(\mathbb{Z}, \text{IntegerAddition}, \text{IntegerOrder}) \) using
  \( \text{Int_ZF_1_T2} \text{ Int_ZF_2_LiB} \text{ Int_ZF_2_L7} \text{ IsAnOrdGroup_def} \)
  by simp
- then show group3(\( \mathbb{Z}, \text{IntegerAddition}, \text{IntegerOrder} \))
  using \( \text{group3_def} \) by simp
- have \( \forall n \in \mathbb{Z}. \forall m \in \mathbb{Z}. n \leq m \lor m \leq n \)
  using \( \text{zle_linear} \text{ Int_ZF_2_L1} \) by auto
- then show \( \text{IntegerOrder} \) \{is total on\} \( \mathbb{Z} \)
  using \( \text{IsTotal_def} \) by simp
- with T1 show IsLinOrder(\( \mathbb{Z}, \text{IntegerOrder} \))
  using \( \text{IsAnOrdGroup_def} \text{ IsPartOrder_def} \text{ IsLinOrder_def} \) by simp

**Qed**

If a pair \((i, m)\) belongs to the order relation on integers and \( i \neq m \), then \( i < m \) in the sense of defined in the standard Isabelle’s Int.thy.

**Lemma (in int0) Int_ZF_2_L9:** assumes A1: \( i \leq m \) and A2: \( i \neq m \)
shows \( i < m \)

**Proof:**
- from A1 have \( i \leq m \) \( i \in \mathbb{Z} \) \( m \in \mathbb{Z} \)
  using \( \text{Int_ZF_2_L1A} \) by auto
- with A2 show \( i < m \) using \( \text{zle_def} \) by simp

**Qed**

This shows how Isabelle’s \(<\) operator translates to IsarMathLib notation.
lemma (in int0) Int_ZF_2_L9AA: assumes A1: \( m \in \mathbb{Z} \) and A2: \( m < n \)
shows \( m \leq n \) and \( m \neq n \)
using assms zle_def Int_ZF_2_L1 by auto

A small technical lemma about putting one on the other side of an inequality.

lemma (in int0) Int_ZF_2_L9A: assumes A1: \( k \in \mathbb{Z} \) and A2: \( m \leq k - (#1) \)
shows \( m + 1 \leq k \)
proof -
from A2 have \( m + 1 \leq (k - (#1)) + 1 \)
using Int_ZF_1_L8A int_ord_transl_inv by simp
with A1 show \( m + 1 \leq k \)
using Int_ZF_1_L13 by simp
qed

We can put any integer on the other side of an inequality reversing its sign.

lemma (in int0) Int_ZF_2_L9B: assumes i: \( i \in \mathbb{Z} \) and m: \( m \in \mathbb{Z} \) and k: \( k \in \mathbb{Z} \)
shows \( i + m \leq k \) if and only if \( i \leq k - m \)
using assms Int_ZF_2_T1 group3.OrderedGroup_ZF_1_L9A by simp

A special case of Int_ZF_2_L9B with weaker assumptions.

lemma (in int0) Int_ZF_2_L9C: assumes i: \( i \in \mathbb{Z} \) and m: \( m \in \mathbb{Z} \) and i: \( i \leq k - m \)
shows \( i \leq k + m \)
using assms Int_ZF_2_T1 group3.OrderedGroup_ZF_1_L9B by simp

Taking (higher order) minus on both sides of inequality reverses it.

lemma (in int0) Int_ZF_2_L10: assumes k: \( k \leq i \)
shows \( -i \leq -k \) and \( -i \leq -k \)
using assms Int_ZF_2_L1A Int_ZF_1_L9A Int_ZF_2_T1

Taking minus on both sides of inequality reverses it, version with a negative

lemma (in int0) Int_ZF_2_L10AB: assumes n: \( n \in \mathbb{Z} \) and m: \( m \leq (-n) \)
shows n: \( n \leq (-m) \)
using assms Int_ZF_2_T1 group3.OrderedGroup_ZF_1_L5AD by simp

We can cancel the same element on on both sides of an inequality, a version
with minus on both sides.
assumes \( m \in \mathbb{Z} \) \( n \in \mathbb{Z} \) \( k \in \mathbb{Z} \) and \( m-n \leq m-k \)
shows \( k \leq n \)
using assms Int_ZF_2_T1 group3.OrderedGroup_ZF_1_L5AF
by simp

If an integer is nonpositive, then its opposite is nonnegative.

lemma (in int0) Int_ZF_2_L10A: assumes \( k \leq 0 \)
shows \( 0 \leq (-k) \)
using assms Int_ZF_2_T1 group3.OrderedGroup_ZF_1_L5A by simp

If the opposite of an integers is nonnegative, then the integer is nonpositive.

lemma (in int0) Int_ZF_2_L10B:
assumes \( k \in \mathbb{Z} \) and \( 0 \leq (-k) \)
shows \( k \leq 0 \)
using assms Int_ZF_2_T1 group3.OrderedGroup_ZF_1_L5AA by simp

Adding one to an integer corresponds to taking a successor for a natural number.

lemma (in int0) Int_ZF_2_L11:
shows \( i + n + (1) = i + \text{succ}(n) \)
proof -
have \( \text{succ}(n) = 1 + n \) using int_succ_int_1 by blast
then have \( i + \text{succ}(n) = i + (1 + n) \)
  using zadd_commute by simp
then show thesis using zadd_assoc by simp
qed

Adding a natural number increases integers.

lemma (in int0) Int_ZF_2_L12: assumes \( A1: i \in \mathbb{Z} \) and \( A2: n \in \mathbb{nat} \)
shows \( i \leq i + n \)
proof -
{ assume \( n = 0 \)
  with \( A1 \) have \( i \leq i + n \) using zadd_int0 int_ord_is_refl refl_def
    by simp }
moreover
{ assume \( n \neq 0 \)
  with \( A2 \) obtain \( k \) where \( k \in \mathbb{nat} \) \( n = \text{succ}(k) \)
    using Nat_ZF_1_L3 by auto
  with \( A1 \) have \( i \leq i + n \)
    using zless_succ_zadd zless_imp_zle Int_ZF_2_L1 by simp }
ultimately show thesis by blast
qed

Adding one increases integers.

lemma (in int0) Int_ZF_2_L12A: assumes \( A1: j \leq k \)
shows \( j \leq k + 1 \)
proof -
  from \( A1 \) have \( T1: j \in \mathbb{Z} \) \( k \in \mathbb{Z} \) \( j \leq k \)
  qed
Adding one increases integers, yet one more version.

lemma (in int0) Int_ZF_2_L12B: assumes A1: m ∈ ℤ shows m ≤ m+1
using assms int_ord_is_refl refl_def Int_ZF_2_L12A by simp

If \( k + 1 = m + n \), where \( n \) is a non-zero natural number, then \( m \leq k \).

lemma (in int0) Int_ZF_2_L13:
assumes A1: k ∈ ℤ m ∈ ℤ and A2: n ∈ nat
and A3: k $+ ($# 1) = m $+ $# succ(n)
shows m ≤ k
proof -
from A1 have k∈ℤ m $+ $# n ∈ ℤ by auto
moreover from assms have k $+ $# 1 = m $+ $# n $+ $#1 using Int_ZF_2_L11 by simp
ultimately have k = m $+ $# n using zadd_right_cancel by simp
with A1 A2 show thesis using Int_ZF_2_L12 by simp
qed

The absolute value of an integer is an integer.

lemma (in int0) Int_ZF_2_L14: assumes A1: m ∈ ℤ
shows abs(m) ∈ ℤ
proof -
have AbsoluteValue(Z,IntegerAddition,IntegerOrder) : ℤ→ ℤ
  using Int_ZF_2_T1 group3.OrderedGroup_ZF_3_L1 by simp
with A1 show thesis using apply_funtype by simp
qed

If two integers are nonnegative, then the opposite of one is less or equal than the other and the sum is also nonnegative.

lemma (in int0) Int_ZF_2_L14A:
assumes 0\leq m  0\leq n
shows (-m) \leq n
0 \leq m + n
using assms Int_ZF_2_T1
  group3.OrderedGroup_ZF_3_L5AC group3.OrderedGroup_ZF_1_L12
by auto

We can increase components in an estimate.
lemma (in int0) Int_ZF_2_L15:
assumes \( b \leq b_1 \) \( c \leq c_1 \) and \( a \leq b+c \)
shows \( a \leq b_1+c_1 \)
proof -
from assms have group3(\( \mathbb{Z} \),IntegerAddition,IntegerOrder)
  ⟨a,IntegerAddition( b,c)⟩ ∈ IntegerOrder
  ⟨b,b_1⟩ ∈ IntegerOrder ⟨c,c_1⟩ ∈ IntegerOrder
using Int_ZF_2_T1 by auto
then have ⟨a,IntegerAddition( b_1,c_1)⟩ ∈ IntegerOrder
  by (rule group3.OrderedGroup_ZF_1_L5E)
thus thesis by simp
qed

We can add or subtract the sides of two inequalities.

lemma (in int0) int_ineq_add_sides:
assumes a \( \leq b \) and c \( \leq d \)
shows a+c \( \leq b+d \)
a-d \( \leq b-c \)
using assms Int_ZF_2_T1
  group3.OrderedGroup_ZF_1_L5B group3.OrderedGroup_ZF_1_L5I
by auto

We can increase the second component in an estimate.

lemma (in int0) Int_ZF_2_L15A:
assumes b \( \in \mathbb{Z} \) and a \( \leq b+c \) and A3: c \( \leq c_1 \)
shows a \( \leq b+c_1 \)
proof -
from assms have group3(\( \mathbb{Z} \),IntegerAddition,IntegerOrder)
  b \( \in \mathbb{Z} \)
  ⟨a,IntegerAddition( b,c)⟩ ∈ IntegerOrder
  ⟨c,c_1⟩ ∈ IntegerOrder
using Int_ZF_2_T1 by auto
then have ⟨a,IntegerAddition( b_1,c_1)⟩ ∈ IntegerOrder
  by (rule group3.OrderedGroup_ZF_1_L5D)
thus thesis by simp
qed

If we increase the second component in a sum of three integers, the whole sum incrases.

lemma (in int0) Int_ZF_2_L15C:
assumes A1: m \( \in \mathbb{Z} \) n\( \in \mathbb{Z} \) and A2: k \( \leq L \)
shows m+k+n \( \leq m+L+n \)
proof -
let P = IntegerAddition
from assms have group3(int,P,IntegerOrder)
  m \( \in \) int n \( \in \) int
\[ \langle k, L \rangle \in \text{IntegerOrder} \]
using Int_ZF_2_T1 by auto
then have \( \langle \langle P(\langle m, k \rangle), n \rangle, \langle P(\langle m, L \rangle), n \rangle \rangle \in \text{IntegerOrder} \)
by (rule group3.OrderedGroup_ZF_1_L10)
then show \( m+k+n \leq m+L+n \) by simp
qed

We don’t decrease an integer by adding a nonnegative one.

**lemma (in int0) Int_ZF_2_L15D:**
assumes \( 0 \leq n \quad m \in \mathbb{Z} \)
shows \( m \leq n+m \)
using assms Int_ZF_2_T1 group3.OrderedGroup_ZF_1_L5F
by simp

Some inequalities about the sum of two integers and its absolute value.

**lemma (in int0) Int_ZF_2_L15E:**
assumes \( m \in \mathbb{Z} \quad n \in \mathbb{Z} \)
shows \( m+n \leq \text{abs}(m)+\text{abs}(n) \)
\( m-n \leq \text{abs}(m)+\text{abs}(n) \)
\( (-m)+n \leq \text{abs}(m)+\text{abs}(n) \)
\( (-m)-n \leq \text{abs}(m)+\text{abs}(n) \)
using assms Int_ZF_2_T1 group3.OrderedGroup_ZF_3_L6A
by auto

We can add a nonnegative integer to the right hand side of an inequality.

**lemma (in int0) Int_ZF_2_L15F:**
assumes \( m \leq k \quad 0 \leq n \)
shows \( m \leq k+n \quad m \leq n+k \)
using assms Int_ZF_2_T1 group3.OrderedGroup_ZF_1_L5G
by auto

Triangle inequality for integers.

**lemma (in int0) Int_triangle_ineq:**
assumes \( m \in \mathbb{Z} \quad n \in \mathbb{Z} \)
shows \( \text{abs}(m+n) \leq \text{abs}(m)+\text{abs}(n) \)
using assms Int_ZF_1_T2 Int_ZF_2_T1 group3.OrdGroup_triangle_ineq
by simp

Taking absolute value does not change nonnegative integers.

**lemma (in int0) Int_ZF_2_L16:**
assumes \( 0 \leq m \) shows \( m \in \mathbb{Z}^+ \) and \( \text{abs}(m) = m \)
using assms Int_ZF_2_T1 group3.OrderedGroup_ZF_1_L2
by auto

\( 0 \leq 1 \), so \( |1| = 1 \).

**lemma (in int0) Int_ZF_2_L16A:**
shows \( 0 \leq 1 \) and \( \text{abs}(1) = 1 \)
proof -
have \( (\# 0) \in \mathbb{Z} \quad (\# 1) \in \mathbb{Z} \) by auto
then have $0 \leq 0$ and $T1: 1 \in \mathbb{Z}$
  using Int_ZF_1_L8 int_ord_is_refl refl_def by auto
then have $0 \leq 0 + 1$ using Int_ZF_2_L12A by simp
with $T1$ show $0 \leq 1$ using Int_ZF_1_T2 group0.group0_2_L2
  by simp
then show $\text{abs}(1) = 1$ using Int_ZF_2_L16 by simp
qed

$1 \leq 2$.

lemma (in int0) Int_ZF_2_L16B: shows $1 \leq 2$
proof -
  have $(\# 1) \in \mathbb{Z}$ by simp
  then show $1 \leq 2$
    using Int_ZF_1_L8 int_ord_is_refl refl_def Int_ZF_2_L12A
    by simp
qed

Integers greater or equal one are greater or equal zero.

lemma (in int0) Int_ZF_2_L16C:
  assumes $1 \leq a$
  shows $0 \leq a$
proof -
  from $A1$ have $0 \leq 1$
    and $1 \leq a$
    using Int_ZF_2_L16A
    by auto
  then show $0 \leq a$
    by (rule Int_order_transitive)
  have $I: 0 \leq 1$
    using Int_ZF_2_L16A
    by simp
  have $1 \leq 2$
    using Int_ZF_1_L8 int_ord_transl_inv
    by simp
  moreover from $A1$
  show $2 \leq a + 1$
    using Int_ZF_1_L8A int_ord_transl_inv
    by simp
  ultimately show $1 \leq a + 1$
    by (rule Int_order_transitive)
  with $I$ show $0 \leq a + 1$
    by (rule Int_order_transitive)
  from $A1$ show $a \neq 0$
    using Int_ZF_2_L16A,
    Int_ZF_2_L3 int_zero_not_one
    by auto
qed

Absolute value is the same for an integer and its opposite.

lemma (in int0) Int_ZF_2_L17:
  assumes $m \in \mathbb{Z}$ shows $\text{abs}(-m) = \text{abs}(m)$
  using assms Int_ZF_2_T1 group3.OrderedGroup_ZF_3_L7A
  by simp

The absolute value of zero is zero.

lemma (in int0) Int_ZF_2_L18: shows $\text{abs}(0) = 0$
  using Int_ZF_2_T1 group3.OrderedGroup_ZF_3_L2A
  by simp

A different version of the triangle inequality.
lemma (in int0) Int_triangle_ineq1:
  assumes A1: \( m, n \in \mathbb{Z} \)
  shows \( |m-n| \leq |n| + |m| \)
  \( |m-n| \leq |m| + |n| \)
proof -
  have \(-n \in \mathbb{Z}\) by simp
  with A1 have \( |m-n| \leq |m| + |n| \)
    using Int_ZF_1_L9A Int_triangle_ineq by simp
  with A1 show \( |m-n| \leq |n| + |m| \)
    \( |m-n| \leq |m| + |n| \)
    using Int_ZF_2_L17 Int_ZF_2_L14 Int_ZF_1_T2 IsCommutative_def
    by auto
qed

Another version of the triangle inequality.

lemma (in int0) Int_triangle_ineq2:
  assumes m \( \in \mathbb{Z}\) n \( \in \mathbb{Z}\)
    and \( |m-n| \leq k\)
  shows \( |m| \leq |n| + k\)
    \( m-k \leq n\)
    \( m \leq n+k\)
    \( n-k \leq m\)
  using assms Int_ZF_1_T2 Int_ZF_2_T1
    group3.OrderedGroup_ZF_3_L7D group3.OrderedGroup_ZF_3_L7E
  by auto

Triangle inequality with three integers. We could use OrdGroup_triangle_ineq3,
but since simp cannot translate the notation directly, it is simpler to reprove
it for integers.

lemma (in int0) Int_triangle_ineq3:
  assumes A1: \( m, n, k \in \mathbb{Z} \)
  shows \( |m+n+k| \leq |m| + |n| + |k| \)
proof -
  from A1 have T: \( m+n \in \mathbb{Z}\) abs(k) \( \in \mathbb{Z}\)
    using Int_ZF_1_L2 T group0.group_op_closed Int_ZF_2_L14
    by auto
  with A1 have \( |m+n+k| \leq |m+n| + abs(k)\)
    using Int_triangle_ineq by simp
  moreover from A1 T have
    abs(m+n) + abs(k) \( \leq |m+n| + abs(k)\)
    using Int_triangle_ineq int_ord_transl_inv by simp
  ultimately show thesis by (rule Int_order_transitive)
qed

The next lemma shows what happens when one integers is not greater or
equal than another.
lemma (in int0) Int_ZF_2_L19: assumes A1: \( m \in \mathbb{Z} \) and A2: \( n \in \mathbb{Z} \) and \( -n \leq m \) shows \( m \leq n \) proof - from A1 A2 show \( m \leq n \) using Int_ZF_2_L10 by simp then show \( (-n) \leq (-m) \) using Int_ZF_2_L19 by blast qed

If one integer is greater or equal and not equal to another, then it is not smaller or equal.

lemma (in int0) Int_ZF_2_L19AA: assumes A1: \( m \leq n \) and A2: \( m \neq n \) shows \( -n \leq m \) proof - from A1 A2 have \( \langle m,n \rangle \in \text{IntegerOrder} \) \( m \neq n \) using Int_ZF_2_T1 by auto thus \( \langle n,m \rangle \notin \text{IntegerOrder} \) by (rule group3.OrderedGroup_ZF_1_L8AA) qed

The next lemma allows to prove theorems for the case of positive and negative integers separately.

lemma (in int0) Int_ZF_2_L19A: assumes A1: \( m \in \mathbb{Z} \) and A2: \( m \neq 0 \) \( m \leq 0 \) \( 0 \leq (-m) \) \( m \neq 0 \) shows \( Q(m) \) proof - from A1 have T: \( 0 \in \mathbb{Z} \) using Int_ZF_1_T2 group0.group0_2_L2 by auto with A1 A2 show \( m \leq 0 \) using Int_ZF_2_L19 by blast from A1 T A2 show \( m \neq 0 \) by (rule Int_ZF_2_L19) from A1 T A2 have \( (-0) \leq (-m) \) by (rule Int_ZF_2_L19) then show \( 0 \leq (-m) \) using Int_ZF_1_T2 group0.group.inv.of_one by simp qed

We can prove a theorem about integers by proving that it holds for \( m = 0 \), \( m \in \mathbb{Z}_+ \) and \( -m \in \mathbb{Z}_+ \).

lemma (in int0) Int_ZF_2_L19B: assumes m\( \in \mathbb{Z} \) and Q(0) and \( \forall n \in \mathbb{Z}_+. \ Q(n) \) and \( \forall n \in \mathbb{Z}_+. \ Q(-n) \) shows Q(m) proof -
let $G = \mathbb{Z}$
let $P = \text{IntegerAddition}$
let $r = \text{IntegerOrder}$
let $b = m$
from assms have
group3($G$, $P$, $r$)  
$r$ {is total on} $G$
$b \in G$  
$Q(\text{TheNeutralElement}(G, P))$
$\forall a \in \text{PositiveSet}(G, P, r). Q(a)$
$\forall a \in \text{PositiveSet}(G, P, r). Q(\text{GroupInv}(G, P)(a))$
using Int_ZF_2_T1 by auto  
then show $Q(b)$ by (rule group3.OrderedGroup_ZF_1_L18)
qed

An integer is not greater than its absolute value.

lemma (in int0) Int_ZF_2_L19C: assumes $m \in \mathbb{Z}$
shows $m \leq \operatorname{abs}(m)$
|(-m)| \leq \operatorname{abs}(m)$
using assms Int_ZF_2_T1
\quad group3.OrderedGroup_ZF_3_L5 group3.OrderedGroup_ZF_3_L6
by auto

$|m - n| = |n - m|$.

lemma (in int0) Int_ZF_2_L20: assumes $m \in \mathbb{Z}$  $n \in \mathbb{Z}$
shows $\operatorname{abs}(m-n) = \operatorname{abs}(n-m)$
using assms Int_ZF_2_T1 group3.OrderedGroup_ZF_3_L7B by simp

We can add the sides of inequalities with absolute values.

lemma (in int0) Int_ZF_2_L21:
assumes $A1: m \in \mathbb{Z}$  $n \in \mathbb{Z}$
and $A2: \operatorname{abs}(m) \leq k$  $\operatorname{abs}(n) \leq l$
shows $\operatorname{abs}(m+n) \leq k + l$
\quad $\operatorname{abs}(m-n) \leq k + l$
using assms Int_ZF_1_T2 Int_ZF_2_T1
\quad group3.OrderedGroup_ZF_3_L7C group3.OrderedGroup_ZF_3_L7CA
by auto

Absolute value is nonnegative.

lemma (in int0) int_abs_nonneg: assumes $A1: m \in \mathbb{Z}$
shows $\operatorname{abs}(m) \in \mathbb{Z}^+$  $0 \leq \operatorname{abs}(m)$
proof -
\quad have AbsoluteValue($\mathbb{Z}$,IntegerAddition,IntegerOrder) : $\mathbb{Z}$→$\mathbb{Z}^+$
\quad using Int_ZF_2_T1 group3.OrderedGroup_ZF_3_L3C by simp
\quad with $A1$ show $\operatorname{abs}(m) \in \mathbb{Z}^+$ using apply_funtype
\quad by simp

631
then show $0 \leq \operatorname{abs}(m)$
using Int_ZF_2_T1 group3.OrderedGroup_ZF_1_L2 by simp
qed

If an nonnegative integer is less or equal than another, then so is its absolute value.

**Lemma (in int0) Int_ZF_2_L23:**
assumes $0 \leq m \leq k$
shows $\operatorname{abs}(m) \leq k$
using assms Int_ZF_2_L16 by simp

### 53.3 Induction on integers.

In this section we show some induction lemmas for integers. The basic tools are the induction on natural numbers and the fact that integers can be written as a sum of a smaller integer and a natural number.

An integer can be written a a sum of a smaller integer and a natural number.

**Lemma (in int0) Int_ZF_3_L2:**
assumes $i \leq m$
shows $\exists n \in \mathbb{N}. \ m = i + n$
proof -
let $n = 0$
{ assume $A2: i = m$
from $A1 A2$ have $n \in \mathbb{N} = i + n$
using Int_ZF_2_L1A zadd_int0_right by auto
hence $\exists n \in \mathbb{N}. \ m = i + n$ by blast }
moreover
{ assume $A3: i \neq m$
with $A1$ have $i < m \in \mathbb{Z} \ m \in \mathbb{Z}$
using Int_ZF_2_L9 Int_ZF_2_L1A by auto
then obtain $k$ where $D1: k \in \mathbb{N} = i + n$ succ($k$)
using zless_imp_succ_zadd_lemma by auto
let $n = \text{succ}(k)$
from $D1$ have $n \in \mathbb{N} = i + n$ by auto
hence $\exists n \in \mathbb{N}. \ m = i + n$ by simp }
ultimately show thesis by blast
qed

Induction for integers, the induction step.

**Lemma (in int0) Int_ZF_3_L6:**
assumes $A1: i \in \mathbb{Z}$
and $A2: \forall m. \ i \leq m \land Q(m) \rightarrow Q(m + (\text{succ}(k)))$
shows $\forall k \in \mathbb{N}. \ Q(i + (\text{succ}(k)))$
proof
fix $k$ assume $A3: k \in \mathbb{N}$ show $Q(i + \text{succ}(k))$
proof
assume $A4: Q(i + k)$
from $A1 A3$ have $i \leq i + (\text{succ}(k))$ using Int_ZF_2_L12 by simp
by simp

632
with A4 A2 have Q(i $+ ($# k) $+ ($# 1)) by simp
    then show Q(i $+ ($# succ(k))) using Int_ZF_2_L11 by simp
qed

Induction on integers, version with higher-order increment function.

lemma (in int0) Int_ZF_3_L7:
  assumes A1: i≤k and A2: Q(i)
  and A3: ∀m. i≤m ∧ Q(m) −→ Q(m $+ ($# 1))
  shows Q(k)
proof -
  from A1 obtain n where D1: n∈nat and D2: k = i $+ $# n
    using Int_ZF_3_L2 by auto
  from A1 have T1: i∈š using Int_ZF_2_L1A by simp
  note ‹n∈nat›
  moreover from A1 A2 have Q(i $+ $#0) using Int_ZF_2_L1A zadd_int0 by simp
  moreover from T1 A3 have ∀k∈nat. Q(i $+ ($# k)) −→ Q(i $+ ($# succ(k)))
    by (rule Int_ZF_3_L6)
  ultimately have Q(i $+ ($# n)) by (rule ind_on_nat)
  with D2 show Q(k) by simp
qed

Induction on integer, implication between two forms of the induction step.

lemma (in int0) Int_ZF_3_L7A: assumes
  A1: ∀m. i≤m ∧ Q(m) −→ Q(m+$1)
  shows ∀m. i≤m ∧ Q(m) −→ Q(m $+ ($# 1))
proof -
  { fix m assume i≤m ∧ Q(m)
    with A1 have T1: m∈Z Q(m+$1) using Int_ZF_2_L1A by auto
    then have m+$1 = m+$($1) using Int_ZF_1_L8 by simp
    with T1 have Q(m $+ ($# 1)) using Int_ZF_1_L2 by simp
  } then show thesis by simp
qed

Induction on integers, version with ZF increment function.

theorem (in int0) Induction_on_int:
  assumes A1: i≤k and A2: Q(i)
  and A3: ∀m. i≤m ∧ Q(m) −→ Q(m+$1)
  shows Q(k)
proof -
  from A3 have ∀m. i≤m ∧ Q(m) −→ Q(m $+ ($# 1))
    by (rule Int_ZF_3_L7A)
  with A1 A2 show thesis by (rule Int_ZF_3_L7)
qed

Another form of induction on integers. This rewrites the basic theorem.
lemma (in int0) Int_ZF_3_L7B: assumes A1: \( i \leq k \) and A2: \( P(-i) \) and A3: \( \forall m. i \leq m \land P(-m) \rightarrow P(-(m + (\# 1))) \) shows \( P(-k) \) proof -
  from A1 A2 A3 show \( P(-k) \) by (rule Int_ZF_3_L7) qed

Another induction on integers. This rewrites Int_ZF_3_L7 substituting \(-k\) for \( k \) and \(-i\) for \( i \).

lemma (in int0) Int_ZF_3_L8: assumes A1: \( k \leq i \) and A2: \( P(i) \) and A3: \( \forall n. n \leq i \land P(n) \rightarrow P(n + (-1)) \) shows \( P(k) \) proof -
  from A1 have T1: \(-i \leq -k\) using Int_ZF_2_L10 by simp
  from A1 A2 have T2: \( P(-(-i)) \) using Int_ZF_2_L1A zminus_zminus by simp
  from T1 T2 A3 show \( P(-(-k)) \) by (rule Int_ZF_3_L7)
  with A1 show \( P(k) \) using Int_ZF_2_L1A zminus_zminus by simp
qed

An implication between two forms of induction steps.

lemma (in int0) Int_ZF_3_L9: assumes A1: \( i \in \mathbb{Z} \) and A2: \( \forall n. n \leq i \land P(n) \rightarrow P(n + (-1)) \) shows \( \forall m. \(-i \leq m \land P(-m) \rightarrow P(-(m + (-1))) \) proof
  fix m show \( \(-i \leq m \land P(-m) \rightarrow P(-(m + (-1))) \) proof
    assume A3: \( \(-i \leq m \land P(-m) \)
    then have \( \(-i \leq m \) by simp
    then have \( \(-m \leq (-i)\) by (rule Int_ZF_2_L10)
    with A1 A2 A3 show \( P(-(m + (-1))) \) using zminus_zminus zminus_zadd_distrib by simp
  qed
  qed

Backwards induction on integers, version with higher-order decrement function.

lemma (in int0) Int_ZF_3_L9A: assumes A1: \( k \leq i \) and A2: \( P(i) \) and A3: \( \forall n. n \leq i \land P(n) \rightarrow P(n + (-1)) \) shows \( P(k) \) proof -
  from A1 have T1: \( i \in \mathbb{Z} \) using Int_ZF_2_L1A by simp
  from T1 A3 have T2: \( \forall m. \(-i \leq m \land P(-m) \rightarrow P(-(m + (-1))) \) by (rule Int_ZF_3_L9)
  from A1 A2 T2 show \( P(k) \) by (rule Int_ZF_3_L8)
  qed

Induction on integers, implication between two forms of the induction step.
lemma (in int0) Int_ZF_3_L10: assumes \( A1: \forall n. n \leq i \land P(n) \to P(n-1) \)
shows \( \forall n. n \leq i \land P(n) \to P(n + -($#1)) \)
proof -
  \{ fix \( n \) assume \( n \leq i \land P(n) \)
      with \( A1 \) have \( T1: n \in \mathbb{Z} \to P(n-1) \)
      using Int_ZF_2_L1A by auto
      then have \( n-1 = n-($#1) \)
      using Int_ZF_1_L8 by simp
      with \( T1 \) have \( P(n + -($#1)) \)
      using Int_ZF_1_L10 by simp
  \} then show thesis by simp
qed

Backwards induction on integers.

theorem (in int0) Back_induct_on_int:
assumes \( A1: k \leq i \) and \( A2: P(i) \)
and \( A3: \forall n. n \leq i \land P(n) \to P(n-1) \)
shows \( P(k) \)
proof -
  from \( A3 \) have \( \forall n. n \leq i \land P(n) \to P(n + -($#1)) \)
  by (rule Int_ZF_3_L10)
  with \( A1 \) show \( P(k) \)
  by (rule Int_ZF_3_L9A)
qed

53.4 Bounded vs. finite subsets of integers

The goal of this section is to establish that a subset of integers is bounded is and only is it is finite. The fact that all finite sets are bounded is already shown for all linearly ordered groups in OrderedGroups_ZF.thy. To show the other implication we show that all intervals starting at 0 are finite and then use a result from OrderedGroups_ZF.thy.

There are no integers between \( k \) and \( k + 1 \).

lemma (in int0) Int_ZF_4_L1:
assumes \( A1: k \in \mathbb{Z} \land m \in \mathbb{Z} \land n \in \mathbb{N} \) and \( A2: k + $#1 = m + $#n \)
shows \( m = k + $#1 \lor m \leq k \)
proof -
  \{ assume \( n=0 \)
      with \( A1 \) have \( m = k + $#1 \lor m \leq k \)
      using Int_ZF_1_L1A by simp \}
      moreover 
  \{ assume \( n\neq 0 \)
      with \( A1 \) obtain \( j \) where \( D1: j \in \mathbb{N} \)
      using Nat_ZF_1_L3 by auto
      with \( A1 \) have \( m = k + $#1 \lor m \leq k \)
      using Int_ZF_2_L13 by simp \}
ultimately show thesis by blast
qed

A trivial calculation lemma that allows to subtract and add one.

635
lemma \textit{IntZF}\_4\_L1A:
\begin{itemize}
\item assumes \( m \in \mathbb{Z} \)
\item shows \( m + 1 = m \)
\end{itemize}
using \textit{assms eq_zdiff_iff} by auto

There are no integers between \( k \) and \( k + 1 \), another formulation.

\begin{proof}
\begin{itemize}
\item let \( k = L \)
\item moreover from \( A_1 \) obtain \( L \) where \( L \in \mathbb{N} \)
\item ultimately have \( m = L \) using \textit{IntZF}\_4\_L1 by simp
\item with \( T_1 \) show \( j = k + 1 \) using \textit{IntZF}\_2\_L9B by auto
\end{itemize}
\end{proof}

Extending an integer interval by one is the same as adding the new endpoint.

\begin{proof}
\begin{itemize}
\item from \( A_2 \) have \( m \leq j \) using \textit{OrderZF}\_2\_L1A
\item with \( T_1 \) show \( m \in \mathbb{Z} \) using \textit{IntZF}\_2\_L1A by auto
\item from \( T_1 \) obtain \( n \) where \( n \in \mathbb{N} \)
\item with \( A_1 \) \( T_1 \) have \( m \leq k \) using \textit{IntZF}\_2\_L9B by auto
\end{itemize}
\end{proof}

636
with T1 A1 show $m..k \cup \{k + \#1\} \subseteq m..(k + \#1)$
  using Int_ZF_2_L12A int_ord_is_refl Order_ZF_2_L2 by auto
qed

Integer intervals are finite - induction step.

lemma (in int0) Int_ZF_4_L4:
  assumes A1: $i \leq m$ and A2: $i..m \in \text{Fin}(\mathbb{Z})$
  shows $i..(m + \#1) \in \text{Fin}(\mathbb{Z})$
  using assms Int_ZF_4_L3 by simp

Integer intervals are finite.

lemma (in int0) Int_ZF_4_L5: assumes A1: $i \in \mathbb{Z}$ $k \in \mathbb{Z}$
  shows $i..k \in \text{Fin}(\mathbb{Z})$
proof -
  { assume A2: $i \leq k$
    moreover from A1 have $i..i \in \text{Fin}(\mathbb{Z})$
      using int_ord_is_refl Int_ZF_2_L4 Order_ZF_2_L4 by simp
    moreover from A2 have $\forall m. i \leq m \land i..m \in \text{Fin}(\mathbb{Z}) \rightarrow i..(m + \#1) \in \text{Fin}(\mathbb{Z})$
      using Int_ZF_4_L4 by simp
    ultimately have $i..k \in \text{Fin}(\mathbb{Z})$ by (rule Int_ZF_3_L7) }
  moreover {
    assume $\neg i \leq k$
    then have $i..k \in \text{Fin}(\mathbb{Z})$ using Int_ZF_2_L6 Order_ZF_2_L5 by simp
  }
  ultimately show thesis by blast
qed

Bounded integer sets are finite.

  shows $A \in \text{Fin}(\mathbb{Z})$
proof -
  have T1: $\forall m \in \text{Nonnegative}(\mathbb{Z},\text{IntegerAddition},\text{IntegerOrder}).$ $\#0..m \in \text{Fin}(\mathbb{Z})$
    proof -
      fix $m$ assume $m \in \text{Nonnegative}(\mathbb{Z},\text{IntegerAddition},\text{IntegerOrder})$
      then have $m \in \mathbb{Z}$ using Int_ZF_2_T1 group3.OrderedGroup_ZF_1_L4E by auto
      then show $\#0..m \in \text{Fin}(\mathbb{Z})$ using Int_ZF_4_L5 by simp
    qed
  moreover from T1 have $\forall m \in \text{Nonnegative}(\mathbb{Z},\text{IntegerAddition},\text{IntegerOrder}).$ $\text{Interval}(\text{IntegerOrder},\text{TheNeutralElement}(\mathbb{Z},\text{IntegerAddition}),m)$
    $\in \text{Fin}(\mathbb{Z})$ using Int_ZF_1_L8 by simp
  moreover note A1
  ultimately show thesis by (rule group3.OrderedGroup_ZF_2_T1) qed

537
A subset of integers is bounded iff it is finite.

**Theorem (in int0)** \( \text{Int}\_\text{bounded\_iff\_fin} \):

- shows \( \text{IsBounded}(A, \text{IntegerOrder}) \iff A \in \text{Fin}(\mathbb{Z}) \)
- using \( \text{Int}\_\text{ZF}\_4\_L6 \) \( \text{Int}\_\text{ZF}\_2\_T1 \) \( \text{group3.ord\_group\_fin\_bounded} \)
  - by blast

The image of an interval by any integer function is finite, hence bounded.

**Lemma (in int0)** \( \text{Int}\_\text{ZF}\_4\_L8 \):

- assumes \( A1: i \in \mathbb{Z}, \ k \in \mathbb{Z} \) and \( A2: f: \mathbb{Z} \rightarrow \mathbb{Z} \)
- shows \( f(i..k) \in \text{Fin}(\mathbb{Z}) \)
  - using \( \text{assms} \) \( \text{Int}\_\text{ZF}\_4\_L5 \) \( \text{Finite1\_L6A} \) \( \text{Int\_bounded\_iff\_fin} \)
  - by auto

If for every integer we can find one in \( A \) that is greater or equal, then \( A \) is not bounded above, hence infinite.

**Lemma (in int0)** \( \text{Int}\_\text{ZF}\_4\_L9 \):

- assumes \( A1: \ \forall m \in \mathbb{Z}. \ \exists k \in A. \ m \leq k \)
- shows \( \neg \text{IsBoundedAbove}(A, \text{IntegerOrder}) \)
  - using \( A \not\in \text{Fin}(\mathbb{Z}) \)

**Proof**

- have \( \mathbb{Z} \neq \{0\} \)
  - using \( \text{Int}\_\text{ZF}\_1\_L3A \) \( \text{int\_zero\_not\_one} \) by blast
  - with \( A1 \) show \( \neg \text{IsBoundedAbove}(A, \text{IntegerOrder}) \)
    - \( A \not\in \text{Fin}(\mathbb{Z}) \)
    - using \( \text{Int}\_\text{ZF}\_2\_T1 \) \( \text{group3.OrderedGroupZF\_2\_L2A} \)
      - by auto

**Qed**

end

54 Integers 1

theory IntZF_1 imports IntZF_IML OrderedRing_ZF

begin

This theory file considers the set of integers as an ordered ring.

54.1 Integers as a ring

In this section we show that integers form a commutative ring.

The next lemma provides the condition to show that addition is distributive with respect to multiplication.
lemma (in int0) Int_ZF_1_1_L1:  \text{assumes A1: } a \in \mathbb{Z}, \ b \in \mathbb{Z}, \ c \in \mathbb{Z} \\
\text{shows} \\
a \cdot (b+c) = a \cdot b + a \cdot c \\
(b+c) \cdot a = b \cdot a + c \cdot a \\
\text{using assms IntZF_1_L2 zadd_zmult_distrib zadd_zmult_distrib2} \\
\text{by auto} \\

Integers form a commutative ring, hence we can use theorems proven in ring0 context (locale).

lemma (in int0) Int_ZF_1_1_L2:  \text{shows} \\
\text{IsAring(\mathbb{Z},IntegerAddition,IntegerMultiplication)} \\
\text{IntegerMultiplication \{is commutative on\} Z} \\
\text{ring0(\mathbb{Z},IntegerAddition,IntegerMultiplication)} \\
\text{proof -} \\
\text{have } \forall a \in \mathbb{Z}. \forall b \in \mathbb{Z}. \forall c \in \mathbb{Z}. \\
\text{a \cdot (b+c) = a \cdot b + a \cdot c } \land (b+c) \cdot a = b \cdot a + c \cdot a \\
\text{using IntZF_1_L1 by simp} \\
\text{then have IsDistributive(\mathbb{Z},IntegerAddition,IntegerMultiplication)} \\
\text{using IsDistributive_def by simp} \\
\text{then show IsAring(\mathbb{Z},IntegerAddition,IntegerMultiplication)} \\
\text{ring0(\mathbb{Z},IntegerAddition,IntegerMultiplication)} \\
\text{using IntZF_1_T1 IntZF_1_T2 IsAring_def ring0_def by auto} \\
\text{have } \forall a \in \mathbb{Z}. \forall b \in \mathbb{Z}. \ a \cdot b = b \cdot a \text{ using IntZF_1_L4 by simp} \\
\text{then show IntegerMultiplication \{is commutative on\} Z} \\
\text{using IsCommutative_def by simp} \\
\text{qed} \\

Zero and one are integers.

lemma (in int0) int_zero_one_are_int:  \text{shows } 0 \in \mathbb{Z}, \ 1 \in \mathbb{Z} \\
\text{using IntZF_1_L2 ring0.RingZF_1_L2 by auto} \\

Negative of zero is zero.

lemma (in int0) int_zero_one_are_intA:  \text{shows } (-0) = 0 \\
\text{using IntZF_1_T2 group0.group_inv_of_one by simp} \\

Properties with one integer.

lemma (in int0) Int_ZF_1_1_L4:  \text{assumes A1: } a \in \mathbb{Z} \\
\text{shows} \\
a+0 = a \\
0+a = a \\
a \cdot 1 = a \quad 1 \cdot a = a \\
0 \cdot a = 0 \quad a \cdot 0 = 0 \\
(-a) \in \mathbb{Z} \quad (-(-a)) = a \\
a-a = 0 \quad a-0 = a \quad 2 \cdot a = a+a \\
\text{proof -} \\
\text{from A1 show} \\
a+0 = a \quad 0+a = a \\
a \cdot 1 = a
1·a = a a-a = 0 a-0 = a
(-a) ∈ Z 2a = a+a (-(-a)) = a
using Int_ZF_1_1_L2 ring0.Ring_ZF_1_L3 by auto
from A1 show 0·a = 0 a·0 = 0
using Int_ZF_1_1_L2 ring0.Ring_ZF_1_L6 by auto
qed

Properties that require two integers.

lemma (in int0) Int_ZF_1_1_L5: assumes a∈Z  b∈Z.
  shows a+b ∈ Z
  a-b ∈ Z
  a·b ∈ Z
  a+b = b+a
  a·b = b·a
  (-b)-a = (-a)-b
  (-a+b)) = (-a)-b
  (-a-b)) = ((-a)+b)
  (-a).b = -(a·b)
  a·(-b) = -(a·b)
  (-a).(-b) = a·b
using assms Int_ZF_1_1_L2 ring0.Ring_ZF_1_L4 ring0.Ring_ZF_1_L9
ring0.Ring_ZF_1_L7 ring0.Ring_ZF_1_L7A Int_ZF_1_L4 by auto

2 and 3 are integers.

lemma (in int0) int_two_three_are_int: shows 2 ∈ Z  3 ∈ Z
  using int_zero_one_are_int Int_ZF_1_1_L5 by auto

Another property with two integers.

lemma (in int0) Int_ZF_1_1_L5B: assumes a∈Z  b∈Z.
  shows a-(-b) = a+b
using assms Int_ZF_1_1_L2 ring0.Ring_ZF_1_L9 by simp

Properties that require three integers.

lemma (in int0) Int_ZF_1_1_L6: assumes a∈Z  b∈Z  c∈Z.
  shows a-(-b+c) = a-b-c
  a-(-b-c) = a-b+c
  a·(b-c) = a·b - a·c
  (b-c)·a = b·a - c·a
using assms Int_ZF_1_1_L2 ring0.Ring_ZF_1_L10 ring0.Ring_ZF_1_L8 by auto

One more property with three integers.

lemma (in int0) Int_ZF_1_1_L6A: assumes a∈Z  b∈Z  c∈Z.
  shows a+(b-c) = a+b-c
using assms Int_ZF_1_1_L2 ring0.Ring_ZF_1_L10A by simp

Associativity of addition and multiplication.

lemma (in int0) Int_ZF_1_1_L7: assumes a∈Z b∈Z c∈Z
  shows
  a+b+c = a+(b+c)
  a·b·c = a·(b·c)
  using assms Int_ZF_1_1_L2 ring0.Ring_ZF_1_L11 by auto

5.4.2 Rearrangement lemmas

In this section we collect lemmas about identities related to rearranging the
terms in expressions

A formula with a positive integer.

lemma (in int0) Int_ZF_1_2_L1: assumes 0≤a
  shows abs(a)+1 = abs(a+1)
  using assms Int_ZF_2_L16 Int_ZF_2_L12A by simp

A formula with two integers, one positive.

lemma (in int0) Int_ZF_1_2_L2: assumes A1: a∈Z and A2: 0≤b
  shows a+(abs(b)+1)·a = (abs(b+1)+1)·a
proof -
  from A2 have abs(b+1) ∈ ℤ
  using Int_ZF_2_L12A Int_ZF_2_L1A Int_ZF_2_L14 by blast
  with A1 A2 show thesis
  using Int_ZF_1_2_L1 Int_ZF_1_1_L2 ring0.Ring_ZF_2_L1
  by simp
  qed

A couple of formulae about canceling opposite integers.

lemma (in int0) Int_ZF_1_2_L3: assumes A1: a∈Z b∈Z
  shows
  a+b-a = b
  a+(b-a) = b
  a+b-b = a
  a-b+b = a
  (-a)+(a+b) = b
  a+(b-a) = b
  (-b)+(a+b) = a
  a-(b+a) = -b
  a-(a+b) = -b
  a-(a-b) = b
  a-b-a = -b
  a-b - (a+b) = (-b)-b
  using assms Int_ZF_1_T2 group0.group0_4_L6A group0.inv_cancel_two
group0.group0_2_L16A group0.group0_4_L6AA group0.group0_4_L6AB
group0.group0_4_L6F group0.group0_4_L6AC by auto
Subtracting one does not increase integers. This may be moved to a theory about ordered rings one day.

**lemma** (in int0) Int_ZF_1_2_L3A: assumes A1: a ≤ b shows a-1 ≤ b
proof -
  from A1 have b+1-1 = b using Int_ZF_2_L1A int_zero_one_are_int Int_ZF_1_2_L3 by simp
moreover from A1 have a-1 ≤ b+1-1 using Int_ZF_2_L12A int_zero_one_are_int Int_ZF_1_1_L4 int_ord_transl_inv by simp
ultimately show a-1 ≤ b by simp
qed

Subtracting one does not increase integers, special case.

**lemma** (in int0) Int_ZF_1_2_L3AA: assumes A1: a ∈ š shows a-1 ≤ a a-1 ≠ a ¬(a ≤ a-1) ¬(a+1 ≤ a) ¬(1+a ≤ a)
proof -
  from A1 have a≤a using int_ord_is_refl refl_def by simp
  then show a-1 ≤ a using Int_ZF_1_2_L3A by simp
moreover from A1 show a-1 ≠ a using Int_ZF_1_1_L4 by simp
ultimately show I: ¬(a≤a-1) using Int_ZF_2_L19AA by blast
with A1 show ¬(a+1 ≤ a) using int_zero_one_are_int Int_ZF_2_L9B by simp
with A1 show ¬(1+a ≤ a) using int_zero_one_are_int Int_ZF_1_1_L5 by simp
qed

A formula with a nonpositive integer.

**lemma** (in int0) Int_ZF_1_2_L4: assumes a ≤ 0 shows abs(a)+1 = abs(a-1)
proof -
  using assms int_zero_one_are_int Int_ZF_1_2_L3A Int_ZF_2_T1 group3.OrderedGroup_ZF_3_L3A Int_ZF_2_L1A int_zero_one_are_int Int_ZF_1_1_L5 by simp

A formula with two integers, one negative.

**lemma** (in int0) Int_ZF_1_2_L5: assumes A1: a ∈ ℤ and A2: b ≤ 0 shows a+(abs(b)+1)·a = (abs(b-1)+1)·a
proof -
  from A2 have abs(b-1) ∈ ℤ using int_zero_one_are_int Int_ZF_1_2_L3A Int_ZF_2_L1A Int_ZF_2_L14
by blast
with A1 A2 show thesis
  using Int_ZF_1_2_L4 Int_ZF_1_1_L4 ring0.Ring_ZF_2_L1 by simp
qed

A rearrangement with four integers.

lemma (in int0) Int_ZF_1_2_L6: assumes A1: a∈Z  b∈Z  c∈Z  d∈Z
  shows  a-(b-1)·c = (d-b·c)-(d-a-c)
proof -
  from A1 have T1: (d-b·c) ∈ Z  d-a ∈ Z  (-b·c) ∈ Z
    using Int_ZF_1_1_L5 Int_ZF_1_1_L4 by auto
  with A1 have (d-b·c)-(d-a-c) = (-b·c)+a+c
    using Int_ZF_1_1_L6 Int_ZF_1_2_L3 by simp
  also from A1 T1 have (-b·c)+a+c = a-(b-1)·c
    using int_zero_one_are_int Int_ZF_1_1_L6 Int_ZF_1_1_L4 Int_ZF_1_1_L5
    by simp
  finally show thesis by simp
qed

Some other rearrangements with two integers.

lemma (in int0) Int_ZF_1_2_L7: assumes a∈Z  b∈Z
  shows  a·b = (a-1)·b+b
        a·(b+1) = a·b+a
        (b+1)·a = b·a+a
        (b+1)·a = a·b+a
    using assms Int_ZF_1_1_L1 Int_ZF_1_1_L5 int_zero_one_are_int
    Int_ZF_1_1_L6 Int_ZF_1_1_L4 Int_ZF_1_1_L2 group0.inv_cancel_two
    by auto

Another rearrangement with two integers.

lemma (in int0) Int_ZF_1_2_L8: assumes A1: a∈Z  b∈Z
    shows a+1+(b+1) = b+a+2
    using assms int_zero_one_are_int Int_ZF_1_T2 group0.group0_4_L8
    by simp

A couple of rearrangement with three integers.

lemma (in int0) Int_ZF_1_2_L9: assumes a∈Z  b∈Z  c∈Z
    shows  (a-b)+(b-c) = a-c
        (a-b)-(a-c) = c-b

643
a + (b + (c - a - b)) = c
(-a) - b + c = c - a - b
(-b) - a + c = c - a - b
(-((-a) + b + c)) = a - b - c
a + b + c - a = b + c
a + b - (a + c) = b - c

using assms Int_ZF_1_T2
  group0.group0_4_L4B group0.group0_4_L6D group0.group0_4_L4D
  group0.group0_4_L6B group0.group0_4_L6E
by auto

Another couple of rearrangements with three integers.

lemma (in int0) Int_ZF_1_2_L9A:
  assumes A1: a ∈ ℤ b ∈ ℤ c ∈ ℤ
  shows (-(a - b - c)) = c + b - a
proof -
  from A1 have T:
    a - b ∈ ℤ (-a) ∈ ℤ (-b) ∈ ℤ using
    Int_ZF_1_1_L4 Int_ZF_1_1_L5 by auto
  with A1 have (-(a - b - c)) = c - ((-b) + a)
    using Int_ZF_1_1_L5 by simp
  also from A1 T have ... = c + b - a
    using Int_ZF_1_1_L6 Int_ZF_1_1_L5B
    by simp
  finally show (-(a - b - c)) = c + b - a
    by simp
qed

Another rearrangement with three integers.

lemma (in int0) Int_ZF_1_2_L10:
  assumes A1: a ∈ ℤ b ∈ ℤ c ∈ ℤ
t shows (a + 1) · b + (c + 1) · b = (c + a + 2) · b
proof -
  from A1 have a + 1 ∈ ℤ c + 1 ∈ ℤ
    using int_zero_one_are_int Int_ZF_1_1_L5 by auto
  with A1 have (a + 1) · b + (c + 1) · b = (a + 1 + (c + 1)) · b
    using Int_ZF_1_1_L1 by simp
  also from A1 have ... = (c + a + 2) · b
    using Int_ZF_1_1_L6 Int_ZF_1_1_L8 by simp
  finally show thesis by simp
qed

A technical rearrangement involing inequalities with absolute value.

lemma (in int0) Int_ZF_1_2_L10A:
  assumes A1: a ∈ ℤ b ∈ ℤ c ∈ ℤ e ∈ ℤ
  and A2: abs(a - b - c) ≤ d abs(b - a - e) ≤ f
  shows abs(c - e) ≤ f + d
proof -
from A1 A2 have T1:
  \( d \in \mathbb{Z} \), \( f \in \mathbb{Z} \), \( a \cdot b \in \mathbb{Z} \), \( a \cdot b - e \in \mathbb{Z} \)
  using Int_ZF_2_L1A Int_ZF_1_1_L5 by auto
with A2 have
  \( \text{abs}((b \cdot a - e) - (a \cdot b - c)) \leq f + d \)
  using Int_ZF_2_L21 by simp
with A1 T1 show \( \text{abs}(c - e) \leq f + d \)
  using Int_ZF_1_1_L5 Int_ZF_1_2_L9 by simp
qed

Some arithmetics.

lemma (in int0) Int_ZF_1_2_L11: assumes A1: \( a \in \mathbb{Z} \)
  shows \( a+1 + 2 = a+3 \)
proof -
  from A1 show \( a+1 + 2 = a+3 \)
    using Int_ZF_1_1_L5 int_zero_one_are_int by auto
  from A1 show \( a = 2 \cdot a - a \)
    using Int_ZF_1_1_L7 Int_ZF_1_2_L7 Int_ZF_1_1_L1 Int_ZF_1_1_L4 Int_ZF_1_T2
    group0.inv_cancel_two by simp
qed

A simple rearrangement with three integers.

lemma (in int0) Int_ZF_1_2_L12:
  assumes \( a \in \mathbb{Z} \), \( b \in \mathbb{Z} \), \( c \in \mathbb{Z} \)
  shows \( (b-c) \cdot a = a \cdot b - a \cdot c \)
using assms Int_ZF_1_1_L6 Int_ZF_1_1_L5 by simp

A big rearrangement with five integers.

lemma (in int0) Int_ZF_1_2_L13:
  assumes A1: \( a \in \mathbb{Z} \), \( b \in \mathbb{Z} \), \( c \in \mathbb{Z} \), \( d \in \mathbb{Z} \), \( x \in \mathbb{Z} \)
  shows \( (x+(a \cdot x+b)+c) \cdot d = d \cdot (a+1) \cdot x + (b \cdot d+c \cdot d) \)
proof -
  from A1 have T1:
    \( a \cdot x \in \mathbb{Z} \), \( (a+1) \cdot x \in \mathbb{Z} \)
    \( (a+1) \cdot x + b \in \mathbb{Z} \)
    using Int_ZF_1_1_L5 Int_zero_one_are_int by auto
  with A1 have \( (x+(a \cdot x+b)+c) \cdot d = ((a+1) \cdot x + b) \cdot d + c \cdot d \)
    using Int_ZF_1_1_L7 Int_ZF_1_2_L7 Int_ZF_1_1_L1 by simp
  also from A1 T1 have \( ... = (a+1) \cdot x \cdot d + b \cdot d + c \cdot d \)
    using Int_ZF_2_1_L1 by simp
  finally have \( (x+(a \cdot x+b)+c) \cdot d = (a+1) \cdot x \cdot d + b \cdot d + c \cdot d \)
    by simp
  moreover from A1 T1 have \( (a+1) \cdot x \cdot d = d \cdot (a+1) \cdot x \)
ultimately have \((x+(a\cdot x+b)+c)\cdot d = d\cdot (a+1)x + b\cdot d + c\cdot d\)
by simp

moreover from A1 T1 have
\[d\cdot (a+1)\in \mathbb{Z} \quad b\cdot d \in \mathbb{Z} \quad c\cdot d \in \mathbb{Z}.
using int_zero_one_are_int Int_ZF_1_1_L5 by auto
ultimately show thesis using Int_ZF_1_1_L7 by simp
qed

Rerrangement about adding linear functions.

lemma (in int0) Int_ZF_1_2_L14:
assumes \(a \in \mathbb{Z}\) \(b \in \mathbb{Z}\) \(c \in \mathbb{Z}\) \(d \in \mathbb{Z}\) \(x \in \mathbb{Z}\)
shows \((a \cdot x + b) + (c \cdot x + d) = (a+c)\cdot x + (b+d)\)
using assms Int_ZF_1_1_L2 ring0.Ring_ZF_2_L3 by simp
A rearrangement with four integers. Again we have to use the generic set
notation to use a theorem proven in different context.

lemma (in int0) Int_ZF_1_2_L15: assumes A1: \(a \in \mathbb{Z}\) \(b \in \mathbb{Z}\) \(c \in \mathbb{Z}\) \(d \in \mathbb{Z}\) and A2: \(a = b-c-d\)
shows \(d = b-a-c\)
\(d = (-a)+b-c\)
\(b = a+d+c\)
proof
- let \(G = \mathbb{Z}\)
  let \(f = \text{IntegerAddition}\)
  from A1 A2 have I:
    group0(G, f) \(\{\text{is commutative on} \} G\)
    \(a \in G\) \(b \in G\) \(c \in G\) \(d \in G\)
    \(a = f(b, \text{GroupInv}(G,f)(c)),\text{GroupInv}(G,f)(d))\)
    using Int_ZF_1_T2 by auto
  then have
    \(d = f(f(b, \text{GroupInv}(G,f)(a)),\text{GroupInv}(G,f)(c))\)
    by (rule group0.group0_4_L9)
  then show \(d = b-a-c\) by simp
from 1 have \(d = f(\text{GroupInv}(G,f)(a)),\text{GroupInv}(G,f)(c))\)
  by (rule group0.group0_4_L9)
thus \(d = (-a)+b-c\)
  by simp
from 1 have \(b = f(a, d),c)\)
  by (rule group0.group0_4_L9)
thus \(b = a+d+c\) by simp
qed

A rearrangement with four integers. Property of groups.

lemma (in int0) Int_ZF_1_2_L16: assumes a\(\in \mathbb{Z}\) \(b \in \mathbb{Z}\) \(c \in \mathbb{Z}\) \(d \in \mathbb{Z}\)
shows \(a+(b-c)+d = a+b+d-c\)
using assms Int_ZF_1_1_L2 group0.group0_4_L9 by simp
Some rearrangements with three integers. Properties of groups.

**Lemma (in int0) Int_ZF_1_2_L17:**

assumes $A1: a \in \mathbb{Z}, b \in \mathbb{Z}, c \in \mathbb{Z}$

shows

- $a+b-c+(c-b) = a$
- $a+(b+c)-c = a+b$

**Proof** -

let $G = \text{int}$
let $f = \text{IntegerAddition}$

from $A1$ have $I:

$\text{group0}(G, f)$

$a \in G, b \in G, c \in G$

using Int_ZF_1_T2 by auto

then have

$f(f(f(a,b), \text{GroupInv}(G, f)(c)), f(c, \text{GroupInv}(G, f)(b))) = a$

by (rule group0.group0_2_L14A)

thus $a+b-c+(c-b) = a$ by simp

from $I$ have

$f(f(a, f(b, c)), \text{GroupInv}(G, f)(c)) = f(a, b)$

by (rule group0.group0_4_L6D)

thus $a+(b+c)-c = a+b$ by simp

qed

Another rearrangement with three integers. Property of abelian groups.

**Lemma (in int0) Int_ZF_1_2_L18:**

assumes $A1: a \in \mathbb{Z}, b \in \mathbb{Z}, c \in \mathbb{Z}$

shows $a+b-c+(c-a) = b$

**Proof** -

let $G = \text{int}$
let $f = \text{IntegerAddition}$

from $A1$ have

$\text{group0}(G, f)$

$f \{\text{is commutative on}\} G$

$a \in G, b \in G, c \in G$

using Int_ZF_1_T2 by auto

then have

$f(f(f(a,b), \text{GroupInv}(G, f)(c)), f(c, \text{GroupInv}(G, f)(a))) = b$

by (rule group0.group0_4_L6D)

thus $a+b-c+(c-a) = b$ by simp

qed

### 54.3 Integers as an ordered ring

We already know from Int_ZF that integers with addition form a linearly ordered group. To show that integers form an ordered ring we need the fact that the set of nonnegative integers is closed under multiplication.

We start with the property that a product of nonnegative integers is nonnegative. The proof is by induction and the next lemma is the induction step.
lemma (in int0) Int_ZF_1_3_L1: assumes A1: $0 \leq a$, $0 \leq b$
  and A3: $0 \leq a \cdot b$
  shows $0 \leq a \cdot (b+1)$
proof -
  from A1 A3 have $0 + 0 \leq a \cdot b + a$
    using int_ineq_add_sides by simp
  with A1 show $0 \leq a \cdot (b+1)$
    using int_zero_one_are_int Int_ZF_1_1_L4 Int_ZF_2_L1A Int_ZF_1_2_L7
    by simp
qed

Product of nonnegative integers is nonnegative.

lemma (in int0) Int_ZF_1_3_L2: assumes A1: $0 \leq a$, $0 \leq b$
  shows $0 \leq a \cdot b$
proof -
  from A1 have $0 \leq b$ by simp
  moreover from A1 have $0 \leq a \cdot 0$
    using Int_ZF_2_L1A Int_ZF_1_1_L4 int_zero_one_are_int int_ord_is_refl refl_def
    by simp
  moreover from A1 have $\forall m. \ 0 \leq m \wedge 0 \leq a \cdot m \longrightarrow 0 \leq a \cdot (m+1)$
    using Int_ZF_1_3_L1 by simp
  ultimately show $0 \leq a \cdot b$ by (rule Induction_on_int)
qed

The set of nonnegative integers is closed under multiplication.

lemma (in int0) Int_ZF_1_3_L2A: shows $\mathbb{Z}^+$ {is closed under} IntegerMultiplication
proof -
  { fix $a \ b$ assume $a \in \mathbb{Z}^+$, $b \in \mathbb{Z}^+$
    then have $a \cdot b \in \mathbb{Z}^+$
      using Int_ZF_1_3_L2 Int_ZF_2_L1A group3.OrderedGroup_ZF_1_L2
      by simp
  } then have $\forall a \in \mathbb{Z}^+. \forall b \in \mathbb{Z}^+. \ a \cdot b \in \mathbb{Z}^+$ by simp
  then show thesis using IsOpClosed_def by simp
qed

Integers form an ordered ring. All theorems proven in the ring1 context are
valid in int0 context.

theorem (in int0) Int_ZF_1_3_T1: shows IsAnOrdRing($\mathbb{Z}$,IntegerAddition,IntegerMultiplication,IntegerOrder)
ring1($\mathbb{Z}$,IntegerAddition,IntegerMultiplication,IntegerOrder)
using Int_ZF_1_1_L2 Int_ZF_2_L1B Int_ZF_1_3_L2A Int_ZF_2_T1
OrdRing_ZF_1_1_L6 OrdRing_ZF_1_2_L2 by auto

Product of integers that are greater that one is greater than one. The proof
is by induction and the next step is the induction step.
lemma (in int0) Int_ZF_1_3_L3_indstep:
assumes A1: \( 1 \leq a \leq b \)
and A2: \( 1 \leq a \cdot b \)
shows \( 1 \leq a \cdot (b+1) \)
proof -
  from A1 A2 have \( 1 \leq 2 \) and \( 2 \leq a \cdot (b+1) \)
  using Int_ZF_2_L1A int_ineq_add_sides Int_ZF_2_L16B Int_ZF_1_2_L7
  by auto
  then show \( 1 \leq a \cdot (b+1) \) by (rule Int_order_transitive)
qed

Product of integers that are greater than one is greater than one.

lemma (in int0) Int_ZF_1_3_L3:
assumes A1: \( 1 \leq a \leq b \)
shows \( 1 \leq a \cdot b \)
proof -
  from A1 have \( 1 \leq b \) and \( 1 \leq a \cdot 1 \)
  using Int_ZF_2_L1A Int_ZF_1_1_L4
  by auto
  moreover from A1 have \( \forall m. \ 1 \leq m \land 1 \leq a \cdot m \longrightarrow 1 \leq a \cdot (m+1) \)
  using Int_ZF_1_3_L3_indstep
  by simp
  ultimately show \( 1 \leq a \cdot b \) by (rule Induction_on_int)
qed

\[ |a \cdot (-b)| = |(-a) \cdot b| = |(-a) \cdot (-b)| = |a \cdot b| \] This is a property of ordered rings..

lemma (in int0) Int_ZF_1_3_L4: assumes a\in \mathbb{Z} \quad b\in \mathbb{Z}
shows \[ |a \cdot b| = |a| \cdot |b| \]
using assms Int_ZF_1_3_L5 Int_ZF_2_L17 by auto

Absolute value of a product is the product of absolute values. Property of ordered rings.

lemma (in int0) Int_ZF_1_3_L5: assumes A1: a\in \mathbb{Z} \quad b\in \mathbb{Z}
shows \[ |a\cdot b| = |a| \cdot |b| \]
using assms Int_ZF_1_3_T1 ring1.OrdRing_ZF_2_L5 by simp

Double nonnegative is nonnegative. Property of ordered rings.

lemma (in int0) Int_ZF_1_3_L5A: assumes 0\leq a
shows 0\leq 2 \cdot a
using assms Int_ZF_1_3_T1 ring1.OrdRing_ZF_1_1_L5A by simp

The next lemma shows what happens when one integer is not greater or equal than another.
lemma (in int0) Int_ZF_1_3_L6:
  assumes A1: \( a \in \mathbb{Z}, \ b \in \mathbb{Z} \)
  shows \( \neg (b \leq a) \iff a + 1 \leq b \)
proof
  assume A3: \( \neg (b \leq a) \)
  with A1 have a\leq b by (rule Int_ZF_2_L19)
  then have a \equiv b \vee a + 1 \leq b
  using Int_ZF_4_L1B by simp
  moreover from A1 A3 have a\neq b by (rule Int_ZF_2_L19)
  ultimately show a + 1 \leq b by simp
next assume A4: a + 1 \leq b
  { assume b\leq a
    with A4 have a + 1 \leq a by (rule Int_order_transitive)
    moreover from A1 have a \leq a + 1
      using Int_ZF_2_L12B by simp
    ultimately have a + 1 = a
      by (rule Int_ZF_2_L3)
    with A1 have False using Int_ZF_1_L14 by simp
  } then show \( \neg (b \leq a) \) by auto
qed

Another form of stating that there are no integers between integers \( m \) and \( m + 1 \).
corollary (in int0) no_int_between: assumes A1: a \in \mathbb{Z}, b \in \mathbb{Z}
  shows b \leq a \vee a + 1 \leq b
  using A1 Int_ZF_1_3_L6 by auto

Another way of saying what it means that one integer is not greater or equal than another.
corollary (in int0) Int_ZF_1_3_L6A:
  assumes A1: a \in \mathbb{Z}, b \in \mathbb{Z} and A2: \( \neg (b \leq a) \)
  shows a \leq b - 1
proof -
  from A1 A2 have a + 1 - 1 \leq b - 1
    using Int_ZF_1_3_L6 int_zero_one_are_int Int_ZF_1_1_L4
    int_ord_transl_inv by simp
  with A1 show a \leq b - 1
    using int_zero_one_are_int Int_ZF_1_2_L3 by simp
qed

Yet another form of stating that there are no integers between \( m \) and \( m + 1 \).

lemma (in int0) no_int_between1:
  assumes A1: a \leq b and A2: a \neq b
  shows a + 1 \leq b
  a \leq b - 1
proof -
from A1 have T: a\in\mathbb{Z} \land b\in\mathbb{Z} using Int_ZF_2_L1A by auto
{ assume b\leq a
  with A1 have a=b by (rule Int_ZF_2_L3)
  with A2 have False by simp }
then have \neg(b\leq a) by auto
with T show a+1 \leq b
  a \leq b-1
  using no_int_between Int_ZF_1_3_L6A by auto
qed

We can decompose proofs into three cases: \( a = b \), \( a \leq b - 1 \) or \( a \geq b + 1 \).

lemma (in int0) Int_ZF_1_3_L6B: assumes A1: a\in\mathbb{Z} \land b\in\mathbb{Z}
  shows a=b \lor (a \leq b-1) \lor (b+1 \leq a)
proof -
  from A1 have a=b \lor (a\leq b \land a\neq b) \lor (b\leq a \land b\neq a)
    using Int_ZF_2_T1 group3.OrderedGroup_ZF_1_L31
    by simp
  then show thesis using no_int_between1 by auto
qed

A special case of Int_ZF_1_3_L6B when \( b = 0 \). This allows to split the proofs
in cases \( a \leq -1 \), \( a = 0 \) and \( a \geq 1 \).

corollary (in int0) Int_ZF_1_3_L6C: assumes A1: a\in\mathbb{Z}
  shows a=0 \lor (a \leq -1) \lor (1\leq a)
proof -
  from A1 have a=0 \lor (a \leq 0 \land a \leq -1) \lor (0 \leq 1 \land 1 \leq a)
    using int_zero_one_are_int Int_ZF_1_3_L6 Int_ZF_1_3_L7 Int_ZF_1_1_L4
    by simp
  then show thesis using Int_ZF_1_1_L4 int_zero_one_are_int
    by simp
qed

An integer is not less or equal zero iff it is greater or equal one.

lemma (in int0) Int_ZF_1_3_L7: assumes a\in\mathbb{Z}
  shows \neg(a\leq 0) \iff 1 \leq a
using assms Int_ZF_1_3_L6 Int_ZF_1_1_L4
by simp

Product of positive integers is positive.

lemma (in int0) Int_ZF_1_3_L8: assumes a\in\mathbb{Z} \land b\in\mathbb{Z}
  and \neg(a\leq 0) \land \neg(b\leq 0)
  shows \neg((a\cdot b) \leq 0)
using assms Int_ZF_1_3_L7 Int_ZF_1_3_L3 Int_ZF_1_1_L5 Int_ZF_1_3_L7
by simp
If \(a \cdot b\) is nonnegative and \(b\) is positive, then \(a\) is nonnegative. Proof by contradiction.

**Lemma (in int0) Int_ZF_1_3_L9:**

- Assumes \(A1: a \in \mathbb{Z}\)
- Assumes \(A2: \neg(b \leq 0)\) and \(A3: a \cdot b \leq 0\)

**Proof**

- \{ assume \(\neg(a \leq 0)\)
  - with \(A1\) and \(A2\) have \(\neg((a \cdot b) \leq 0)\) using Int_ZF_1_3_L8 by simp
- \} with \(A3\) show \(a \leq 0\) by auto

**Qed**

One integer is less or equal another iff the difference is nonpositive.

**Lemma (in int0) Int_ZF_1_3_L10:**

- Assumes \(a \in \mathbb{Z}\)
- Assumes \(b \in \mathbb{Z}\)

**Shows** \(a \leq b \iff a - b \leq 0\)

**Using** asms Int_ZF_2_T1 group3.OrderedGroup_ZF_1_L9 by simp

**Some conclusions from the fact that one integer is less or equal than another.**

**Lemma (in int0) Int_ZF_1_3_L10A:**

- Assumes \(a \leq b\)
- Shows \(0 \leq b - a\)

**Using** asms Int_ZF_2_T1 group3.OrderedGroup_ZF_1_L12A by simp

**We can simplify out a positive element on both sides of an inequality.**

**Lemma (in int0) Int_ineq_simpl_positive:**

- Assumes \(A1: a \in \mathbb{Z}\)
- Assumes \(b \in \mathbb{Z}\)
- Assumes \(c \in \mathbb{Z}\)

**And** \(A2: a \cdot c \leq b \cdot c\) and \(A4: \neg(c \leq 0)\)

**Shows** \(a \leq b\)

**Proof**

- from \(A1\) and \(A4\) have \(a - b \in \mathbb{Z}\) \(c \in \mathbb{Z}\) \(\neg(c \leq 0)\)
  - using Int_ZF_1_1_L5 by auto

- moreover from \(A1\) and \(A2\) have \((a - b) \cdot c \leq 0\)
  - using Int_ZF_1_1_L5 Int_ZF_1_3_L10 Int_ZF_1_1_L6 by simp

- ultimately have \(a - b \leq 0\) by (rule Int_ZF_1_3_L9)
  - with \(A1\) show \(a \leq b\) using Int_ZF_1_3_L10 by simp

**Qed**

A technical lemma about conclusion from an inequality between absolute values. This is a property of ordered rings.

**Lemma (in int0) Int_ZF_1_3_L11:**

- Assumes \(A1: a \in \mathbb{Z}\)
- Assumes \(b \in \mathbb{Z}\)

**And** \(A2: \neg(a \leq b)\) and \(A3: \neg(a \leq 0)\)

**Shows** \(\neg(a \leq \text{abs}(b))\)

**Proof**

-
\{ assume abs(a) ≤ 0
  moreover from A1 have 0 ≤ abs(a) using int_abs_nonneg
  by simp
  ultimately have abs(a) = 0 by (rule Int_ZF_2_L3)
  with A1 A2 have False using int_abs_nonneg by simp
\} then show ¬(abs(a) ≤ 0) by auto
qed

Negative times positive is negative. This a property of ordered rings.

lemma (in int0) Int_ZF_1_3_L12:
  assumes a ≤ 0 and 0 ≤ b
  shows a · b ≤ 0
  using assms Int_ZF_1_3_T1 ring1.OrdRing_ZF_1_L8
  by simp

We can multiply an inequality by a nonnegative number. This is a property of ordered rings.

lemma (in int0) Int_ZF_1_3_L13:
  assumes A1: a ≤ b and A2: 0 ≤ c
  shows a · c ≤ b · c
  c · a ≤ c · b
  using assms Int_ZF_1_3_T1 ring1.OrdRing_ZF_1_L9 by auto

A technical lemma about decreasing a factor in an inequality.

lemma (in int0) Int_ZF_1_3_L13A:
  assumes 1 ≤ a and b ≤ c and (a+1) · c ≤ d
  shows (a+1) · b ≤ d
proof -
  from assms have
    (a+1) · b ≤ (a+1) · c
    (a+1) · c ≤ d
    using Int_ZF_2_L16C Int_ZF_1_3_L13 by auto
  then show (a+1) · b ≤ d by (rule Int_order_transitive)
qed

We can multiply an inequality by a positive number. This is a property of ordered rings.

lemma (in int0) Int_ZF_1_3_L13B:
  assumes A1: a ≤ b and A2: c ∈ ℤ⁺
  shows a · c ≤ b · c
  c · a ≤ c · b
proof -
  let R = ℤ
  let A = IntegerAddition
  let M = IntegerMultiplication
  let r = IntegerOrder

653
from A1 A2 have 
  ring1(R, A, M, r) 
  ⟨a,b⟩ ∈ r 
  c ∈ PositiveSet(R, A, r) 
  using Int_ZF_1_3_T1 by auto 
then show 
  a·c ≤ b·c 
  c·a ≤ c·b 
  using ring1.OrdRing_ZF_1_L9A by auto 
qed

A rearrangement with four integers and absolute value.

lemma (in int0) Int_ZF_1_3_L14: 
  assumes A1: a ∈ š b ∈ š c ∈ š d ∈ š 
  shows abs(a·b)+(abs(a)+c)·d = (d+abs(b))·abs(a)+c·d 
proof - 
  from A1 have T1: abs(a) ∈ š abs(b) ∈ š 
    abs(a)·abs(b) ∈ š abs(a)·d ∈ š 
    c·d ∈ š 
    abs(b)+d ∈ š 
    using Int_ZF_2_L14 Int_ZF_1_1_L5 by auto 
  with A1 have abs(a·b)+(abs(a)+c)·d = (d+abs(b))·abs(a)+c·d 
    using Int_ZF_1_3_L5 Int_ZF_1_1_L1 Int_ZF_1_1_L7 by simp 
  with A1 T1 show thesis 
    using Int_ZF_1_1_L5 by simp 
qed

A technical lemma about what happens when one absolute value is not 
greater or equal than another.

lemma (in int0) Int_ZF_1_3_L15: 
  assumes A1: m ∈ š n ∈ š 
  and A2: ¬(abs(m) ≤ abs(n)) 
  shows n ≤ abs(m) m≠0 
proof - 
  from A1 have T1: n ≤ abs(n) 
    using Int_ZF_2_L19C by simp 
  from A1 have abs(n) ∈ š abs(m) ∈ š 
    using Int_ZF_2_L14 by auto 
  moreover note A2 
  ultimately have abs(n) ≤ abs(m) 
    by (rule Int_ZF_2_L19) 
  with T1 show n ≤ abs(m) by (rule Int_order_transitive) 
  from A1 A2 show m≠0 using Int_ZF_2_L18 int_abs_nonneg by auto 
qed

Negative of a nonnegative is nonpositive.

lemma (in int0) Int_ZF_1_3_L16: 
  assumes A1: 0 ≤ m 
  shows (¬m) ≤ 0 
proof - 

654
from A1 have \((-m) \leq (-0)\)
  using Int_ZF_2_L10 by simp
then show \((-m) \leq 0\) using Int_ZF_1_L11 by simp
qed

Some statements about intervals centered at 0.

lemma (in int0) Int_ZF_1_3_L17: assumes A1: \(m \in \mathbb{Z}\)
  shows
  \((-\text{abs}(m)) \leq \text{abs}(m)\)
  \((-\text{abs}(m)) .. \text{abs}(m) \neq 0\)
proof -
  from A1 have \((-\text{abs}(m)) \leq 0\) \(0 \leq \text{abs}(m)\)
    using int_abs_nonneg Int_ZF_1_3_L16 by auto
  then show \((-\text{abs}(m)) \leq \text{abs}(m)\) by (rule Int_order_transitive)
  then have \(\text{abs}(m) \in (-\text{abs}(m)) .. \text{abs}(m)\)
    using int_ord_is_refl Int_ZF_2_L1A Order_ZF_2_L2 by simp
  thus \((-\text{abs}(m)) .. \text{abs}(m) \neq 0\) by auto
qed

The greater of two integers is indeed greater than both, and the smaller one
is smaller that both.

lemma (in int0) Int_ZF_1_3_L18: assumes A1: \(m \in \mathbb{Z}\) \(n \in \mathbb{Z}\)
  shows
  \(m \leq \text{GreaterOf}(\text{IntegerOrder},m,n)\)
  \(n \leq \text{GreaterOf}(\text{IntegerOrder},m,n)\)
  \(\text{SmallerOf}(\text{IntegerOrder},m,n) \leq m\)
  \(\text{SmallerOf}(\text{IntegerOrder},m,n) \leq n\)
using assms Int_ZF_2_T1 Order_ZF_3_L2 by auto

If \(|m| \leq n\), then \(m \in -n .. n\).

lemma (in int0) Int_ZF_1_3_L19:
  assumes A1: \(m \in \mathbb{Z}\) and A2: \(\text{abs}(m) \leq n\)
  shows
  \((-n) \leq m \leq n\)
  \(m \in (-n) .. n\)
  \(0 \leq n\)
using assms Int_ZF_2_T1 group3.OrderedGroup_ZF_3_L8B
  group3.OrderedGroup_ZF_3_L8A Order_ZF_2_L1 by auto

A slight generalization of the above lemma.

lemma (in int0) Int_ZF_1_3_L19A:
  assumes A1: \(m \in \mathbb{Z}\) and A2: \(\text{abs}(m) \leq n\) and A3: \(0 \leq k\)
  shows \((-\text{abs}(m)) \leq m\)
using assms Int_ZF_2_T1 group3.OrderedGroup_ZF_3_L8B
  by simp

Sets of integers that have absolute value bounded are bounded.
lemma (in int0) Int_ZF_1_3_L20:
assumes A1: ∀x∈X. b(x) ∈ ℤ ∧ abs(b(x)) ≤ L
shows IsBounded({b(x). x∈X},IntegerOrder)
proof -
let G = ℤ
let P = IntegerAddition
let r = IntegerOrder
from A1 have
group3(G, P, r)
r {is total on} G
∀x∈X. b(x) ∈ G ∧ ⟨AbsoluteValue(G, P, r)(b(x)), L⟩ ∈ r
using Int_ZF_2_T1 by auto
then show IsBounded({b(x). x∈X},IntegerOrder)
by (rule group3.OrderedGroup_ZF_3_L9A)
qed

If a set is bounded, then the absolute values of the elements of that set are bounded.

lemma (in int0) Int_ZF_1_3_L20A: assumes IsBounded(A,IntegerOrder)
shows ∃L. ∀a∈A. abs(a) ≤ L
using assms Int_ZF_2_T1 group3.OrderedGroup_ZF_3_L10A
by simp

Absolute values of integers from a finite image of integers are bounded by an integer.

lemma (in int0) Int_ZF_1_3_L20AA:
assumes A1: {b(x). x∈ℤ} ∈ Fin(ℤ)
shows ∃L∈ℤ. ∀x∈ℤ. abs(b(x)) ≤ L
using assms int_not_empty Int_ZF_2_T1 group3.OrderedGroup_ZF_3_L11A
by simp

If absolute values of values of some integer function are bounded, then the image a set from the domain is a bounded set.

lemma (in int0) Int_ZF_1_3_L20B:
assumes f:X→ℤ and A⊆X and ∀x∈A. abs(f(x)) ≤ L
shows IsBounded(f(A),IntegerOrder)
proof -
let G = ℤ
let P = IntegerAddition
let r = IntegerOrder
from assms have
group3(G, P, r)
r {is total on} G
f:X→G
A⊆X
∀x∈A. ⟨AbsoluteValue(G, P, r)(f(x)), L⟩ ∈ r
using Int_ZF_2_T1 by auto
then show IsBounded(f(A), r)
A special case of the previous lemma for a function from integers to integers.

**Corollary** (in int0) *Int_ZF_1_3_L20C:*
- Assumes \( f: \mathbb{Z} \to \mathbb{Z} \) and \( \forall m \in \mathbb{Z}. \, \text{abs}(f(m)) \leq L \)
- Shows \( f(\mathbb{Z}) \in \text{Fin}(\mathbb{Z}) \)

**Proof** -
- From assumptions have \( f: \mathbb{Z} \to \mathbb{Z} \subseteq \mathbb{Z} \) \( \forall m \in \mathbb{Z}. \, \text{abs}(f(m)) \leq L \)
- By auto
- Then have \( \text{IsBounded}(f(\mathbb{Z})), \text{IntegerOrder} \)
- By (rule *Int_ZF_1_3_L20B*)
- Then show \( f(\mathbb{Z}) \in \text{Fin}(\mathbb{Z}) \) using \( \text{Int_bounded_iff_fin} \)
- By simp

**QED**

A triangle inequality with three integers. Property of linearly ordered abelian groups.

**Lemma** (in int0) *Int_triangle_ineq3:*
- Assumes \( A1: a \in \mathbb{Z} \, b \in \mathbb{Z} \, c \in \mathbb{Z} \)
- Shows \( \text{abs}(a-b-c) \leq \text{abs}(a) + \text{abs}(b) + \text{abs}(c) \)

**Proof** -
- From \( A1 \) have \( T: a-b \in \mathbb{Z} \, \text{abs}(c) \in \mathbb{Z} \)
- Using *Int_ZF_1_1_L5* *Int_ZF_2_L14* by auto
- With \( A1 \) have \( \text{abs}(a-b-c) \leq \text{abs}(a-b) + \text{abs}(c) \)
- Using *Int_triangle_ineq1* by simp
- Moreover from \( A1 \) \( T \) have
- \( \text{abs}(a-b) + \text{abs}(c) \leq \text{abs}(a) + \text{abs}(b) + \text{abs}(c) \)
- Using *Int_triangle_ineq1* *int_ord_transl_inv* by simp
- Ultimately show thesis by (rule *Int_order_transitive*)

**QED**

If \( a \leq c \) and \( b \leq c \), then \( a + b \leq 2 \cdot c \). Property of ordered rings.

**Lemma** (in int0) *Int_ZF_1_3_L21:*
- Assumes \( A1: a \leq c \, b \leq c \) shows \( a+b \leq 2 \cdot c \)
- Using assumptions *Int_ZF_1_3_T1* *ring1.OrdRing_ZF_2_L6* by simp

If an integer \( a \) is between \( b \) and \( b + c \), then \( |b - a| \leq c \). Property of ordered groups.

**Lemma** (in int0) *Int_ZF_1_3_L22:*
- Assumes \( a \leq b \) and \( c \in \mathbb{Z} \) and \( b \leq c+a \)
- Shows \( \text{abs}(b-a) \leq c \)
- Using assumptions *Int_ZF_2_T1* *group3.OrderedGroup_ZF_3_L8C* by simp

An application of the triangle inequality with four integers. Property of linearly ordered abelian groups.

**Lemma** (in int0) *Int_ZF_1_3_L22A:
assumes \( a \in \mathbb{Z}, \ b \in \mathbb{Z}, \ c \in \mathbb{Z}, \ d \in \mathbb{Z} \)
shows \( |a-c| \leq |a+b| + |c+d| + |b-d| \)
using assms Int_ZF_1_T2 Int_ZF_2_T1 group3.OrderedGroup_ZF_3_L7F by simp

If an integer \( a \) is between \( b \) and \( b + c \), then \( |b - a| \leq c \). Property of ordered groups. A version of Int_ZF_1_3_L22 with slightly different assumptions.

**Lemma (in int0) Int_ZF_1_3_L23:**
assumes \( A1: \ a \leq b \) and \( A2: \ c \in \mathbb{Z} \) and \( A3: \ b \leq a+c \)
shows \( |b-a| \leq c \)
proof -
from \( A1 \) have \( a \in \mathbb{Z} \)
using Int_ZF_2_L1A by simp
with \( A2 \) \( A3 \) have \( b \leq c+a \)
using Int_ZF_1_1_L5 by simp
with \( A1 \) \( A2 \) show \( |b-a| \leq c \)
using Int_ZF_1_3_L22 by simp
qed

54.4 Maximum and minimum of a set of integers

In this section we provide some sufficient conditions for integer subsets to have extrema (maxima and minima).

Finite nonempty subsets of integers attain maxima and minima.

**Theorem (in int0) Int_fin_have_max_min:**
assumes \( A1: \ A \in \text{Fin}(\mathbb{Z}) \) and \( A2: \ A \neq 0 \)
shows
- \( \text{HasAmaximum}(\text{IntegerOrder},A) \)
- \( \text{HasAminimum}(\text{IntegerOrder},A) \)
- \( \text{Maximum}(\text{IntegerOrder},A) \in A \)
- \( \text{Minimum}(\text{IntegerOrder},A) \in A \)
- \( \forall x \in A. \ x \leq \text{Maximum}(\text{IntegerOrder},A) \)
- \( \forall x \in A. \ \text{Minimum}(\text{IntegerOrder},A) \leq x \)
- \( \text{Maximum}(\text{IntegerOrder},A) \in \mathbb{Z} \)
- \( \text{Minimum}(\text{IntegerOrder},A) \in \mathbb{Z} \)
proof -
from \( A1 \) have
- \( A=0 \lor \text{HasAmaximum}(\text{IntegerOrder},A) \) and
- \( A=0 \lor \text{HasAminimum}(\text{IntegerOrder},A) \)
using Int_ZF_2_T1 Int_ZF_2_L6 Finite_ZF_1_1_T1A Finite_ZF_1_1_T1B by auto
with \( A2 \) show
- \( \text{HasAmaximum}(\text{IntegerOrder},A) \)
- \( \text{HasAminimum}(\text{IntegerOrder},A) \)
by auto
from \( A1 \) \( A2 \) show
- \( \text{Maximum}(\text{IntegerOrder},A) \in A \)
- \( \text{Minimum}(\text{IntegerOrder},A) \in A \)
∀x∈A. x ≤ \text{Maximum}(\text{IntegerOrder},A)
∀x∈A. \text{Minimum}(\text{IntegerOrder},A) ≤ x

using Int_ZF_2_T1 Finite_ZF_1_T2 by auto

moreover from A1 have A⊆\mathbb{Z} using FinD by simp

ultimately show
\text{Maximum}(\text{IntegerOrder},A) ∈ \mathbb{Z}
\text{Minimum}(\text{IntegerOrder},A) ∈ \mathbb{Z}

by auto

qed

Bounded nonempty integer subsets attain maximum and minimum.

**Theorem (in int0) Int_bounded_have_max_min:**

assumes IsBounded(A,\text{IntegerOrder}) and A≠0

shows
\text{HasAmaximum}(\text{IntegerOrder},A)
\text{HasAminimum}(\text{IntegerOrder},A)
\text{Maximum}(\text{IntegerOrder},A) ∈ A
\text{Minimum}(\text{IntegerOrder},A) ∈ A
∀x∈A. x ≤ \text{Maximum}(\text{IntegerOrder},A)
∀x∈A. \text{Minimum}(\text{IntegerOrder},A) ≤ x

Maximum(\text{IntegerOrder},A) ∈ \mathbb{Z}
Minimum(\text{IntegerOrder},A) ∈ \mathbb{Z}

using assms Int_fin_have_max_min Int_bounded_iff_fin

by auto

Nonempty set of integers that is bounded below attains its minimum.

**Theorem (in int0) int_bounded_below_has_min:**

assumes A1: IsBoundedBelow(A,\text{IntegerOrder}) and A2: A≠0

shows
\text{HasAminimum}(\text{IntegerOrder},A)
\text{Minimum}(\text{IntegerOrder},A) ∈ A

∀x∈A. \text{Minimum}(\text{IntegerOrder},A) ≤ x

proof -

from A1 A2 have
\text{IntegerOrder} \{\text{is total on}\} \mathbb{Z}
\text{trans}(\text{IntegerOrder})
\text{IntegerOrder} ⊆ \mathbb{Z}×\mathbb{Z}
∀A. \text{IsBounded}(A,\text{IntegerOrder}) ∧ A≠0 → \text{HasAminimum}(\text{IntegerOrder},A)
A≠0 \text{IsBoundedBelow}(A,\text{IntegerOrder})

using Int_ZF_2_T1 Int_ZF_2_L6 Int_ZF_2_L1B Int_bounded_have_max_min

by auto

then show \text{HasAminimum}(\text{IntegerOrder},A)

by (rule Order_ZF_4_L11)

then show
\text{Minimum}(\text{IntegerOrder},A) ∈ A
∀x∈A. \text{Minimum}(\text{IntegerOrder},A) ≤ x

using Int_ZF_2_L4 Order_ZF_4_L4 by auto

qed
Nonempty set of integers that is bounded above attains its maximum.

**theorem (in int0) int_bounded_above_has_max:**

**assumes** A1: IsBoundedAbove(A,IntegerOrder) and A2: A\neq 0

**shows**

HasAmaximum(IntegerOrder,A)

Maximum(IntegerOrder,A) \in A

Maximum(IntegerOrder,A) \in \mathbb{Z}

\forall x \in A. x \leq Maximum(IntegerOrder,A)

**proof**

from A1 A2 have

IntegerOrder \{is total on\} \mathbb{Z}

trans(IntegerOrder) and

I: IntegerOrder \subseteq \mathbb{Z} \times \mathbb{Z} and

\forall A. IsBounded(A,IntegerOrder) \land A\neq 0 \implies HasAmaximum(IntegerOrder,A)

A\neq 0 IsBoundedAbove(A,IntegerOrder)

using Int_ZF_2_T1 Int_ZF_2_L6 Int_ZF_2_L1B Int_bounded_have_max_min

by auto

then show HasAmaximum(IntegerOrder,A)

by (rule Order_ZF_4_L11A)

then show

II: Maximum(IntegerOrder,A) \in A and

\forall x \in A. x \leq Maximum(IntegerOrder,A)

using Int_ZF_2_L4 Order_ZF_4_L3 by auto

from I A1 have A \subseteq \mathbb{Z} by (rule Order_ZF_3_L1A)

with II show Maximum(IntegerOrder,A) \in \mathbb{Z} by auto

qed

A set defined by separation over a bounded set attains its maximum and minimum.

**lemma (in int0) Int_ZF_1_4_L1:**

**assumes** A1: IsBounded(A,IntegerOrder) and A2: A\neq 0 and A3: \forall q \in \mathbb{Z}. F(q) \in \mathbb{Z} and A4: K = \{F(q). q \in A\}

**shows**

HasAmaximum(IntegerOrder,K)

HasAminimum(IntegerOrder,K)

Maximum(IntegerOrder,K) \in K

Minimum(IntegerOrder,K) \in K

Maximum(IntegerOrder,K) \in \mathbb{Z}

Minimum(IntegerOrder,K) \in \mathbb{Z}

\forall q \in A. F(q) \leq Maximum(IntegerOrder,K)

\forall q \in A. Minimum(IntegerOrder,K) \leq F(q)

IsBounded(K,IntegerOrder)

**proof**

from A1 have A \in Fin(\mathbb{Z}) using Int_bounded_iff_fin

by simp

with A3 have \{F(q). q \in A\} \in Fin(\mathbb{Z})

by (rule fin_image_fin)

with A2 A4 have T1: K \in Fin(\mathbb{Z}) K\neq 0 by auto
then show T2:

\begin{align*}
\text{HasAmaximum}(\text{IntegerOrder}, K) \\
\text{HasAminimum}(\text{IntegerOrder}, K) \\
\text{and } \text{Maximum}(\text{IntegerOrder}, K) \in K \\
\text{Minimum}(\text{IntegerOrder}, K) \in K \\
\text{Maximum}(\text{IntegerOrder}, K) \in \mathbb{Z} \\
\text{Minimum}(\text{IntegerOrder}, K) \in \mathbb{Z}
\end{align*}

using \text{Int_fin_have_max_min} by auto

\{ fix \ q assume \ q \in A \\
with \ A4 have \ F(q) \in K by auto \\
with \ T1 have \\
\quad \text{F}(q) \leq \text{Maximum}(\text{IntegerOrder}, K) \\
\quad \text{Minimum}(\text{IntegerOrder}, K) \leq \text{F}(q) \\
\quad \text{using } \text{Int_fin_have_max_min} \text{ by auto} \\
\} then show \\
\forall q \in A. F(q) \leq \text{Maximum}(\text{IntegerOrder}, K) \\
\forall q \in A. \text{Minimum}(\text{IntegerOrder}, K) \leq F(q) \\
by \text{auto} \\
from \ T2 show \ \text{IsBounded}(K, \text{IntegerOrder}) \\
using \text{Order_ZF_4_L7} \text{ Order_ZF_4_L8A} \text{ IsBounded_def} \\
by \text{simp} \\
\text{qed}

A three element set has a maximume and minimum.

\text{lemma} (in int0) \text{Int_ZF_1_4_L1A}: \text{assumes } A1: a \in \mathbb{Z} \text{ b} \in \mathbb{Z} \text{ c} \in \mathbb{Z} \\
\text{shows } \\
\text{Maximum}(\text{IntegerOrder}, \{a,b,c\}) \in \mathbb{Z} \\
a \leq \text{Maximum}(\text{IntegerOrder}, \{a,b,c\}) \\
b \leq \text{Maximum}(\text{IntegerOrder}, \{a,b,c\}) \\
c \leq \text{Maximum}(\text{IntegerOrder}, \{a,b,c\}) \\
\text{using } \text{assms} \text{ Int_ZF_2_T1 Finite_ZF_1_L2A} \text{ by auto}

Integer functions attain maxima and minima over intervals.

\text{lemma} (in int0) \text{Int_ZF_1_4_L2}: \\
\text{assumes } A1: f: \mathbb{Z} \to \mathbb{Z} \text{ and } A2: a\leq b \\
\text{shows } \\
\text{maxf}(f,a..b) \in \mathbb{Z} \\
\forall c \in a..b. f(c) \leq \text{maxf}(f,a..b) \\
\exists c \in a..b. f(c) = \text{maxf}(f,a..b) \\
\text{minf}(f,a..b) \in \mathbb{Z} \\
\forall c \in a..b. \text{minf}(f,a..b) \leq f(c) \\
\exists c \in a..b. f(c) = \text{minf}(f,a..b) \\
\text{proof} - \\
from \ A2 have T: a \in \mathbb{Z} \ b \in \mathbb{Z} \ a..b \subseteq \mathbb{Z} \\
\text{using } \text{Int_ZF_2_L1A Int_ZF_2_L1B Order_ZF_2_L6} \text{ by auto} \\
with \ A1 A2 have \\
\text{Maximum}(\text{IntegerOrder}, f(a..b)) \in f(a..b) \\
\forall x \in f(a..b). x \leq \text{Maximum}(\text{IntegerOrder}, f(a..b))

661
Maximum(IntegerOrder, f(a..b)) ∈ ℤ
Minimum(IntegerOrder, f(a..b)) ∈ f(a..b)
∀x∈f(a..b). Minimum(IntegerOrder, f(a..b)) ≤ x
Minimum(IntegerOrder, f(a..b)) ∈ ℤ
using Int_ZF_4_L8 Int_ZF_2_T1 group3.OrderedGroup_ZF_2_L6
Int_fin_have_max_min by auto

with A1 T show
maxf(f,a..b) ∈ ℤ
∀c ∈ a..b. f(c) ≤ maxf(f,a..b)
∃c ∈ a..b. f(c) = maxf(f,a..b)
minf(f,a..b) ∈ ℤ
∀c ∈ a..b. minf(f,a..b) ≤ f(c)
∃c ∈ a..b. f(c) = minf(f,a..b)
using func_imagedef by auto

qed

54.5 The set of nonnegative integers

The set of nonnegative integers looks like the set of natural numbers. We explore that in this section. We also rephrase some lemmas about the set of positive integers known from the theory of ordered groups.

The set of positive integers is closed under addition.

lemma (in int0) pos_int_closed_add:
  shows ℤ⁺ {is closed under} IntegerAddition
  using Int_ZF_2_T1 group3.OrderedGroup_ZF_1_L13 by simp

Text expended version of the fact that the set of positive integers is closed under addition

lemma (in int0) pos_int_closed_add_unfolded:
  assumes a∈ℤ⁺ b∈ℤ⁺ shows a+b ∈ ℤ⁺
  using assms pos_int_closed_add IsOpClosed_def by simp

ℤ⁺ is bounded below.

lemma (in int0) Int_ZF_1_5_L1: shows
  IsBoundedBelow(ℤ⁺, IntegerOrder)
  IsBoundedBelow(ℤ⁺, IntegerOrder)
  using Nonnegative_def PositiveSet_def IsBoundedBelow_def by auto

Subsets of ℤ⁺ are bounded below.

lemma (in int0) Int_ZF_1_5_L1A: assumes A ⊆ ℤ⁺
  shows IsBoundedBelow(A, IntegerOrder)
  using assms Int_ZF_1_5_L1 Order_ZF_3_L12 by blast

Subsets of ℤ⁺ are bounded below.

lemma (in int0) Int_ZF_1_5_L1B: assumes A1: A ⊆ ℤ⁺
shows \( \text{IsBoundedBelow}(A, \text{IntegerOrder}) \)
using \( A1 \) \( \text{IntZF}_1.5.L1 \) \( \text{OrderZF}_3.L12 \) by blast

Every nonempty subset of positive integers has a minimum.

**lemma** (in int0) \( \text{IntZF}_1.5.L1C \):
assumes \( A \subseteq \mathbb{Z}_+ \) and \( A \neq \emptyset \)
shows
\( \text{HasAminimum}(\text{IntegerOrder}, A) \)
\( \text{Minimum}(\text{IntegerOrder}, A) \in A \)
\( \forall x \in A. \text{Minimum}(\text{IntegerOrder}, A) \leq x \)
using assms \( \text{IntZF}_1.5.L1B \) \( \text{int_bounded_below_has_min} \) by auto

Infinite subsets of \( \mathbb{Z}_+ \) do not have a maximum - If \( A \subseteq \mathbb{Z}_+ \) then for every integer we can find one in the set that is not smaller.

**lemma** (in int0) \( \text{IntZF}_1.5.L2 \):
assumes \( A1: A \subseteq \mathbb{Z}_+ \) and \( A2: A \notin \text{Fin}(\mathbb{Z}) \) and \( A3: D \in \mathbb{Z} \)
shows \( \exists n \in A. D \leq n \)
proof -
{ assume \( \forall n \in A. \neg (D \leq n) \)
  moreover from \( A1 \) \( A3 \) have \( D \in \mathbb{Z} \) \( \forall n \in A. n \in \mathbb{Z} \)
  using \( \text{Nonnegative_def} \) by auto
  ultimately have \( \forall n \in A. n \leq D \)
  using \( \text{IntZF}_2.L19 \) by blast
  hence \( \forall n \in A. (n, D) \in \text{IntegerOrder} \) by simp
  then have \( \text{IsBoundedAbove}(A, \text{IntegerOrder}) \)
  by (rule \( \text{OrderZF}_3.L10 \))
  with \( A1 \) have \( \text{IsBounded}(A, \text{IntegerOrder}) \)
  using \( \text{IntZF}_1.5.L1A \) \( \text{IsBounded_def} \) by simp
  with \( A2 \) have False using \( \text{Int_bounded_iff_fin} \) by auto
} thus thesis by auto
qed

Infinite subsets of \( \mathbb{Z}_+ \) do not have a maximum - If \( A \subseteq \mathbb{Z}_+ \) then for every integer we can find one in the set that is not smaller. This is very similar to \( \text{IntZF}_1.5.L2 \), except we have \( \mathbb{Z}_+ \) instead of \( \mathbb{Z}_+ \) here.

**lemma** (in int0) \( \text{IntZF}_1.5.L2A \):
assumes \( A1: A \subseteq \mathbb{Z}_+ \) and \( A2: A \notin \text{Fin}(\mathbb{Z}) \) and \( A3: D \in \mathbb{Z} \)
shows \( \exists n \in A. D \leq n \)
proof -
{ assume \( \forall n \in A. \neg (D \leq n) \)
  moreover from \( A1 \) \( A3 \) have \( D \in \mathbb{Z} \) \( \forall n \in A. n \in \mathbb{Z} \)
  using \( \text{PositiveSet_def} \) by auto
  ultimately have \( \forall n \in A. n \leq D \)
  using \( \text{IntZF}_2.L19 \) by blast
  hence \( \forall n \in A. (n, D) \in \text{IntegerOrder} \) by simp
  then have \( \text{IsBoundedAbove}(A, \text{IntegerOrder}) \)
  by (rule \( \text{OrderZF}_3.L10 \))
  with \( A1 \) have \( \text{IsBounded}(A, \text{IntegerOrder}) \)
  using \( \text{IntZF}_1.5.L1B \) \( \text{IsBounded_def} \) by simp
} 663
with A2 have False using Int_bounded_iff_fin by auto
} thus thesis by auto
qed

An integer is either positive, zero, or its opposite is positive.

lemma (in int0) Int_decomp: assumes m∈Z
  shows Exactly_1_of_3_holds (m=0,m∈Z+,(-m)∈Z+)
  using assms Int_ZF_2_T1 group3.OrdGroup_decomp
  by simp

An integer is zero, positive, or it’s inverse is positive.

lemma (in int0) int_decomp_cases: assumes m∈Z
  shows m=0 ∨ m∈Z+ ∨ (-m)∈Z+
  using assms Int_ZF_2_T1 group3.OrderedGroup_ZF_1_L14
  by simp

An integer is in the positive set iff it is greater or equal one.

lemma (in int0) Int_ZF_1_5_L3: shows m∈Z+ ←→ 1≤m
proof
  assume m∈Z+
  then have 0≤m m≠0
    using PositiveSet_def by auto
  then have 0+1 ≤ m
    using Int_ZF_4_L1B by auto
  then show 1≤m
    using int_zero_one_are_int Int_ZF_1_T2 group0.group0_2_L2
    by simp
next
  assume 1≤m
  then have m∈Z 0≤m m≠0
    using Int_ZF_2_L1A Int_ZF_2_L16C by auto
  then show m∈Z+ using PositiveSet_def by auto
qed

The set of positive integers is closed under multiplication. The unfolded form.

lemma (in int0) pos_int_closed_mul_unfold: assumes a∈Z+ b∈Z+
  shows a·b ∈ Z+
  using assms Int_ZF_1_5_L3 Int_ZF_1_3_L3 by simp

The set of positive integers is closed under multiplication.

lemma (in int0) pos_int_closed_mul: shows Z+ {is closed under} IntegerMultiplication
  using pos_int_closed_mul_unfold IsOpClosed_def
  by simp

It is an overkill to prove that the ring of integers has no zero divisors this way, but why not?

lemma (in int0) int_has_no_zero_divs:
shows \text{HasNoZeroDivs}(\mathbb{Z}, \text{IntegerAddition}, \text{IntegerMultiplication})
using \text{pos_int_closed_mul} \text{Int}_\text{ZF}_1 \text{T1} \text{ring1.OrdRing}_\text{ZF}_3 \text{L3}
by simp

Nonnegative integers are positive ones plus zero.

lemma (in int0) Int_ZF_1_5_L3A: shows \(\mathbb{Z}^+ = \mathbb{Z}_+ \cup \{0\}\)
using Int_ZF_2_T1 \text{group3.OrderedGroup}_\text{ZF}_1 \text{L24} by simp

We can make a function smaller than any constant on a given interval of positive integers by adding another constant.

lemma (in int0) Int_ZF_1_5_L4:
assumes A1: \(f: \mathbb{Z} \to \mathbb{Z}\) and A2: \(K \in \mathbb{Z}, N \in \mathbb{Z}\)
shows \(\exists C \in \mathbb{Z}. \forall n \in \mathbb{Z}_+. \ K \leq f(n) + C \rightarrow N \leq n\)

proof -
from A2 have \(N \leq 1 \lor 2 \leq N\)
using int_zero_one_are_int no_int_between
by simp
moreover
\{ assume A3: \(N \leq 1\)
let \(C = 0\)
have \(C \in \mathbb{Z}\) using int_zero_one_are_int
by simp
moreover
\{ fix n assume \(n \in \mathbb{Z}_+\)
then have \(1 \leq n\) using Int_ZF_1_5_L3
by simp
with A3 have \(N \leq n\) by (rule Int_order_transitive)
\} then have \(\forall n \in \mathbb{Z}_+. \ K \leq f(n) + C \rightarrow N \leq n\)
by auto
ultimately have \(\exists C \in \mathbb{Z}. \forall n \in \mathbb{Z}_+. \ K \leq f(n) + C \rightarrow N \leq n\)
by auto 
moreover
\{ let \(C = K - 1 - \text{maxf}(f, 1 .. (N-1))\)
assume \(2 \leq N\)
then have \(2 - 1 \leq N - 1\)
using int_zero_one_are_int Int_ZF_1_1_L5 int_ord_transl_inv
by simp
then have I: \(1 \leq N - 1\)
using int_zero_one_are_int Int_ZF_1_1_L4 by simp
with A1 A2 have T:
\(\text{maxf}(f, 1 .. (N-1)) \in \mathbb{Z}, K - 1 \in \mathbb{Z}, C \in \mathbb{Z}\)
using Int_ZF_1_4_L2 Int_ZF_1_1_L5 int_zero_one_are_int
by auto
moreover
\{ fix n assume A4: \(n \in \mathbb{Z}_+\)
\{ assume A5: \(K \leq f(n) + C \) and \(\neg (N \leq n)\)
with A2 A4 have \(n \leq N - 1\)
using PositiveSet_def Int_ZF_1_3_L6A by simp
with A4 have \(n \in 1 .. (N-1)\)
}\}
Absolute value is identity on positive integers.

**lemma (in int0) Int_ZF_1_5_L4A:**

assumes \( a \in \mathbb{Z}^+ \)

shows \( \text{abs}(a) = a \)

using assms Int_ZF_2_T1 group3.OrderedGroup_ZF_3_L2B

by simp

One and two are in \( \mathbb{Z}^+ \).

**lemma (in int0) Int_ZF_1_5_L5:**

assumes \( A1: f : \mathbb{Z}^+ \rightarrow X \)

shows \( f(\mathbb{Z}^+) \neq 0 \)

proof -

have \( \mathbb{Z}^+ \subseteq \mathbb{Z} \) using PositiveSet_def by auto

with \( A1 \) show \( f(\mathbb{Z}^+) \neq 0 \)

using Int_ZF_2_L16B by auto

The image of \( \mathbb{Z}^+ \) by a function defined on integers is not empty.

**lemma (in int0) Int_ZF_1_5_L6:**

assumes \( A1: f : \mathbb{Z} \rightarrow X \)

shows \( f(\mathbb{Z}^+) \neq 0 \)

proof -

have \( 0 \leq n-1 \) using Int_ZF_1_5_L3 by auto

then have \( 1 \leq n \) using Int_ZF_1_5_L3 by auto

proof -

from \( A1 \) have \( 1 \leq n \) \((-1) \in \mathbb{Z} \)

by auto

then have \( 1-1 \leq n-1 \)
using int_ord_transl_inv by simp
then show 0 ≤ n-1
  using int_zero_one_are_int Int_ZF_1_1_L4 by simp
then show 0 ∈ 0..(n-1)
  using int_zero_one_are_int int_ord_is_refl refl_def Order_ZF_2_L1B
  by simp
show 0..(n-1) ⊆ ℤ
  using Int_ZF_2_L1B Order_ZF_2_L6 by simp
qed

Intgers greater than one in ℤ⁺ belong to ℤ⁺. This is a property of ordered
groups and follows from OrderedGroup_ZF_1_L19, but Isabelle’s simplifier has
problems using that result directly, so we reprove it specifically for integers.

lemma (in int0) Int_ZF_1_5_L7: assumes a ∈ ℤ⁺ and a≤b
shows b ∈ ℤ⁺
proof-
  from assms have 1≤a a≤b
    using Int_ZF_1_5_L3 by auto
  then have 1≤b by (rule Int_order_transitive)
  then show b ∈ ℤ⁺ using Int_ZF_1_5_L3 by simp
qed

Adding a positive integer increases integers.

lemma (in int0) Int_ZF_1_5_L7A: assumes a∈ℤ b∈ℤ⁺
shows a ≤ a+b a ≠ a+b a+b ∈ ℤ
  using assms Int_ZF_2_T1 group3.OrderedGroup_ZF_1_L22
  by auto

For any integer m the greater of m and 1 is a positive integer that is greater
or equal than m. If we add 1 to it we get a positive integer that is strictly
greater than m.

lemma (in int0) Int_ZF_1_5_L7B: assumes a∈ℤ
shows a ≤ GreaterOf(IntegerOrder,1,a)
  GreaterOf(IntegerOrder,1,a) ∈ ℤ⁺
  GreaterOf(IntegerOrder,1,a) + 1 ∈ ℤ⁺
  a ≤ GreaterOf(IntegerOrder,1,a) + 1
  a ≠ GreaterOf(IntegerOrder,1,a) + 1
  using assms int_zero_not_one Int_ZF_1_3_T1 ring1.OrdRing_ZF_3_L12
  by auto

The opposite of an element of ℤ⁺ cannot belong to ℤ⁺.

lemma (in int0) Int_ZF_1_5_L8: assumes a ∈ ℤ⁺
shows (¬a) ∉ ℤ⁺
  using assms Int_ZF_2_T1 group3.OrderedGroup_ZF_1_L20
  by simp

For every integer there is one in ℤ⁺ that is greater or equal.
lemma (in int0) Int_ZF_1_5_L9: assumes a∈š shows ∃b∈š. a≤b
using assms int_not_trivial Int_ZF_2_T1 group3.OrderedGroup_ZF_1_L23 by simp

A theorem about odd extensions. Recall from OrdereGroup_ZF.thy that the odd extension of an integer function \( f \) defined on \( \mathbb{Z}_+ \) is the odd function on \( \mathbb{Z} \) equal to \( f \) on \( \mathbb{Z}_+ \). First we show that the odd extension is defined on \( \mathbb{Z} \).

lemma (in int0) Int_ZF_1_5_L10: assumes f : \( \mathbb{Z}_+ \rightarrow \mathbb{Z} \)
shows OddExtension(š,IntegerAddition,IntegerOrder,f) : Š→Š
using assms Int_ZF_2_T1 group3.odd_ext_props by simp

On \( \mathbb{Z}_+ \), the odd extension of \( f \) is the same as \( f \).

lemma (in int0) Int_ZF_1_5_L11: assumes f : \( \mathbb{Z}_+ \rightarrow \mathbb{Z} \) and a ∈ \( \mathbb{Z}_+ \) and
g = OddExtension(š,IntegerAddition,IntegerOrder,f)
shows g(a) = f(a)
using assms Int_ZF_2_T1 group3.odd_ext_props by simp

On \( -\mathbb{Z}_+ \), the value of the odd extension of \( f \) is the negative of \( f(-a) \).

lemma (in int0) Int_ZF_1_5_L12:
assumes f : \( \mathbb{Z}_+ \rightarrow \mathbb{Z} \) and a ∈ \( \mathbb{Z}_+ \) and
g = OddExtension(š,IntegerAddition,IntegerOrder,f)
shows g(a) = -(f(-a))
using assms Int_ZF_2_T1 group3.odd_ext_props by simp

Odd extensions are odd on \( \mathbb{Z} \).

lemma (in int0) int_oddext_is_odd:
assumes f : \( \mathbb{Z}_+ \rightarrow \mathbb{Z} \) and a∈\( \mathbb{Z} \) and
g = OddExtension(š,IntegerAddition,IntegerOrder,f)
shows g(-a) = -(g(a))
using assms Int_ZF_2_T1 group3.odd_ext_props by simp

Alternative definition of an odd function.

lemma (in int0) Int_ZF_1_5_L13: assumes A1: f: \( \mathbb{Z} \rightarrow \mathbb{Z} \) shows \((\forall a∈\mathbb{Z}. f(-a) = -(f(a))) \leftrightarrow (\forall a∈\mathbb{Z}. -(f(-a))) = f(a))\)
using assms Int_ZF_1_T2 group0.group0_6_L2 by simp

Another way of expressing the fact that odd extensions are odd.

lemma (in int0) int_oddext_is_odd_alt:
assumes f : \( \mathbb{Z}_+ \rightarrow \mathbb{Z} \) and a∈\( \mathbb{Z} \) and
g = OddExtension(š,IntegerAddition,IntegerOrder,f)
shows \(-g(-a)) = g(a)
using assms Int_ZF_2_T1 group3.odd_ext_is_odd_alt by simp

54.6 Functions with infinite limits

In this section we consider functions (integer sequences) that have infinite limits. An integer function has infinite positive limit if it is arbitrarily large
for large enough arguments. Similarly, a function has infinite negative limit
if it is arbitrarily small for small enough arguments. The material in this
come mostly from the section in OrderedGroup_ZF.thy with he same title.
Here we rewrite the theorems from that section in the notation we use for
integers and add some results specific for the ordered group of integers.

If an image of a set by a function with infinite positive limit is bounded
above, then the set itself is bounded above.

**Lemma (in int0) Int_ZF_1_6_L1**

assumes \( f : \mathbb{Z} \rightarrow \mathbb{Z} \)
and \( \forall a \in \mathbb{Z}. \exists b \in \mathbb{Z}_+. \forall x. b \leq x \rightarrow a \leq f(x) \) and \( A \subseteq \mathbb{Z} \)
and \( \text{IsBoundedAbove}(f(A), \text{IntegerOrder}) \)
shows \( \text{IsBoundedAbove}(A, \text{IntegerOrder}) \)
using assms int_not_trivial Int_ZF_2_T1 group3.OrderedGroup_ZF_7_L1
by simp

If an image of a set defined by separation by a function with infinite positive
limit is bounded above, then the set itself is bounded above.

**Lemma (in int0) Int_ZF_1_6_L2**

assumes \( A1: X \neq \emptyset \) and \( A2: f : \mathbb{Z} \rightarrow \mathbb{Z} \)
and \( A3: \forall a \in \mathbb{Z}. \exists b \in \mathbb{Z}_+. \forall x. b \leq x \rightarrow a \leq f(x) \) and
\( A4: \forall x \in X. b(x) \in \mathbb{Z} \land f(b(x)) \leq U \)
shows \( \exists u. \forall x \in X. b(x) \leq u \)
proof -
let \( G = \mathbb{Z} \)
let \( P = \text{IntegerAddition} \)
let \( r = \text{IntegerOrder} \)
from \( A1 \ A2 \ A3 \ A4 \) have
\( \exists u. \forall x \in X. b(x) \leq u \)
by simp

If an image of a set defined by separation by a function with infinite negative
limit is bounded below, then the set itself is bounded above. This
is dual to Int_ZF_1_6_L2.

**Lemma (in int0) Int_ZF_1_6_L3**

assumes \( A1: X \neq \emptyset \) and \( A2: f : \mathbb{Z} \rightarrow \mathbb{Z} \)
and \( A3: \forall a \in \mathbb{Z}. \exists b \in \mathbb{Z}_+ \forall x. b \leq x \rightarrow f(-x) \leq a \) and
\( A4: \forall x \in X. b(x) \in \mathbb{Z} \land L \leq f(b(x)) \)
shows \( \exists l. \forall x \in X. 1 \leq b(x) \)
proof -
let \( G = \mathbb{Z} \)

669
let $P =$ IntegerAddition
let $r =$ IntegerOrder
from $A1$ $A2$ $A3$ $A4$ have
  group3($G$, $P$, $r$)
    $r$ {is total on} $G$
    $G \neq \{\text{TheNeutralElement}(G, P)\}$
    $X \neq 0$
    $f: G \rightarrow G$
  $\forall a \in G. \exists b \in \text{PositiveSet}(G, P, r) . \forall y. (b, y) \in r \rightarrow (f(\text{GroupInv}(G, P)(y)), a) \in r$
  $\forall x \in X. b(x) \in G \land (L, f(b(x))) \in r$
using int_not_trivial Int_ZF_2_T1 by auto
then have $\exists l. \forall x \in X. (l, b(x)) \in r$ by (rule group3.OrderedGroup_ZF_7_L3)
thus thesis by simp
qed

The next lemma combines Int_ZF_1_6_L2 and Int_ZF_1_6_L3 to show that if the image of a set defined by separation by a function with infinite limits is bounded, then the set itself is bounded. The proof again uses directly a fact from OrderedGroup_ZF.

lemma (in int0) Int_ZF_1_6_L4:
  assumes $A1: X \neq 0$ and $A2: f: \mathbb{Z} \rightarrow \mathbb{Z}$ and
  $A3: \forall a \in \mathbb{Z}. \exists b \in \mathbb{Z}^+. \forall x. b \leq x \rightarrow a \leq f(x)$ and
  $A4: \forall a \in \mathbb{Z}. \exists b \in \mathbb{Z}^+. \forall y. b \leq y \rightarrow f(-y) \leq a$ and
  $A5: \forall x \in X. (b(x) \in \mathbb{Z} \land f(b(x)) \leq U \land L \leq f(b(x)))$
shows $\exists M. \forall x \in X. \text{abs}(b(x)) \leq M$
proof -
  let $G = \mathbb{Z}$
  let $P =$ IntegerAddition
  let $r =$ IntegerOrder
  from $A1$ $A2$ $A3$ $A4$ $A5$ have
  group3($G$, $P$, $r$)
  $r$ {is total on} $G$
  $G \neq \{\text{TheNeutralElement}(G, P)\}$
  $X \neq 0$
  $f: G \rightarrow G$
  $\forall a \in G. \exists b \in \text{PositiveSet}(G, P, r). \forall y. (b, y) \in r \rightarrow (a, f(y)) \in r$
  $\forall a \in G. \exists b \in \text{PositiveSet}(G, P, r). \forall y. (b, y) \in r \rightarrow (a, f(y)) \in r$
  $\forall a \in G. \exists b \in \text{PositiveSet}(G, P, r). \forall y. (b, y) \in r \rightarrow (a, f(y)) \in r$
using int_not_trivial Int_ZF_2_T1 by auto
then have $\exists M. \forall x \in X. (\text{AbsoluteValue}(G, P, r) b(x), M) \in r$
  by (rule group3.OrderedGroup_ZF_7_L3)
thus thesis by simp
qed

If a function is larger than some constant for arguments large enough, then the image of a set that is bounded below is bounded below. This is not true for ordered groups in general, but only for those for which bounded sets are finite. This does not require the function to have infinite limit, but such functions do have this property.
lemma (in int0) Int_ZF_1_6_L5:
assumes A1: f: ℤ→ ℤ and A2: N∈ ℤ and A3: ∀ m. N≤ m ⟹ L ≤ f(m) and A4: IsBoundedBelow(A,IntegerOrder)
shows IsBoundedBelow(f(A),IntegerOrder)

proof -
from A2 A4 have A = {x∈ A. x≤ N} ∪ {x∈ A. N≤ x}
using Int_ZF_2_T1 Int_ZF_2_L1C Order_ZF_1_L5
by simp
moreover have
f({x∈ A. x≤ N} ∪ {x∈ A. N≤ x}) = f{x∈ A. x≤ N} ∪ f{x∈ A. N≤ x}
by (rule image_Un)
ultimately have f(A) = f{x∈ A. x≤ N} ∪ f{x∈ A. N≤ x}
by simp
moreover have IsBoundedBelow(f{x∈ A. x≤ N},IntegerOrder)
proof -
let B = {x∈ A. x≤ N}
from A4 have B ∈ Fin(ℤ)
using Order_ZF_3_L16 Int_bounded_iff_fin by auto
with A1 have IsBounded(f(B),IntegerOrder)
using Finite1_L6A Int_bounded_iff_fin by simp
then show IsBoundedBelow(f(B),IntegerOrder)
using IsBounded_def by simp
qed
moreover have IsBoundedBelow(f{x∈ A. N≤ x},IntegerOrder)
proof -
let C = {x∈ A. N≤ x}
from A4 have C ⊆ ℤ using Int_ZF_2_L1C by auto
with A1 A3 have ∀ y ∈ f(C). ⟨L,y⟩ ∈ IntegerOrder
using func_imagedef by simp
then show IsBoundedBelow(f(C),IntegerOrder)
by (rule Order_ZF_3_L9)
qed
ultimately show IsBoundedBelow(f(A),IntegerOrder)
using Int_ZF_2_T1 Int_ZF_2_L6 Int_ZF_2_L1B Order_ZF_3_L6
by simp
qed

A function that has an infinite limit can be made arbitrarily large on positive integers by adding a constant. This does not actually require the function to have infinite limit, just to be larger than a constant for arguments large enough.

lemma (in int0) Int_ZF_1_6_L6: assumes A1: N∈ ℤ and A2: ∀ m. N≤ m ⟹ L ≤ f(m) and A3: f: ℤ→ ℤ and A4: K∈ ℤ
shows ∃ c∈ ℤ. ∀ n∈ ℤ+. K ≤ f(n)+c

proof -
have IsBoundedBelow(ℤ+,IntegerOrder)
using Int_ZF_1_5_L1 by simp
with A3 A1 A2 have IsBoundedBelow(f(Z_+),IntegerOrder)
  by (rule Int_ZF_1_6_L5)
with A1 obtain l where I: \forall y \in f(Z_+). l \leq y
  using Int_ZF_1_5_L5 IsBoundedBelow_def by auto
let c = K - l
from A3 have f(Z_+) \neq 0 using Int_ZF_1_5_L5
  by simp
then have \exists y. y \in f(Z_+) by (rule nonempty_has_element)
then obtain y where I: \forall x. y \in f(Z_+) \leq x
  using Int_ZF_2_L1A Int_ZF_1_1_L5 by auto
\{ fix n assume A5: n \in Z_+ \
  have Z_+ \subseteq Z using PositiveSet_def by auto \
  with A3 I T A5 have l + c \leq f(n) + c \
    using func_imagedef int_ord_transl_inv by auto \
  with I T have l + c \leq f(n) + c \
    using int_ord_transl_inv by simp \
  with A4 T have K \leq f(n) + c \
    using Int_ZF_1_2_L3 by simp \
  \} then have \forall n \in Z_+. K \leq f(n) + c by simp 
with T show thesis by auto
qed

If a function has infinite limit, then we can add such constant such that
minimum of those arguments for which the function (plus the constant) is
larger than another given constant is greater than a third constant. It is not
as complicated as it sounds.

lemma (in int0) Int_ZF_1_6_L7:
  assumes A1: f: Z_+ \rightarrow Z and A2: K \in Z_+ N \in Z
  and A3: \forall a \in Z_+. \exists b \in Z_+. \forall x. b \leq x \longrightarrow a \leq f(x)
  shows \exists C \in Z. N \leq \text{Minimum}(IntegerOrder,\{n \in Z_+. K \leq f(n)+C\})
proof -
  from A1 A2 have \exists C \in Z. \forall n \in Z_+. K \leq f(n) + C \longrightarrow N \leq n
    using Int_ZF_1_5_L4 by simp 
then obtain C where I: C \in Z and
    II: \forall n \in Z_+. K \leq f(n) + C \longrightarrow N \leq n 
    by auto
have antisym(IntegerOrder) using Int_ZF_2_L4 by simp 
moreover have HasAminimum(IntegerOrder,\{n \in Z_+. K \leq f(n)+C\}) 
proof -
  from A2 A3 I have \exists n \in Z_+. \forall x. n \leq x \longrightarrow K - C \leq f(x) 
    using Int_ZF_1_1_L5 by simp 
then obtain n where
    n \in Z_+ and \forall x. n \leq x \longrightarrow K - C \leq f(x) 
    by auto 
with A2 I have \{n \in Z_+. K \leq f(n)+C\} \neq \emptyset
    \{n \in Z_+. K \leq f(n)+C\} \subseteq Z_+
using int_ord_is_refl refl_def PositiveSet_def Int_ZF_2_L9C by auto
then show HasAminimum(IntegerOrder,\{n∈\mathbb{Z}_+. K ≤ f(n)+C\})
using Int_ZF_1_5_L1C by simp
qed
moreover from II have \∀ n ∈ \{n∈\mathbb{Z}_+. K ≤ f(n)+C\}. (N,n) ∈ IntegerOrder
by auto
ultimately have \langle N,Minimum(IntegerOrder,\{n∈\mathbb{Z}_+. K ≤ f(n)+C\}) \rangle ∈ IntegerOrder
by (rule Order_ZF_4_L12)
with I show thesis by auto
qed

For any integer \( m \) the function \( k \mapsto m \cdot k \) has an infinite limit (or negative of that). This is why we put some properties of these functions here, even though they properly belong to a (yet nonexistent) section on homomorphisms. The next lemma shows that the set \( \{a \cdot x : x ∈ \mathbb{Z}\} \) can finite only if \( a = 0 \).

lemma (in int0) Int_ZF_1_6_L8:
  assumes A1: \( a ∈ \mathbb{Z} \) and A2: \{a \cdot x. x ∈ \mathbb{Z}\} ∈ Fin(\mathbb{Z})
  shows a = 0
proof -
  from A1 have a=0 ∨ (a ≤ -1) ∨ (1≤a)
    using Int_ZF_1_3_L6C by simp
  moreover
  { assume a ≤ -1
    then have \{a \cdot x. x∈\mathbb{Z}\} /∈ Fin(\mathbb{Z})
      using int_zero_not_one Int_ZF_1_3_T1 ring1.OrdRing_ZF_3_L6
    by simp
    with A2 have False by simp }
  moreover
  { assume 1≤a
    then have \{a \cdot x. x∈\mathbb{Z}\} /∈ Fin(\mathbb{Z})
      using int_zero_not_one Int_ZF_1_3_T1 ring1.OrdRing_ZF_3_L5
    by simp
    with A2 have False by simp }
  ultimately show a = 0 by auto
qed

54.7 Miscellaneous

In this section we put some technical lemmas needed in various other places that are hard to classify.

Suppose we have an integer expression (a meta-function) \( F \) such that \( F(p)|p| \) is bounded by a linear function of \( |p| \), that is for some integers \( A, B \) we have \( F(p)|p| ≤ A|p| + B \). We show that \( F \) is then bounded. The proof is easy, we
just divide both sides by \(|p|\) and take the limit (just kidding).

**Lemma (in int0) Int_ZF_1_7_L1:**

**Assumes:**

- \( \forall q \in \mathbb{Z}. \ F(q) \in \mathbb{Z} \) and
- \( \forall q \in \mathbb{Z}. \ F(q) \cdot \abs{q} \leq A \cdot \abs{q} + B \) and
- \( A \in \mathbb{Z}, \ B \in \mathbb{Z} \)

**Shows:** \( \exists L. \ \forall p \in \mathbb{Z}. \ F(p) \leq L \)

**Proof:**

- Let \( I = (-\abs{B})..\abs{B} \)
- Let \( K = \{ F(q). \ q \in I \} \)
- Let \( M = \text{Maximum}(\text{IntegerOrder}, K) \)
- Let \( L = \text{GreaterOf}(\text{IntegerOrder}, M, A+1) \)

From A3 A1 have C1: IsBounded(I, IntegerOrder)

- \( I \neq 0 \)
- \( \forall q \in \mathbb{Z}. \ F(q) \in \mathbb{Z} \)
- \( K = \{ F(q). \ q \in I \} \)

Using Order_ZF_3_L11 Int_ZF_1_3_L17 by auto

Then have \( M \in \mathbb{Z} \) by (rule Int_ZF_1_4_L1)

From C1 have T2: \( \forall q \in I. \ F(q) \leq M \)

By (rule Int_ZF_1_4_L1)

- \{ fix p assume A4: p \in \mathbb{Z} have \( F(p) \leq L \)

Proof -

- \{ assume abs(p) \leq abs(B) with A3 T1 T2 have \( F(p) \leq M \) M \leq L using Int_ZF_2_L14 Int_ZF_1_1_L5 by auto

Then have \( F(p) \leq L \) by (rule Int_order_transitive) \}

Moreover

- \{ assume A5: \( \neg(\abs{p} \leq \abs{B}) \)

From A3 A2 A4 have

- \( A \cdot \abs{p} \in \mathbb{Z} \)
- \( F(p) \cdot \abs{p} \leq A \cdot \abs{p} + B \)

Using Int_ZF_2_L14 Int_ZF_1_1_L5 by auto

Moreover from A3 A4 A5 have B \leq abs(p)

Using Int_ZF_2_L15 by simp

Ultimately have

- \( F(p) \cdot \abs{p} \leq A \cdot \abs{p} + \abs{p} \)

Using Int_ZF_2_L15A by blast

With A3 A4 have \( F(p) \cdot \abs{p} \leq (A+1) \cdot \abs{p} \)

Using Int_ZF_2_L14 Int_ZF_1_2_L7 by simp

Moreover from A3 A1 A4 A5 have

- \( F(p) \in \mathbb{Z} \)
- \( A+1 \in \mathbb{Z} \)
- \( \abs{p} \in \mathbb{Z} \)
- \( \neg(\abs{p} \leq 0) \)

Using int_zero_one_are_int Int_ZF_1_1_L5 Int_ZF_2_L14 Int_ZF_1_3_L11 by auto

Ultimately have \( F(p) \leq A+1 \)

Using Int_ineq_simpl_positive by simp

Moreover from T1 have \( A+1 \leq L \) by simp

674
ultimately have $F(p) \leq L$ by (rule Int_order_transitive) 
ultimately show thesis by blast
qed 
} then have $\forall p \in \mathbb{Z}$. $F(p) \leq L$ by simp
thus thesis by auto
qed

A lemma about splitting (not really, there is some overlap) the $\mathbb{Z} \times \mathbb{Z}$ into six subsets (cases). The subsets are as follows: first and third quadrant, and second and fourth quadrant farther split by the $b = -a$ line.

lemma (in int0) int_plane_split_in6: assumes $a \in \mathbb{Z}$ $b \in \mathbb{Z}$
shows $0 \leq a \land 0 \leq b \lor a \leq 0 \land b \leq 0 \lor a \leq 0 \land 0 \leq b \land a + b \leq 0 \lor 0 \leq a \land b \geq 0 \land a + b \leq 0$
using assms Int_ZF_2_T1 group3.OrdGroup_6cases by simp

55 Division on integers

theory IntDiv_ZF_IML imports Int_ZF_1 ZF.IntDiv
begin

This theory translates some results form the Isabelle’s IntDiv.thy theory to the notation used by IsarMathLib.

55.1 Quotient and reminder

For any integers $m,n$, $n > 0$ there are unique integers $q,p$ such that $0 \leq p < n$ and $m = n \cdot q + p$. Number $p$ in this decomposition is usually called $m$ mod $n$. Standard Isabelle denotes numbers $q,p$ as $m \text{ zdiv } n$ and $m \text{ zmod } n$, resp., and we will use the same notation.

The next lemma is sometimes called the "quotient-reminder theorem".

lemma (in int0) IntDiv_ZF_1_L1: assumes $m \in \mathbb{Z}$ $n \in \mathbb{Z}$
shows $m = n \cdot (m \text{ zdiv } n) + (m \text{ zmod } n)$
using assms Int_ZF_1_L2 raw_zmod_zdiv_equality
by simp

If $n$ is greater than 0 then $m \text{ zmod } n$ is between 0 and $n - 1$.

lemma (in int0) IntDiv_ZF_1_L2: assumes $A1: m \in \mathbb{Z}$ and $A2: 0 \leq n \land n \neq 0$
shows $0 \leq m \text{ zmod } n$
$m \text{ zmod } n \leq n \land m \text{ zmod } n \neq n$
\( m \mod n \leq n-1 \)

**proof -**

from \( A2 \) have \( T: n \in \mathbb{Z} \)
using Int_ZF_2_L1A by simp

from \( A2 \) have \( \#0 \leq n \) using Int_ZF_2_L9 IntZF_1_L8
by auto

with \( T \) show
\( 0 \leq m \mod n \)
\( m \mod n \leq n \)
\( m \mod n \neq n \)
using pos_mod Int_ZF_1_L8 Int_ZF_1_L8A zmod_type
Int_ZF_2_L1 Int_ZF_2_L9AA
by auto

then show \( m \mod n \leq n-1 \)
using Int_ZF_4_L1B by auto

qed

\((m \cdot k) \div k = m.\)

**lemma** (in int0) IntDiv_ZF_1_L3:

assumes \( m \in \mathbb{Z}, k \in \mathbb{Z} \) and \( k \neq 0 \)

shows
\( (m \cdot k) \div k = m \)
\( (k \cdot m) \div k = m \)
using assms zdiv_zmult_self1 zdiv_zmult_self2
Int_ZF_1_L8 Int_ZF_1_L2
by auto

The next lemma essentially translates \texttt{zdiv_mono1} from standard Isabelle to our notation.

**lemma** (in int0) IntDiv_ZF_1_L4:

assumes \( A1: m \leq k \) and \( A2: 0 \leq n \neq 0 \)

shows \( m \div n \leq k \div n \)

**proof -**

from \( A2 \) have \( \#0 \leq n \) \( \#0 \neq n \)
using Int_ZF_1_L8 by auto

with \( A1 \) have
\( m \div n \leq k \div n \)
\( m \div n \in \mathbb{Z} \)
\( m \div k \in \mathbb{Z} \)
using Int_ZF_2_L1A Int_ZF_2_L9 zdiv_mono1
by auto

then show \( (m \div n) \leq (k \div n) \)
using Int_ZF_2_L1 by simp

qed

A quotient-reminder theorem about integers greater than a given product.

**lemma** (in int0) IntDiv_ZF_1_L5:

assumes \( A1: n \in \mathbb{Z}_+ \) and \( A2: n \leq k \) and \( A3: k \cdot n \leq m \)

shows
\( m = n \cdot (m \div n) \)
\( m = (m \mod n) \cdot n + (m \mod n) \)
\[(m \mod n) \in 0 \ldots (n-1)\]
\[k \leq (m \div n)\]
\[m \div n \in \mathbb{Z}_+\]

**proof**
- from A2 A3 have T:
  \[m \in \mathbb{Z} \quad n \in \mathbb{Z} \quad k \in \mathbb{Z} \quad m \div n \in \mathbb{Z}\]
  using Int_ZF_2_L1A by auto
then show \(m = n \cdot (m \div n) + (m \mod n)\)
  using IntDiv_ZF_1_L1 by simp
with T show \(m = (m \div n) \cdot n + (m \mod n)\)
  using Int_ZF_1_L4 by simp
from A1 have I: \(0 \leq n \quad n \neq 0\)
  using PositiveSet_def by auto
with T show \((m \mod n) \in 0 \ldots (n-1)\)
  using IntDiv_ZF_1_L2 Order_ZF_2_L1 by simp
from A3 I have \((k \cdot n \div n) \leq (m \div n)\)
  using IntDiv_ZF_1_L4 by simp
with I T show \(k \leq (m \div n)\)
  using IntDiv_ZF_1_L3 by simp
with A1 A2 show \(m \div n \in \mathbb{Z}_+\)
  using Int_ZF_1_5_L7 by blast

qed

end

56 Integers 2

theory Int_ZF_2 imports func_ZF_1 Int_ZF_1 IntDiv_ZF_IML Group_ZF_3

begin

In this theory file we consider the properties of integers that are needed for
the real numbers construction in Real_ZF series.

56.1 Slopes

In this section we study basic properties of slopes - the integer almost homo-
omorphisms. The general definition of an almost homomorphism \(f\) on a group
\(G\) written in additive notation requires the set \(\{f(m + n) - f(m) - f(n) : m, n \in G\}\) to be finite. In this section we establish a definition that is equiva-
 lent for integers: that for all integer \(m, n\) we have \(|f(m+n) - f(m) - f(n)| \leq L\)
for some \(L\).

First we extend the standard notation for integers with notation related to
slopes. We define slopes as almost homomorphisms on the additive group
of integers. The set of slopes is denoted \(S\). We also define "positive" slopes
as those that take infinite number of positive values on positive integers.
We write $\delta(s,m,n)$ to denote the homomorphism difference of $s$ at $m,n$ (i.e. the expression $s(m+n) - s(m) - s(n)$). We denote $\max\delta(s)$ the maximum absolute value of homomorphism difference of $s$ as $m,n$ range over integers.
If $s$ is a slope, then the set of homomorphism differences is finite and this maximum exists. In Group\_ZF\_3 we define the equivalence relation on almost homomorphisms using the notion of a quotient group relation and use "$\sim$" to denote it. As here this symbol seems to be hogged by the standard Isabelle, we will use "$\approx$" instead "$\sim$". We show in this section that $s \sim r$ iff for some $L$ we have $|s(m) - r(m)| \leq L$ for all integer $m$. The "$\circ$" denotes the first operation on almost homomorphisms. For slopes this is addition of functions defined in the natural way. The "$\circ$" symbol denotes the second operation on almost homomorphisms (see Group\_ZF\_3 for definition), defined for the group of integers. In short "$\circ$" is the composition of slopes. The "$\cdot$" symbol acts as an infix operator that assigns the value $\min\{n \in \mathbb{Z}_+: p \leq f(n)\}$ to a pair (of sets) $f$ and $p$. In application $f$ represents a function defined on $\mathbb{Z}_+$ and $p$ is a positive integer. We choose this notation because we use it to construct the right inverse in the ring of classes of slopes and show that this ring is in fact a field. To study the homomorphism difference of the function defined by $p \mapsto f^{-1}(p)$ we introduce the symbol $\varepsilon$ defined as $\varepsilon(f,\langle m,n \rangle) = f^{-1}(m + n) - f^{-1}(m) - f^{-1}(n)$. Of course the intention is to use the fact that $\varepsilon(f,\langle m,n \rangle)$ is the homomorphism difference of the function $g$ defined as $g(m) = f^{-1}(m)$. We also define $\gamma(s,m,n)$ as the expression $\delta(f,m,-n) + s(0) - \delta(f,n,-n)$. This is useful because of the identity $f(m-n) = \gamma(m,n) + f(m) - f(n)$ that allows to obtain bounds on the value of a slope at the difference of of two integers. For every integer $m$ we introduce notation $mS$ defined by $mE(n) = m \cdot n$. The mapping $q \mapsto qS$ embeds integers into $S$ preserving the order, (that is, maps positive integers into $S_+$).

locale int1 = int0 +

fixes slopes ($S$ )
defines slopes_def[simp]: $S \equiv \text{AlmostHoms(\mathbb{Z},\text{IntegerAddition})}$

fixes posslopes ($S_+$)
defines posslopes_def[simp]: $S_+ \equiv \{s \in S. s(\mathbb{Z}_+) \cap \mathbb{Z}_+ \notin \text{Fin(\mathbb{Z})}\}$

fixes $\delta$
defines $\delta$ _def[simp]: $\delta(s,m,n) \equiv s(m+n)-s(m)-s(n)$

fixes maxhomdiff ($\max\delta$ )
defines maxhomdiff_def[simp]: $\max\delta(s) \equiv \text{Maximum(IntegerOrder,}\{\abs(\delta(s,m,n)). (m,n) \in \mathbb{Z}\times\mathbb{Z}\})$

fixes AlEqRel
defines AlEqRel_def[simp]:
AlEqRel ≡ QuotientGroupRel(S, AlHomOp1(Z, IntegerAddition), FinRangeFunctions(Z, Z))

fixes AlEq (infix ~ 68)
defines AlEq_def[simp]: s ~ r ≡ (s, r) ∈ AlEqRel

fixes slope_add (infix + 70)
defines slope_add_def[simp]: s + r ≡ AlHomOp1(Z, IntegerAddition)(s, r)

fixes slope_comp (infix o 70)
defines slope_comp_def[simp]: s o r ≡ AlHomOp2(Z, IntegerAddition)(s, r)

fixes neg (~ [90] 91)
defines neg_def[simp]: -s ≡ GroupInv(Z, IntegerAddition) O s

fixes slope_inv (infix − 71)
defines slope_inv_def[simp]: f −1(p) ≡ Minimum(IntegerOrder, {n∈Z+. p ≤ f(n)})

fixes ε
defines ε_def[simp]:
ε(f, p) ≡ f −1(fst(p) + snd(p)) − f −1(fst(p)) − f −1(snd(p))

fixes γ
defines γ_def[simp]:
γ(s, m, n) ≡ δ(s, m, −n) − δ(s, n, −n) + s(0)

fixes intembed (_S)
defines intembed_def[simp]: m^S ≡ {(n, m·n). n∈Z}

We can use theorems proven in the group1 context.

lemma (in int1) Int_ZF_2_1_L1: shows group1(Z, IntegerAddition)
    using Int_ZF_1_T2 group1_axioms.intro group1_def by simp

Type information related to the homomorphism difference expression.

lemma (in int1) Int_ZF_2_1_L2: assumes f∈S and n∈Z m∈Z
    shows m+n ∈ Z
    f(m+n) ∈ Z
    f(m) ∈ Z  f(n) ∈ Z
    f(m) + f(n) ∈ Z
    HomDiff(Z, IntegerAddition, f, (m, n)) ∈ Z
    using assms Int_ZF_2_1_L1 group1.Group_ZF_3_2_L4A
    by auto

Type information related to the homomorphism difference expression.

lemma (in int1) Int_ZF_2_1_L2A:
    assumes f:Z→Z and n∈Z m∈Z
    shows
\[ m+n \in \mathbb{Z} \]
\[ f(m+n) \in \mathbb{Z} \quad f(m) \in \mathbb{Z} \quad f(n) \in \mathbb{Z} \]
\[ f(m) + f(n) \in \mathbb{Z} \]
\[ \text{HomDiff}(\mathbb{Z}, \text{IntegerAddition}, f, (m, n)) \in \mathbb{Z} \]
\[ \text{using asms Int_ZF_2_1_L1 group1.Group_ZF_3_2_L4 by auto} \]

Slopes map integers into integers.

**lemma (in int1) Int_ZF_2_1_L2B:**

assumes A1: \( f \in S \) and A2: \( m \in \mathbb{Z} \)

shows \( f(m) \in \mathbb{Z} \)

**proof**

- from A1 have \( f : \mathbb{Z} \rightarrow \mathbb{Z} \) using AlmostHoms_def by simp
- with A2 show \( f(m) \in \mathbb{Z} \) using apply_funtype by simp

**qed**

The homomorphism difference in multiplicative notation is defined as the expression \( s(m \cdot n) \cdot (s(m) \cdot s(n))^{-1} \). The next lemma shows that in the additive notation used for integers the homomorphism difference is \( f(m+n) - f(m) - f(n) \) which we denote as \( \delta(f,m,n) \).

**lemma (in int1) Int_ZF_2_1_L3:**

assumes \( f : \mathbb{Z} \rightarrow \mathbb{Z} \) and \( m \in \mathbb{Z} \quad n \in \mathbb{Z} \)

shows \( \text{HomDiff}(\mathbb{Z}, \text{IntegerAddition}, f, (m, n)) = \delta(f,m,n) \)

**proof**

- from A1 A2 have \( f(m) \in \mathbb{Z} \quad f(n) \in \mathbb{Z} \quad \delta(f,m,n) \in \mathbb{Z} \) and
- \( \text{HomDiff}(\mathbb{Z}, \text{IntegerAddition}, f, (m, n)) = \delta(f,m,n) \)
- using Int_ZF_2_1_L2 AlmostHoms_def Int_ZF_2_1_L3 by auto

with A1 A2 show \( f(m+n) = f(m) + (f(n) + \delta(f,m,n)) \)

using Int_ZF_2_1_L3 Int_ZF_1_L3 Int_ZF_2_1_L1 group1.Group_ZF_3_4_L1 by simp

**qed**

The next formula restates the definition of the homomorphism difference to express the value an almost homomorphism on a sum.

**lemma (in int1) Int_ZF_2_1_L3A:**

assumes A1: \( f \in S \) and A2: \( m \in \mathbb{Z} \quad n \in \mathbb{Z} \)

shows \( f(m+n) = f(m)+(f(n)+\delta(f,m,n)) \)

**proof**

- from A1 A2 have \( f(m) \in \mathbb{Z} \quad f(n) \in \mathbb{Z} \quad \delta(f,m,n) \in \mathbb{Z} \) and
- \( \text{HomDiff}(\mathbb{Z}, \text{IntegerAddition}, f, (m, n)) = \delta(f,m,n) \)
- using Int_ZF_2_1_L2 AlmostHoms_def Int_ZF_2_1_L3 by auto

with A1 A2 show \( f(m+n) = f(m)+(f(n)+\delta(f,m,n)) \)

using Int_ZF_2_1_L3 Int_ZF_1_L3 Int_ZF_2_1_L1 group1.Group_ZF_3_4_L1 by simp

**qed**

The homomorphism difference of any integer function is integer.

**lemma (in int1) Int_ZF_2_1_L3B:**

assumes \( f : \mathbb{Z} \rightarrow \mathbb{Z} \) and \( m \in \mathbb{Z} \quad n \in \mathbb{Z} \)

shows \( \delta(f,m,n) \in \mathbb{Z} \)

**proof**

- using asms Int_ZF_2_1_L2A Int_ZF_2_1_L3 by simp

680
The value of an integer function at a sum expressed in terms of $\delta$.

**Lemma (in int1)** IntZF_2_1_L3C: assumes $A1: f:\mathbb{Z} \rightarrow \mathbb{Z}$ and $A2: m \in \mathbb{Z}, n \in \mathbb{Z}$

shows $f(m+n) = \delta(f,m,n) + f(n) + f(m)$

**Proof** -
- from $A1 A2$ have $T$:
  - $\delta(f,m,n) \in \mathbb{Z}$
  - $f(m+n) \in \mathbb{Z}$
  - $f(m) \in \mathbb{Z}$
  - $f(n) \in \mathbb{Z}$
  - using IntZF_1_1_L5 apply_funtype by auto

then show $f(m+n) = \delta(f,m,n) + f(n) + f(m)$
  - using IntZF_1_2_L15 by simp

qed

The next lemma presents two ways the set of homomorphism differences can be written.

**Lemma (in int1)** IntZF_2_1_L4: assumes $A1: f:\mathbb{Z} \rightarrow \mathbb{Z}$

shows $\{ \text{abs}(\text{HomDiff}(\mathbb{Z}, \text{IntegerAddition}, f, x)). \ x \in \mathbb{Z} \times \mathbb{Z} \} = \{ \text{abs}(\delta(f,m,n)). \ (m, n) \in \mathbb{Z} \times \mathbb{Z} \}$

**Proof** -
- from $A1$ have $\forall m \in \mathbb{Z}. \forall n \in \mathbb{Z}$.
  - $\text{abs}(\text{HomDiff}(\mathbb{Z}, \text{IntegerAddition}, f, (m, n))) = \text{abs}(\delta(f,m,n))$
  - using IntZF_2_1_L3 by simp

then show thesis by (rule ZF1_1_L4A)

qed

If $f$ maps integers into integers and for all $m, n \in \mathbb{Z}$ we have $|f(m + n) - f(m) - f(n)| \leq L$ for some $L$, then $f$ is a slope.

**Lemma (in int1)** IntZF_2_1_L5: assumes $A1: f:\mathbb{Z} \rightarrow \mathbb{Z}$ and $A2: \forall m \in \mathbb{Z}. \forall n \in \mathbb{Z}. \ |\delta(f,m,n)| \leq L$

shows $f \in S$

**Proof** -
- let $\text{Abs} = \text{AbsoluteValue}(\mathbb{Z}, \text{IntegerAddition}, \text{IntegerOrder})$
  - have $\text{group3}(\mathbb{Z}, \text{IntegerAddition}, \text{IntegerOrder})$
  - $\text{IntegerOrder} \{\text{is total on}\} \mathbb{Z}$
  - using IntZF_2_1_T1 by auto

moreover from $A1 A2$ have
  - $\forall x \in \mathbb{Z} \times \mathbb{Z}. \ \text{HomDiff}(\mathbb{Z}, \text{IntegerAddition}, f, x) \in \mathbb{Z}$
  - $(\text{Abs}(\text{HomDiff}(\mathbb{Z}, \text{IntegerAddition}, f, x)), L) \in \text{IntegerOrder}$
  - using IntZF_2_1_L2A IntZF_2_1_L3 by auto

ultimately have $\text{IsBounded}((\{\text{HomDiff}(\mathbb{Z}, \text{IntegerAddition}, f, x). \ x \in \mathbb{Z} \times \mathbb{Z}\}, \text{IntegerOrder}$)
  - by (rule group3.OrderedGroup_ZF_3_L9A)

with $A1$ show $f \in S$ using Int_bounded_iff_fin AlmostHoms_def by simp

qed

The absolute value of homomorphism difference of a slope $s$ does not exceed $\max \delta(s)$.

**Lemma (in int1)** IntZF_2_1_L7:
- assumes $A1: s \in S$ and $A2: n \in \mathbb{Z}, m \in \mathbb{Z}$
shows
abs(δ(s,m,n)) ≤ maxΔ(s)
δ(s,m,n) ∈ \mathbb{Z} \quad \text{maxΔ(s)} ∈ \mathbb{Z}
(-maxΔ(s)) ≤ δ(s,m,n)

proof -
from A1 A2 show T: δ(s,m,n) ∈ \mathbb{Z}
using Int_ZF_2_1_L2 Int_ZF_1_1_L5 by simp

moreover have A ∈ Fin(\mathbb{Z})
proof -
have ∀k∈\mathbb{Z}. abs(k) ∈ \mathbb{Z}
using Int_ZF_2_L14 by simp
moreover from A1 have
\{HomDiff(\mathbb{Z},IntegerAddition,s,x). x∈\mathbb{Z}×\mathbb{Z}\} ∈ Fin(\mathbb{Z})

ultimately show A ∈ Fin(\mathbb{Z}) by (rule Finite1_L6C)

qed

moreover have \text{A} \neq 0 by auto
ultimately have ∀k∈\text{A}. \langle k,\text{Maximum}(\text{IntegerOrder},A) \rangle ∈ \text{IntegerOrder}
by (rule Finite_ZF_1_T2)

moreover from A1 A2 have d∈A using AlmostHoms_def Int_ZF_2_1_L4
by auto

ultimately have d ≤ \text{Maximum}(\text{IntegerOrder},A) by auto
with A1 show d ≤ maxΔ(s) maxΔ(s) ∈ \mathbb{Z}

by auto

ultimately have d ≤ \text{Maximum}(\text{IntegerOrder},A) by auto
with T show (-maxΔ(s)) ≤ δ(s,m,n)
using Int_ZF_1_3_L19 by simp

qed

A useful estimate for the value of a slope at 0, plus some type information
for slopes.

lemma (in int1) Int_ZF_2_1_L8: assumes A1: s∈S
shows
abs(s(0)) ≤ maxΔ(s)
0 ≤ maxΔ(s)
abs(s(0)) ∈ \mathbb{Z} \quad \text{maxΔ(s)} ∈ \mathbb{Z}
abs(s(0)) + maxΔ(s) ∈ \mathbb{Z}

proof -
from A1 have s(0) ∈ \mathbb{Z}
using int_zero_one_are_int Int_ZF_2_1_L2B by simp
then have I: 0 ≤ abs(s(0))
and abs(δ(s,0,0)) = abs(s(0))
using Int_abs_nonneg int_zero_one_are_int Int_ZF_1_1_L4
Int_ZF_2_L17 by auto
moreover from A1 have abs(δ(s,0,0)) ≤ maxΔ(s)

682
ultimately show \( II: \text{abs}(s(0)) \leq \text{max}\delta(s) \)
by simp
with I show \( 0 \leq \text{max}\delta(s) \)
by (rule Int_order_transitive)
with II show \( \text{max}\delta(s) \in \mathbb{Z} \)
by simp
abs(s(0)) + \text{max}\delta(s) \in \mathbb{Z}
using Int_ZF_2_L1A Int_ZF_1_1_L5
by auto
qed

Int Group_ZF_3.thy we show that finite range functions valued in an abelian group form a normal subgroup of almost homomorphisms. This allows to define the equivalence relation between almost homomorphisms as the relation resulting from dividing by that normal subgroup. Then we show in Group_ZF_3_4_L12 that if the difference of \( f \) and \( g \) has finite range (actually \( f(n) \cdot g(n)^{-1} \) as we use multiplicative notation in Group_ZF_3.thy), then \( f \) and \( g \) are equivalent. The next lemma translates that fact into the notation used in int1 context.

lemma (in int1) Int_ZF_2_1_L9: assumes A1: \( s \in S \)
and A2: \( \forall m \in S. \text{abs}(s(m)-r(m)) \leq L \)
shows \( s \sim r \)
proof -
from A1 A2 have \( \forall m \in S. s(m)-r(m) \in \mathbb{Z} \land \text{abs}(s(m)-r(m)) \leq L \)
using Int_ZF_2_1_L2B Int_ZF_1_1_L5
by simp
then have \( \text{IsBounded}({(s(n)-r(n). n \in \mathbb{Z})}, \text{IntegerOrder}) \)
by (rule Int_ZF_1_3_L20)
with A1 show \( s \sim r \)
using Int_bounded_iff_fin
Int_ZF_2_1_L1 group1.Group_ZF_3_4_L12
by simp
qed

A necessary condition for two slopes to be almost equal. For slopes the definition postulates the set \( \{f(m) - g(m) : m \in \mathbb{Z} \} \) to be finite. This lemma shows that this implies that \( |f(m) - g(m)| \) is bounded (by some integer) as \( m \) varies over integers. We also mention here that in this context \( s \sim r \) implies that both \( s \) and \( r \) are slopes.

lemma (in int1) Int_ZF_2_1_L9A: assumes \( s \sim r \)
shows \( \exists L \in \mathbb{Z}. \forall m \in \mathbb{Z}. \text{abs}(s(m)-r(m)) \leq L \)
\( s \in S \land r \in S \)
using assms Int_ZF_2_1_L1 group1.Group_ZF_3_4_L11
Int_ZF_1_3_L20AA QuotientGroupRel_def
by auto

Let’s recall that the relation of almost equality is an equivalence relation on the set of slopes.

lemma (in int1) Int_ZF_2_1_L9B: shows
Another version of sufficient condition for two slopes to be almost equal: if
the difference of two slopes is a finite range function, then they are almost
equal.

Lemma (in int1) Int_ZF_2_1_L9C: assumes $s \in S$ $r \in S$ and
$s + (-r) \in \text{FinRangeFunctions}(\mathbb{Z}, \mathbb{Z})$
shows $s \sim r$
$r \sim s$
using assms Int_ZF_2_1_L1
group1.Group_ZF_3_2_L13 group1.Group_ZF_3_4_L12A
by auto

If two slopes are almost equal, then the difference has finite range. This is
the inverse of Int_ZF_2_1_L9C.

Lemma (in int1) Int_ZF_2_1_L9D: assumes $A1: s \sim r$
shows $s + (-r) \in \text{FinRangeFunctions}(\mathbb{Z}, \mathbb{Z})$
proof -
let $G = \mathbb{Z}$
let $f = \text{IntegerAddition}$
from $A1$ have $\text{AlHomOp1}(G, f)\langle s, \text{GroupInv}(\text{AlmostHoms}(G, f), \text{AlHomOp1}(G,
f))(r) \rangle \in \text{FinRangeFunctions}(G, G)$
using Int_ZF_2_1_L1 Group_ZF_3_4_L12B by auto
with $A1$ show $s + (-r) \in \text{FinRangeFunctions}(\mathbb{Z}, \mathbb{Z})$
using Int_ZF_2_1_L9A Int_ZF_2_1_L1 Group_ZF_3_2_L13
by simp
qed

What is the value of a composition of slopes?

Lemma (in int1) Int_ZF_2_1_L10: assumes $s \in S$ $r \in S$ and $m \in \mathbb{Z}$
shows $(s \circ r)(m) = s(r(m))$ $s(r(m)) \in \mathbb{Z}$
using assms Int_ZF_2_1_L1 Group_ZF_3_4_L2 by auto

Composition of slopes is a slope.

Lemma (in int1) Int_ZF_2_1_L11: assumes $s \in S$ $r \in S$
shows $s \circ r \in S$
using assms Int_ZF_2_1_L1 Group_ZF_3_4_T1 by simp

Negative of a slope is a slope.

Lemma (in int1) Int_ZF_2_1_L12: assumes $s \in S$ shows $-s \in S$
using assms Int_ZF_1_T2 Int_ZF_2_1_L1 Group_ZF_3_2_L13
by simp
What is the value of a negative of a slope?

**Lemma (in int1) Int_ZF_2_1_L12A:**

assumes \( s \in S \) and \( m \in \mathbb{Z} \)

shows \((-s)(m) = -(s(m))\)

using assms Int_ZF_2_1_L1 group1.Group_ZF_3_2_L5
by simp

What are the values of a sum of slopes?

**Lemma (in int1) Int_ZF_2_1_L12B:**

assumes \( s \in S \) and \( r \in S \) and \( m \in \mathbb{Z} \)

shows \((s+r)(m) = s(m) + r(m)\)

using assms Int_ZF_2_1_L1 group1.Group_ZF_3_2_L12
by simp

Sum of slopes is a slope.

**Lemma (in int1) Int_ZF_2_1_L12C:**

assumes \( s \in S \) and \( r \in S \)

shows \( s+r \in S \)

using assms Int_ZF_2_1_L1 group1.Group_ZF_3_2_L16
by simp

A simple but useful identity.

**Lemma (in int1) Int_ZF_2_1_L13:**

assumes \( s \in S \) and \( n \in \mathbb{Z} \)

shows \( s(n \cdot m) + (s(m) + \delta(s,n \cdot m,m)) = s((n+1) \cdot m)\)

using assms Int_ZF_1_1_L5 Int_ZF_2_1_L2B Int_ZF_1_2_L9 Int_ZF_1_2_L7
by simp

Some estimates for the absolute value of a slope at the opposite integer.

**Lemma (in int1) Int_ZF_2_1_L14:**

assumes A1: \( s \in S \) and A2: \( m \in \mathbb{Z} \)

shows

\[
\begin{align*}
abs(s(0) - \delta(s,m,-m) - s(m)) & \leq 2 \cdot \max \delta(s) \\
abs(s(0) - s(m)) & \leq 2 \cdot \max \delta(s) + \abs(s(m)) \\
s(-m) & \leq \abs(s(0)) + \max \delta(s) - s(m)
\end{align*}
\]

**Proof**

- from A1 A2 have T:

\[
(-m) \in \mathbb{Z} \quad abs(s(m)) \in \mathbb{Z} \quad s(0) \in \mathbb{Z} \quad abs(s(0)) \in \mathbb{Z} \\
\delta(s,m,-m) \in \mathbb{Z} \quad s(m) \in \mathbb{Z} \quad s(-m) \in \mathbb{Z} \\
(\delta(s(m))) \in \mathbb{Z} \quad s(0) - \delta(s,m,-m) \in \mathbb{Z}
\]

using Int_ZF_1_1_L4 Int_ZF_2_1_L2B Int_ZF_2_1_L14 Int_ZF_2_1_L2
Int_ZF_1_1_L5 int_zero_one_are_int by auto

with A2 show I: \( s(-m) = s(0) - \delta(s,m,-m) - s(m) \)

using Int_ZF_1_1_L4 Int_ZF_1_2_L15 by simp

from T have \( abs(s(0) - \delta(s,m,-m)) \leq \abs(s(0)) + \abs(\delta(s,m,-m)) \)

using Int_triangle_ineq1 by simp

moreover from A1 A2 T have \( \abs(s(0)) + \abs(\delta(s,m,-m)) \leq 2 \cdot \max \delta(s) \)

using Int_ZF_2_1_L7 Int_ZF_2_1_L8 Int_ZF_1_3_L21 by simp

ultimately have \( \abs(s(0) - \delta(s,m,-m)) \leq 2 \cdot \max \delta(s) \)

by (rule Int_order_transitive)

moreover
from I have $s(m) + s(-m) = s(m) + (s(0) - \delta(s,m,-m) - s(m))$
   by simp
with T have $\text{abs}(s(m) + s(-m)) = \text{abs}(s(0) - \delta(s,m,-m))$
   using Int_ZF_1_2_L3 by simp
ultimately show $\text{abs}(s(m)+s(-m)) \leq 2 \cdot \text{max}\delta(s)$
   by simp
from I have $\text{abs}(s(-m)) = \text{abs}(s(0) - \delta(s,m,-m) - s(m))$
   by simp
with T have $\text{abs}(s(-m)) \leq \text{abs}(s(0)) + \text{abs}(\delta(s,m,-m)) + \text{abs}(s(m))$
   using Int_ZF_1_2_L3 by simp
moreover from A1 A2 T have
   $\text{abs}(s(0)) + \text{abs}(\delta(s,m,-m)) + \text{abs}(s(m)) \leq 2 \cdot \text{max}\delta(s) + \text{abs}(s(m))$
   using Int_ZF_2_1_L7 Int_ZF_2_1_L8 Int_ZF_1_3_L21 int_ord_transl_inv
by simp
ultimately show $\text{abs}(s(-m)) \leq 2 \cdot \text{max}\delta(s) + \text{abs}(s(m))$
   by (rule Int_order_transitive)
from T have $s(0) - \delta(s,m,-m) \leq \text{abs}(s(0)) + \text{abs}(\delta(s,m,-m))$
   using Int_ZF_2_1_L15E by simp
moreover from A1 A2 T have
   $\text{abs}(s(0)) + \text{abs}(\delta(s,m,-m)) \leq \text{abs}(s(0)) + \text{max}\delta(s)$
   using Int_ZF_2_1_L7 int_ord_transl_inv by simp
ultimately have $s(0) - \delta(s,m,-m) \leq \text{abs}(s(0)) + \text{max}\delta(s)$
   by (rule Int_order_transitive)
with T have $s(0) - \delta(s,m,-m) - s(m) \leq \text{abs}(s(0)) + \text{max}\delta(s) - s(m)$
   using int_ord_transl_inv by simp
with I show $s(-m) \leq \text{abs}(s(0)) + \text{max}\delta(s) - s(m)$
   by simp
qed

An identity that expresses the value of an integer function at the opposite
integer in terms of the value of that function at the integer, zero, and the
homomorphism difference. We have a similar identity in Int_ZF_2_1_L14,
but over there we assume that $f$ is a slope.

lemma (in int1) Int_ZF_2_1_L14A: assumes A1: $f: \mathbb{Z} \rightarrow \mathbb{Z}$ and A2: $m \in \mathbb{Z}$
   shows $f(-m) = (-\delta(f,m,-m)) + f(0) - f(m)$
proof -
  from A1 A2 have T:
    $f(-m) \in \mathbb{Z}$, $\delta(f,m,-m) \in \mathbb{Z}$, $f(0) \in \mathbb{Z}$, $f(m) \in \mathbb{Z}$
    using Int_ZF_1_1_L4 Int_ZF_1_1_L5 int_zero_one_are_int apply_funtype
    by auto
  with A2 show $f(-m) = (-\delta(f,m,-m)) + f(0) - f(m)$
    using Int_ZF_1_1_L4 Int_ZF_1_2_L15 by simp
qed

The next lemma allows to use the expression $\text{maxf}(f,0..M-1)$. Recall that $\text{maxf}(f,A)$ is the maximum of (function) $f$ on (the set) $A$.
lemma (in int1) Int_ZF_2_1_L15: 
asumes s ∈ S and M ∈ ℤ⁺ 
shows 
\[ \text{maxf}(s,0..(M-1)) ∈ ℤ \] 
\[ \forall n ∈ 0..(M-1). \ s(n) ≤ \text{maxf}(s,0..(M-1)) \] 
\[ \text{minf}(s,0..(M-1)) ∈ ℤ \] 
\[ \forall n ∈ 0..(M-1). \ \text{minf}(s,0..(M-1)) ≤ s(n) \] 
using assms AlmostHoms_def Int_ZF_1_5_L6 Int_ZF_1_4_L2 
by auto

A lower estimate for the value of a slope at \( nM + k \).

lemma (in int1) Int_ZF_2_1_L16: 
asumes A1: s ∈ S and A2: m ∈ ℤ and A3: M ∈ ℤ⁺ and A4: k ∈ 0..(M-1) 
shows 
\[ s(m·M) + (\text{minf}(s,0..(M-1)) - \text{maxδ}(s)) ≤ s(m·M+k) \] 
proof - 
from A3 have 0..(M-1) ⊆ ℤ 
using Int_ZF_1_5_L6 by simp
with A1 A2 A3 A4 have T: m·M ∈ ℤ 
k ∈ ℤ 
s(m·M) ∈ ℤ 
using PositiveSet_def Int_ZF_1_1_L5 Int_ZF_2_1_L2B 
by auto
with A1 A3 A4 have 
\[ s(m·M) + (\text{minf}(s,0..(M-1)) - \text{maxδ}(s)) ≤ s(m·M) + (s(k) + \delta(s,m·M,k)) \] 
using Int_ZF_2_1_L15 Int_ZF_2_1_L7 int_ineq_add_sides int_ord_transl_inv 
by simp
with A1 T show thesis using Int_ZF_2_1_L3A by simp
qed

Identity is a slope.

lemma (in int1) Int_ZF_2_1_L17: 
shows id(ℤ) ∈ S 
using Int_ZF_2_1_L1 group1.Group_ZF_3_4_L15 by simp

Simple identities about (absolute value of) homomorphism differences.

lemma (in int1) Int_ZF_2_1_L18: 
asumes A1: f:ℤ→ℤ and A2: m∈ℤ n∈ℤ 
shows 
\[ \|f(n) + f(m) - f(m+n)\| = \|\delta(f,m,n)\| \] 
\[ \|f(m) + f(n) - f(m+n)\| = \|\delta(f,m,n)\| \] 
\[ -(f(m)) - f(n) + f(m+n) = \delta(f,m,n) \] 
\[ -(f(n)) - f(m) + f(m+n) = \delta(f,m,n) \] 
\[ \|-(f(m+n)) + f(m) + f(n)\| = \|\delta(f,m,n)\| \] 
proof - 
from A1 A2 have T: 
f(m+n) ∈ ℤ 
f(m) ∈ ℤ 
f(n) ∈ ℤ 
f(m+n) - f(m) - f(n) ∈ ℤ 
(-f(m)) ∈ ℤ 
(-f(m+n)) + f(m) + f(n) ∈ ℤ 
using apply_funtype Int_ZF_1_1_L4 Int_ZF_1_1_L5 by auto
then have 
\[ \|-(f(m+n) - f(m) - f(n))\| = \|f(m+n) - f(m) - f(n)\| \] 
687
Some identities about the homomorphism difference of odd functions.

lemma (in int1) Int_ZF_2_1_L19:
assumes A1: f: \mathbb{Z} \rightarrow \mathbb{Z} and A2: \forall x \in \mathbb{Z}. \ (f(-x)) = (x)
and A3: m \in \mathbb{Z} n \in \mathbb{Z}
shows abs(\delta(f,-m,-n)) = abs(\delta(f,m,n))
proof -
from A1 A2 A3 show abs(\delta(f,-m,-n)) = abs(\delta(f,m,n))
using Int_ZF_1_2_L3 Int_ZF_2_1_L18 by auto
from A3 have T: m+n \in \mathbb{Z} using Int_ZF_1_1_L5 by simp
from A1 A2 have I: \forall x \in \mathbb{Z}. \ f(-x) = (-f(x))
using Int_ZF_1_5_L13 by simp
with A1 A2 A3 T show \delta(f,n,-(m+n)) = \delta(f,m,n)
\delta(f,m,-(m+n)) = \delta(f,m,n)
using Int_ZF_1_2_L3 Int_ZF_2_1_L18 by auto
from A3 have abs(\delta(f,-m,-n)) = abs(\delta(-f(m+n)) - f(-m) - f(-n))
using Int_ZF_1_1_L5 by simp 
also from A1 A2 A3 T I have ... = abs(δ(f,m,n))
using Int_ZF_2_1_L18 by simp 
finally show abs(δ(f,-m,-n)) = abs(δ(f,m,n)) by simp 
qed

Recall that \( f \) is a slope iff \( f(m+n) - f(m) - f(n) \) is bounded as \( m, n \) ranges over integers. The next lemma is the first step in showing that we only need to check this condition as \( m,n \) ranges over positive intergers. Namely we show that if the condition holds for positive integers, then it holds if one integer is positive and the second one is nonnegative.

lemma (in int1) Int_ZF_2_1_L20: assumes A1: f: \( \mathbb{Z} \rightarrow \mathbb{Z} \) and  A2: \( \forall a \in \mathbb{Z}_+. \ \forall b \in \mathbb{Z}_+. \ abs(\delta(f,a,b)) \leq L \) and  A3: \( m \in \mathbb{Z}_+ \ \ n \in \mathbb{Z}_+ \)  shows 0 \leq L \abs(\delta(f,m,n)) \leq L + \abs(f(0))  proof  -  from A1 A2 have \( \delta(f,1,1) \in \mathbb{Z} \ and \ abs(\delta(f,1,1)) \leq L \)  using int_one_two_are_pos PositiveSet_def Int_ZF_2_1_L3B  by auto  then show I: 0 \leq L using Int_ZF_1_3_L19 by simp  from A1 A3 have T:  \( n \in \mathbb{Z} \ f(n) \in \mathbb{Z} \ f(0) \in \mathbb{Z} \\
\delta(f,m,n) \in \mathbb{Z} \ abs(\delta(f,m,n)) \in \mathbb{Z} \)  using PositiveSet_def int_zero_one_are_int apply_funtype Nonnegative_def Int_ZF_2_1_L3B Int_ZF_2_L14 by auto  from A3 have m=0 \vee m\in\mathbb{Z}_+ using Int_ZF_1_5_L3A by auto  moreover  \{  assume m = 0  with T I have abs(\delta(f,m,n)) \leq L + \abs(f(0))  using Int_ZF_1_1_L4 Int_ZF_1_2_L3 Int_ZF_2_L17 int_ord_is_refl refl_def Int_ZF_2_L15F by simp  \}  moreover  \{  assume m\in\mathbb{Z}_+  with A2 A3 T have abs(\delta(f,m,n)) \leq L + \abs(f(0))  using int_abs_nonneg Int_ZF_2_L15F by simp  \}  ultimately show abs(\delta(f,m,n)) \leq L + \abs(f(0))  by auto  qed

If the slope condition holds for all pairs of integers such that one integer is positive and the second one is nonnegative, then it holds when both integers are nonnegative.

lemma (in int1) Int_ZF_2_1_L21: assumes A1: f: \( \mathbb{Z} \rightarrow \mathbb{Z} \) and A2: \( \forall a \in \mathbb{Z}_+. \ \forall b \in \mathbb{Z}_+. \ abs(\delta(f,a,b)) \leq L \) and  A3: \( n \in \mathbb{Z}_+ \ \ m \in \mathbb{Z}_+ \)
shows $\text{abs}(\delta(f,m,n)) \leq L + \text{abs}(f(0))$

proof -
from A1 A2 have

\[ \delta(f,1,1) \in \mathbb{Z} \quad \text{and} \quad \text{abs}(\delta(f,1,1)) \leq L \]
using int_one_two_are_pos PositiveSet_def Nonnegative_def Int_ZF_2_1_L3B
by auto
then have I: $0 \leq L$ using Int_ZF_1_3_L19 by simp
from A1 A3 have T:

\[ m \in \mathbb{Z} \quad f(m) \in \mathbb{Z} \quad f(0) \in \mathbb{Z} \quad (-f(0)) \in \mathbb{Z} \quad \delta(f,m,n) \in \mathbb{Z} \quad \text{abs}(\delta(f,m,n)) \leq L \]
using int_zero_one_are_int apply_funtype Nonnegative_def Int_ZF_2_1_L3B Int_ZF_2_L14 Int_ZF_1_1_L4
by auto
from A3 have n=0 ∨ n∈\mathbb{Z}_+ using Int_ZF_1_5_L3A by auto
moreover
\{ assume n=0

with T have $\delta(f,m,n) = -f(0)$
using Int_ZF_2_1_L14 by simp
with T have $\text{abs}(\delta(f,m,n)) = \text{abs}(f(0))$
using Int_ZF_2_L17 by simp
with T I have $\text{abs}(\delta(f,m,n)) \leq L + \text{abs}(f(0))$
using Int_ZF_2_L15F by simp
\}
moreover
\{ assume n∈\mathbb{Z}_+

with A2 A3 T have $\text{abs}(\delta(f,m,n)) \leq L + \text{abs}(f(0))$
using int_abs_nonneg Int_ZF_2_L15F by simp
\}
ultimately show $\text{abs}(\delta(f,m,n)) \leq L + \text{abs}(f(0))$
by auto
qed

If the homomorphism difference is bounded on $\mathbb{Z}_+ \times \mathbb{Z}_+$, then it is bounded on $\mathbb{Z}^+ \times \mathbb{Z}^+$.

lemma (in int1) Int_ZF_2_1_L22: assumes A1: $f: \mathbb{Z} \rightarrow \mathbb{Z}$ and
A2: $\forall a \in \mathbb{Z}_+. \forall b \in \mathbb{Z}_+. \text{abs}(\delta(f,a,b)) \leq L$
shows $\exists M. \forall m \in \mathbb{Z}^+. \forall n \in \mathbb{Z}^+. \text{abs}(\delta(f,m,n)) \leq M$
proof -
from A1 A2 have

$\forall m \in \mathbb{Z}^+. \forall n \in \mathbb{Z}^+. \text{abs}(\delta(f,m,n)) \leq L + \text{abs}(f(0)) + \text{abs}(f(0))$
using Int_ZF_2_1_L20 Int_ZF_2_1_L21 by simp
then show thesis by auto
qed

For odd functions we can do better than in Int_ZF_2_1_L22: if the homomorphism difference of $f$ is bounded on $\mathbb{Z}^+ \times \mathbb{Z}^+$, then it is bounded on $\mathbb{Z} \times \mathbb{Z}$, hence $f$ is a slope. Loong prof by splitting the $\mathbb{Z} \times \mathbb{Z}$ into six subsets.

lemma (in int1) Int_ZF_2_1_L23: assumes A1: $f: \mathbb{Z} \rightarrow \mathbb{Z}$ and
A2: $\forall a \in \mathbb{Z}_+. \forall b \in \mathbb{Z}_+. \text{abs}(\delta(f,a,b)) \leq L$
and A3: $\forall x \in \mathbb{Z}. (-f(-x)) = f(x)$
shows $f \in S$

proof -

from $A1$ $A2$ have

\[ \exists M. \forall a \in \mathbb{Z}^+. \forall b \in \mathbb{Z}^+. \abs(\delta(f,a,b)) \leq M \]

by (rule Int_ZF_2_1_L22)

then obtain $M$ where $I$: $\forall m \in \mathbb{Z}^+. \forall n \in \mathbb{Z}^+. \abs(\delta(f,m,n)) \leq M$

by auto

\{ fix $a$ $b$ assume $A4$: $a \in \mathbb{Z}$ $b \in \mathbb{Z}$

then have

$0 \leq a \wedge 0 \leq b \vee a \leq 0 \wedge b \leq 0 \vee$

$a \leq 0 \wedge 0 \leq b \wedge 0 \leq a + b \wedge a \leq 0 \wedge 0 \leq b \wedge a + b \leq 0 \vee$

$0 \leq a \wedge b \leq 0 \wedge 0 \leq a + b \vee 0 \leq a \wedge b \leq 0 \wedge a + b \leq 0$

using int_plane_split_in6 by simp

moreover

\{ assume $0 \leq a \wedge 0 \leq b$

then have $a \in \mathbb{Z}^+$ $b \in \mathbb{Z}^+$

using Int_ZF_2_L16 by auto

with I have $\abs(\delta(f,a,b)) \leq M$ by simp \}

moreover

\{ assume $a \leq 0 \wedge b \leq 0$

with I have $\abs(\delta(f,-a,-b)) \leq M$

using Int_ZF_2_L10A Int_ZF_2_L16 by simp

with $A1$ $A3$ $A4$ have $\abs(\delta(f,a,b)) \leq M$

using Int_ZF_2_1_L19 by simp \}

moreover

\{ assume $a \leq 0 \wedge 0 \leq b \wedge 0 \leq a + b$

with I have $\abs(\delta(f,-a,a+b)) \leq M$

using Int_ZF_2_L10A Int_ZF_2_L16 by simp

with $A1$ $A3$ $A4$ have $\abs(\delta(f,a,b)) \leq M$

using Int_ZF_2_1_L19 by simp \}

moreover

\{ assume $a \leq 0 \wedge 0 \leq b \wedge a + b \leq 0$

with I have $\abs(\delta(f,b,-(a+b))) \leq M$

using Int_ZF_2_L10A Int_ZF_2_L16 by simp

with $A1$ $A3$ $A4$ have $\abs(\delta(f,a,b)) \leq M$

using Int_ZF_2_1_L19 by simp \}

moreover

\{ assume $0 \leq a \wedge b \leq 0 \wedge 0 \leq a + b$

with I have $\abs(\delta(f,-b,a+b)) \leq M$

using Int_ZF_2_L10A Int_ZF_2_L16 by simp

with $A1$ $A3$ $A4$ have $\abs(\delta(f,a,b)) \leq M$

using Int_ZF_2_1_L19 by simp \}

moreover

\{ assume $0 \leq a \wedge b \leq 0 \wedge a + b \leq 0$

with I have $\abs(\delta(f,a,-(a+b))) \leq M$

using Int_ZF_2_L10A Int_ZF_2_L16 by simp

with $A1$ $A3$ $A4$ have $\abs(\delta(f,a,b)) \leq M$

using Int_ZF_2_1_L19 by simp \}

ultimately have $\abs(\delta(f,a,b)) \leq M$ by auto \}

691
then have $\forall m \in \mathbb{Z}$. $\forall n \in \mathbb{Z}$. $\text{abs}(\delta(f,m,n)) \leq M$ by simp
with A1 show $f \in S$ by (rule Int_ZF_2_1_L5)
qed

If the homomorphism difference of a function defined on positive integers is bounded, then the odd extension of this function is a slope.

lemma (in int1) Int_ZF_2_1_L24:
assumes A1: $f: \mathbb{Z}^+ \rightarrow \mathbb{Z}$ and A2: $\forall a \in \mathbb{Z}^+. \forall b \in \mathbb{Z}^+. \text{abs}(\delta(f,a,b)) \leq L$
shows $\text{OddExtension}(\mathbb{Z}, \text{IntegerAddition}, \text{IntegerOrder}, f) \in S$
proof -
let $g = \text{OddExtension}(\mathbb{Z}, \text{IntegerAddition}, \text{IntegerOrder}, f)$
from A1 have $g : \mathbb{Z} \rightarrow \mathbb{Z}$
using Int_ZF_1_5_L10 by simp
moreover have $\forall a \in \mathbb{Z}^+. \forall b \in \mathbb{Z}^+. \text{abs}(\delta(g,a,b)) \leq L$
proof -
{ fix $a$ $b$
assume A3: $a \in \mathbb{Z}^+$ $b \in \mathbb{Z}^+$
with A1 have $\text{abs}(\delta(f,a,b)) = \text{abs}(\delta(g,a,b))$
using pos_int_closed_add_unfolded Int_ZF_1_5_L11 by simp
moreover from A2 A3 have $\text{abs}(\delta(f,a,b)) \leq L$ by simp
ultimately have $\text{abs}(\delta(g,a,b)) \leq L$ by simp
} then show thesis by simp
qed
moreover from A1 have $\forall x \in \mathbb{Z}$. $(-g(-x)) = g(x)$
using int_oddext_is_odd_alt by simp
ultimately show $g \in S$ by (rule Int_ZF_2_1_L23)
qed

Type information related to $\gamma$.

lemma (in int1) Int_ZF_2_1_L25:
assumes A1: $f: \mathbb{Z} \rightarrow \mathbb{Z}$ and A2: $m \in \mathbb{Z}$ $n \in \mathbb{Z}$
shows
$\delta(f,m,-n) \in \mathbb{Z}$
$\delta(f,n,-n) \in \mathbb{Z}$
$(-\delta(f,n,-n)) \in \mathbb{Z}$
$f(0) \in \mathbb{Z}$
$\gamma(f,m,n) \in \mathbb{Z}$
proof -
from A1 A2 show T1: $\delta(f,m,-n) \in \mathbb{Z}$ $f(0) \in \mathbb{Z}$
using Int_ZF_1_1_L4 Int_ZF_2_1_L3B int_zero_one_are_int apply_funtype by auto
from A2 have $(-n) \in \mathbb{Z}$
using Int_ZF_1_1_L4 by simp
with A1 A2 show $\delta(f,n,-n) \in \mathbb{Z}$
using Int_ZF_2_1_L3B by simp
then show $(-\delta(f,n,-n)) \in \mathbb{Z}$
using Int_ZF_1_1_L4 by simp
with T1 show $\gamma(f,m,n) \in \mathbb{Z}$

692
A couple of formulae involving \( f(m - n) \) and \( \gamma(f, m, n) \).

**Lemma (in int1) Int_ZF_2_1_L26:**
assumes A1: \( f: \mathbb{Z} \rightarrow \mathbb{Z} \) and A2: \( m \in \mathbb{Z} \quad n \in \mathbb{Z} \)
shows
\[
\begin{align*}
f(m-n) &= \gamma(f, m, n) + f(m) - f(n) \\
f(m-n) &= \gamma(f, m, n) + (f(m) - f(n)) \\
f(m-n) &= (f(n) - \gamma(f, m, n)) = f(m)
\end{align*}
\]
proof -
from A1 A2 have T:
\[
\begin{align*}
(-n) &\in \mathbb{Z} \\
\delta(f, m, -n) &\in \mathbb{Z} \\
f(0) &\in \mathbb{Z} \\
f(m) &\in \mathbb{Z} \\
f(n) &\in \mathbb{Z} \\
(-f(n)) &\in \mathbb{Z} \\
(-\delta(f, n, -n)) &+ f(0) \in \mathbb{Z} \\
\gamma(f, m, n) &\in \mathbb{Z}
\end{align*}
\]
using Int_ZF_1_1_L4 Int_ZF_2_1_L25 apply_funtype Int_ZF_1_1_L5
by auto
with A1 A2 have f(m-n) = \[
\delta(f, m, -n) + ((-\delta(f, n, -n)) + f(0)) + f(m)
\]
using Int_ZF_2_1_L3C Int_ZF_2_1_L14A by simp
with T have f(m-n) = \[
\delta(f, m, -n) + ((-\delta(f, n, -n)) + f(0)) + f(m) - f(n)
\]
using Int_ZF_1_2_L16 by simp
moreover from T have \[
\delta(f, m, -n) + ((-\delta(f, n, -n)) + f(0)) = \gamma(f, m, n)
\]
using Int_ZF_1_1_L7 by simp
ultimately show I: \( f(m-n) = \gamma(f, m, n) + f(m) - f(n) \)
by simp
then have f(m-n) + (f(n) - \gamma(f, m, n)) = \[
(\gamma(f, m, n) + f(m) - f(n)) + (f(n) - \gamma(f, m, n))
\]
by simp
moreover from T have ... = f(m) using Int_ZF_1_2_L18
by simp
ultimately show f(m-n) + (f(n) - \gamma(f, m, n)) = f(m)
by simp
from T have \( \gamma(f, m, n) \in \mathbb{Z} \) and \( -f(n) \in \mathbb{Z} \)
by auto
then have \[
\gamma(f, m, n) + f(m) + (-f(n)) = \gamma(f, m, n) + (f(m) + (-f(n)))
\]
by rule Int_ZF_1_1_L7
with I show f(m-n) = \[
\gamma(f, m, n) + (f(m) - f(n))
\]
by simp
qed

A formula expressing the difference between \( f(m - n - k) \) and \( f(m) - f(n) - f(k) \) in terms of \( \gamma \).

**Lemma (in int1) Int_ZF_2_1_L26A:**
assumes A1: \( f: \mathbb{Z} \rightarrow \mathbb{Z} \) and A2: \( m \in \mathbb{Z} \quad n \in \mathbb{Z} \quad k \in \mathbb{Z} \)
shows
\[ f(m-n-k) - (f(m) - f(n) - f(k)) = \gamma(f,m-n,k) + \gamma(f,m,n) \]

proof -

from A1 A2 have
  T: \( m-n \in \mathbb{Z} \) \( \gamma(f,m-n,k) \in \mathbb{Z} \) \( f(m) - f(n) - f(k) \in \mathbb{Z} \) and
  T1: \( \gamma(f,m,n) \in \mathbb{Z} \) \( f(m) - f(n) \in \mathbb{Z} \) \( (-f(k)) \in \mathbb{Z} \)
  using Int_ZF_1_1_L4 Int_ZF_1_1_L5 Int_ZF_2_1_L25 apply_funtype by auto

from A1 A2 have
  \( f(m-n) - f(k) = \gamma(f,m,n) + (f(m) - f(n)) + (-f(k)) \)
  using Int_ZF_2_1_L26 by simp

also from T1 have \( \ldots = \gamma(f,m,n) + (f(m) - f(n) + (-f(k))) \)
  by (rule Int_ZF_1_1_L7)

finally have
  \( f(m-n) - f(k) = \gamma(f,m,n) + (f(m) - f(n) - f(k)) \)
  by simp

moreover from A1 A2 T have
  \( f(m-n-k) = \gamma(f,m-n,k) + (f(m-n)-f(k)) \)
  using Int_ZF_2_1_L26 by simp

ultimately have
  \( f(m-n-k) - (f(m) - f(n) - f(k)) = \gamma(f,m-n,k) + (\gamma(f,m,n) + (f(m) - f(n) - f(k))) = \)
  \( - (f(m) - f(n) - f(k)) \)
  by simp

with T T1 show thesis
  using Int_ZF_1_2_L17 by simp

qed

If \( s \) is a slope, then \( \gamma(s,m,n) \) is uniformly bounded.

lemma (in int1) Int_ZF_2_1_L27: assumes A1: \( s \in \mathcal{S} \)
shows \( \exists L \in \mathbb{Z}. \forall m \in \mathbb{Z}. \forall n \in \mathbb{Z}. \ abs(\gamma(s,m,n)) \leq L \)

proof -

let \( L = \max \delta(s) + \max \delta(s) + \abs(s(0)) \)

from A1 have T:
  \( \max \delta(s) \in \mathbb{Z}. \ abs(s(0)) \in \mathbb{Z} \) \( L \in \mathbb{Z} \)
  using Int_ZF_2_1_L8 int_zero_one_are_int Int_ZF_2_1_L2B Int_ZF_2_1_L14 Int_ZF_1_1_L5 by auto

moreover
  \{ fix \ m
  fix n
  assume A2: \( m \in \mathbb{Z} \) \( n \in \mathbb{Z} \)
  with A1 have T:
    \( (-n) \in \mathbb{Z} \)
    \( \delta(s,m,-n) \in \mathbb{Z} \)
    \( \delta(s,n,-n) \in \mathbb{Z} \)
    \( (-\delta(s,n,-n)) \in \mathbb{Z} \)
    \( s(0) \in \mathbb{Z} \) \( abs(s(0)) \in \mathbb{Z} \)
    using Int_ZF_1_1_L4 AlmostHoms_def Int_ZF_2_1_L25 Int_ZF_2_1_L14 by auto

694
with $T$ have
\[
\text{abs}(\delta(s,m,-n) - \delta(s,n,-n) + s(0)) \leq \\
\text{abs}(\delta(s,m,-n)) + \text{abs}(-\delta(s,n,-n)) + \text{abs}(s(0))
\]
using Int_triangle_ineq3 by simp
moreover from $A1$ $A2$ $T$ have
\[
\text{abs}(\delta(s,m,-n)) + \text{abs}(-\delta(s,n,-n)) + \text{abs}(s(0)) \leq L
\]
using Int_ZF_2_1_L7 int_ineq_add_sides int_ord_transl_inv Int_ZF_2_L17 by simp
ultimately have $\text{abs}(\delta(s,m,-n) - \delta(s,n,-n) + s(0)) \leq L$
by (rule Int_order_transitive)
then have $\text{abs}(\gamma(s,m,n)) \leq L$
by simp
ultimately show $\exists L \in \mathbb{Z}. \forall m \in \mathbb{Z}. \forall n \in \mathbb{Z}. \text{abs}(\gamma(s,m,n)) \leq L$
by auto
qed

If $s$ is a slope, then $s(m) \leq s(m-1) + M$, where $L$ does not depend on $m$.

**Lemma (in int1)** Int_ZF_2_1_L28: assumes $A1$: $s \in S$
shows $\exists M \in \mathbb{Z}. \forall m \in \mathbb{Z}. s(m) \leq s(m-1) + M$

**Proof** -
from $A1$ have
\[
\exists L \in \mathbb{Z}. \forall m \in \mathbb{Z}. \forall n \in \mathbb{Z}. \text{abs}(\gamma(s,m,n)) \leq L
\]
using Int_ZF_2_1_L27 by simp
then obtain $L$ where $T$: $L \in \mathbb{Z}$ and $\forall m \in \mathbb{Z}. \forall n \in \mathbb{Z}. \text{abs}(\gamma(s,m,n)) \leq L$
using Int_ZF_2_1_L27 by auto
then have $I$: $\forall m \in \mathbb{Z}. \text{abs}(\gamma(s,m,1)) \leq L$
using int_zero_one_are_int by simp
let $M = s(1) + L$
from $A1$ $T$ have $M \in \mathbb{Z}$
using int_zero_one_are_int Int_ZF_2_1_L2B Int_ZF_1_1_L5 by simp
moreover
\{ fix $m$ assume $A2$: $m \in \mathbb{Z}$
with $A1$ have
\[
T1: s: \mathbb{Z} \rightarrow \mathbb{Z}. m \in \mathbb{Z}. 1 \in \mathbb{Z} \text{ and } \\
T2: \gamma(s,m,1) \in \mathbb{Z}. s(1) \in \mathbb{Z}
\]
using int_zero_one_are_int AlmostHoms_def Int_ZF_2_1_L25 by auto
from $A2$ $T1$ have $T3$: $s(m-1) \in \mathbb{Z}$
using Int_ZF_1_1_L5 apply_funtype by simp
from $I$ $A1$ $T2$ have
\[
(\gamma(s,m,1)) \leq \text{abs}(\gamma(s,m,1))
\]
\[
\text{abs}(\gamma(s,m,1)) \leq L
\]
using Int_ZF_2_1_L19C by auto
then have $(\gamma(s,m,1)) \leq L$
by (rule Int_order_transitive)
with $T2$ $T3$ have
\[
s(m-1) + (s(1) - \gamma(s,m,1)) \leq s(m-1) + M
\]
using int_ord_transl_inv by simp
moreover from $T1$ have
\[ s(m-1) + (s(1) - \gamma(s,m,1)) = s(m) \]

by (rule Int_ZF_2_1_L26)

ultimately have \( s(m) \leq s(m-1) + M \) by simp 

ultimately show \( \exists M \in \mathbb{Z}. \forall m \in \mathbb{Z}. \ s(m) \leq s(m-1) + M \)

by auto

qed

If \( s \) is a slope, then the difference between \( s(m-n-k) \) and \( s(m) - s(n) - s(k) \) is uniformly bounded.

lemma (in int1) Int_ZF_2_1_L29: assumes A1: \( s \in S \)
shows \( \exists M \in \mathbb{Z}. \forall m \in \mathbb{Z}. \forall n \in \mathbb{Z}. \forall k \in \mathbb{Z}. \ abs(s(m-n-k) - (s(m) - s(n) - s(k))) \leq M \)

proof -

from A1 have \( \exists L \in \mathbb{Z}. \forall m \in \mathbb{Z}. \forall n \in \mathbb{Z}. \ abs(\gamma(s,m,n)) \leq L \)
using Int_ZF_2_1_L27 simp -
then obtain \( L \) where I: \( L \in \mathbb{Z} \) and II: \( \forall m \in \mathbb{Z}. \forall n \in \mathbb{Z}. \ abs(\gamma(s,m,n)) \leq L \)
by auto

from I have \( L+L \in \mathbb{Z} \)
using Int_ZF_1_1_L5 by simp -
moreover
\{ fix \( m, n, k \) assume A2: \( m \in \mathbb{Z} \), \( n \in \mathbb{Z} \), \( k \in \mathbb{Z} \)
with A1 have T:
\( m-n \in \mathbb{Z} \); \( \gamma(s,m-n,k) \in \mathbb{Z} \); \( \gamma(s,m,n) \in \mathbb{Z} \)
using Int_ZF_1_1_L5 AlmostHoms_def Int_ZF_2_1_L25
by auto
then have I: \( abs(\gamma(s,m-n,k) + \gamma(s,m,n)) \leq abs(\gamma(s,m-n,k)) + abs(\gamma(s,m,n)) \)
using Int_triangle_ineq simp from II A2 T have
\( abs(\gamma(s,m-n,k)) \leq L \)
\( abs(\gamma(s,m,n)) \leq L \)
by auto
then have \( abs(\gamma(s,m-n,k)) + abs(\gamma(s,m,n)) \leq L+L \)
using int_ineq_add_sides simp -
with I have \( abs(\gamma(s,m-n,k) + \gamma(s,m,n)) \leq L+L \)
by (rule Int_order_transitive)
moreover from A1 A2 have
\( s(m-n-k) - (s(m) - s(n) - s(k)) = \gamma(s,m-n,k) + \gamma(s,m,n) \)
using AlmostHoms_def Int_ZF_2_1_L26A by simp
ultimately have \( abs(s(m-n-k) - (s(m) - s(n) - s(k))) \leq L+L \)
by simp \}
ultimately show thesis by auto

qed

If \( s \) is a slope, then we can find integers \( M, K \) such that \( s(m-n-k) \leq s(m) - s(n) - s(k) + M \) and \( s(m) - s(n) - s(k) + K \leq s(m-n-k) \), for all integer \( m, n, k \).
lemma (in int1) Int_ZF_2_1_L30: assumes A1: \( s \in S \)
shows
\[ \exists M \in \mathbb{Z}. \forall m \in \mathbb{Z} \cdot \forall n \in \mathbb{Z} \cdot \forall k \in \mathbb{Z}. s(m-n-k) \leq s(m) - s(n) - s(k) + M \]

proof -
from A1 have
\[ \exists M \in \mathbb{Z}. \forall m \in \mathbb{Z} \cdot \forall n \in \mathbb{Z} \cdot \forall k \in \mathbb{Z}. \text{abs}(s(m-n-k) - (s(m)-s(n)-s(k))) \leq M \]
using Int_ZF_2_1_L29 by simp
then obtain \( M \) where I: \( M \in \mathbb{Z} \) and II: \( \forall m \in \mathbb{Z} \cdot \forall n \in \mathbb{Z} \cdot \forall k \in \mathbb{Z}. \text{abs}(s(m-n-k) - (s(m)-s(n)-s(k))) \leq M \)
by auto
from I have III: \( (-M) \in \mathbb{Z} \) using Int_ZF_1_1_L4 by simp
{ fix \( m \), \( n \), \( k \)
assume A2: \( m \in \mathbb{Z} \)
\( n \in \mathbb{Z} \)
\( k \in \mathbb{Z} \)
with A1 have \( s(m-n-k) \in \mathbb{Z} \) and \( s(m)-s(n)-s(k) \in \mathbb{Z} \)
using Int_ZF_2_1_L5 Int_ZF_2_1_L2B by auto
moreover from II A2 have \( \text{abs}(s(m-n-k) - (s(m)-s(n)-s(k))) \leq M \)
by simp
ultimately have
\[ s(m-n-k) \leq s(m) - s(n) - s(k) + M \wedge \]
\[ s(m)-s(n)-s(k) - M \leq s(m-n-k) \]
using Int_triangle_ineq2 by simp
} then have
\[ \forall m \in \mathbb{Z} \cdot \forall n \in \mathbb{Z} \cdot \forall k \in \mathbb{Z}. s(m-n-k) \leq s(m) - s(n) - s(k) + M \]
\[ \forall m \in \mathbb{Z} \cdot \forall n \in \mathbb{Z} \cdot \forall k \in \mathbb{Z}. s(m)-s(n)-s(k) - M \leq s(m-n-k) \]
by auto
with I III show
\[ \exists M \in \mathbb{Z} \cdot \forall m \in \mathbb{Z} \cdot \forall n \in \mathbb{Z} \cdot \forall k \in \mathbb{Z}. s(m-n-k) \leq s(m) - s(n) - s(k) + M \]
\[ \exists K \in \mathbb{Z} \cdot \forall m \in \mathbb{Z} \cdot \forall n \in \mathbb{Z} \cdot \forall k \in \mathbb{Z}. s(m)-s(n)-s(k) + K \leq s(m-n-k) \]
by auto
qed

By definition functions \( f,g \) are almost equal if \( f - g^* \) is bounded. In the next lemma we show it is sufficient to check the boundedness on positive integers.

lemma (in int1) Int_ZF_2_1_L31: assumes A1: \( s \in S \)
and A2: \( \forall m \in \mathbb{Z}_+. \cdot \text{abs}(s(m)-r(m)) \leq L \)
shows \( s \sim r \)

proof -
let \( a = \text{abs}(s(0) - r(0)) \)
let \( c = 2 \cdot \text{max} \delta(s) + 2 \cdot \text{max} \delta(r) + L \)
let \( M = \text{Maximum}(\text{IntegerOrder}, \{a,L,c\}) \)
from A2 have \( \text{abs}(s(1)-r(1)) \leq L \)
using int_one_two_are_pos by simp
then have T: \( L \in \mathbb{Z} \) using Int_ZF_2_1_L1A by simp
moreover from A1 have \( a \in \mathbb{Z} \)
using int_zero_one_are_int Int_ZF_2_1_L2B
Int_ZF_1_1_L5 Int_ZF_2_1_L14 by simp
moreover from A1 T have \( c \in \mathbb{Z} \)

697
ultimately have
I: \( a \leq M \) and
II: \( L \leq M \) and
III: \( c \leq M \)

using IntZF_1_4_L1A by auto

\[
\{ \text{fix } m \text{ assume } A5: m \in \mathbb{Z} \\
\text{with } A1 \text{ have } T:\ \\
s(m) \in \mathbb{Z}, \ s(m) - r(m) \in \mathbb{Z} \\
s(-m) \in \mathbb{Z}, \ r(-m) \in \mathbb{Z} \\
\text{using IntZF}_2_1_L2B \text{ IntZF}_1_1_L4 \text{ IntZF}_1_1_L5 \\
\text{by auto} \\
\text{from } A5 \text{ have } m=0 \lor m \in \mathbb{Z}^+ \lor (-m) \in \mathbb{Z}^+ \\
\text{using int_decomp_cases by simp} \\
moreover \}
\text{assume } m=0 \\
\text{with I have } \|s(m) - r(m)\| \leq M \\
\text{by simp} \\
moreover \}
\text{assume } m \in \mathbb{Z}^+ \\
\text{with } A2 \text{ II have} \\
\|s(m) - r(m)\| \leq L \text{ and } L \leq M \\
\text{by auto} \\
\text{then have } \|s(m) - r(m)\| \leq M \\
\text{by (rule Int_order_transitive)} \}
moreover \}
\text{assume } A6: (-m) \in \mathbb{Z}^+ \\
\text{from } T \text{ have } \|s(m) - r(m)\| \leq \\
\|s(m) + s(-m)\| + \|r(m) + r(-m)\| + \|s(-m) - r(-m)\| \\
\text{using IntZF}_1_3_L22A by simp \\
moreover \}
\text{from } A1 \text{ A2 III A5 A6 have} \\
\|s(m) + s(-m)\| + \|r(m) + r(-m)\| + \|s(-m) - r(-m)\| \leq c \\
c \leq M \\
\text{using IntZF}_2_1_L14 \text{ int_ineq_add_sides by auto} \\
\text{then have} \\
\|s(m) + s(-m)\| + \|r(m) + r(-m)\| + \|s(-m) - r(-m)\| \leq M \\
\text{by (rule Int_order_transitive)} \\
ultimately have \|s(m) - r(m)\| \leq M \\
\text{by (rule Int_order_transitive)} \}
ultimately have \|s(m) - r(m)\| \leq M \\
\text{by auto} \\
\text{then have } \forall m \in \mathbb{Z}. \ \|s(m) - r(m)\| \leq M \\
\text{by simp} \\
\text{with } A1 \text{ show } s \sim r \text{ by (rule IntZF}_2_1_L9) \\
\text{qed}

A sufficient condition for an odd slope to be almost equal to identity: If for
all positive integers the value of the slope at \( m \) is between \( m \) and \( m + M \) plus some constant independent of \( m \), then the slope is almost identity.

**Lemma (in int1) Int_ZF_2_1_L32:** assumes 
\( A1: s \in S \quad M \in \mathbb{Z} \)
\( A2: \forall m \in \mathbb{Z}_+. \ m \leq s(m) \land s(m) \leq m + M \)
shows \( s \sim id(\mathbb{Z}) \)

**proof -**
- let \( r = id(\mathbb{Z}) \)
- from \( A1 \) have \( s \in S \) \( r \in S \)
  - using Int_ZF_2_1_L17 by auto
- moreover from \( A1 \ A2 \) have \( \forall m \in \mathbb{Z}_+. \ |s(m) - r(m)| \leq M \)
  - using Int_ZF_2_1_L23 PositiveSet_def id_conv by simp
- ultimately show \( s \sim id(\mathbb{Z}) \) by (rule Int_ZF_2_1_L31)

**qed**

A lemma about adding a constant to slopes. This is actually proven in Group_ZF_3_5_L1, in Group_ZF_3.thy here we just refer to that lemma to show it in notation used for integers. Unfortunately we have to use raw set notation in the proof.

**Lemma (in int1) Int_ZF_2_1_L33:**
assumes \( A1: s \in S \) and \( A2: c \in \mathbb{Z} \) and \( A3: r = \{ \langle m, s(m) + c \rangle . m \in \mathbb{Z}\} \)
shows 
\( \forall m \in \mathbb{Z}. \ r(m) = s(m) + c \)
\( r \in S \)
\( s \sim r \)

**proof -**
- let \( G = \mathbb{Z} \)
- let \( f = \text{IntegerAddition} \)
- let \( AH = \text{AlmostHoms}(G, f) \)
- from \( \text{assms} \) have I:
  - group1(G, f)
  - \( s \in \text{AlmostHoms}(G, f) \)
  - \( c \in G \)
  - \( r = \{ \langle x, f(s(x)), c \rangle . x \in G\} \)
  - using Int_ZF_2_1_L1 by auto
- then have \( \forall x \in G. \ r(x) = f(s(x),c) \)
  - by (rule group1.Group_ZF_3_5_L1)
- moreover from I have \( r \in \text{AlmostHoms}(G, f) \)
  - by (rule group1.Group_ZF_3_5_L1)
- moreover from I have
  - \( \langle s, r \rangle \in \text{QuotientGroupRel}(\text{AlmostHoms}(G, f), \text{AlHomOp1}(G, f), \text{FinRangeFunctions}(G, G)) \)
  - by (rule group1.Group_ZF_3_5_L1)
- ultimately show \( \forall m \in \mathbb{Z}. \ r(m) = s(m) + c \)
  - \( r \in S \)
  - \( s \sim r \)
  - by auto

699
56.2 Composing slopes

Composition of slopes is not commutative. However, as we show in this section if \( f \) and \( g \) are slopes then the range of \( f \circ g - g \circ f \) is bounded. This allows to show that the multiplication of real numbers is commutative.

Two useful estimates.

lemma (in int1) Int_ZF_2_2_L1:
assumes A1: \( f:\mathbb{Z} \rightarrow \mathbb{Z} \) and A2: \( p \in \mathbb{Z} \) \( q \in \mathbb{Z} \)
shows
\[
\text{abs}(f((p+1) \cdot q)-(p+1) \cdot f(q)) \leq \text{abs}(\delta(f,p \cdot q,q))+\text{abs}(f(p \cdot q)-p \cdot f(q))
\]
\[
\text{abs}(f((p-1) \cdot q)-(p-1) \cdot f(q)) \leq \text{abs}(\delta(f,(p-1) \cdot q,q))+\text{abs}(f(p \cdot q)-p \cdot f(q))
\]

proof -
let \( R = \mathbb{Z} \)
let \( A = \text{IntegerAddition} \)
let \( M = \text{IntegerMultiplication} \)
let \( I = \text{GroupInv}(R, A) \)
let \( a = f((p+1) \cdot q) \)
let \( b = p \)
let \( c = f(q) \)
let \( d = f(p \cdot q) \)
from A1 A2 have T1:
\( \text{ring0}(R, A, M) a \in R \) \( b \in R \) \( c \in R \) \( d \in R \)
using Int_ZF_1_1_L2 int_zero_one_are_int Int_ZF_1_1_L5 apply_funtype by auto
then have \( A \langle a, I(M A \langle A \langle b, TheNeutralElement(R, M) \rangle, c \rangle) \rangle = A \langle A \langle A \langle a, I(d) \rangle, I(c) \rangle, A \langle d, I(M A \langle b, c \rangle) \rangle \rangle \)
by (rule ring0.Ring_ZF_2_L2)
with A2 have \( f((p+1) \cdot q)-(p+1) \cdot f(q) = \delta(f,p \cdot q,q)+(f(p \cdot q)-p \cdot f(q)) \)
using int_zero_one_are_int Int_ZF_1_1_L1 Int_ZF_1_1_L4 by simp
moreover from A1 A2 T1 have \( \delta(f,p \cdot q,q) \in \mathbb{Z} \) \( f(p \cdot q)-p \cdot f(q) \in \mathbb{Z} \)
using Int_ZF_1_1_L5 apply_funtype by auto
ultimately show
\[
\text{abs}(f((p+1) \cdot q)-(p+1) \cdot f(q)) \leq \text{abs}(\delta(f,p \cdot q,q))+\text{abs}(f(p \cdot q)-p \cdot f(q))
\]
using Int_triangle_ineq by simp
from A1 A2 have T1:
\( f((p-1) \cdot q) \in \mathbb{Z} \) \( p \in \mathbb{Z} \) \( f(q) \in \mathbb{Z} \) \( f(p \cdot q) \in \mathbb{Z} \)
using int_zero_one_are_int Int_ZF_1_1_L5 apply_funtype by auto
then have \( f((p-1) \cdot q)-(p-1) \cdot f(q) = (f(p \cdot q)-p \cdot f(q))-(f(p-1) \cdot q)-f(q) \)
by (rule Int_ZF_1_2_L6)
with A2 have \( f((p-1) \cdot q)-(p-1) \cdot f(q) = (f(p \cdot q)-p \cdot f(q))-\delta(f,(p-1) \cdot q,q) \)
using Int_ZF_1_2_L7 by simp
moreover from A1 A2 have
If $f$ is a slope, then $|f(p \cdot q) - p \cdot f(q)| \leq (|p| + 1) \cdot \max \delta(f)$. The proof is by induction on $p$ and the next lemma is the induction step for the case when $0 \leq p$.

**Lemma (in int1) Int_ZF_2_2_L2:**

assumes $A1: f \in S$ and $A2: 0 \leq p \cdot q \in \mathbb{Z}$

and $A3: \text{abs}(f(p \cdot q) - p \cdot f(q)) \leq (\text{abs}(p) + 1) \cdot \max \delta(f)$

shows $\text{abs}(f((p+1) \cdot q) - (p+1) \cdot f(q)) \leq (\text{abs}(p+1) + 1) \cdot \max \delta(f)$

**Proof:**

- from $A2$ have $q \in \mathbb{Z}$, $p \cdot q \in \mathbb{Z}$
  
  using Int_ZF_2_L1A int_zero_one_are_int Int_ZF_1_1_L5 by auto

with $A1$ have $\text{I}: \text{abs}(\delta(f,(p-1) \cdot q,q)) \leq \max \delta(f)$ by (rule Int_ZF_2_1_L7)

moreover note $A3$

moreover from $A1$, $A2$ have

$\text{abs}(f((p+1) \cdot q) - (p+1) \cdot f(q)) \leq \text{abs}(\delta(f,(p-1) \cdot q,q)) + \text{abs}(f(p \cdot q) - p \cdot f(q))$

using AlmostHoms_def Int_ZF_2_L1A Int_ZF_2_2_L1 by simp

ultimately have

$\text{abs}(f((p+1) \cdot q) - (p+1) \cdot f(q)) \leq \text{max}(\delta(f) + (\text{abs}(p+1) \cdot \max \delta(f))$

by (rule Int_ZF_2_2_L15)

moreover from $\text{I}$, $A2$ have

$\text{max}(\delta(f) + (\text{abs}(p+1) \cdot \max \delta(f)) = (\text{abs}(p+1) + 1) \cdot \max \delta(f)$

using Int_ZF_2_L1A Int_ZF_1_2_L2 by simp

ultimately show $\text{abs}(f((p+1) \cdot q) - (p+1) \cdot f(q)) \leq (\text{abs}(p+1) + 1) \cdot \max \delta(f)$

by simp

**Qed**

If $f$ is a slope, then $|f(p \cdot q) - p \cdot f(q)| \leq (|p| + 1) \cdot \max \delta$. The proof is by induction on $p$ and the next lemma is the induction step for the case when $p \leq 0$.

**Lemma (in int1) Int_ZF_2_2_L3:**

assumes $A1: f \in S$ and $A2: p \leq 0 \cdot q \in \mathbb{Z}$

and $A3: \text{abs}(f(p \cdot q) - p \cdot f(q)) \leq (\text{abs}(p) + 1) \cdot \max \delta(f)$

shows $\text{abs}(f((p-1) \cdot q) - (p-1) \cdot f(q)) \leq (\text{abs}(p-1) + 1) \cdot \max \delta(f)$

**Proof:**

- from $A2$ have $q \in \mathbb{Z}$, $(p-1) \cdot q \in \mathbb{Z}$
  
  using Int_ZF_2_L1A int_zero_one_are_int Int_ZF_1_1_L5 by auto

with $A1$ have $\text{I}: \text{abs}(\delta(f,(p-1) \cdot q,q)) \leq \max \delta(f)$ by (rule Int_ZF_2_1_L7)

moreover note $A3$

moreover from $A1$, $A2$ have

$\text{abs}(f((p-1) \cdot q) - (p-1) \cdot f(q)) \leq \text{abs}(\delta(f,(p-1) \cdot q,q)) + \text{abs}(f(p \cdot q) - p \cdot f(q))$

using AlmostHoms_def Int_ZF_2_L1A Int_ZF_2_2_L1 by simp
ultimately have
\[ \text{ultimately have} \]
\[ \abs{f((p-1) \cdot q) - (p-1) \cdot f(q)} \leq \max \delta(f) + (\abs{p}+1) \cdot \max \delta(f) \]
by \text{(rule IntZF_2_L15)}

with \text{I A2 show thesis using IntZF_2_L1A IntZF_1_2_L5 by simp}

If \( f \) is a slope, then \( |f(p \cdot q) - p \cdot f(q)| \leq (|p| + 1) \cdot \max \delta(f) \). Proof by cases on \( 0 \leq p \).

**lemma** (in int1) IntZF_2_2_L4:
\[ \text{assumes A1: } f \in S \text{ and A2: } p \in \mathbb{Z}, q \in \mathbb{Z} \]
\[ \text{shows abs}(f(p \cdot q) - p \cdot f(q)) \leq (\abs{p}+1) \cdot \max \delta(f) \]
**proof** -
\[
\begin{align*}
\{ & \text{ assume } 0 \leq p \\
\{ & \text{ moreover from A1 A2 have } \abs{f(0 \cdot q) - 0 \cdot f(q)} \leq (\abs{0}+1) \cdot \max \delta(f) \\
\{ & \text{ using int_zero_one_are_int IntZF_2_1_L2B IntZF_1_1_L4 IntZF_2_1_L8 IntZF_2_L18 by simp} \\
\{ & \text{ moreover from A1 A2 have } \\
\{ & \quad \forall p. \ 0 \leq p \land \abs{f(p \cdot q) - p \cdot f(q)} \leq (\abs{p}+1) \cdot \max \delta(f) \rightarrow \\
\{ & \quad \abs{f((p+1) \cdot q) - (p+1) \cdot f(q)} \leq (\abs{p+1} + 1) \cdot \max \delta(f) \\
\{ & \quad \text{ using IntZF_2_2_L2 by simp} \\
\{ & \quad \text{ ultimately have } \abs{f(p \cdot q) - p \cdot f(q)} \leq (\abs{p}+1) \cdot \max \delta(f) \\
\{ & \quad \text{ by (rule Induction_on_int)} \\
\{ & \text{ moreover } \\
\{ & \quad \text{ assume } \neg (0 \leq p) \\
\{ & \quad \text{ with A2 have } p \leq 0 \text{ using IntZF_2_L19A by simp} \\
\{ & \quad \text{ moreover from A1 A2 have } \abs{f(0 \cdot q) - 0 \cdot f(q)} \leq (\abs{0}+1) \cdot \max \delta(f) \\
\{ & \quad \text{ using int_zero_one_are_int IntZF_2_1_L2B IntZF_1_1_L4 IntZF_2_1_L8 IntZF_2_L18 by simp} \\
\{ & \quad \text{ moreover from A1 A2 have } \\
\{ & \quad \forall p. \ p \leq 0 \land \abs{f(p \cdot q) - p \cdot f(q)} \leq (\abs{p}+1) \cdot \max \delta(f) \rightarrow \\
\{ & \quad \abs{f((p+1) \cdot q) - (p+1) \cdot f(q)} \leq (\abs{p+1} + 1) \cdot \max \delta(f) \\
\{ & \quad \text{ using IntZF_2_2_L3 by simp} \\
\{ & \quad \text{ ultimately have } \abs{f(p \cdot q) - p \cdot f(q)} \leq (\abs{p}+1) \cdot \max \delta(f) \\
\{ & \quad \text{ by (rule Back_induct_on_int)} \\
\{ & \text{ ultimately show thesis by blast} \\
\text{qed}
\end{align*}
\]

The next elegant result is Lemma 7 in the Arthan’s paper [2].

**lemma** (in int1) Arthan_Lem_7:
\[ \text{assumes A1: } f \in S \text{ and A2: } p \in \mathbb{Z}, q \in \mathbb{Z} \]
\[ \text{shows } \abs{q \cdot f(p) - p \cdot f(q)} \leq (\abs{p}+\abs{q}+2) \cdot \max \delta(f) \]
**proof** -
\[
\begin{align*}
\text{from A1 A2 have } T: \\
\text{q f(p) - f(p q) } \in \mathbb{Z} \\
f(p q) - p f(q) \in \mathbb{Z} \\
f(q p) \in \mathbb{Z}, f(p q) \in \mathbb{Z} \\
q f(p) \in \mathbb{Z}, p f(q) \in \mathbb{Z} \\
\max \delta(f) \in \mathbb{Z} \\
\abs{q} \in \mathbb{Z}, \abs{p} \in \mathbb{Z}
\end{align*}
\]
using Int_ZF_1_1_L5 Int_ZF_2_1_L2B Int_ZF_2_1_L7 Int_ZF_2_L14 by auto
moreover have \( \text{abs}(qf(p) - f(pq)) \leq (\text{abs}(q) + 1) \max \delta(f) \)
proof -
from A1 A2 have \( \text{abs}(f(qp) - qf(p)) \leq (\text{abs}(q) + 1) \max \delta(f) \)
using Int_ZF_2_2_L4 by simp
with T A2 show thesis
using Int_ZF_2_L20 Int_ZF_1_1_L5 by simp
qed
moreover have \( \text{abs}(fp(pq) - pqf(pq)) \leq (\text{abs}(pq) + 1) \max \delta(f) \)
proof -
from A1 A2 have \( \text{abs}(fp(pq) - pqf(pq)) \leq (\text{abs}(pq) + 1) \max \delta(f) \)
using Int_ZF_2_2_L4 by simp
ultimately have \( \text{abs}(qf(p) - f(pq) + (f(pq) - pqf(pq))) \leq (\text{abs}(q) + 1) \max \delta(f) + (\text{abs}(pq) + 1) \max \delta(f) \)
using Int_ZF_2_L21 by simp
with T show thesis using Int_ZF_1_2_L9 int_zero_one_are_int Int_ZF_1_2_L10
by simp
qed

This is Lemma 8 in the Arthan’s paper.

lemma (in int1) Arthan_Lem_8: assumes A1: \( f \in S \)
shows \( \exists A B. A \in \mathbb{Z} \land B \in \mathbb{Z} \land (\forall p \in \mathbb{Z}. \text{abs}(f(p)) \leq A \cdot \text{abs}(p) + B) \)
proof -
let \( A = \max \delta(f) + \text{abs}(f(1)) \)
let \( B = 3 \cdot \max \delta(f) \)
from A1 have \( A \in \mathbb{Z} \land B \in \mathbb{Z} \)
using int_zero_one_are_int Int_ZF_1_1_L5 Int_ZF_2_1_L2B
Int_ZF_2_1_L7 Int_ZF_2_L14 by auto
moreover have \( \forall p \in \mathbb{Z}. \text{abs}(f(p)) \leq A \cdot \text{abs}(p) + B \)
proof -
fix \( p \) assume A2: \( p \in \mathbb{Z} \)
with A1 have T: \( f(p) \in \mathbb{Z} \land \text{abs}(p) \in \mathbb{Z} \land f(1) \in \mathbb{Z} \)
\( p \cdot f(1) \in \mathbb{Z} \land 3 \cdot \max \delta(f) \in \mathbb{Z} \)
using Int_ZF_2_1_L2B Int_ZF_2_1_L14 int_zero_one_are_int
Int_ZF_1_1_L5 Int_ZF_2_1_L7 by auto
from A1 A2 have \( \text{abs}(1 \cdot f(p) - p \cdot f(1)) \leq (\text{abs}(p) + \text{abs}(1) + 2) \max \delta(f) \)
using int_zero_one_are_int Arthan_Lem_7 by simp
with T have \( \text{abs}(f(p)) \leq \text{abs}(p \cdot f(1)) + (\text{abs}(p) + 3) \max \delta(f) \)
using Int_ZF_2_L16A Int_ZF_1_1_L4 Int_ZF_1_2_L11
Int_triangle_ineq2 by simp
with A2 T show \( \text{abs}(f(p)) \leq A \cdot \text{abs}(p) + B \)
using Int_ZF_1_3_L14 by simp
qed
ultimately show thesis by auto
qed

If \( f \) and \( g \) are slopes, then \( f \circ g \) is equivalent (almost equal) to \( g \circ f \). This is Theorem 9 in Arthan’s paper [2].

theorem (in int1) Arthan_Th_9: assumes A1: \( f \in S \land g \in S \)

shows $f \circ g \sim g \circ f$

proof -

from $A_1$ have
\[ \exists A \ B \ C \ D \in \mathbb{Z} \land (\forall p \in \mathbb{Z}. \ abs(f(p)) \leq A \cdot abs(p) + B) \]
using Arthan_Lem_6 by auto
then obtain $A \ B \ C \ D$ where $D_1: A \in \mathbb{Z}$ $B \in \mathbb{Z}$ $C \in \mathbb{Z}$ $D \in \mathbb{Z}$ and $D_2$:
\[ \forall p \in \mathbb{Z}. \ abs(f(p)) \leq A \cdot abs(p) + B \]
\[ \forall p \in \mathbb{Z}. \ abs(g(p)) \leq C \cdot abs(p) + D \]
by auto
let $E = \max\delta(g) \cdot (A+1) + \max\delta(f) \cdot (C+1)$
let $F = (B \cdot \max\delta(g) + 2 \cdot \max\delta(g)) + (D \cdot \max\delta(f) + 2 \cdot \max\delta(f))$

{ fix $p$ assume $A_2$: $p \in \mathbb{Z}$
with $A_1$ have $T_1$:
\[ g(p) \in \mathbb{Z} \land f(p) \in \mathbb{Z} \land abs(p) \in \mathbb{Z} \land 2 \in \mathbb{Z} \]
\[ f(g(p)) \in \mathbb{Z} \land g(f(p)) \in \mathbb{Z} \land f(g(p)) - g(f(p)) \in \mathbb{Z} \]
\[ p \cdot f(g(p)) \in \mathbb{Z} \land p \cdot g(f(p)) \in \mathbb{Z} \]
\[ \max\delta(g) \cdot \max\delta(f) \leq \max\delta(g) \cdot (A+1) + \max\delta(f) \cdot (C+1) \]
by auto
with $A_1$ $A_2$ have
\[ \max\delta(g) \cdot \max\delta(f) \leq \max\delta(g) \cdot (A+1) + \max\delta(f) \cdot (C+1) \]
using $\max\delta(g) \cdot (A+1) + \max\delta(f) \cdot (C+1)$
moreover have
\[ \max\delta(g) \cdot (A+1) + \max\delta(f) \cdot (C+1) \]
proof -

from $D_2$ $A_2$ $T_1$ have
\[ abs(p) + abs(f(p)) + 2 \leq abs(p) + (A \cdot abs(p) + B) + 2 \]
\[ abs(p) + abs(g(p)) + 2 \leq abs(p) + (C \cdot abs(p) + D) + 2 \]
using $\max\delta(g) \cdot (A+1) + \max\delta(f) \cdot (C+1)$
moreover from $A_1$ $D_1$ $T_1$ have
\[ (A \cdot abs(p) + B) + 2 \cdot \max\delta(g) = \max\delta(g) \cdot (A+1) - abs(p) + (B \cdot \max\delta(g) + 2 \cdot \max\delta(g)) \]
\[ (C \cdot abs(p) + D) + 2 \cdot \max\delta(f) = \max\delta(f) \cdot (C+1) - abs(p) + (D \cdot \max\delta(f) + 2 \cdot \max\delta(f)) \]
using $\max\delta(g) \cdot (A+1) - abs(p) + (B \cdot \max\delta(g) + 2 \cdot \max\delta(g))$
moreover from $A_1$ $D_1$ $T_1$ have
ultimately have
\[ (A \cdot abs(p) + B) + 2 \cdot \max\delta(g) + (C \cdot abs(p) + D) + 2 \cdot \max\delta(f) \leq \max\delta(g) \cdot (A+1) - abs(p) + (B \cdot \max\delta(g) + 2 \cdot \max\delta(g)) + (D \cdot \max\delta(f) + 2 \cdot \max\delta(f)) \]
using $\max\delta(g) \cdot (A+1)$ $\max\delta(f)$$\max\delta(g) + 2 \cdot \max\delta(g)) + (D \cdot \max\delta(f) + 2 \cdot \max\delta(f))$
moreover from $A_1$ $A_2$ $D_1$ have $abs(p) \in \mathbb{Z}$
maxδ(g)·(A+1) ∈ ℤ. B·maxδ(g) + 2·maxδ(g) ∈ ℤ
maxδ(f)·(C+1) ∈ ℤ. D·maxδ(f) + 2·maxδ(f) ∈ ℤ
using Int_ZF_2_L14 Int_ZF_2_1_L8 int_zero_one_are_int
Int_ZF_1_1_L5 int_two_three_are_int by auto
ultimately show thesis using Int_ZF_1_2_L14 by simp
qed
ultimately have
abs((f(g(p))−g(f(p)))·p) ≤ E·abs(p) + F
by (rule Int_order_transitive)
with A2 T1 have
abs(f(g(p))−g(f(p)))·abs(p) ≤ E·abs(p) + F
abs(f(g(p))−g(f(p))) ∈ ℤ
using Int_ZF_1_3_L5
by auto
} then have
∀p∈ℤ. abs(f(g(p))−g(f(p))) ∈ ℤ
∀p∈ℤ. abs(f(g(p))−g(f(p)))·abs(p) ≤ E·abs(p) + F
by auto
moreover from A1 D1 have E ∈ ℤ F ∈ ℤ
using int_zero_one_are_int int_two_three_are_int Int_ZF_2_1_L8 Int_ZF_1_1_L5
by auto
ultimately have
∃L. ∀p∈ℤ. abs(f(g(p))−g(f(p))) ≤ L
by (rule Int_ZF_1_7_L1)
with A1 obtain L where ∀p∈ℤ. abs((fog)(p)−(gof)(p)) ≤ L
using Int_ZF_2_1_L10 by auto
moreover from A1 have fog ∈ S gof ∈ S
using Int_ZF_2_1_L11 by auto
ultimately show fog ∼ gof using Int_ZF_2_1_L9 by auto
qed
end

57 Integers 3

theory Int_ZF_3 imports Int_ZF_2

begin

This theory is a continuation of Int_ZF_2. We consider here the properties
of slopes (almost homomorphisms on integers) that allow to define the order
relation and multiplicative inverse on real numbers. We also prove theorems
that allow to show completeness of the order relation of real numbers we
define in Real_ZF.

57.1 Positive slopes

This section provides background material for defining the order relation on
real numbers.

705
Positive slopes are functions (of course.)

lemma (in int1) Int_ZF_2_3_L1: assumes A1: \( f \in \mathcal{S}_+ \) shows \( f : \mathbb{Z} \rightarrow \mathbb{Z} \)
using assms AlmostHoms_def PositiveSet_def by simp

A small technical lemma to simplify the proof of the next theorem.

lemma (in int1) Int_ZF_2_3_L1A: assumes A1: \( f \in \mathcal{S}_+ \) and A2: \( \exists n \in f(\mathbb{Z}_+) \cap \mathbb{Z}_+, \ a \leq n \)
shows \( \exists M \in \mathbb{Z}_+. \ a \leq f(M) \)
proof -
from A1 have f: \( \mathbb{Z} \rightarrow \mathbb{Z} \)
using assms AlmostHoms_def PositiveSet_def by simp
from A1 A2 show thesis using func_imagedef by auto
qed

The next lemma is Lemma 3 in the Arthan’s paper.

lemma (in int1) Arthan_Lem_3: assumes A1: \( f \in \mathcal{S}_+ \) and A2: \( D \in \mathbb{Z}_+ \)
shows \( \exists M \in \mathbb{Z}_+. \ \forall m \in \mathbb{Z}_+. (m+1) \cdot D \leq f(m \cdot M) \)
proof -
let E = max(\delta(f)) + D
let A = f(\mathbb{Z}_+) \cap \mathbb{Z}_+\nfrom A1 A2 have I: \( D \leq E \)
using Int_ZF_1_5_L3 Int_ZF_2_1_L8 Int_ZF_2_L1A Int_ZF_2_L15D by simp
from A1 A2 have A: \( \mathbb{Z}_+ \subseteq \mathbb{Z}_+ \)
using int_two_three_are_int Int_ZF_2_1_L8 PositiveSet_def Int_ZF_1_1_L5 by auto
with A1 have \( \exists M \in \mathbb{Z}_+. \ 2 \cdot E \leq f(M) \)
using Int_ZF_2_L1A Int_ZF_2_3_L1A by simp
then obtain M where II: \( M \in \mathbb{Z}_+ \) and III: \( 2 \cdot E \leq f(M) \)
by auto
{ fix m assume m \( \in \mathbb{Z}_+ \) then have A4: \( 1 \leq m \)
using Int_ZF_2_L1_5_L3 by simp
moreover from I I I have \( (1+1) \cdot E \leq f(1 \cdot M) \)
using PositiveSet_def Int_ZF_2_1_L4 by simp
moreover have \( \forall k. \ 1 \leq k \land (k+1) \cdot E \leq f(k \cdot M) \rightarrow (k+1) \cdot E \leq f((k+1) \cdot M) \)
proof -
{ fix k assume A5: \( 1 \leq k \) and A6: \( (k+1) \cdot E \leq f(k \cdot M) \)
with A1 A2 II have T:
\( k \in \mathbb{Z} \) \( M \in \mathbb{Z} \) \( k+1 \in \mathbb{Z} \) \( E \in \mathbb{Z} \) \( (k+1) \cdot E \in \mathbb{Z} \) \( 2 \cdot E \in \mathbb{Z} \)
using Int_ZF_2_L1A PositiveSet_def int_zero_one_are_int
Int_ZF_1_1_L5 Int_ZF_2_1_L8 by auto
from A1 A2 A5 II have \( \delta(f,k,M,M) \in \mathbb{Z} \)
abs(\delta(f,k,M,M)) \leq max(\delta(f)) \( 0 \leq D \)
using Int_ZF_2_L1A PositiveSet_def Int_ZF_2_1_L5
Int_ZF_2_1_L7 Int_ZF_2_L16C by auto
with III A6 have \( (k+1) \cdot E + (2 \cdot E - E) \leq f(k \cdot M) + (f(M) + \delta(f,k,M,M)) \)
}
using Int_ZF_1_3_L19A int_ineq_add_sides by simp
with A1 T have \((k+1+1) \cdot E \leq f((k+1) \cdot M)\)
  using Int_ZF_1_1_L1 int_zero_one_are_int Int_ZF_1_1_L4
  Int_ZF_1_2_L11 Int_ZF_2_1_L13 by simp
  \} then show thesis by simp
qed
ultimately have \((k+1+1) \cdot E \leq f((k+1) \cdot M)\) by simp
with A4 I have \((m+1) \cdot D \leq f(m \cdot M)\) by (rule Induction_on_int)
with II show thesis by auto
qed

A special case of Arthan_Lem_3 when \(D = 1\).
corollary (in int1) Arthan_L_3_spec: assumes A1: \(f \in S_+\)
  shows \(\exists M \in \mathbb{Z}_+. \forall n \in \mathbb{Z}_+. \; n+1 \leq f(n \cdot M)\)
proof -
  have \(\forall n \in \mathbb{Z}_+. \; n+1 \in \mathbb{Z}\)
    using PositiveSet_def int_zero_one_are_int Int_ZF_1_1_L5
    by simp
  then have \(\forall n \in \mathbb{Z}_+. \; (n+1) \cdot 1 = n+1\)
    using Int_ZF_1_1_L4 by simp
  moreover from A1 have \(\exists M \in \mathbb{Z}_+. \; \forall n \in \mathbb{Z}_+. \; (n+1) \cdot 1 \leq f(n \cdot M)\)
    using int_one_two_are_pos Arthan_Lem_3 by simp
  ultimately show thesis by simp
qed

We know from Group_ZF_3.thy that finite range functions are almost homomorphisms. Besides reminding that fact for slopes the next lemma shows that finite range functions do not belong to \(S_+\). This is important, because the projection of the set of finite range functions defines zero in the real number construction in Real_ZF.x.thy series, while the projection of \(S_+\) becomes the set of (strictly) positive reals. We don’t want zero to be positive, do we? The next lemma is a part of Lemma 5 in the Arthan’s paper [2].

lemma (in int1) Int_ZF_2_3_L1B:
  assumes A1: \(f \in \text{FinRangeFunctions}(\mathbb{Z}, \mathbb{Z})\)
  shows \(f \in S \quad f \not\in S_+\)
proof -
  from A1 show \(f \in S\) using Int_ZF_2_1_L1 group1.Group_ZF_3_3_L1
    by auto
  have \(\mathbb{Z}_+ \subseteq \mathbb{Z}\) using PositiveSet_def by auto
  with A1 have \(f(\mathbb{Z}_+) \in \text{Fin}(\mathbb{Z})\)
    using Finite1_L21 by simp
  then have \(f(\mathbb{Z}_+) \cap \mathbb{Z}_+ \in \text{Fin}(\mathbb{Z})\)
    using Fin_subset_lemma by blast
  thus \(f \not\in S_+\) by auto
qed

We want to show that if \(f\) is a slope and neither \(f\) nor \(-f\) are in \(S_+\), then
is bounded. The next lemma is the first step towards that goal and shows that if slope is not in $S_+$ then $f(Z_+)$ is bounded above.

**Lemma (in int1) Int_ZF_2_3_L2:**

Assumes $A_1: f \in S$ and $A_2: f \notin S_+$.

Shows $\text{IsBoundedAbove}(f(Z_+), \text{IntegerOrder})$.

**Proof** -

- From $A_1$ have $f: Z \to Z$ using $\text{AlmostHoms_def}$ by simp
- Then have $f(Z_+) \subseteq Z$ using $\text{func1_1_L6}$ by simp
- Moreover from $A_1$ $A_2$ have $f(Z_+) \cap Z_+ \in \text{Fin}(Z)$ by auto
- Ultimately show thesis using $\text{Int_ZF_2_T1 \ group3.OrderedGroup_ZF_2_L4}$ by simp

**Qed**

If $f$ is a slope and $-f \notin S_+$, then $f(Z_+)$ is bounded below.

**Lemma (in int1) Int_ZF_2_3_L3:**

Assumes $A_1: f \in S$ and $A_2: -f \notin S_+$.

Shows $\text{IsBoundedBelow}(f(Z_+), \text{IntegerOrder})$.

**Proof** -

- From $A_1$ have $T: f: Z \to Z$ using $\text{AlmostHoms_def}$ by simp
- Then have $-(f(Z_+))) = (-f)(Z_+)$ using $\text{Int_ZF_1_T2 \ group0_2_T2 \ PositiveSet_def \ func1_1_L15C}$ by auto
- With $A_1$ $A_2$ show $\text{IsBoundedBelow}(f(Z_+), \text{IntegerOrder})$ using $\text{Int_ZF_2_1_L12 \ Int_ZF_2_3_L2 \ PositiveSet_def \ func1_1_L6}$
- $\text{Int_ZF_2_T1 \ group3.OrderedGroup_ZF_2_L5}$ by simp

**Qed**

A slope that is bounded on $Z_+$ is bounded everywhere.

**Lemma (in int1) Int_ZF_2_3_L4:**

Assumes $A_1: f \in S$ and $A_2: m \in Z$

And $A_3: \forall n \in Z_+. \ \text{abs}(f(n)) \leq L$.

Shows $\text{abs}(f(m)) \leq 2 \cdot \text{max} \delta(f) + L$.

**Proof** -

- From $A_1$ $A_3$ have $0 \leq \text{abs}(f(1)) \ \text{abs}(f(1)) \leq L$
- Using $\text{int_zero_one_are_int \ Int_ZF_2_1_L2B \ int_abs_nonneg \ int_one_two_are_pos}$ by auto
- Then have II: $0 \leq L$ by (rule $\text{Int_order_transitive}$)

**Note** $A_2$

Moreover have $\text{abs}(f(0)) \leq 2 \cdot \text{max} \delta(f) + L$

**Proof** -

- From $A_1$ have $\text{abs}(f(0)) \leq \text{max} \delta(f) \ 0 \leq \text{max} \delta(f)$
- And $T: \text{max} \delta(f) \in Z$
- Using $\text{Int_ZF_2_1_L18}$ by auto
- With II have $\text{abs}(f(0)) \leq \text{max} \delta(f) + \text{max} \delta(f) + L$
- Using $\text{Int_ZF_2_L15F}$ by simp
- With $T$ show thesis using $\text{Int_ZF_1_1_L4}$ by simp

**Qed**

Moreover from $A_1$ $A_3$ II have

708
∀n∈\mathbb{Z}^+. \quad \text{abs}(f(n)) \leq 2 \cdot \max\delta(f) + L \quad \text{using} \quad \text{Int}_{ZF\_2\_1\_L8} \quad \text{Int}_{ZF\_1\_3\_L5A} \quad \text{Int}_{ZF\_2\_L15F} \quad \text{by} \quad \text{simp}

moreover have \quad \forall n\in\mathbb{Z}^+. \quad \text{abs}(f(-n)) \leq 2 \cdot \max\delta(f) + L

\text{proof}

fix n assume n\in\mathbb{Z}^+

with A1 A3 have

\text{abs}(f(-n)) \leq 2 \cdot \max\delta(f) + \text{abs}(f(n))

\text{abs}(f(n)) \leq L \quad \text{using} \quad \text{Int}_{ZF\_2\_L15A} \quad \text{by} \quad \text{blast}

ultimately show thesis by (rule Int_{ZF\_2\_L19B})

qed

A slope whose image of the set of positive integers is bounded is a finite range function.

**lemma (in int1) Int_{ZF\_2\_3\_L4A}**

assumes A1: f\in S and A2: IsBounded(f(\mathbb{Z}^+_)), IntegerOrder

shows f \in \text{FinRangeFunctions}(\mathbb{Z}, \mathbb{Z})

\text{proof -}

have T1: \mathbb{Z}^+_\subseteq \mathbb{Z} using PositiveSet_def by auto

from A1 have T2: f:\mathbb{Z}\rightarrow \mathbb{Z} using AlmostHoms_def by simp

from A2 obtain L where \forall a\in f(\mathbb{Z}^+_). \quad \text{abs}(a) \leq L

\text{using} \quad \text{Int}_{ZF\_1\_3\_L20A} \quad \text{by} \quad \text{auto}

with T2 T1 have \forall n\in \mathbb{Z}^+. \quad \text{abs}(f(n)) \leq L

by (rule func1_1_L15B)

with A1 have \forall m\in \mathbb{Z}^+. \quad \text{abs}(f(m)) \leq 2 \cdot \max\delta(f) + L

\text{using} \quad \text{Int}_{ZF\_2\_3\_L4} \quad \text{by} \quad \text{simp}

with T2 have f(\mathbb{Z}) \in \text{Fin}(\mathbb{Z})

by (rule Int_{ZF\_1\_3\_L20C})

with T2 show f \in \text{FinRangeFunctions}(\mathbb{Z}, \mathbb{Z})

using FinRangeFunctions_def by simp

qed

A slope whose image of the set of positive integers is bounded below is a finite range function or a positive slope.

**lemma (in int1) Int_{ZF\_2\_3\_L4B}**

assumes f\in S and IsBoundedBelow(f(\mathbb{Z}^+_)), IntegerOrder

shows f \in \text{FinRangeFunctions}(\mathbb{Z}, \mathbb{Z}) \lor f\in S^+_\quad \text{using} \quad \text{assms} \quad \text{Int}_{ZF\_2\_3\_L2} \quad \text{IsBounded_def} \quad \text{Int}_{ZF\_2\_3\_L4A} \quad \text{by} \quad \text{auto}

If one slope is not greater then another on positive integers, then they are almost equal or the difference is a positive slope.

**lemma (in int1) Int_{ZF\_2\_3\_L4C}**

assumes A1: f\in S \quad g\in S and
\( \forall n \in \mathbb{Z}_+. f(n) \leq g(n) \) shows \( f \sim g \lor g + (-f) \in S_+ \)

**proof** - 
let \( h = g + (-f) \)
from A1 have \((-f) \in S\) using Int_ZF_2_1_L12
by simp
with A1 have I: \( h \in S \) using Int_ZF_2_1_L12C
by simp
moreover have IsBoundedBelow(h(\(\mathbb{Z}_+\)), IntegerOrder)
proof - 
from I have \( h: \mathbb{Z}_+ \rightarrow \mathbb{Z} \) and \( \mathbb{Z}_+ \subseteq \mathbb{Z} \) using AlmostHoms_def PositiveSet_def
by auto
moreover from A1 A2 have \( \forall n \in \mathbb{Z}_+. (0, h(n)) \in \text{IntegerOrder} \)
using Int_ZF_2_1_L2B PositiveSet_def Int_ZF_1_3_L10A
Int_ZF_2_1_L12 Int_ZF_2_1_L12B Int_ZF_2_1_L12A
by simp
ultimately show IsBoundedBelow(h(\(\mathbb{Z}_+\)), IntegerOrder)
by (rule func_ZF_8_L1)
qed
ultimately have \( h \in \text{FinRangeFunctions}(\mathbb{Z}, \mathbb{Z}) \lor h \in S_+ \)
using Int_ZF_2_3_L4B by simp
with A1 show \( f \sim g \lor g + (-f) \in S_+ \)
using Int_ZF_2_1_L9C by auto
qed

Positive slopes are arbitrarily large for large enough arguments.

**lemma (in int1) Int_ZF_2_3_L5:**
assumes A1: \( f \in S_+ \) and A2: \( K \in \mathbb{Z} \)
shows \( \exists N \in \mathbb{Z}_+. \forall m. N \leq m \rightarrow K \leq f(m) \)
proof - 
from A1 obtain M where I: \( M \in \mathbb{Z}_+ \) and II: \( \forall n \in \mathbb{Z}_+. n+1 \leq f(n \cdot M) \)
using Arthan_L_3_spec by auto
let \( j = \text{GreaterOf}(\text{IntegerOrder}, M, K - (\text{minf}(f, 0..(M-1)) - \text{max}(f)) - 1) \)
from A1 I have T1: \( \text{minf}(f, 0..(M-1)) - \text{max}(f) \in \mathbb{Z} \)
using Int_ZF_2_1_L15 Int_ZF_2_1_L8 Int_ZF_1_1_L5 PositiveSet_def by auto
with A2 I have T2: \( K - (\text{minf}(f, 0..(M-1)) - \text{max}(f)) \in \mathbb{Z} \)
\( K - (\text{minf}(f, 0..(M-1)) - \text{max}(f)) - 1 \in \mathbb{Z} \)
using Int_ZF_1_1_L5 int_zero_one_are_int by auto
with T1 have III: \( M \leq j \) and \( K - (\text{minf}(f, 0..(M-1)) - \text{max}(f)) - 1 \leq j \)
using Int_ZF_1_3_L18 by auto
with A2 T1 T2 have IV: \( K \leq j+1 + (\text{minf}(f, 0..(M-1)) - \text{max}(f)) \)
using int_zero_one_are_int Int_ZF_2_L9C by simp
let $N = \text{GreaterOf}(\text{IntegerOrder}, 1, j \cdot M)$

from T1 III have T3: $j \in \mathbb{Z}$, $j \cdot M \in \mathbb{Z}$

using Int_ZF_2_L1A Int_ZF_1_1_L5 by auto

then have V: $N \in \mathbb{Z}^+$ and VI: $j \cdot M \leq N$

using int_zero_one_are_int Int_ZF_1_5_L3 Int_ZF_1_3_L18 by auto

{ fix $m$
  let $n = m \div M$
  let $k = m \mod M$
  assume $N \leq m$
  with VI have $j \cdot M \leq m$ by (rule Int_order_transitive)
  with I III have
  VII: $m = n \cdot M + k$
  j $\leq n$ and
  VIII: $n \in \mathbb{Z}^+$, $k \in 0..(M-1)$
  using IntDiv_ZF_1_L5 by auto
  with II have
  $j + 1 \leq n + 1$ $n+1 \leq f(n \cdot M)$
  using int_zero_one_are_int int_ord_transl_inv by auto
  then have $j + 1 \leq f(n \cdot M)$
  by (rule Int_order_transitive)
  with T1 have
  $j + 1 + (\text{minf}(f, 0..(M-1)) - \text{max}(f)) \leq f(n \cdot M) + (\text{minf}(f, 0..(M-1)) - \text{max}(f))$
  using int_ord_transl_inv by simp
  with IV have $K \leq f(n \cdot M) + (\text{minf}(f, 0..(M-1)) - \text{max}(f))$
  by (rule Int_order_transitive)
  moreover from A1 I VIII have
  $f(n \cdot M) + (\text{minf}(f, 0..(M-1)) - \text{max}(f)) \leq f(n \cdot M + k)$
  using PositiveSet_def Int_ZF_2_1_L16 by simp
  ultimately have $K \leq f(n \cdot M + k)$
  by (rule Int_order_transitive)
  with VII have $K \leq f(m)$ by simp
  } then have $\forall m. N \leq m \longrightarrow K \leq f(m)$
  by simp
  with V show thesis by auto

qed

Positive slopes are arbitrarily small for small enough arguments. Kind of dual to Int_ZF_2_3_L5.

lemma (in int1) Int_ZF_2_3_L5A: assumes A1: $f \in \mathcal{S}^+$ and A2: $K \in \mathbb{Z}$
shows $\exists N \in \mathbb{Z}^+. \forall m. N \leq m \longrightarrow f(-m) \leq K$

proof -
  from A1 have T1: $\text{abs}(f(0)) + \text{max}(f) \in \mathbb{Z}$
  using Int_ZF_2_L1A Int_ZF_1_1_L8 by auto
  with A2 have $\text{abs}(f(0)) + \text{max}(f) - K \in \mathbb{Z}$
  using Int_ZF_1_1_L5 by simp
  with A1 have
  $\exists N \in \mathbb{Z}^+. \forall m. N \leq m \longrightarrow \text{abs}(f(0)) + \text{max}(f) - K \leq f(m)$
  by auto

711
using \texttt{IntZF.2.3_L5} by simp
then obtain \( N \) where \( \text{I}: N \in \mathbb{Z}_+ \) and \( \text{II} \):
\[ \forall m. \ N \leq m \rightarrow \text{abs}(f(0)) + \max \delta(f) - K \leq f(m) \]
by auto
\{ fix \( m \) assume \( \text{A3}: N \leq m \)
with \( \text{A1} \) have
\[ f(-m) \leq \text{abs}(f(0)) + \max \delta(f) - f(m) \]
using \texttt{IntZF.2.1_L14} by simp
moreover from \( \text{II} \ T1 \ \text{A3} \) have
\[ \text{abs}(f(0)) + \max \delta(f) - f(m) \leq K \]
using \texttt{IntZF.2.1_L10} by simp
ultimately have \( f(-m) \leq K \)
by (rule \texttt{Int_order_transitive})
\}
then have \( \forall m. \ N \leq m \rightarrow f(-m) \leq K \)
by simp
with \( \text{I} \) show thesis by auto
qed

A special case of \texttt{IntZF.2.3_L5} where \( K = 1 \).

corollary (in \texttt{int}) \texttt{IntZF.2.3_L6}:
assumes \( f \in S_+ \)
shows \( \exists N \in \mathbb{Z}_+. \ \forall m. \ N \leq m \rightarrow f(m) \in \mathbb{Z}_+ \)
using \texttt{assms \texttt{int_zero_one_are_int \texttt{IntZF.2.3_L5 \texttt{IntZF.1_S_L3}}} by simp

A special case of \texttt{IntZF.2.3_L5} where \( m = N \).

corollary (in \texttt{int}) \texttt{IntZF.2.3_L6A}:
assumes \( f \in S_+ \) and \( K \in \mathbb{Z} \)
shows \( \exists N \in \mathbb{Z}_+. \ K \leq f(N) \)
proof -
from \texttt{assms} have \( \exists N \in \mathbb{Z}_+. \ \forall m. \ N \leq m \rightarrow K \leq f(m) \)
using \texttt{IntZF.2.3_L5} by simp
then obtain \( N \) where \( \text{I}: N \in \mathbb{Z}_+ \) and \( \text{II} \): \( \forall m. \ N \leq m \rightarrow K \leq f(m) \)
by auto
then show thesis using \texttt{PositiveSet_def \texttt{int_is_refl \texttt{refl_def}}} by auto
qed

If values of a slope are not bounded above, then the slope is positive.

lemma (in \texttt{int}) \texttt{IntZF.2.3_L7}:
assumes \( A1: f \in S \)
and \( A2: \ \forall K \in \mathbb{Z}. \ \exists n \in \mathbb{Z}_+. \ K \leq f(n) \)
shows \( f \in S_+ \)
proof -
\{ fix \( k \) assume \( K \in \mathbb{Z} \)
with \( A2 \) obtain \( n \) where \( n \in \mathbb{Z}_+ \) \( K \leq f(n) \)
by auto
moreover from \( A1 \) have \( \mathbb{Z}_+ \subseteq \mathbb{Z}. \ f: \mathbb{Z} \rightarrow \mathbb{Z} \)
using \texttt{PositiveSet_def \texttt{AlmostHoms_def}} by auto
\}

712
ultimately have $\exists m \in f(Z_+) \cdot K \leq m$
using $\text{func1_1_L15D by auto}$

} then have $\forall K \in Z. \exists m \in f(Z_+) \cdot K \leq m$ by simp
with $A1$ show $f \in S_+$ using $\text{Int_ZF_4_L9}$ $\text{Int_ZF_2_3_L2}$
by auto

qed

For unbounded slope $f$ either $f \in S_+$ of $-f \in S_+$.

theorem (in int1) $\text{Int_ZF_2_3_L8}$:
assumes $A1: f \in S$ and $A2: f \notin \text{FinRangeFunctions}(Z,Z)$
shows $(f \in S_+) \lor (-f) \in S_+$
proof -

{ have $T1: Z_+ \subseteq Z$ using $\text{PositiveSet_def by auto}$
from $A1$ have $T2: f: Z \rightarrow Z$ using $\text{AlmostHoms_def by simp}$
then have $I: f(Z_+) \subseteq Z$ using $\text{func1_1_L6 by auto}$
from $A1$ $A2$ have $f \in S_+ \lor (-f) \in S_+$
using $\text{Int_ZF_2_3_L2}$ $\text{Int_ZF_2_3_L3}$ $\text{IsBounded_def}$ $\text{Int_ZF_2_3_L4A}$
by blast
moreover have $\neg (f \in S_+ \land (-f) \in S_+)$
proof -

{ assume $A3: f \in S_+$ and $A4: (-f) \in S_+$
from $A3$ obtain $N1$ where
$I: N1 \in Z_+$ and $II: \forall m. N1 \leq m \rightarrow f(m) \in Z_+$
using $\text{Int_ZF_2_3_L6 by auto}$
from $A4$ obtain $N2$ where
$III: N2 \in Z_+$ and $IV: \forall m. N2 \leq m \rightarrow (-f)(m) \in Z_+$
using $\text{Int_ZF_2_3_L6 by auto}$
let $N = \text{GreaterOf}(\text{IntegerOrder},N1,N2)$
from $I$ $III$ have $N1 \leq N$ $N2 \leq N$
using $\text{PositiveSet_def}$ $\text{Int_ZF_1_3_L18 by auto}$
with $A1$ $II$ $IV$ have
$f(N) \in Z_+$ $(-f)(N) \in Z_+$ $(-f)(N) = -(f(N))$
using $\text{Int_ZF_2_L1A}$ $\text{PositiveSet_def}$ $\text{Int_ZF_2_1_L12A}$
by auto
then have False using $\text{Int_ZF_1_5_L8 by simp}$
} thus thesis by auto

qed

ultimately show $(f \in S_+) \lor (-f) \in S_+$
using $\text{xor_def by simp}$

qed

The sum of positive slopes is a positive slope.

theorem (in int1) $\text{sum_of_pos_sls_is_pos_sl}$:
assumes $A1: f \in S_+$ $g \in S_+$
shows $f+g \in S_+$
proof -

{ fix $K$ assume $K \in Z$
with $A1$ have $\exists N \in Z_+. \forall m. N \leq m \rightarrow K \leq f(m)$
using $\text{Int_ZF_2_3_L5 by simp}$

713
then obtain \( N \) where I: \( N \in \mathbb{Z}_+ \) and II: \( \forall m. \ N \leq m \rightarrow K \leq f(m) \)
by auto
from A1 have \( \exists M \in \mathbb{Z}_+. \ \forall m. \ M \leq m \rightarrow 0 \leq g(m) \)
using \text{int_zero_one_are_int} \ Int_ZF_2_3_L5 by simp
then obtain \( M \) where III: \( M \in \mathbb{Z}_+ \) and IV: \( \forall m. \ M \leq m \rightarrow 0 \leq g(m) \)
by auto
let \( L = \text{GreaterOf}(\text{IntegerOrder}, N, M) \)
from I III have V: \( L \in \mathbb{Z}_+ \)
III \( \subseteq \mathbb{Z} \)
using \text{GreaterOf_def} \ PositiveSet_def by auto
moreover from A1 V have \( (f \circ g)(L) = f(L) + g(L) \)
using \text{Int_ZF_2_1_L12B} by auto
moreover from I II III IV have \( K \leq f(L) + g(L) \)
using \text{PositiveSet_def} \ \text{Int_ZF_1_3_L18} \ \text{Int_ZF_2_L15F} by simp
ultimately have \( L \in \mathbb{Z}_+ \) \( K \leq (f \circ g)(L) \)
by auto
then have \( \exists n \in \mathbb{Z}_+. \ K \leq (f \circ g)(n) \)
by auto
\}
with A1 show \( f \circ g \in S_+ \)
using \text{Int_ZF_2_1_L12C} \ \text{Int_ZF_2_3_L7} by simp
qed

The composition of positive slopes is a positive slope.

\text{theorem} (in \text{int1}) \ \text{comp_of_pos_sls_is_pos_sl}:
assumes A1: \( f \in S_+ \) \( g \in S_+ \)
shows \( f \circ g \in S_+ \)
proof -
\{ fix \( K \) assume \( K \in \mathbb{Z} \)
with A1 have \( \exists N \in \mathbb{Z}_+. \ \forall m. \ N \leq m \rightarrow K \leq f(m) \)
using \text{Int_ZF_2_3_L5} by simp
then obtain \( N \) where \( N \in \mathbb{Z}_+ \) and I: \( \forall m. \ N \leq m \rightarrow K \leq f(m) \)
by auto
with A1 have \( \exists M \in \mathbb{Z}_+. \ N \leq g(M) \)
using \text{PositiveSet_def} \ \text{Int_ZF_2_3_L6A} by simp
then obtain \( M \) where \( M \in \mathbb{Z}_+ \) \( N \leq g(M) \)
by auto
with A1 I have \( \exists M \in \mathbb{Z}_+. \ K \leq (f \circ g)(M) \)
using \text{PositiveSet_def} \ \text{Int_ZF_2_1_L10} by auto
\}
with A1 show \( f \circ g \in S_+ \)
using \text{Int_ZF_2_1_L11} \ \text{Int_ZF_2_3_L7} by simp
qed

A slope equivalent to a positive one is positive.

\text{lemma} (in \text{int1}) \ \text{Int_ZF_2_3_L9}:
assumes A1: \( f \in S_+ \) and A2: \( (f, g) \in A1EqRel \)
shows \( g \in S_+ \)
proof -
from A2 have T: \( g \in S \) and \( \exists L \in \mathbb{Z}. \ \forall m \in \mathbb{Z}. \ \text{abs}(f(m)-g(m)) \leq L \)
714
using Int_ZF_2_1_L9A by auto
then obtain $L$ where
I: $L \in \mathbb{Z}$ and II: $\forall m \in \mathbb{Z}$. $|f(m) - g(m)| \leq L$
by auto

\{ fix $K$ assume A3: $K \in \mathbb{Z}$
with I have $K + L \in \mathbb{Z}$
using Int_ZF_1_1_L5 by simp
with A1 obtain $M$ where III: $M \in \mathbb{Z}^+$ and IV: $K + L \leq f(M)$
using Int_ZF_2_3_L6A by auto
with A1 A3 I have $K \leq f(M) - L$
using PositiveSet_def Int_ZF_2_1_L2B Int_ZF_2_L9B
by simp
moreover from A1 T II III have
$f(M) - L \leq g(M)$
using PositiveSet_def Int_ZF_2_1_L2B Int_triangle_ineq2
by simp
ultimately have $K \leq g(M)$
by (rule Int_order_transitive)
with III have \exists n \in \mathbb{Z}^+. K \leq g(n)
by auto
\} with T show $g \in S^+$
using Int_ZF_2_3_L7 by simp
qed

The set of positive slopes is saturated with respect to the relation of equivalence of slopes.

lemma (in int1) pos_slopes_saturated: shows IsSaturated(AlEqRel, $S^+$)
proof -
have\equiv(S, AlEqRel)
AlEqRel \subseteq S \times S
using Int_ZF_2_1_L9B by auto
moreover have $S^+ \subseteq S$ by auto
moreover have $\forall f \in S^+. \forall g \in S$. $(f, g) \in AlEqRel \Longrightarrow g \in S^+$
using Int_ZF_2_3_L9 by blast
ultimately show IsSaturated(AlEqRel, $S^+$)
by (rule EquivClass_3_L3)
qed

A technical lemma involving a projection of the set of positive slopes and a logical expression with exclusive or.

lemma (in int1) Int_ZF_2_3_L10: assumes A1: $f \in S$ g\in S
and A2: $R = \{AlEqRel \{a\}. a \in S^+\}$
and A3: $(f \in S^+ \wedge (g \in S^+) \wedge (f \in S^+) \wedge (g \in S^+))$
shows $(AlEqRel \{f\} \in R) \wedge (AlEqRel \{g\} \in R)$
proof -
from A1 A2 A3 have $equiv(S, AlEqRel)$
IsSaturated(AlEqRel, S+)
S+ ⊆ S
f ∈ S, g ∈ S
R = {AlEqRel{s}. s ∈ S+}
(f ∈ S+) Xor (g ∈ S+)

using pos_slopes_saturated Int_ZF_2_1_L9B by auto
then show thesis by (rule EquivClass_3_L7)
qed

Identity function is a positive slope.

lemma (in int1) Int_ZF_2_3_L11: shows id(\mathbb{Z}) ∈ S+
proof -
  let f = id(\mathbb{Z})
  { fix K assume K∈\mathbb{Z}
    then obtain n where T: n∈\mathbb{Z}_+ and K≤n
      using Int_ZF_1_5_L9 by auto
    moreover from T have f(n) = n
      using PositiveSet_def by simp
    ultimately have n∈\mathbb{Z}_+ and K≤f(n)
      by auto
    then have \exists n∈\mathbb{Z}_+. K≤f(n) by auto
  } then show f ∈ S+.
    using Int_ZF_2_1_L17 Int_ZF_2_3_L7 by simp
qed

The identity function is not almost equal to any bounded function.

lemma (in int1) Int_ZF_2_3_L12: assumes A1: f ∈ FinRangeFunctions(\mathbb{Z},\mathbb{Z})
  shows ¬(id(\mathbb{Z}) ∼ f)
proof -
  { from A1 have id(\mathbb{Z}) ∈ S+
    using Int_ZF_2_3_L11 by simp
    moreover assume (id(\mathbb{Z}),f) ∈ AlEqRel
    ultimately have f ∈ S+
      by (rule Int_ZF_2_3_L9)
    with A1 have False using Int_ZF_2_3_L1B
      by simp
  } then show ¬(id(\mathbb{Z}) ∼ f) by auto
qed

57.2 Inverting slopes

Not every slope is a 1:1 function. However, we can still invert slopes in the
sense that if f is a slope, then we can find a slope g such that f ◦ g is almost
equal to the identity function. The goal of this this section is to establish
this fact for positive slopes.

If f is a positive slope, then for every positive integer p the set \{n ∈ \mathbb{Z}_+: p ≤ f(n)\} is a nonempty subset of positive integers. Recall that f^{-1}(p) is
the notation for the smallest element of this set.
lemma (in int1) Int_ZF_2_4_L1:
  assumes A1: \( f \in S \) and A2: \( p \in \mathbb{Z}_+ \) and A3: \( A = \{ n \in \mathbb{Z}_+. \ p \leq f(n) \} \)
  shows
  \( A \subseteq \mathbb{Z}_+ \)
  \( A \neq 0 \)
  \( f^{-1}(p) \in A \)
  \( \forall m \in A. \ f^{-1}(p) \leq m \)
proof -
  from A3 show I: \( A \subseteq \mathbb{Z}_+ \) by auto
  from A1 A2 have \( \exists n \in \mathbb{Z}_+. \ p \leq f(n) \)
    using PositiveSet_def Int_ZF_2_3_L6A by simp
  with A3 show II: \( A \neq 0 \) by auto
  from A3 I II show
    \( f^{-1}(p) \in A \)
    \( \forall m \in A. \ f^{-1}(p) \leq m \)
    using Int_ZF_1_5_L1C by auto
qed

If \( f \) is a positive slope and \( p \) is a positive integer \( p \), then \( f^{-1}(p) \) (defined as
the minimum of the set \( \{ n \in \mathbb{Z}_+: p \leq f(n) \} \) ) is a (well defined) positive
integer.

lemma (in int1) Int_ZF_2_4_L2:
  assumes f \( \in S \) and p \( \in \mathbb{Z}_+ \)
  shows
    \( f^{-1}(p) \in \mathbb{Z}_+ \)
    \( p \leq f(f^{-1}(p)) \)
  using assms Int_ZF_2_4_L1 by auto

If \( f \) is a positive slope and \( p \) is a positive integer such that \( n \leq f(p) \), then
\( f^{-1}(n) \leq p \).

lemma (in int1) Int_ZF_2_4_L3:
  assumes f \( \in S \) and \( m \in \mathbb{Z}_+ \) \( p \in \mathbb{Z}_+ \) and \( m \leq f(p) \)
  shows \( f^{-1}(m) \leq p \)
  using assms Int_ZF_2_4_L1 by simp

An upper bound \( f(f^{-1}(m) - 1) \) for positive slopes.

lemma (in int1) Int_ZF_2_4_L4:
  assumes A1: \( f \in S \) and A2: \( m \in \mathbb{Z}_+ \) and A3: \( f^{-1}(m)-1 \in \mathbb{Z}_+ \)
  shows \( f(f^{-1}(m)-1) \leq m \ \ f(f^{-1}(m)-1) \neq m \)
proof -
  from A1 A2 have T: \( f^{-1}(m) \in \mathbb{Z} \) using Int_ZF_2_4_L2 PositiveSet_def
    by simp
  from A1 A3 have f: \( \mathbb{Z} \to \mathbb{Z} \) and f^{-1}(m)-1 \( \in \mathbb{Z} \)
    using Int_ZF_2_3_L1 PositiveSet_def by auto
  with A1 A2 have T1: \( f(f^{-1}(m)-1) \in \mathbb{Z} \ \ m \in \mathbb{Z} \)
    using apply_funtype PositiveSet_def by auto
  \{ assume m \( \leq f(f^{-1}(m)-1) \)
    with A1 A2 A3 have \( f^{-1}(m) \leq f^{-1}(m)-1 \)
  \}
The (candidate for) the inverse of a positive slope is nondecreasing.

lemma (in int1) Int_ZF_2_4_L5:
assumes A1: f ∈ Sₜ and A2: m ∈ ℜ₊ and A3: m ≤ n
shows f⁻¹(m) ≤ f⁻¹(n)
proof -
  from A2 A3 have T: n ∈ ℜ₊ using Int_ZF_1_5_L7 by blast
  with A1 have n ≤ f(f⁻¹(n)) using Int_ZF_2_4_L2
    by simp
  with A3 have m ≤ f(f⁻¹(n)) by (rule Int_order_transitive)
  with A1 A2 T show f⁻¹(m) ≤ f⁻¹(n)
    using Int_ZF_1_5_L7A Int_ZF_2_4_L2 Int_ZF_2_4_L3 by simp
qed

If f⁻¹(m) is positive and n is a positive integer, then, then f⁻¹(m+n)−1 is positive.

lemma (in int1) Int_ZF_2_4_L6:
assumes A1: f ∈ Sₜ and A2: m ∈ ℜ₊ n ∈ ℜ₊ and A3: f⁻¹(m+1) ∈ ℜ₊
shows f⁻¹(m+n+1) ∈ ℜ₊
proof -
  from A1 A2 have f⁻¹(m+1) ≤ f⁻¹(m+n) + 1
    using PositiveSet_def Int_ZF_1_5_L7A Int_ZF_2_4_L2 Int_ZF_2_4_L5
    int_zero_one_are_int Int_ZF_1_1_L4 Int_ZF_1_2_L19
    int_ord_transl_inv by simp
  with A3 show f⁻¹(m+n+1) ∈ ℜ₊ using Int_ZF_1_5_L7
    by blast
qed

If f is a slope, then f(f⁻¹(m+n)−f⁻¹(m)−f⁻¹(n)) is uniformly bounded above and below. Will it be the messiest IsarMathLib proof ever? Only time will tell.

lemma (in int1) Int_ZF_2_4_L7:  assumes A1: f ∈ Sₜ and A2: ∀m ∈ ℜ₊. f⁻¹(m)−1 ∈ ℜ₊
shows ∃U ∈ ℜ₊. ∀m ∈ ℜ₊. ∀n ∈ ℜ₊. f(f⁻¹(m+n)−f⁻¹(m)−f⁻¹(n)) ≤ U
  ∃N ∈ ℜ₊. ∀m ∈ ℜ₊. ∀n ∈ ℜ₊. N ≤ f(f⁻¹(m+n)−f⁻¹(m)−f⁻¹(n))
proof -
  from A1 have ∃L ∈ ℜ₊. ∀r ∈ ℜ₊. f(r) ≤ f(r−1) + L

718
using Int_ZF_2_1_L28 by simp 

then obtain \( L \) where

I: \( L \in \mathbb{Z} \) and II: \( \forall r \in \mathbb{Z}. \ f(r) \leq f(r-1) + L \)

by auto

from A1 have

\[ \exists M \in \mathbb{Z}. \ \forall r \in \mathbb{Z}. \ \forall p \in \mathbb{Z}. \forall q \in \mathbb{Z}. \ f(r-p-q) \leq f(r)-f(p)-f(q)+M \]

\[ \exists K \in \mathbb{Z}. \ \forall r \in \mathbb{Z}. \ \forall p \in \mathbb{Z}. \forall q \in \mathbb{Z}. \ f(r-p-q) \leq f(r)-f(p)-f(q)+K \]

using Int_ZF_2_1_L30 by auto

then obtain \( M \) \( K \) where

III: \( M \in \mathbb{Z} \) and

IV: \( \forall r \in \mathbb{Z}. \forall p \in \mathbb{Z}. \forall q \in \mathbb{Z}. \ f(r-p-q) \leq f(r)-f(p)-f(q)+M \)

and

V: \( K \in \mathbb{Z} \) and VI: \( \forall r \in \mathbb{Z}. \forall p \in \mathbb{Z}. \forall q \in \mathbb{Z}. \ f(r-p-q) \leq f(r)-f(p)-f(q)+K \)

by auto

from I III V have

\( L+M \in \mathbb{Z} \) \( (-L) - L + K \in \mathbb{Z} \)

using Int_ZF_1_1_L4 Int_ZF_1_1_L5 by auto

moreover

\{ \ \{ \ \}

assume A3: \( m \in \mathbb{Z}_+ \) \( n \in \mathbb{Z}_+ \)

have \( f(f^{-1}(m)+f^{-1}(n)) \leq L+M \wedge (-L)-L+K \leq f(f^{-1}(m)+f^{-1}(n)) \)

proof -

let \( r = f^{-1}(m+n) \)

let \( p = f^{-1}(m) \)

let \( q = f^{-1}(n) \)

from A1 A3 have T1:

\( p \in \mathbb{Z}_+ \) \( q \in \mathbb{Z}_+ \) \( r \in \mathbb{Z}_+ \)

using Int_ZF_2_4_L2 pos_int_closed_add_unfolded by auto

with A3 have T2:

\( m \in \mathbb{Z} \) \( n \in \mathbb{Z} \) \( p \in \mathbb{Z} \) \( q \in \mathbb{Z} \) \( r \in \mathbb{Z} \)

using PositiveSet_def by auto

from A2 A3 have T3:

\( r-1 \in \mathbb{Z}_+ \) \( p-1 \in \mathbb{Z}_+ \) \( q-1 \in \mathbb{Z}_+ \)

using pos_int_closed_add_unfolded by auto

from A1 A3 have VII:

\( m+n \leq f(r) \)

\( m \leq f(p) \)

\( n \leq f(q) \)

using Int_ZF_2_4_L2 pos_int_closed_add_unfolded by auto

from A1 A3 T3 have VIII:

\( f(r-1) \leq m+n \)

\( f(p-1) \leq m \)

\( f(q-1) \leq n \)

using pos_int_closed_add_unfolded Int_ZF_2_4_L4 by auto

have \( f(r-p-q) \leq L+M \)

proof -

from IV T2 have \( f(r-p-q) \leq f(r)-f(p)-f(q)+M \)

by simp

moreover

719
from I II T2 VIII have 
  \( f(r) \leq f(r-1) + L \) 
  \( f(r-1) + L \leq m+n+L \) 
  using int_ord_transl_inv by auto
then have \( f(r) \leq m+n+L \)
  by (rule Int_order_transitive) 
with VII have \( f(r) - f(p) \leq m+n+L-m \)
  using int_ineq_add_sides by simp 
with I T2 have \( f(r) - f(p) - f(q) \leq n+L-n \)
  using Int_ZF_1_2_L9 int_ineq_add_sides by simp 
with I III T2 have \( f(r) - f(p) - f(q) + M \leq L+M \)
  using int_ord_transl_inv by simp 
ultimately show \( f(r-p-q) \leq L+M \)
  by (rule Int_order_transitive)
qed
moreover have \( (-L)-L +K \leq f(r-p-q) \)
proof - 
from I II T2 VIII have 
  \( f(p) \leq f(p-1) + L \) 
  \( f(p-1) + L \leq m +L \) 
  using int_ord_transl_inv by auto
then have \( f(p) \leq m +L \)
  by (rule Int_order_transitive) 
with VII have \( m+n -(m+L) \leq f(r) - f(p) \)
  using int_ineq_add_sides by simp 
with I T2 have \( n - L \leq f(r) - f(p) \)
  using Int_ZF_1_2_L9 by simp 
moreover 
from I II T2 VIII have 
  \( f(q) \leq f(q-1) + L \) 
  \( f(q-1) + L \leq n +L \) 
  using int_ord_transl_inv by auto
then have \( f(q) \leq n +L \)
  by (rule Int_order_transitive) 
ultimately have \( n - L - (n+L) \leq f(r) - f(p) - f(q) \)
  using int_ineq_add_sides by simp 
with V T2 have \( (-L)-L +K \leq f(r) - f(p) - f(q) + K \)
  using Int_ZF_1_2_L3 int_ord_transl_inv by simp 
moreover from VI T2 have 
  \( f(r) - f(p) - f(q) + K \leq f(r-p-q) \)
  by simp 
ultimately show \( (-L)-L +K \leq f(r-p-q) \)
  by (rule Int_order_transitive)
qed
ultimately show 
  \( f(r-p-q) \leq L+M \wedge \)
  \( (-L)+K \leq f(f^{-1}(m+n)-f^{-1}(m)-f^{-1}(n)) \)
proof

using Int_ZF_2_4_L2 PositiveSet_def

f \in \langle f, x \rangle 

proof

moreover have ... from A1

moreover from A1

have \exists U \in \mathbb{Z} \times \mathbb{Z}^+ : \{ x \in \mathbb{Z} : f(x) \leq a \} 

by auto

have \mathbb{Z}^+ \times \mathbb{Z}^+ \neq 0 \text{ using int_one_two_are_pos by auto}

moreover from A1 have f: \mathbb{Z} \rightarrow \mathbb{Z}

moreover from A1 have 

\forall a \in \mathbb{Z} \exists b \in \mathbb{Z} : \forall x. b \leq x \rightarrow a \leq f(x)

using Int_ZF_2_3_L5 by simp

moreover from A1 have 

\forall a \in \mathbb{Z} \exists b \in \mathbb{Z} : \forall y. b \leq y \rightarrow f(-y) \leq a

using Int_ZF_2_3_L5A by simp

moreover have 

\forall x \in \mathbb{Z}^+ \times \mathbb{Z}^+. \epsilon(f,x) \in \mathbb{Z} \land f(\epsilon(f,x)) \leq U \land N \leq f(\epsilon(f,x))

proof - 

\{ fix x assume A3: x \in \mathbb{Z}^+ \times \mathbb{Z}^+ 

let m = fst(x) 

let n = snd(x) \}

from A3 have T: m \in \mathbb{Z}^+ \land n \in \mathbb{Z}^+ \land m+n \in \mathbb{Z}^+

using pos_int_closed_add_unfolded by auto

with A1 have 

f^{-1}(m+n) \in \mathbb{Z} \land f^{-1}(m) \in \mathbb{Z} \land f^{-1}(n) \in \mathbb{Z}

using Int_ZF_2_4_L2 PositiveSet_def by auto

with I T have 

\epsilon(f,x) \in \mathbb{Z} \land f(\epsilon(f,x)) \leq U \land N \leq f(\epsilon(f,x))

using Int_ZF_1_1_L5 by auto

} thus thesis by simp

qed
ultimately show $\exists M. \forall x \in \mathbb{Z}_+ \times \mathbb{Z}_+. \text{abs}(\varepsilon(f,x)) \leq M$

by (rule Int_ZF_1_6_L4)

qed

The (candidate for) inverse of a positive slope is a (well defined) function
on $\mathbb{Z}_+$.

lemma (in int1) Int_ZF_2_4_L9:
assumes $A1: f \in S_+$ and $A2: g = \{(p,f^{-1}(p)). p \in \mathbb{Z}_+\}$
shows $g : \mathbb{Z}_+ \rightarrow \mathbb{Z}_+$

proof -
from A1 have $\forall p \in \mathbb{Z}_+. f^{-1}(p) \in \mathbb{Z}_+$
  $\forall p \in \mathbb{Z}_+. f^{-1}(p) \in \mathbb{Z}$
  using Int_ZF_2_4_L2 PositiveSet_def by auto
with A2 show $g : \mathbb{Z}_+ \rightarrow \mathbb{Z}_+$
  using Int_ZF_2_4_L9 by simp

qed

What are the values of the (candidate for) the inverse of a positive slope?

lemma (in int1) Int_ZF_2_4_L10:
assumes $A1: f \in S_+$ and $A2: g = \{(p,f^{-1}(p)). p \in \mathbb{Z}_+\}$ and $A3: p \in \mathbb{Z}_+$
shows $g(p) = f^{-1}(p)$

proof -
from A1 A2 have $g : \mathbb{Z}_+ \rightarrow \mathbb{Z}_+$ using Int_ZF_2_4_L9 by simp
with A2 A3 show $g(p) = f^{-1}(p)$ using ZF_fun_from_tot_val by simp

qed

The (candidate for) the inverse of a positive slope is a slope.

lemma (in int1) Int_ZF_2_4_L11:
assumes $A1: f \in S_+$ and $A2: \forall m \in \mathbb{Z}_+. f^{-1}(m) - 1 \in \mathbb{Z}_+$ and
$A3: g = \{(p,f^{-1}(p)). p \in \mathbb{Z}_+\}$
shows $\text{OddExtension}(\mathbb{Z}_+,\text{IntegerAddition},\text{IntegerOrder},g) \in S$

proof -
from A1 A2 have $\exists L. \forall x \in \mathbb{Z}_+ \times \mathbb{Z}_+. \text{abs}(\varepsilon(f,x)) \leq L$
  using Int_ZF_2_4_L8 by simp
then obtain L where I: $\forall x \in \mathbb{Z}_+ \times \mathbb{Z}_+. \text{abs}(\varepsilon(f,x)) \leq L$
  by auto
from A1 A3 have $g : \mathbb{Z}_+ \rightarrow \mathbb{Z}$ using Int_ZF_2_4_L9
  by simp
moreover have $\forall m \in \mathbb{Z}_+. \forall n \in \mathbb{Z}_+. \text{abs}(\delta(g,m,n)) \leq L$
  by (rule Int_ZF_1_6_L4)
proof-
{ fix m n
  assume A4: $m \in \mathbb{Z}_+ \quad n \in \mathbb{Z}_+$
  then have $\langle m,n \rangle \in \mathbb{Z}_+ \times \mathbb{Z}_+$ by simp
  with I have $\text{abs}(\varepsilon(f,\langle m,n \rangle)) \leq L$ by simp
  moreover have $\varepsilon(f,\langle m,n \rangle) = f^{-1}(mn) - f^{-1}(m) - f^{-1}(n)$

722
by simp
  moreover from A1 A3 A4 have
  \( f^{-1}(m+n) = g(m+n) \) \( f^{-1}(m) = g(m) \) \( f^{-1}(n) = g(n) \)
  using pos_int_closed_add_unfolded Int_ZF_2_4_L10 by auto
  ultimately have \( \text{abs}(\delta(g,m,n)) \leq L \) by simp
  } thus \( \forall m\in\mathbb{Z}_+. \ \forall n\in\mathbb{Z}_+. \ \text{abs}(\delta(g,m,n)) \leq L \) by simp
qed
ultimately have thesis by (rule Int_ZF_2_1_L24)
qed

Every positive slope that is at least 2 on positive integers almost has an inverse.

lemma (in int1) Int_ZF_2_4_L12: assumes A1: \( f \in S \) and
  A2: \( \forall m\in\mathbb{Z}_+. \ f^{-1}(m)-1 \in \mathbb{Z}_+ \)
shows \( \exists h \in S . \ f\circ h \sim \text{id}(\mathbb{Z}) \)
proof -
  let \( g = \{ (p,f^{-1}(p)) . \ p \in \mathbb{Z}_+ \} \)
  let \( h = \text{OddExtension}(\mathbb{Z}, \text{IntegerAddition}, \text{IntegerOrder}, g) \)
  from A1 have
  \( \exists M\in\mathbb{Z}. \ \forall n\in\mathbb{Z}. \ f(n) \leq f(n-1) + M \)
  using Int_ZF_2_1_L28 by simp
  then obtain M where
  I: \( M\in\mathbb{Z} \) and \( II: \forall n\in\mathbb{Z}. \ f(n) \leq f(n-1) + M \)
  by auto
  from A1 A2 have T: \( h \in S \)
  using Int_ZF_2_4_L11 by simp
  moreover have \( f\circ h \sim \text{id}(\mathbb{Z}) \)
  proof -
  from A1 T have \( f\circ h \in S \) using Int_ZF_2_1_L11
  by simp
  moreover note I
  moreover
  \{ fix \( m \) assume A3: \( m\in\mathbb{Z}_+ \)
  with A1 have \( f^{-1}(m) \in \mathbb{Z} \)
  using Int_ZF_2_4_L2 PositiveSet_def by simp
  with II have \( f(f^{-1}(m)) \leq f(f^{-1}(m)-1) + M \)
  by simp
  moreover from A1 A2 I A3 have \( f(f^{-1}(m)-1) + M \leq m+M \)
  using Int_ZF_2_4_L4 int_ord_transl_inv by simp
  ultimately have \( f(f^{-1}(m)) \leq m+M \)
  by (rule Int_order_transitive)
  moreover from A1 A3 have \( m \leq f(f^{-1}(m)) \)
  using Int_ZF_2_4_L2 by simp
  moreover from A1 A2 T A3 have \( f(f^{-1}(m)) = (f\circ h)(m) \)
  using Int_ZF_2_4_L9 Int_ZF_1_5_L11
  Int_ZF_2_4_L10 PositiveSet_def Int_ZF_2_1_L10
  by simp
  ultimately have \( m \leq (f\circ h)(m) \land (f\circ h)(m) \leq m+M \)
  by simp \}

723
ultimately show $f \circ h \sim \text{id}(Z)$ using Int_ZF_2_1_L32 
by simp 
qd
ultimately show $\exists h \in S. f \circ h \sim \text{id}(Z)$ 
by auto
qd

Int_ZF_2_4_L12 is almost what we need, except that it has an assumption that the values of the slope that we get the inverse for are not smaller than 2 on positive integers. The Arthan’s proof of Theorem 11 has a mistake where he says “note that for all but finitely many $m, n \in N$ $p = g(m)$ and $q = g(n)$ are both positive”. Of course there may be infinitely many pairs $(m, n)$ such that $p, q$ are not both positive. This is however easy to workaround: we just modify the slope by adding a constant so that the slope is large enough on positive integers and then look for the inverse.

theorem (in int1) pos_slope_has_inv: assumes A1: $f \in S_+$ shows $\exists g \in S. f \sim g \land (\exists h \in S. g \circ h \sim \text{id}(Z))$
proof -
  from A1 have $f: Z \rightarrow Z. \ 1 \in Z. \ 2 \in Z.$
    using AlmostHoms_def int_zero_one_are_int int_two_three_are_int
    by auto
moreover from A1 have $\forall a \in Z. \exists b \in Z_+. \forall x. \ b \leq x \longrightarrow a \leq f(x)$
    using Int_ZF_2_3_L5
    by simp
ultimately have $\exists c \in Z. \ 2 \leq \text{Minimum(IntegerOrder}, \{n \in Z_+. \ 1 \leq f(n)+c\})$
    by (rule Int_ZF_1_6_L7)
than obtain $c$ where I: $c \in Z$ and
II: $2 \leq \text{Minimum(IntegerOrder}, \{n \in Z_+. \ 1 \leq f(n)+c\})$
    by auto
let $g = \{<m,f(m)+c>. \ m \in Z\}$
from A1 I II have III: $g \in S$ and IV: $f \sim g$ using Int_ZF_2_1_L33
    by auto
from IV have $<f,g> \in \text{AlEqRel}$ by simp
with A1 have T: $g \in S_+$ by (rule Int_ZF_2_3_L9)
moreover have $\forall m \in Z_+. \ g^{-1}(m) - 1 \in Z_+$
proof
  fix $m$ assume A2: $m \in Z_+$
  from A1 I II have $V: \ 2 \leq g^{-1}(1)$
    using Int_ZF_2_1_L33 PositiveSet_def by simp
moreover from A2 T have $g^{-1}(1) \leq g^{-1}(m)$
    using Int_ZF_1_5_L3 int_one_two_are_pos Int_ZF_2_4_L5
    by simp
ultimately have $2 \leq g^{-1}(m)$
    by (rule Int_order_transitive)
then have $2-1 \leq g^{-1}(m)-1$
    using int_zero_one_are_int Int_ZF_1_1_L4 int_ord_transl_inv
    by simp

724
then show \( g^{-1}(m)-1 \in \mathbb{Z}_+ \)
using \( \text{Int_ZF}_1.2 \_L3 \ \text{Int_ZF}_1.5 \_L3 \)
by simp
qed
ultimately have \( \exists h \in S. \ g \circ h \sim \text{id}(\mathbb{Z}) \)
by (rule \( \text{Int_ZF}_2.4 \_L12 \))
with III IV show thesis by auto
qed

57.3 Completeness

In this section we consider properties of slopes that are needed for the proof of completeness of real numbers constructed in \( \text{Real}_2 \_1\). In particular we consider properties of embedding of integers into the set of slopes by the mapping \( m \mapsto m^S \), where \( m^S \) is defined by \( m^S(n) = m \cdot n \).

If \( m \) is an integer, then \( m^S \) is a slope whose value is \( m \cdot n \) for every integer.

**Lemma (in int1) \( \text{Int_ZF}_2.5 \_L1 \):** assumes \( A1: \ m \in \mathbb{Z} \)
shows
\( \forall n \in \mathbb{Z}. \ (m^S)(n) = m \cdot n \)
\( m^S \in S \)
proof -
from \( A1 \) have \( I: m^S: \mathbb{Z} \rightarrow \mathbb{Z} \)
using \( \text{Int}_2 \_1 \_1 \_L5 \ \text{ZF}_\_\_fun_from_total \) by simp
then show \( II: \forall n \in \mathbb{Z}. \ (m^S)(n) = m \cdot n \) using \( \text{ZF}_\_fun_from_tot_val \)
by simp
{ fix n k
assume \( A2: \ n \in \mathbb{Z} \ \ k \in \mathbb{Z} \)
with \( A1 \) have \( T: \ m \cdot n \in \mathbb{Z} \ \ m \cdot k \in \mathbb{Z} \)
using \( \text{Int}_2 \_1 \_1 \_L5 \) by auto
from \( A1 \ \ A2 \ \ T \) have \( \delta(m^S,n,k) = m \cdot k - m \cdot k \)
using \( \text{Int}_2 \_1 \_1 \_L5 \ \text{Int}_2 \_1 \_1 \_L1 \ \text{Int}_2 \_1 \_2 \_L3 \)
by simp
also from \( T \) have \( \ldots = 0 \) using \( \text{Int}_2 \_1 \_1 \_L4 \)
by simp
finally have \( \delta(m^S,n,k) = 0 \) by simp
then have \( \text{abs}(\delta(m^S,n,k)) \leq 0 \)
using \( \text{Int}_2 \_2 \_L18 \ \text{Int}_\_zero_one_\_are_int \ \text{Int}_\_ord_is_refl \ \text{refl}_\_def \)
by simp
} then have \( \forall n \in \mathbb{Z}. \forall k \in \mathbb{Z}. \ \text{abs}(\delta(m^S,n,k)) \leq 0 \)
by simp
with I show \( m^S \in S \) by (rule \( \text{Int}_2 \_1 \_1 \_L5 \))
qed

For any slope \( f \) there is an integer \( m \) such that there is some slope \( g \) that is almost equal to \( m^S \) and dominates \( f \) in the sense that \( f \leq g \) on positive integers (which implies that either \( g \) is almost equal to \( f \) or \( g - f \) is a positive slope. This will be used in \( \text{Real}_2 \_1\_\_thy \) to show that for any real number there is an integer that (whose real embedding) is greater or equal.
lemma (in int1) Int_ZF_2_5_L2: assumes \( f \in S \) shows \( \exists m \in \mathbb{Z}. \exists g \in S. (m^S \sim g \land (f \sim g \lor g + (-f) \in S^+) \)

proof -
  from A1 have
    \( \exists m, k. m \in \mathbb{Z} \land k \in \mathbb{Z} \land (\forall p \in \mathbb{Z}. \text{abs}(f(p)) \leq m \cdot \text{abs}(p) + k) \)
    using Arthan_Lem_8 by simp
  then obtain \( m, k \) where I: \( m \in \mathbb{Z} \) and II: \( k \in \mathbb{Z} \) and
    III: \( \forall p \in \mathbb{Z}. \text{abs}(f(p)) \leq m \cdot \text{abs}(p) + k \)
    by auto
  let \( g = \{ (n, m^S(n) + k). n \in \mathbb{Z} \} \)
  from I have \( IV: m^S \in S \) using Int_ZF_2_5_L1 by simp
  with II have \( V: g \in S \) and VI: \( m^S \sim g \) using Int_ZF_2_1_L33 by auto
  { fix \( n \) assume A2: \( n \in \mathbb{Z}_+ \)
    with A1 have \( f(n) \in \mathbb{Z} \)
      using Int_ZF_2_1_L2B PositiveSet_def by simp
    then have \( f(n) \leq \text{abs}(f(n)) \) using Int_ZF_2_L19C by simp
    moreover from III A2 have \( \text{abs}(f(n)) \leq m \cdot \text{abs}(n) + k \)
      using PositiveSet_def by simp
    with A2 have \( \text{abs}(f(n)) \leq m \cdot n + k \)
      using Int_ZF_1_5_L4A by simp
    ultimately have \( f(n) \leq g(n) \)
      by (rule Int_order_transitive)
    moreover from II IV A2 have \( g(n) = (m^S)(n) + k \)
      using Int_ZF_2_1_L33 PositiveSet_def by simp
    with I A2 have \( g(n) = m \cdot n + k \)
      using Int_ZF_2_5_L1 PositiveSet_def by simp
    ultimately have \( f(n) \leq g(n) \)
      by simp
  } then have \( \forall n \in \mathbb{Z}_+. f(n) \leq g(n) \)
    by simp
  with A1 V have \( f \sim g \lor g + (-f) \in S^+ \)
    using Int_ZF_2_3_L4C by simp
  with I V VI show thesis by auto
qed

The negative of an integer embeds in slopes as a negative of the original embedding.

lemma (in int1) Int_ZF_2_5_L3: assumes \( m \in \mathbb{Z} \) shows \( (-m)^S = -m^S \)

proof -
  from A1 have \( (-m)^S: \mathbb{Z} \rightarrow \mathbb{Z} \) and \( (-m^S)): \mathbb{Z} \rightarrow \mathbb{Z} \)
    using Int_ZF_1_1_L4 Int_ZF_2_5_L1 AlmostHoms_def Int_ZF_2_1_L12 by auto
  moreover have \( \forall n \in \mathbb{Z}. ((-m)^S)(n) = (-m^S)(n) \)
    proof

The sum of embeddings is the embedding of the sum.

**Lemma (in int1) Int_ZF_2_5_L3A:** assumes \( A1: m \in \mathbb{Z} \) \( k \in \mathbb{Z} \)
shows \( (m^S) + (k^S) = (m+k)^S \)
proof -
from \( A1 \) have \( T1: m+k \in \mathbb{Z} \) using Int_ZF_1_1_L5
by simp
with \( A1 \) have \( T2: 
(\exists S. (m^S) \in S \land (k^S) \in S) 
\land (m+k)^S \in S 
\land (m^S) + (k^S) \in S 
\land \neg (m+k)^S \in S 
\land Int_ZF_2_5_L1 \land Int_ZF_2_1_L12C \) by auto
then have \( (\exists S. (m^S) + (k^S)) : \mathbb{Z} \rightarrow \mathbb{Z} \)
\( (m+k)^S : \mathbb{Z} \rightarrow \mathbb{Z} \)
using AlmostHoms_def by auto
moreover have \( \forall n \in \mathbb{Z}. (m^S) + (k^S))(n) = ((m+k)^S)(n) \)
proof
fix \( n \) assume \( A2: n \in \mathbb{Z} \)
with \( A1 \) \( T1 \) \( T2 \) have \( (m^S) + (k^S))(n) = (m+k)n 
using Int_ZF_2_1_L12B Int_ZF_2_5_L1 Int_ZF_1_1_L1
by simp
also from \( T1 \) \( A2 \) have \( (m^S) + (k^S))(n) = ((m+k)^S)(n) 
using Int_ZF_2_5_L1 by simp
finally show \( (m^S) + (k^S))(n) = ((m+k)^S)(n) 
by simp
qed
ultimately show \( (m^S) + (k^S) = ((m+k)^S) \)
using fun_extension_iff by simp
qed

The composition of embeddings is the embedding of the product.

**Lemma (in int1) Int_ZF_2_5_L3B:** assumes \( A1: m \in \mathbb{Z} \) \( k \in \mathbb{Z} \)
shows \( (m^S) \circ (k^S) = ((m\cdot k)^S) \)
proof -
from \( A1 \) have \( T1: m \cdot k \in \mathbb{Z} \) using Int_ZF_1_1_L5
by simp

727
with A1 have T2:

\[(m^S) \in S \quad (k^S) \in S\]

\[(m \cdot k)^S \in S\]

using Int_ZF_2_5_L1 Int_ZF_2_1_L11 by auto

then have

\[(m^S) \circ (k^S) : \mathbb{Z} \rightarrow \mathbb{Z}\]

\[(m \cdot k)^S : \mathbb{Z} \rightarrow \mathbb{Z}\]

using AlmostHoms_def by auto

moreover have \(\forall n \in \mathbb{Z}. \ ((m^S) \circ (k^S))(n) = ((m \cdot k)^S)(n)\)

proof

fix \(n\) assume A2: \(n \in \mathbb{Z}\)

with A1 T2 have

\[((m^S) \circ (k^S))(n) = (m^S)(k \cdot n)\]

using Int_ZF_2_1_L10 Int_ZF_2_5_L1 by simp

moreover from A1 A2 have \(k \cdot n \in \mathbb{Z}\) using Int_ZF_1_1_L5 by simp

ultimately have \(((m^S) \circ (k^S))(n) = m \cdot k \cdot n\)

by simp

also from T1 A2 have \(m \cdot k \cdot n = ((m \cdot k)^S)(n)\)

using Int_ZF_2_5_L1 by simp

finally show \(((m^S) \circ (k^S))(n) = ((m \cdot k)^S)(n)\)

by simp

qed

ultimately show \(((m^S) \circ (k^S)) = ((m \cdot k)^S)\)

using fun_extension_iff by simp

qed

Embedding integers in slopes preserves order.

lemma (in int1) Int_ZF_2_5_L4: assumes A1: \(m \leq n\)

shows \((m^S) \sim (n^S) \lor (n^S) + (- (m^S)) \in S_+\)

proof -

from A1 have \(m^S \in S\) and \(n^S \in S\)

using Int_ZF_2_5_L1A by auto

moreover from A1 have \(\forall k \in \mathbb{Z}_+. \ (m^S)(k) \leq (n^S)(k)\)

using Int_ZF_2_3_L4C PositiveSet_def Int_ZF_2_5_L1 by simp

ultimately show thesis using Int_ZF_2_3_L4C by simp

qed

We aim at showing that \(m \mapsto m^S\) is an injection modulo the relation of almost equality. To do that we first show that if \(m^S\) has finite range, then \(m = 0\).

lemma (in int1) Int_ZF_2_5_L5:

assumes m\(\in\mathbb{Z}\) and \(m^S \in \text{FinRangeFunctions}(\mathbb{Z}, \mathbb{Z})\)

728
Embeddings of two integers are almost equal only if the integers are equal.

**Lemma (in int1) Int_ZF_2_5_L6:**

Assumes \( A1: m \in \mathbb{Z}, k \in \mathbb{Z} \) and \( A2: (m^S) \sim (k^S) \).

Shows \( m = k \)

**Proof:**

1. From \( A1 \) have \( T: m-k \in \mathbb{Z} \) using Int_ZF_1_1_L5 by simp
2. Using Int_ZF_2_5_L3 by simp
3. Then have \( m^S + (-k^S) = (m^S) + ((-k)^S) \) by simp
4. With \( A1 \) have \( m^S + (-k^S) = ((m-k)^S) \) using Int_ZF_1_1_L4 Int_ZF_2_5_L3A by simp
5. Moreover from \( A1 A2 \) have \( m^S + ((-k)^S) \in \text{FinRangeFunctions}(\mathbb{Z}, \mathbb{Z}) \)
6. Using Int_ZF_2_5_L1 Int_ZF_2_1_L9D by simp
7. Ultimately have \( (m-k)^S \in \text{FinRangeFunctions}(\mathbb{Z}, \mathbb{Z}) \) by simp
8. With \( T \) have \( m-k = 0 \) using Int_ZF_2_5_L5 by simp
9. With \( A1 \) show \( m = k \) by (rule Int_ZF_1_L15)

**QED**

Embedding of 1 is the identity slope and embedding of zero is a finite range function.

**Lemma (in int1) Int_ZF_2_5_L7:**

Shows

\( 1^S = \text{id}(\mathbb{Z}) \)
\( 0^S \in \text{FinRangeFunctions}(\mathbb{Z}, \mathbb{Z}) \)

**Proof:**

1. Have \( \text{id}(\mathbb{Z}) = \{(x,x). x \in \mathbb{Z}\} \) using id_def by blast
2. Then show \( 1^S = \text{id}(\mathbb{Z}) \) using Int_ZF_1_1_L4 by simp
3. Have \( \{0^S(n). n \in \mathbb{Z}\} = \{0.n. n \in \mathbb{Z}\} \)
   Using int_zero_one_are_int Int_ZF_2_5_L1 by simp
4. Also have \( \ldots = \{0\} \) using Int_ZF_1_1_L4 int_not_empty by simp
5. Finally have \( \{0^S(n). n \in \mathbb{Z}\} = \{0\} \) by simp
6. Then have \( \{0^S(n). n \in \mathbb{Z}\} \in \text{Fin}(\mathbb{Z}) \)
   Using int_zero_one_are_int Finite1_L16 by simp
7. Moreover have \( 0^S: \mathbb{Z} \rightarrow \mathbb{Z} \)
   Using int_zero_one_are_int Int_ZF_2_5_L1 AlmostHoms_def by simp
8. Ultimately show \( 0^S \in \text{FinRangeFunctions}(\mathbb{Z}, \mathbb{Z}) \)
   Using Finite1_L19 by simp

**QED**

A somewhat technical condition for a embedding of an integer to be "less or
Another technical condition for the composition of a slope and an integer (embedding) to be "less or equal" (in the sense appropriate for slopes) than embedding of another integer.

lemma (in int1) Int_ZF_2_5_L8:
  assumes A1: \( f \in S \) and A2: \( N \in \mathbb{Z} \) \( M \in \mathbb{Z} \) and 
  A3: \( \forall n\in\mathbb{Z}^+. \ M \cdot n \leq f(N \cdot n) \)
  shows \( M^S \sim f \circ (N^S) \vee (f \circ (N^S)) + (-(M^S)) \in S^+ \)

proof -
  from A1 A2 have \( M^S \in S \) \( f \circ (N^S) \in S \)
  using Int_ZF_2_5_L1 Int_ZF_2_1_L11 by auto
  moreover from A1 A2 A3 have \( \forall n\in\mathbb{Z}^+. \ (M^S)(n) \leq (f \circ (N^S))(n) \)
  using Int_ZF_2_5_L1 PositiveSet_def Int_ZF_2_1_L10 by simp
  ultimately show thesis using Int_ZF_2_3_L4C by simp
qed

Another technical condition for the composition of a slope and an integer (embedding) to be "less or equal" (in the sense appropriate for slopes) than embedding of another integer.

lemma (in int1) Int_ZF_2_5_L9:
  assumes A1: \( f \in S \) and A2: \( N \in \mathbb{Z} \) \( M \in \mathbb{Z} \) and 
  A3: \( \forall n\in\mathbb{Z}^+. \ f(N \cdot n) \leq M \cdot n \)
  shows \( f \circ (N^S) \sim (M^S) \vee (M^S) + (-(f \circ (N^S))) \in S^+ \)

proof -
  from A1 A2 have \( f \circ (N^S) \in S \) \( M^S \in S \)
  using Int_ZF_2_5_L1 Int_ZF_2_1_L11 by auto
  moreover from A1 A2 A3 have \( \forall n\in\mathbb{Z}^+. \ (f \circ (N^S))(n) \leq (M^S)(n) \)
  using Int_ZF_2_5_L1 PositiveSet_def Int_ZF_2_1_L10 by simp
  ultimately show thesis using Int_ZF_2_3_L4C by simp
qed

end

58  Construction real numbers - the generic part

theory Real_ZF imports Int_ZF_IML Ring_ZF_1

begin

The goal of the Real_ZF series of theory files is to provide a construction of the set of real numbers. There are several ways to construct real numbers. Most common start from the rational numbers and use Dedekind cuts or Cauchy sequences. Real_ZF_x.thy series formalizes an alternative approach that constructs real numbers directly from the group of integers. Our formalization is mostly based on [2]. Different variants of this construction are
also described in [1] and [3]. I recommend to read these papers, but for the impatient here is a short description: we take a set of maps \( s : \mathbb{Z} \to \mathbb{Z} \) such that the set \( \{ s(m + n) - s(m) - s(n) \}_{n,m \in \mathbb{Z}} \) is finite (\( \mathbb{Z} \) means the integers here). We call these maps slopes. Slopes form a group with the natural addition \((s + r)(n) = s(n) + r(n)\). The maps such that the set \( s(\mathbb{Z}) \) is finite (finite range functions) form a subgroup of slopes. The additive group of real numbers is defined as the quotient group of slopes by the (sub)group of finite range functions. The multiplication is defined as the projection of the composition of slopes into the resulting quotient (coset) space.

58.1 The definition of real numbers

This section contains the construction of the ring of real numbers as classes of slopes - integer almost homomorphisms. The real definitions are in \texttt{Group_ZF_2} theory, here we just specialize the definitions of almost homomorphisms, their equivalence and operations to the additive group of integers from the general case of abelian groups considered in \texttt{Group_ZF_2}.

The set of slopes is defined as the set of almost homomorphisms on the additive group of integers.

\begin{definition}
\text{Slopes} \equiv \text{AlmostHoms}(\mathbb{Z}, \mathbb{Z})
\end{definition}

The first operation on slopes (pointwise addition) is a special case of the first operation on almost homomorphisms.

\begin{definition}
\text{SlopeOp1} \equiv \text{AlHomOp1}(\mathbb{Z}, \mathbb{Z})
\end{definition}

The second operation on slopes (composition) is a special case of the second operation on almost homomorphisms.

\begin{definition}
\text{SlopeOp2} \equiv \text{AlHomOp2}(\mathbb{Z}, \mathbb{Z})
\end{definition}

Bounded integer maps are functions from integers to integers that have finite range. They play a role of zero in the set of real numbers we are constructing.

\begin{definition}
\text{BoundedIntMaps} \equiv \text{FinRangeFunctions}(\mathbb{Z}, \mathbb{Z})
\end{definition}

Bounded integer maps form a normal subgroup of slopes. The equivalence relation on slopes is the (group) quotient relation defined by this subgroup.

\begin{definition}
\text{SlopeEquivalenceRel} \equiv \text{QuotientGroupRel}(\text{Slopes}, \text{SlopeOp1}, \text{BoundedIntMaps})
\end{definition}

The set of real numbers is the set of equivalence classes of slopes.

\begin{definition}
\text{RealNumbers}
\end{definition}
\textbf{RealNumbers} \equiv \text{Slopes} /\!\!/ \text{SlopeEquivalenceRel}

The addition on real numbers is defined as the projection of pointwise addition of slopes on the quotient. This means that the additive group of real numbers is the quotient group: the group of slopes (with pointwise addition) defined by the normal subgroup of bounded integer maps.

\textbf{definition}
\textbf{RealAddition} \equiv \text{ProjFun2(Slopes, SlopeEquivalenceRel, SlopeOp1)}

Multiplication is defined as the projection of composition of slopes on the quotient. The fact that it works is probably the most surprising part of the construction.

\textbf{definition}
\textbf{RealMultiplication} \equiv \text{ProjFun2(Slopes, SlopeEquivalenceRel, SlopeOp2)}

We first show that we can use theorems proven in some proof contexts (locales). The locale \texttt{group1} requires assumption that we deal with an abelian group. The next lemma allows to use all theorems proven in the context called \texttt{group1}.

\textbf{lemma} \texttt{Real_ZF_1_L1:} \textbf{shows} \texttt{group1(int, IntegerAddition)}
\textbf{using} \texttt{group1_axioms.intro group1_def Int_ZF_1_T2} \textbf{by} \texttt{simp}

Real numbers form a ring. This is a special case of the theorem proven in \texttt{Ring_ZF_1.thy}, where we show the same in general for almost homomorphisms rather than slopes.

\textbf{theorem} \texttt{Real_ZF_1_T1:} \textbf{shows} \texttt{IsAring(RealNumbers, RealAddition, RealMultiplication)}
\textbf{proof} -
\begin{itemize}
\item let \texttt{AH} = \texttt{AlmostHoms(int, IntegerAddition)}
\item let \texttt{Op1} = \texttt{AlHomOp1(int, IntegerAddition)}
\item let \texttt{FR} = \texttt{FinRangeFunctions(int, int)}
\item let \texttt{Op2} = \texttt{AlHomOp2(int, IntegerAddition)}
\item let \texttt{R} = \texttt{QuotientGroupRel(AH, Op1, FR)}
\item let \texttt{A} = \texttt{ProjFun2(AH, R, Op1)}
\item let \texttt{M} = \texttt{ProjFun2(AH, R, Op2)}
\item have \texttt{IsAring(AH//R, A, M)} using \texttt{Real_ZF_1_L1 group1.Ring_ZF_1_1_T1} \textbf{by} \texttt{simp}
\item then show thesis using \texttt{Slopes_def SlopeOp2_def SlopeOp1_def BoundedIntMaps_def SlopeEquivalenceRel_def RealNumbers_def RealAddition_def RealMultiplication_def} \textbf{by} \texttt{simp}
\end{itemize}
\textbf{qed}

We can use theorems proven in \texttt{group0} and \texttt{group1} contexts applied to the group of real numbers.

\textbf{lemma} \texttt{Real_ZF_1_L2:} \textbf{shows}
\begin{itemize}
\item \texttt{group0(RealNumbers, RealAddition)}
\item \texttt{RealAddition (is commutative on) RealNumbers}
\item \texttt{group1(RealNumbers, RealAddition)}
\end{itemize}
proof -

have IsAgroup(RealNumbers,RealAddition)
  RealAddition {is commutative on} RealNumbers
  using Real_ZF_1_T1 IsAring_def by auto
then show group0(RealNumbers,RealAddition)
  RealAddition {is commutative on} RealNumbers
  group1(RealNumbers,RealAddition)
  using group1_axioms.intro group0_def group1_def by auto
qed

Let's define some notation.

locale real0 =

  fixes real (R)
  defines real_def [simp]: R ≡ RealNumbers

  fixes ra (infixl + 69)
  defines ra_def [simp]: a + b ≡ RealAddition(a,b)

  fixes rminus (- _ 72)
  defines rminus_def [simp]: -a ≡ GroupInv(R,RealAddition)(a)

  fixes rsub (infixl - 69)
  defines rsub_def [simp]: a - b ≡ a + (-b)

  fixes rm (infixl · 70)
  defines rm_def [simp]: a · b ≡ RealMultiplication(a,b)

  fixes rzero (0)
  defines rzero_def [simp]:
    0 ≡ TheNeutralElement(RealNumbers,RealAddition)

  fixes rone (1)
  defines rone_def [simp]:
    1 ≡ TheNeutralElement(RealNumbers,RealMultiplication)

  fixes rtwo (2)
  defines rtwo_def [simp]: 2 ≡ 1 + 1

  fixes non_zero (R₀)
  defines non_zero_def[simp]: R₀ ≡ R - {0}

  fixes inv (_⁻¹ [90] 91)
  defines inv_def[simp]:
    a⁻¹ ≡ GroupInv(R₀,restrict(RealMultiplication,R₀×R₀))(a)

In real0 context all theorems proven in the ring0 context are valid.
lemma (in real0) Real_ZF_1_L3: shows ring0(R,RealAddition,RealMultiplication) using Real_ZF_1_T1 ring0_def ring0.Ring_ZF_1_L1 by auto

Let's try out our notation to see that zero and one are real numbers.

lemma (in real0) Real_ZF_1_L4: shows 0∈R 1∈R using Real_ZF_1_L3 ring0.Ring_ZF_1_L2 by auto

The lemma below lists some properties that require one real number to state.

lemma (in real0) Real_ZF_1_L5: assumes A1: a∈R shows (-a) ∈ R (-(-a)) = a a+0 = a 0+a = a a·1 = a 1·a = a a-a = 0 a-0 = a using assms Real_ZF_1_L3 ring0.Ring_ZF_1_L3 by auto

The lemma below lists some properties that require two real numbers to state.

lemma (in real0) Real_ZF_1_L6: assumes a∈R b∈R shows a+b ∈ R a-b ∈ R a·b ∈ R a+b = b+a (-a)·b = -(a·b) a·(-b) = -(a·b) using assms Real_ZF_1_L3 ring0.Ring_ZF_1_L4 ring0.Ring_ZF_1_L7 by auto

Multiplication of reals is associative.

lemma (in real0) Real_ZF_1_L6A: assumes a∈R b∈R c∈R shows a·(b·c) = (a·b)·c using assms Real_ZF_1_L3 ring0.Ring_ZF_1_L11 by simp

Addition is distributive with respect to multiplication.

lemma (in real0) Real_ZF_1_L7: assumes a∈R b∈R c∈R shows a·(b+c) = a·b + a·c (b+c)·a = b·a + c·a a·(b-c) = a·b - a·c (b-c)·a = b·a - c·a

734
using assms Real_ZF_1_L3 ring0.ring_oper_distr ring0.Ring_ZF_1_L8
by auto

A simple rearrangement with four real numbers.

lemma (in real0) Real_ZF_1_L7A:
assumes a∈R b∈R c∈R d∈R
shows a-b + (c-d) = a+c-b-d
using assms Real_ZF_1_L2 group0.group0_4_L8A by simp

RealAddition is defined as the projection of the first operation on slopes
(that is, slope addition) on the quotient (slopes divided by the "almost
equal" relation. The next lemma plays with definitions to show that this
is the same as the operation induced on the appropriate quotient group.
The names AH, Op1 and FR are used in group1 context to denote almost
homomorphisms, the first operation on AH and finite range functions resp.

lemma Real_ZF_1_L8: assumes
AH = AlmostHoms(int,IntegerAddition) and
Op1 = AlHomOp1(int,IntegerAddition) and
FR = FinRangeFunctions(int,int)
shows RealAddition = QuotientGroupOp(AH,Op1,FR)
using assms RealAddition_def SlopeEquivalenceRel_def
QuotientGroupOp_def Slopes_def SlopeOp1_def BoundedIntMaps_def
by simp

The symbol 0 in the real0 context is defined as the neutral element of real
addition. The next lemma shows that this is the same as the neutral element
of the appropriate quotient group.

lemma (in real0) Real_ZF_1_L9: assumes
AH = AlmostHoms(int,IntegerAddition) and
Op1 = AlHomOp1(int,IntegerAddition) and
FR = FinRangeFunctions(int,int) and
r = QuotientGroupRel(AH,Op1,FR)
shows TheNeutralElement(AH//r,QuotientGroupOp(AH,Op1,FR)) = 0
SlopeEquivalenceRel = r
using assms Slopes_def Real_ZF_1_L8 RealNumbers_def
SlopeEquivalenceRel_def SlopeOp1_def BoundedIntMaps_def
by auto

Zero is the class of any finite range function.

lemma (in real0) Real_ZF_1_L10: assumes A1: s ∈ Slopes
shows SlopeEquivalenceRel(s) = 0 ↔ s ∈ BoundedIntMaps
proof -
let AH = AlmostHoms(int,IntegerAddition)
let Op1 = AlHomOp1(int,IntegerAddition)
let FR = FinRangeFunctions(int,int)
let \( r = \text{QuotientGroupRel}(AH,Op1,FR) \)
let \( e = \text{TheNeutralElement}(AH//r,\text{QuotientGroupOp}(AH,Op1,FR)) \)
from A1 have
  group1(int,IntegerAddition)
s \in AH
  using Real_ZF_1_L1 Slopes_def
by auto
then have \( r\{s\} = e \iff s \in FR \)
  using group1.Group_ZF_3_3_L5 by simp
moreover have
  \( r = \text{SlopeEquivalenceRel} \)
e = 0
FR = BoundedIntMaps
  using SlopeEquivalenceRel_def Slopes_def SlopeOp1_def
BoundedIntMaps_def Real_ZF_1_L9 by auto
ultimately show thesis by simp
qed

We will need a couple of results from Group_ZF_3.thy The first two that state that the definition of addition and multiplication of real numbers are consistent, that is the result does not depend on the choice of the slopes representing the numbers. The second one implies that what we call SlopeEquivalenceRel is actually an equivalence relation on the set of slopes. We also show that the neutral element of the multiplicative operation on reals (in short number 1) is the class of the identity function on integers.

lemma Real_ZF_1_L11: shows
  Congruent2(SlopeEquivalenceRel,SlopeOp1)
  Congruent2(SlopeEquivalenceRel,SlopeOp2)
  SlopeEquivalenceRel \( \subseteq \) Slopes \( \times \) Slopes
equiv(Slopes, SlopeEquivalenceRel)
  SlopeEquivalenceRel(id(int)) =
  TheNeutralElement(RealNumbers,RealMultiplication)
  BoundedIntMaps \( \subseteq \) Slopes
proof -
  let \( G = \) int
  let \( f = \) IntegerAddition
  let \( AH = \) AlmostHoms(int,IntegerAddition)
  let \( Op1 = \) AlHomOp1(int,IntegerAddition)
  let \( Op2 = \) AlHomOp2(int,IntegerAddition)
  let \( FR = \) FinRangeFunctions(int,int)
  let \( R = \) QuotientGroupRel(int,int)
  have
    Congruent2(R,Op1)
    Congruent2(R,Op2)
    using Real_ZF_1_L1 group1.Group_ZF_3_4_L13A group1.Group_ZF_3_3_L4
by auto
then show
  Congruent2(SlopeEquivalenceRel,SlopeOp1)
Congruent2(SlopeEquivalenceRel,SlopeOp2)
using SlopeEquivalenceRel_def SlopeOp1_def Slopes_def
BoundedIntMaps_def SlopeOp2_def by auto
have have equiv(AH,R)
  using Real_ZF_1_L1 group1.Group_ZF_3_3_L3 by simp
then show equiv(Slopes,SlopeEquivalenceRel)
  using BoundedIntMaps_def SlopeEquivalenceRel_def SlopeOp1_def Slopes_def
by simp
then show SlopeEquivalenceRel ⊆ Slopes × Slopes
  using equiv_type by simp
have R{id(int)} = TheNeutralElement(AH//R,ProjFun2(AH,R,Op2))
  using Real_ZF_1_L11 Real_ZF_1_L10 by auto
then show SlopeEquivalenceRel{id(int)} = TheNeutralElement(RealNumbers,RealMultiplication)
  using Slopes_def RealNumbers_def
SlopeEquivalenceRel_def SlopeOp1_def BoundedIntMaps_def
RealMultiplication_def SlopeOp2_def by simp
have FR ⊆ AH using Real_ZF_1_L1 group1.Group_ZF_3_3_L1
  by simp
then show BoundedIntMaps ⊆ Slopes
  using BoundedIntMaps_def Slopes_def by simp
qed

A one-side implication of the equivalence from Real_ZF_1_L10: the class of a bounded integer map is the real zero.

lemma (in real0) Real_ZF_1_L11A: assumes s ∈ BoundedIntMaps
shows SlopeEquivalenceRel(s) = 0
  using assms Real_ZF_1_L11 Real_ZF_1_L10 by auto

The next lemma is rephrases the result from Group_ZF_3.thy that says that the negative (the group inverse with respect to real addition) of the class of a slope is the class of that slope composed with the integer additive group inverse. The result and proof is not very readable as we use mostly generic set theory notation with long names here. Real_ZF_1.thy contains the same statement written in a more readable notation: \([-s] = -[s]\).

lemma (in real0) Real_ZF_1_L12: assumes A1: s ∈ Slopes and
Dr: r = QuotientGroupRel(Slopes,SlopeOp1,BoundedIntMaps)
shows r{GroupInv(int,IntegerAddition) 0 s} = -(r{s})

proof -
  let G = int
  let f = IntegerAddition
  let AH = AlmostHoms(int,IntegerAddition)
  let Op1 = AlHomOp1(int,IntegerAddition)
  let FR = FinRangeFunctions(int,int)
  let F = ProjFun2(Slopes,r,SlopeOp1)
  from A1 Dr have
group1(G, f)
s ∈ AlmostHoms(G, f)

r = QuotientGroupRel(
  AlmostHoms(G, f), AlHomOp1(G, f), FinRangeFunctions(G, G))

and F = ProjFun2(AlmostHoms(G, f), r, AlHomOp1(G, f))

using Real_ZF_1_L1 Slopes_def SlopeOp1_def BoundedIntMaps_def
by auto

then have
  r{GroupInv(G, f) O s} =
    GroupInv(AlmostHoms(G, f) // r, F)(r {s})
using group1.Group_ZF_3_3_L6
by simp

with Dr show thesis
  using RealNumbers_def Slopes_def SlopeEquivalenceRel_def RealAddition_def
  by simp

qed

Two classes are equal iff the slopes that represent them are almost equal.

lemma Real_ZF_1_L13: assumes s ∈ Slopes p ∈ Slopes
  and r = SlopeEquivalenceRel
  shows r{s} = r{p} ↔ ⟨s,p⟩ ∈ r
using assms Real_ZF_1_L11 eq_equiv_class equiv_class_eq
by blast

Identity function on integers is a slope. This lemma concludes the easy part of the construction that follows from the fact that slope equivalence classes form a ring. It is easy to see that multiplication of classes of almost homomorphisms is not commutative in general. The remaining properties of real numbers, like commutativity of multiplication and the existence of multiplicative inverses have to be proven using properties of the group of integers, rather that in general setting of abelian groups. This is done in the Real_ZF_1 theory.

lemma Real_ZF_1_L14: shows id(int) ∈ Slopes
proof -
  have id(int) ∈ AlmostHoms(int,IntegerAddition)
    using Real_ZF_1_L1 group1.Group_ZF_3_4_L15
    by simp
  then show thesis using Slopes_def by simp
qed

end

59 Construction of real numbers

theory Real_ZF_1 imports Real_ZF Int_ZF_3 OrderedField_ZF

begin

In this theory file we continue the construction of real numbers started in Real_ZF to a successful conclusion. We put here those parts of the construc-

738
tion that can not be done in the general settings of abelian groups and require integers.

59.1 Definitions and notation

In this section we define notions and notation needed for the rest of the construction.

We define positive slopes as those that take an infinite number of positive values on the positive integers (see Int_ZF_2 for properties of positive slopes).

**definition**

PositiveSlopes ≡ \{s ∈ Slopes. \(s(\text{PositiveIntegers}) \cap \text{PositiveIntegers} \neq \text{Fin(int)}\}\}

The order on the set of real numbers is constructed by specifying the set of positive reals. This set is defined as the projection of the set of positive slopes.

**definition**

PositiveReals ≡ \{\text{SlopeEquivalenceRel}\{s\}. s ∈ PositiveSlopes\}

The order relation on real numbers is constructed from the set of positive elements in a standard way (see section ”Alternative definitions” in OrderedGroup_ZF.)

**definition**

OrderOnReals ≡ OrderFromPosSet(RealNumbers,RealAddition,PositiveReals)

The next locale extends the locale real0 to define notation specific to the construction of real numbers. The notation follows the one defined in Int_ZF_2.thy. If \(m\) is an integer, then the real number which is the class of the slope \(n \mapsto m \cdot n\) is denoted \(m^R\). For a real number \(a\) notation \([a]\) means the largest integer \(m\) such that the real version of it (that is, \(m^R\)) is not greater than \(a\). For an integer \(m\) and a subset of reals \(S\) the expression \(\Gamma(S,m)\) is defined as \(\max\{[p^R \cdot x] : x \in S\}\). This is plays a role in the proof of completeness of real numbers. We also reuse some notation defined in the int0 context, like \(\mathbb{Z}_+\) (the set of positive integers) and \(\text{abs}(m)\) (the absolute value of an integer, and some defined in the int1 context, like the addition (+) and composition (\(\circ\) of slopes).

locale real1 = real0 +

fixes A1Eq (infix ~ 68)
defines A1Eq_def[simp]: \(s \sim r \equiv (s,r) \in \text{SlopeEquivalenceRel}\)

fixes slope_add (infix + 70)
defines slope_add_def[simp]: \(s + r \equiv \text{SlopeOp1}(s,r)\)
fixes slope_comp (infix \circ 71)
defines slope_comp_def[simp]: s \circ r \equiv \text{SlopeOp2}(s,r)

fixes slopes (S)
defines slopes_def[simp]: S \equiv \text{AlmostHoms}(\text{int},\text{IntegerAddition})

fixes posslopes (S_+)
defines posslopes_def[simp]: S_+ \equiv \text{PositiveSlopes}

fixes slope_class ([ _ ])
defines slope_class_def[simp]: [f] \equiv \text{SlopeEquivalenceRel}\{f\}

fixes slope_neg (-_ [90] 91)
defines slope_neg_def[simp]: -s \equiv \text{GroupInv}(\text{int},\text{IntegerAddition}) \circ s

fixes lesseqr (infix \leq 60)
defines lesseqr_def[simp]: a \leq b \equiv \langle a,b \rangle \in \text{OrderOnReals}

fixes sless (infix < 60)
defines sless_def[simp]: a < b \equiv a \leq b \land a \neq b

fixes positivereals (R_+)
defines positivereals_def[simp]: R_+ \equiv \text{PositiveSet}(\text{R},\text{RealAddition},\text{OrderOnReals})

fixes intembed (_R [90] 91)
defines intembed_def[simp]: m_R \equiv \{\langle n,\text{IntegerMultiplication}(m,n) \rangle. n \in \text{int}\}

fixes floor (\lfloor _ \rfloor)
defines floor_def[simp]: \lfloor a \rfloor \equiv \text{Maximum}(\text{IntegerOrder},\{m \in \text{int}. m_R \leq a\})

fixes \Gamma
defines \Gamma\_def[simp]: \Gamma(S,p) \equiv \text{Maximum}(\text{IntegerOrder},\{p^R.x. x \in S\})

fixes ia (infixl + 69)
defines ia_def[simp]: a+b \equiv \text{IntegerAddition}(a,b)

fixes iminus (-_ [72])
defines iminus_def[simp]: -a \equiv \text{GroupInv}(\text{int},\text{IntegerAddition})(a)

fixes isub (infixl - 69)
defines isub_def[simp]: a-b \equiv a - (\neg b)

fixes intpositives (Z_+)
defines intpositives_def[simp]: Z_+ \equiv \text{PositiveSet}(\text{int},\text{IntegerAddition},\text{IntegerOrder})
**59.2 Multiplication of real numbers**

Multiplication of real numbers is defined as a projection of composition of slopes onto the space of equivalence classes of slopes. Thus, the product of the real numbers given as classes of slopes \( s \) and \( r \) is defined as the class of \( s \circ r \). The goal of this section is to show that multiplication defined this way is commutative.

Let’s recall a theorem from Int_ZF_2.thy that states that if \( f, g \) are slopes, then \( f \circ g \) is equivalent to \( g \circ f \). Here we conclude from that that the classes of \( f \circ g \) and \( g \circ f \) are the same.

**lemma (in real1) Real_ZF_1_1_L2:** assumes \( f \in S \) \( g \in S \) shows \([f \circ g] = [g \circ f]\)

**proof**
- from A1 have \( f \circ g \sim g \circ f \)
  - using Slopes_def int1.Arthan_Th_9 SlopeOp1_def BoundedIntMaps_def SlopeEquivalenceRel_def SlopeOp2_def by simp
  - then show thesis using Real_ZF_1_L11 equiv_class_eq by simp

**qed**

Classes of slopes are real numbers.

**lemma (in real1) Real_ZF_1_1_L3:** assumes \( f \in S \) shows \([f] \in R\)

**proof**

...
from A1 have \([f] \in \text{Slopes}//\text{SlopeEquivalenceRel}\)
using \text{Slopes_def} \text{quotientI} \text{by simp}
then show \([f] \in R\) using \text{RealNumbers_def} \text{by simp}
qed

Each real number is a class of a slope.

**lemma (in real1) Real_ZF_1_1_L3A:** assumes A1: \(a \in R\)
shows \(\exists f \in S . \ a = [f]\)
proof -
  from A1 have \(a \in S//\text{SlopeEquivalenceRel}\)
  using \text{RealNumbers_def} \text{Slopes_def} \text{by simp}
  then show thesis using \text{quotient_def} \text{by simp}
qed

It is useful to have the definition of addition and multiplication in the \texttt{real1}
context notation.

**lemma (in real1) Real_ZF_1_1_L4:** assumes A1: \(f \in S\ g \in S\)
shows \([f] + [g] = [f+g]\)
\([f] \cdot [g] = [f \circ g]\)
proof -
  let \(r = \text{SlopeEquivalenceRel}\)
  have \([f] \cdot [g] = \text{ProjFun2}(S,r,\text{SlopeOp2})(\langle f, g \rangle)\)
  using \text{RealMultiplication_def} \text{Slopes_def} \text{by simp}
  also from A1 have ... = \([f \circ g]\)
  using \text{Real_ZF_1_L11} \text{EquivClass_1_L10} \text{Slopes_def} \text{by simp}
  finally show \([f] \cdot [g] = [f \circ g]\) \text{by simp}
  have \([f] + [g] = \text{ProjFun2}(S,r,\text{SlopeOp1})(\langle f, g \rangle)\)
  using \text{RealAddition_def} \text{Slopes_def} \text{by simp}
  also from A1 have ... = \([f+g]\)
  using \text{Real_ZF_1_L11} \text{EquivClass_1_L10} \text{Slopes_def} \text{by simp}
  finally show \([f] + [g] = [f+g]\) \text{by simp}
qed

The next lemma is essentially the same as \texttt{Real_ZF_1_1_L12}, but written in the
notation defined in the \texttt{real1} context. It states that if \(f\) is a slope, then
\(-[f] = [-f]\).

**lemma (in real1) Real_ZF_1_1_L4A:** assumes f \(\in S\)
shows \([-f] = [-f]\)
using assms \text{Slopes_def} \text{SlopeEquivalenceRel_def} \text{Real_ZF_1_L12} \text{by simp}

Subtracting real numbers corresops to adding the opposite slope.

**lemma (in real1) Real_ZF_1_1_L4B:** assumes A1: \(f \in S\ g \in S\)
shows \([f] - [g] = [f+(-g)]\)

proof -

from \(A1\) have \([f+(-g)] = [f] + [-g]\)
using Slopes_def BoundedIntMaps_def int1.Int_ZF_2_1_L12

Real_ZF_1_1_L4 by simp

with \(A1\) show \([f] - [g] = [f+(-g)]\)
using Real_ZF_1_1_L4A by simp

qed

Multiplication of real numbers is commutative.

theorem (in real1) real_mult_commute: assumes \(A1\): \(a \in R \quad b \in R\)
shows \(a \cdot b = b \cdot a\)
proof -

from \(A1\) have \(\exists f \in S . \ a = [f]\)
\(\exists g \in S . \ b = [g]\)
using Real_ZF_1_1_L3A by auto

then obtain \(f \ g\) where
\(f \in S \quad g \in S\) and \(a = [f]\) \(b = [g]\)
by auto

then show \(a \cdot b = b \cdot a\)
using Real_ZF_1_1_L4 Real_ZF_1_1_L2 by simp

qed

Multiplication is commutative on reals.

lemma real_mult_commutative: shows RealMultiplication \{is commutative on\} RealNumbers
using real1.real_mult_commute IsCommutative_def
by simp

The neutral element of multiplication of reals (denoted as \(1\) in the real1 context) is the class of identity function on integers. This is really shown in Real_ZF_1_L11, here we only rewrite it in the notation used in the real1 context.

lemma (in real1) real_one_cl_identity: shows \([id(int)] = 1\)
using Real_ZF_1_L11 by simp

If \(f\) is bounded, then its class is the neutral element of additive operation on reals (denoted as \(0\) in the real1 context).

lemma (in real1) real_zero_cl_bounded_map: assumes \(f \in BoundedIntMaps\) shows \([f] = 0\)
using assms Real_ZF_1_L11A by simp

Two real numbers are equal iff the slopes that represent them are almost equal. This is proven in Real_ZF_1_L13, here we just rewrite it in the notation used in the real1 context.

lemma (in real1) Real_ZF_1_1_L5:
assumes \( f \in S \) \( g \in S \)
shows [\( f \)] = [\( g \)] \( \iff f \sim g \)
using assms Slopes_def Real_ZF_1_L13 by simp

If the pair of function belongs to the slope equivalence relation, then their classes are equal. This is convenient, because we don’t need to assume that \( f, g \) are slopes (follows from the fact that \( f \sim g \)).

**Lemma (in real1) Real_ZF_1_1_L5A:** \( f \sim g \)
shows [\( f \)] = [\( g \)]
using assms Real_ZF_1_L11 Slopes_def Real_ZF_1_1_L5 by auto

Identity function on integers is a slope. This is proven in Real_ZF_1_L13, here we just rewrite it in the notation used in the real1 context.

**Lemma (in real1) id_on_int_is_slope:** \( \text{id(int)} \in S \)
using Real_ZF_1_L14 Slopes_def by simp

A result from Int_ZF_2.thy: the identity function on integers is not almost equal to any bounded function.

**Lemma (in real1) Real_ZF_1_1_L7:**
assumes \( A1: \ f \in \text{BoundedIntMaps} \)
shows \( \neg(\text{id(int)} \sim f) \)
using assms Slopes_def SlopeOp1_def BoundedIntMaps_def SlopeEquivalenceRel_def BoundedIntMaps_def int1.Int_ZF_2_3_L12 by simp

Zero is not one.

**Lemma (in real1) real_zero_not_one:** \( 1 \neq 0 \)
proof -
\[
\begin{align*}
& \{ \text{assume } A1: \ 1=0 \\
& \quad \text{have } \exists f \in S. \ 0 = [f] \\
& \quad \quad \text{using Real_ZF_1_L4 Real_ZF_1_1_L3A by simp} \\
& \quad \quad \text{with A1 have} \\
& \quad \quad \quad \exists f \in S. \ [\text{id(int)}] = [f] \land [f] = 0 \\
& \quad \quad \quad \text{using real_one_cl_identity by auto} \\
& \quad \quad \quad \text{then have False using Real_ZF_1_1_L5 Slopes_def} \\
& \quad \quad \quad \text{Real_ZF_1_L10 Real_ZF_1_1_L7 id_on_int_is_slope} \\
& \quad \quad \quad \text{by auto} \\
& \} \text{ then show } 1 \neq 0 \text{ by auto} \\
\end{align*}
\]
qed

Negative of a real number is a real number. Property of groups.

**Lemma (in real1) Real_ZF_1_1_L8:** \( a \in \mathbb{R} \)
\( (-a) \in \mathbb{R} \)
using assms Real_ZF_1_L2 group0.inverse_in_group by simp

An identity with three real numbers.

**Lemma (in real1) Real_ZF_1_1_L9:** \( a \in \mathbb{R} \) \( b \in \mathbb{R} \) \( c \in \mathbb{R} \)
shows $a \cdot (b \cdot c) = a \cdot c \cdot b$

using assms real_mult_commutative Real_ZF_1_L3 ring0.Ring_ZF_2_L4
by simp

59.3 The order on reals

In this section we show that the order relation defined by prescribing the
set of positive reals as the projection of the set of positive slopes makes the
ring of real numbers into an ordered ring. We also collect the facts about
ordered groups and rings that we use in the construction.

Positive slopes are slopes and positive reals are real.

lemma Real_ZF_1_2_L1: shows
\[\text{PositiveSlopes} \subseteq \text{Slopes} \quad \text{PositiveReals} \subseteq \text{RealNumbers}\]
proof -
  have \[\text{PositiveSlopes} = \{s \in \text{Slopes}. s(\text{PositiveIntegers}) \cap \text{PositiveIntegers} \notin \text{Fin(int)}\}\]
    using PositiveSlopes_def by simp
  then show \[\text{PositiveSlopes} \subseteq \text{Slopes}\] by (rule subset_with_property)
  then have \[\{\text{SlopeEquivalenceRel}\{s\}. s \in \text{PositiveSlopes}\} \subseteq \text{Slopes}/\text{SlopeEquivalenceRel}\]
    using EquivClass_1_L1A by simp
  then show \[\text{PositiveReals} \subseteq \text{RealNumbers}\]
    using PositiveReals_def RealNumbers_def by simp
qed

Positive reals are the same as classes of a positive slopes.

lemma (in real1) Real_ZF_1_2_L2: shows $a \in \text{PositiveReals} \iff (\exists f \in S_+. a = [f])$
proof
  assume $a \in \text{PositiveReals}$
  then have $a \in \{([s]). s \in S_+\}$ using PositiveReals_def
    by simp
  then show $\exists f \in S_+. a = [f]$ by auto
  next assume $\exists f \in S_+. a = [f]$
    then have $a \in \{([s]). s \in S_+\}$ by auto
    then show $a \in \text{PositiveReals}$ using PositiveReals_def
      by simp
qed

Let’s recall from Int_ZF_2.thy that the sum and composition of positive
slopes is a positive slope.

lemma (in real1) Real_ZF_1_2_L3: assumes $f \in S_+ \quad g \in S_+$
  shows $f+g \in S_+$

745
\( f \circ g \in S_+ \)

using assms Slopes_def PositiveSlopes_def PositiveIntegers_def
  SlopeOp1_def int1.sum_of_pos_sls_is_pos_sl
  SlopeOp2_def int1.comp_of_pos_sls_is_pos_sl
by auto

Bounded integer maps are not positive slopes.

**lemma** (in real1) Real_ZF_1_2_L5:
  assumes \( f \in \text{BoundedIntMaps} \)
  shows \( f/ \in S_+ \)
using assms BoundedIntMaps_def Slopes_def PositiveSlopes_def
  PositiveIntegers_def int1.Int_ZF_2_3_L1B
by simp

The set of positive reals is closed under addition and multiplication. Zero (the neutral element of addition) is not a positive number.

**lemma** (in real1) Real_ZF_1_2_L6: shows
  \( \text{PositiveReals} \) {is closed under} RealAddition
  \( \text{PositiveReals} \) {is closed under} RealMultiplication
  \( 0 \not\in \text{PositiveReals} \)
proof -
  { fix \( a \) \( b \)
    assume \( a \in \text{PositiveReals} \) and \( b \in \text{PositiveReals} \)
    then obtain \( f \) \( g \) where
      I: \( f \in S_+ \) \( g \in S_+ \) and
      II: \( a = [f] \) \( b = [g] \)
      using Real_ZF_1_2_L2 by auto
    then have \( f \in S \) \( g \in S \) using Real_ZF_1_2_L1 Slopes_def
      by auto
    with I II have \( a+b \in \text{PositiveReals} \) \& \( a\cdot b \in \text{PositiveReals} \)
      using Real_ZF_1_1_L4 Real_ZF_1_2_L3 Real_ZF_1_2_L2
      by auto
    } then show \( \text{PositiveReals} \) {is closed under} RealAddition
      \( \text{PositiveReals} \) {is closed under} RealMultiplication
      using IsOpClosed_def
      by auto
  { assume \( 0 \in \text{PositiveReals} \)
    then obtain \( f \) where \( f \in S_+ \) and \( 0 = [f] \)
      using Real_ZF_1_2_L2 by auto
    then have False
      using Real_ZF_1_2_L1 Slopes_def Real_ZF_1_1_L10 Real_ZF_1_2_L5
      by auto
    } then show \( 0 \not\in \text{PositiveReals} \) by auto
qed

If a class of a slope \( f \) is not zero, then either \( f \) is a positive slope or \( -f \) is a positive slope. The real proof is in Int_ZF_2.thy.

**lemma** (in real1) Real_ZF_1_2_L7:
assumes $A_1: f \in S$ and $A_2: [f] \neq 0$
shows $(f \in S^+) \text{ Xor } (-f) \in S^+$
using assms Slopes_def SlopeEquivalenceRel_def BoundedIntMaps_def PositiveSlopes_def PositiveIntegers_def Real_ZF_1_L10 int1.Int_ZF_2_3_L8 by simp

The next lemma rephrases Int_ZF_2_3_L10 in the notation used in real1 context.

lemma (in real1) Real_ZF_1_2_L8:
assumes $A_1: f \in S \quad g \in S$
and $A_2: (f \in S^+) \text{ Xor } (g \in S^+)$
shows $([f] \in \text{PositiveReals}) \text{ Xor } ([g] \in \text{PositiveReals})$
using assms PositiveReals_def SlopeEquivalenceRel_def Slopes_def SlopeOp1_def BoundedIntMaps_def PositiveSlopes_def PositiveIntegers_def int1.Int_ZF_2_3_L8 by simp

The trichotomy law for the (potential) order on reals: if $a \neq 0$, then either $a$ is positive or $-a$ is positive.

lemma (in real1) Real_ZF_1_2_L9:
assumes $A_1: a \in \mathbb{R} \quad a \neq 0$
shows $(a \in \text{PositiveReals}) \text{ Xor } ((-a) \in \text{PositiveReals})$
proof -
from $A_1$ obtain $f$ where $I: f \in S \quad a = [f]$
using Real_ZF_1_1_L3A by auto
with $A_2$ have $([f] \in \text{PositiveReals}) \text{ Xor } ([f] \in \text{PositiveReals})$
using Slopes_def BoundedIntMaps_def int1.Int_ZF_2_3_L8 by simp
with $I$ show $(a \in \text{PositiveReals}) \text{ Xor } ((-a) \in \text{PositiveReals})$
using Real_ZF_1_1_L4A by simp
qed

Finally we are ready to prove that real numbers form an ordered ring with no zero divisors.

theorem reals_are_ord_ring: shows
IsAnOrdRing(RealNumbers,RealAddition,RealMultiplication,OrderOnReals)
OrderOnReals {is total on} RealNumbers
PositiveSet(RealNumbers,RealAddition,OrderOnReals) = PositiveReals
HasNoZeroDivs(RealNumbers,RealAddition,RealMultiplication)
proof -
let $R = \text{RealNumbers}$
let $A = \text{RealAddition}$
let $M = \text{RealMultiplication}$
let $P = \text{PositiveReals}$
let $r = \text{OrderOnReals}$
let $z = \text{TheNeutralElement}(R, A)$
have $I$: ring0$(R, A, M)$
    $M \{\text{is commutative on}\} R$
P ⊆ R
P \{is closed under\} A
TheNeutralElement(R, A) \notin P
\forall a \in R. \ a \neq z \rightarrow (a \in P) \lor (\text{GroupInv}(R, A)(a) \in P)
P \{is closed under\} M
r = \text{OrderFromPosSet}(R, A, P)
\text{using real0.Real_ZF_1_L3 real_mult_commutative Real_ZF_1_2_L1}
\text{real1.Real_ZF_1_2_L6 real1.Real_ZF_1_2_L9 OrderOnReals_def}
by auto
then show IsAnOrdRing(R, A, M, r)
  by (rule ring0.ring_ord_by_positive_set)
from I show r \{is total on\} R
  by (rule ring0.ring_ord_by_positive_set)
from I show PositiveSet(R, A, r) = P
  by (rule ring0.ring_ord_by_positive_set)
from I show HasNoZeroDivs(R, A, M)
  by (rule ring0.ring_ord_by_positive_set)
qed

All theorems proven in the ring1 (about ordered rings), group3 (about ordered groups) and group1 (about groups) contexts are valid as applied to ordered real numbers with addition and (real) order.

**lemma** (in real1) Real_ZF_1_2_L10: shows
ring1(RealNumbers,RealAddition,RealMultiplication,OrderOnReals)
IsAnOrdGroup(RealNumbers,RealAddition,OrderOnReals)
group3(RealNumbers,RealAddition,OrderOnReals)
OrderOnReals \{is total on\} RealNumbers
proof -
  show ring1(RealNumbers,RealAddition,RealMultiplication,OrderOnReals)
    using reals_are_ord_ring OrdRing_ZF_1_2_L2 by simp
  then show
    IsAnOrdGroup(RealNumbers,RealAddition,OrderOnReals)
    group3(RealNumbers,RealAddition,OrderOnReals)
    OrderOnReals \{is total on\} RealNumbers
    using ring1.OrdRing_ZF_1_2_L4 by auto
qed

If a = b or b - a is positive, then a is less or equal b.

**lemma** (in real1) Real_ZF_1_2_L11: assumes A1: a∈R \ b∈R and
A3: a=b \lor b-a ∈ PositiveReals
shows a≤b
  using assms reals_are_ord_ring Real_ZF_1_2_L10
  group3.OrdGroup_ZF_1_2_L30 by simp

A sufficient condition for two classes to be in the real order.

**lemma** (in real1) Real_ZF_1_2_L12: assumes A1: f ∈ S \ g ∈ S and
A2: f\sim \lor (g + (-f)) ∈ S
shows [f] ≤ [g]
proof -
  from A1 A2 have \([f] = [g] \lor [g] - [f] \in \text{PositiveReals}\)
  using Real_ZF_1_1_L5A Real_ZF_1_2_L2 Real_ZF_1_1_L4B
  by auto
with A1 show \([f] \leq [g]\) using Real_ZF_1_1_L3 Real_ZF_1_2_L11
  by simp
qed

Taking negative on both sides reverses the inequality, a case with an inverse
on one side. Property of ordered groups.

lemma (in real1) Real_ZF_1_2_L13:
  assumes A1: \(a \in \mathbb{R}\) and A2: \((-a) \leq b\)
  shows \((-b) \leq a\)
  using assms Real_ZF_1_2_L10 group3.OrderedGroup_ZF_1_L5AG
  by simp

Real order is antisymmetric.

lemma (in real1) real_ord_antisyms:
  assumes A1: \(a \leq b\) \(b \leq a\) shows \(a = b\)
  proof -
    from A1 have group3(RealNumbers,RealAddition,OrderOnReals)
      \(\langle a,b \rangle \in \text{OrderOnReals}\) \(\langle b,a \rangle \in \text{OrderOnReals}\)
      using Real_ZF_1_2_L10 by auto
    then have \(\langle a,c \rangle \in \text{OrderOnReals}\)
      by (rule group3.group_order_antisym)
    then show \(a = b\) by simp
  qed

Real order is transitive.

lemma (in real1) real_ord_transitive: assumes A1: \(a \leq b\) \(b \leq c\)
  shows \(a \leq c\)
  proof -
    from A1 have group3(RealNumbers,RealAddition,OrderOnReals)
      \(\langle a,b \rangle \in \text{OrderOnReals}\) \(\langle b,c \rangle \in \text{OrderOnReals}\)
      using Real_ZF_1_2_L10 by auto
    then have \(\langle a,c \rangle \in \text{OrderOnReals}\)
      by (rule group3.Group_order_transitive)
    then show \(a \leq c\) by simp
  qed

We can multiply both sides of an inequality by a nonnegative real number.

lemma (in real1) Real_ZF_1_2_L14:
  assumes a\(\leq b\) and \(0 \leq c\)
  shows \(a \cdot c \leq b \cdot c\) \(c \cdot a \leq c \cdot b\)
  using assms Real_ZF_1_2_L10 ring1.OrdRing_ZF_1_L9
  by auto
A special case of RealZF_1_2_L14: we can multiply an inequality by a real number.

**lemma (in real1) RealZF_1_2_L14A:**

**assumes**

A1: \( a \leq b \) and
A2: \( c \in \mathbb{R}_+ \)

**shows**

\( c \cdot a \leq c \cdot b \)

**using**

assms RealZF_1_2_L10 ring1.OrdRingZF_1_L9A
by simp

In the real1 context notation \( a \leq b \) implies that \( a \) and \( b \) are real numbers.

**lemma (in real1) RealZF_1_2_L15:**

**assumes**

\( a \leq b \)

**shows**

\( 0 \leq b - a \)

**using**

assms RealZF_1_2_L10 group3.OrderedGroupZF_1_L4
by auto

\( a \leq b \) implies that \( 0 \leq b - a \).

**lemma (in real1) RealZF_1_2_L16:**

**assumes**

\( a \leq b \)

**shows**

\( 0 \leq a + b \)

**using**

assms RealZF_1_2_L10 group3.OrderedGroupZF_1_L12
by simp

A sum of nonnegative elements is nonnegative.

**lemma (in real1) RealZF_1_2_L17:**

**assumes**

\( 0 \leq a \) \( 0 \leq b \)

**shows**

\( 0 \leq a + b \)

**using**

assms RealZF_1_2_L10 group3.OrderedGroupZF_1_L12
by simp

We can add sides of two inequalities

**lemma (in real1) RealZF_1_2_L18:**

**assumes**

\( a \leq b \) \( c \leq d \)

**shows**

\( a + c \leq b + d \)

**using**

assms RealZF_1_2_L10 group3.OrderedGroupZF_1_L5
by simp

The order on real is reflexive.

**lemma (in real1) real_ord_refl:**

**assumes**

\( a \in \mathbb{R} \)

**shows**

\( a \leq a \)

**using**

assms RealZF_1_2_L10 group3.OrderedGroupZF_1_L3
by simp

We can add a real number to both sides of an inequality.

**lemma (in real1) add_num_to_ineq:**

**assumes**

\( a \leq b \) and
\( c \in \mathbb{R} \)

**shows**

\( a + c \leq b + c \)

**using**

assms RealZF_1_2_L10 IsAnOrdGroup_def by simp

We can put a number on the other side of an inequality, changing its sign.

**lemma (in real1) RealZF_1_2_L19:**

**assumes**

\( a \in \mathbb{R} \) \( b \in \mathbb{R} \) and
\( c \leq a + b \)

**shows**

\( c - b \leq a \)

**using**

assms RealZF_1_2_L10 group3.OrderedGroupZF_1_L9
by simp
What happens when one real number is not greater or equal than another?

**lemma (in real1) Real_ZF_1_2_L20:** assumes $a \in \mathbb{R}$ $b \in \mathbb{R}$ and $\neg (a \leq b)$

shows $b < a$

**proof**

- from assmhs have I:
  - group3$(\mathbb{R}, \text{RealAddition}, \text{OrderOnReals})$
  - OrderOnReals is total on $\mathbb{R}$
  - $a \in \mathbb{R}$ $b \in \mathbb{R}$ $\neg ((a,b) \in \text{OrderOnReals})$
  - using Real_ZF_1_2_L10 by auto

then have $(b,a) \in \text{OrderOnReals}$

by (rule group3.OrderedGroup_ZF_1_L8)

then have $b \leq a$ by simp

moreover from I have $a \neq b$ by (rule group3.OrderedGroup_ZF_1_L8)

ultimately show $b < a$ by auto

qed

We can put a number on the other side of an inequality, changing its sign, version with a minus.

**lemma (in real1) Real_ZF_1_2_L21:**

assumes $a \in \mathbb{R}$ $b \in \mathbb{R}$ and $c \leq a - b$

shows $c + b \leq a$

using assmhs Real_ZF_1_2_L10 group3.OrderedGroup_ZF_1_L5J

by simp

The order on reals is a relation on reals.

**lemma (in real1) Real_ZF_1_2_L22:** shows OrderOnReals $\subseteq \mathbb{R} \times \mathbb{R}$

using Real_ZF_1_2_L10 IsAnOrdGroup_def

by simp

A set that is bounded above in the sense defined by order on reals is a subset of real numbers.

**lemma (in real1) Real_ZF_1_2_L23:**

assumes A1: IsBoundedAbove$(A, \text{OrderOnReals})$

shows $A \subseteq \mathbb{R}$

using A1 Real_ZF_1_2_L22 Order_ZF_3_L1A

by blast

Properties of the maximum of three real numbers.

**lemma (in real1) Real_ZF_1_2_L24:**

assumes A1: $a \in \mathbb{R}$ $b \in \mathbb{R}$ $c \in \mathbb{R}$

shows

- Maximum$(\text{OrderOnReals}, \{a,b,c\}) \in \{a,b,c\}$
- Maximum$(\text{OrderOnReals}, \{a,b,c\}) \in \mathbb{R}$
- $a \leq \text{Maximum}(\text{OrderOnReals}, \{a,b,c\})$
- $b \leq \text{Maximum}(\text{OrderOnReals}, \{a,b,c\})$
- $c \leq \text{Maximum}(\text{OrderOnReals}, \{a,b,c\})$

**proof**

- have IsLinOrder$(\mathbb{R}, \text{OrderOnReals})$
using Real_ZF_1_2_L10 group3.group_ord_total_is_lin
by simp
with A1 show
Maximum(OrderOnReals,"a,b,c") ∈ {a,b,c}
Maximum(OrderOnReals,"a,b,c") ∈ \mathbb{R}
a ≤ Maximum(OrderOnReals,"a,b,c")
b ≤ Maximum(OrderOnReals,"a,b,c")
c ≤ Maximum(OrderOnReals,"a,b,c")
using Finite_ZF_1_L2A by auto

qed

A form of transitivity for the order on reals.

lemma (in real1) real_strict_ord_transit:
assumes A1: a ≤ b and A2: b < c
shows a < c
proof -
from A1 A2 have I:
group3(R,RealAddition,OrderOnReals)
\langle a,b \rangle ∈ OrderOnReals \langle b,c \rangle ∈ OrderOnReals ∧ b ≠ c
using Real_ZF_1_2_L10 by auto
then have \langle a,c \rangle ∈ OrderOnReals ∧ a ≠ c by (rule group3.group_strict_ord_transit)
then show a < c by simp
qed

We can multiply a right hand side of an inequality between positive real
numbers by a number that is greater than one.

lemma (in real1) Real_ZF_1_2_L25:
assumes b ∈ R_+ and a ≤ b and 1 < c
shows a < b · c
using assms reals_are_ord_ring Real_ZF_1_2_L10 ring1.OrdRing_ZF_3_L17
by simp

We can move a real number to the other side of a strict inequality, changing
its sign.

lemma (in real1) Real_ZF_1_2_L26:
assumes a ∈ R_+ b ∈ R_+ and a-b < c
shows a < c+b
using assms Real_ZF_1_2_L10 group3.OrderedGroup_ZF_1_L12B
by simp

Real order is translation invariant.

lemma (in real1) real_ord_transl_inv:
assumes a ≤ b and c ∈ R
shows c+a ≤ c+b
using assms Real_ZF_1_2_L10 IsAnOrdGroup_def
by simp

It is convenient to have the transitivity of the order on integers in the nota-
tion specific to real1 context. This may be confusing for the presentation
readers: even though \( \leq \) and \( \leq \) are printed in the same way, they are different symbols in the source. In the real1 context the former denotes inequality between integers, and the latter denotes inequality between real numbers (classes of slopes). The next lemma is about transitivity of the order relation on integers.

**lemma (in real1) int_order_transitive:**

assumes \( A1: a \leq b \) \( b \leq c \)

shows \( a \leq c \)

**proof**

- from \( A1 \) have
  
  \( (a,b) \in \text{IntegerOrder} \) and \( (b,c) \in \text{IntegerOrder} \)
  
  by auto

  then have \( (a,c) \in \text{IntegerOrder} \)

  by (rule Int_ZF_2_L5)

  then show \( a \leq c \) by simp

qed

A property of nonempty subsets of real numbers that don’t have a maximum: for any element we can find one that is (strictly) greater.

**lemma (in real1) Real_ZF_1_2_L27:**

assumes \( A \subseteq \mathbb{R} \) and \( \neg \text{HasAmaximum(OrderOnReals,A)} \) and \( x \in A \)

shows \( \exists y \in A. x < y \)

using assms Real_ZF_1_2_L10 group3.OrderedGroup_ZF_2_L2B by simp

The next lemma shows what happens when one real number is not greater or equal than another.

**lemma (in real1) Real_ZF_1_2_L28:**

assumes \( a \in \mathbb{R} \) \( b \in \mathbb{R} \) and \( \neg (a \leq b) \)

shows \( b < a \)

**proof**

- from assms have

  group3(\( \mathbb{R} \),RealAddition,OrderOnReals)

  OrderOnReals {is total on} \( \mathbb{R} \)

  \( a \in \mathbb{R} \) \( b \in \mathbb{R} \) \( (a,b) \notin \text{OrderOnReals} \)

  using Real_ZF_1_2_L10 by auto

  then have \( (b,a) \in \text{OrderOnReals} \) \( \land b \neq a \)

  by (rule group3.OrderedGroup_ZF_1_L8)

  then show \( b < a \) by simp

qed

If a real number is less than another, then the second one can not be less or equal that the first.

**lemma (in real1) Real_ZF_1_2_L29:**

assumes \( a < b \)

shows \( \neg (b \leq a) \)

**proof**

- from assms have
group3(R,RealAddition,OrderOnReals) 
(a,b) ∈ OrderOnReals  a≠b 
using Real_ZF_1_2_L10 by auto 
then have ⟨b,a⟩ ∉ OrderOnReals 
  by (rule group3.OrderedGroup_ZF_1_L8AA) 
then show ¬(b≤a) by simp 

qed

59.4 Inverting reals

In this section we tackle the issue of existence of (multiplicative) inverses of real numbers and show that real numbers form an ordered field. We also restate here some facts specific to ordered fields that we need for the construction. The actual proofs of most of these facts can be found in Field_ZF.thy and OrderedField_ZF.thy.

We rewrite the theorem from Int_ZF_2.thy that shows that for every positive slope we can find one that is almost equal and has an inverse.

lemma (in real1) pos_slopes_have_inv: assumes f ∈ S+ shows ∃g ∈ S. f∼g ∧ (∃h ∈ S. g◦h ∼ id(int)) 
  using assms PositiveSlopes_def Slopes_def PositiveIntegers_def 
    int1.pos_slope_has_inv SlopeOp1_def SlopeOp2_def 
    BoundedIntMaps_def SlopeEquivalenceRel_def 
  by simp

The set of real numbers we are constructing is an ordered field.

theorem (in real1) reals_are_ord_field: shows IsAnOrdField(RealNumbers,RealAddition,RealMultiplication,OrderOnReals) 
proof - 
  let R = RealNumbers 
  let A = RealAddition 
  let M = RealMultiplication 
  let r = OrderOnReals 
  have ring1(R,A,M,r) and 0 ≠ 1 
    using reals_are_ord_ring OrdRing_ZF_1_L2 real_zero_not_one 
    by auto 
  moreover have M {is commutative on} R 
    using real_mult_commutative by simp 
  moreover have ∀a∈PositiveSet(R,A,r). ∃b∈R. a·b = 1 
    proof 
      fix a assume a ∈ PositiveSet(R,A,r) 
      then obtain f where I: f∈S_and II: a = [f] 
        using reals_are_ord_ring Real_ZF_1_2_L2 
        by auto 
      then have ∃g∈S. f∼g ∧ (∃h∈S. g◦h ∼ id(int)) 
        using pos_slopes_have_inv by simp 
      then obtain g where
III: $g \in S$ and IV: $f \sim g$ and V: $\exists h \in S$. $g \circ h \sim \text{id(int)}$
by auto
from V obtain $h$ where VII: $h \in S$ and VIII: $g \circ h \sim \text{id(int)}$
by auto
from I III IV have $[f] = [g]$
using Real_ZF_1_2_L1 Slopes_def Real_ZF_1_1_L5
by auto
with II III VII VIII have $a \cdot [h] = 1$
using Real_ZF_1_1_L4 Real_ZF_1_1_L5A real_one_cl_identity
by simp
with VII show $\exists b \in \mathbb{R}. a \cdot b = 1$ using Real_ZF_1_1_L3
by auto
qed
ultimately show thesis using ring1.OrdField_ZF_1_L4
by simp
qed

Reals form a field.

lemma reals_are_field:
shows IsAfield(RealNumbers,RealAddition,RealMultiplication)
using real1.reals_are_ord_field OrdField_ZF_1_L1A
by simp

Theorem proven in field0 and field1 contexts are valid as applied to real numbers.

lemma field_cntxts_ok: shows
field0(RealNumbers,RealAddition,RealMultiplication)
field1(RealNumbers,RealAddition,RealMultiplication,OrderOnReals)
using reals_are_field real1.reals_are_ord_field
field_field0 OrdField_ZF_1_L2 by auto

If $a$ is positive, then $a^{-1}$ is also positive.

lemma (in real1) Real_ZF_1_3_L1: assumes $a \in \mathbb{R}_{+}$
shows $a^{-1} \in \mathbb{R}_{+} \quad a^{-1} \in \mathbb{R}$
using assms field_cntxts_ok field1.OrdField_ZF_1_L8 PositiveSet_def
by auto

A technical fact about multiplying strict inequality by the inverse of one of the sides.

lemma (in real1) Real_ZF_1_3_L2:
assumes $a \in \mathbb{R}_{+}$ and $a^{-1} < b$
shows $1 < b \cdot a$
using assms field_cntxts_ok field1.OrdField_ZF_2_L2
by simp

If $a$ is smaller than $b$, then $(b - a)^{-1}$ is positive.

lemma (in real1) Real_ZF_1_3_L3: assumes $a < b$
shows $(b - a)^{-1} \in \mathbb{R}_{+}$
We can put a positive factor on the other side of a strict inequality, changing it to its inverse.

**lemma (in real1) Real_ZF_1_3_L4:**
\[ \text{assumes } A1: a \in \mathbb{R}, \ b \in \mathbb{R}^+ \text{ and } A2: a \cdot b < c \]
\[ \text{shows } a < c/b^{-1} \]
using assms field_cntxts_ok field1.OrdField_ZF_2_L6 by simp

We can put a positive factor on the other side of a strict inequality, changing it to its inverse, version with the product initially on the right hand side.

**lemma (in real1) Real_ZF_1_3_L4A:**
\[ \text{assumes } A1: b \in \mathbb{R}, \ c \in \mathbb{R}^+ \text{ and } A2: a < b \cdot c \]
\[ \text{shows } a \cdot c^{-1} < b \]
using assms field_cntxts_ok field1.OrdField_ZF_2_L6A by simp

We can put a positive factor on the other side of an inequality, changing it to its inverse, version with the product initially on the right hand side.

**lemma (in real1) Real_ZF_1_3_L4B:**
\[ \text{assumes } A1: b \in \mathbb{R}, \ c \in \mathbb{R}^+ \text{ and } A2: a \leq b \cdot c \]
\[ \text{shows } a \cdot c^{-1} \leq b \]
using assms field_cntxts_ok field1.OrdField_ZF_2_L5A by simp

We can put a positive factor on the other side of an inequality, changing it to its inverse, version with the product initially on the left hand side.

**lemma (in real1) Real_ZF_1_3_L4C:**
\[ \text{assumes } A1: a \in \mathbb{R}, \ b \in \mathbb{R}^+ \text{ and } A2: a \cdot b \leq c \]
\[ \text{shows } a \leq c \cdot b^{-1} \]
using assms field_cntxts_ok field1.OrdField_ZF_2_L5 by simp

A technical lemma about solving a strict inequality with three real numbers and inverse of a difference.

**lemma (in real1) Real_ZF_1_3_L5:**
\[ \text{assumes } a < b \text{ and } (b-a)^{-1} < c \]
\[ \text{shows } 1 + a \cdot c < b \cdot c \]
using assms field_cntxts_ok field1.OrdField_ZF_2_L9 by simp

We can multiply an inequality by the inverse of a positive number.

**lemma (in real1) Real_ZF_1_3_L6:**
\[ \text{assumes } a \leq b \text{ and } c \in \mathbb{R}^+ \text{ shows } a \cdot c^{-1} \leq b \cdot c^{-1} \]
using assms field_cntxts_ok field1.OrdField_ZF_2_L3
We can multiply a strict inequality by a positive number or its inverse.

**Lemma (in real1) Real_ZF_1_3_L7:**

**Assumptions:**
- $a < b$
- $c \in \mathbb{R}_+$

**Shows:**
- $a \cdot c < b \cdot c$
- $c \cdot a < c \cdot b$
- $a \cdot c^{-1} < b \cdot c^{-1}$

**Using:**
- `assms`
- `field_cntxts_ok`
- `field1.OrdField_ZF_2_L4`

**Proof:**

by auto

An identity with three real numbers, inverse and cancelling.

**Lemma (in real1) Real_ZF_1_3_L8:**

**Assumptions:**
- $a \in \mathbb{R}$
- $b \in \mathbb{R}$
- $b \neq 0$
- $c \in \mathbb{R}$

**Shows:**
- $a \cdot b \cdot (c \cdot b^{-1}) = a \cdot c$

**Using:**
- `assms`
- `field_cntxts_ok`
- `field0.Field_ZF_2_L6`

**Proof:**

by simp

### 59.5 Completeness

This goal of this section is to show that the order on real numbers is complete, that is every subset of reals that is bounded above has a smallest upper bound.

If $m$ is an integer, then $m^R$ is a real number. Recall that in `real1` context $m^R$ denotes the class of the slope $n \mapsto m \cdot n$.

**Lemma (in real1) real_int_is_real:**

**Assumptions:**
- $m \in \mathbb{Z}$

**Shows:**
- $m^R \in \mathbb{R}$

**Using:**
- `assms`
- `int1.Int_ZF_2_5_L1`
- `Real_ZF_1_1_L3`

by simp

The negative of the real embedding of an integer is the embedding of the negative of the integer.

**Lemma (in real1) Real_ZF_1_4_L1:**

**Assumptions:**
- $m \in \mathbb{Z}$

**Shows:**
- $(-m)^R = -(m^R)$

**Using:**
- `assms`
- `int1.Int_ZF_2_5_L3`
- `int1.Int_ZF_2_5_L1`
- `Real_ZF_1_1_L4A`

by simp

The embedding of sum of integers is the sum of embeddings.

**Lemma (in real1) Real_ZF_1_4_L1A:**

**Assumptions:**
- $m \in \mathbb{Z}$
- $k \in \mathbb{Z}$

**Shows:**
- $m^R + k^R = ((m+k)^R)$

**Using:**
- `assms`
- `int1.Int_ZF_2_5_L1`
- `SlopeOp1_def`
- `int1.Int_ZF_2_5_L3A`
- `Real_ZF_1_1_L4`

by simp

The embedding of a difference of integers is the difference of embeddings.

**Lemma (in real1) Real_ZF_1_4_L1B:**

**Assumptions:**
- $m \in \mathbb{Z}$
- $k \in \mathbb{Z}$

**Shows:**
- $m^R - k^R = (m-k)^R$

**Proof:**

- from `A1` have $(-k) \in \mathbb{Z}$ using `int0.Int_ZF_1_1_L4`
by simp
with A1 have \((m-k)^R = m^R + (-k)^R\)
  using Real_ZF_1_4_L1A by simp
with A1 show \(m^R - k^R = (m-k)^R\)
  using Real_ZF_1_4_L1 by simp
qed

The embedding of the product of integers is the product of embeddings.

lemma (in real1) Real_ZF_1_4_L1C: assumes \(m \in \text{int} \quad k \in \text{int}\)
  shows \(m^R \cdot k^R = (m \cdot k)^R\)
  using assms int1.Int_ZF_2_5_L1 SlopeOp2_def int1.Int_ZF_2_5_L3B
  Real_ZF_1_1_L4 by simp

For any real numbers there is an integer whose real version is greater or equal.

lemma (in real1) Real_ZF_1_4_L2: assumes a \(\in \)' shows \(\exists m \in \text{int}. a \leq m^R\)
proof -
  from A1 obtain f where I: \(f \in \mathcal{S}\) and II: \(a = \lfloor f \rfloor\)
    using Real_ZF_1_1_L3A by auto
  then have \(\exists g \in \mathcal{S}. \{\langle n,m \cdot n \rangle. n \in \text{int}\} \sim g \land (f \sim g \lor (g + (-f)) \in S_+)\)
    using int1.Int_ZF_2_5_L2 Slopes_def SlopeOp1_def
    BoundedIntMaps_def SlopeEquivalenceRel_def
    PositiveIntegers_def PositiveSlopes_def
    by simp
  then obtain m g where III: \(m \in \text{int}\) and IV: \((-a) \leq m^R\)
    using Real_ZF_1_4_L2 by auto
  let k = GroupInv(int,IntegerAddition)(m)
  from A1 I II IV have \(k \in \text{int} \quad k^R \leq a\)
    using Real_ZF_1_2_L13 Real_ZF_1_4_L1 by simp
  qed

For any real numbers there is an integer whose real version (embedding) is less or equal.

lemma (in real1) Real_ZF_1_4_L3: assumes A1: a\(\in\)R
  shows \(\{m \in \text{int}. m^R \leq a\} \neq 0\)
proof -
  from A1 have \((-a) \in \mathbb{R}\) using Real_ZF_1_1_L8
    by simp
  then obtain m where I: \(m \in \text{int}\) and II: \((-a) \leq m^R\)
    using Real_ZF_1_4_L2 by auto
  let k = GroupInv(int,IntegerAddition)(m)
  from A1 I II have \(k \in \text{int} \quad k^R \leq a\)
    using Real_ZF_1_2_L13 Real_ZF_1_4_L1 by simp
  qed

758
Embeddings of two integers are equal only if the integers are equal.

lemma (in real1) Real_ZF_1_4_L4:
assumes A1: m ∈ int k ∈ int and A2: m^R = k^R
shows m=k
proof -
  let r = {⟨n, IntegerMultiplication ⟨m, n⟩⟩ . n ∈ int}
  let s = {⟨n, IntegerMultiplication ⟨k, n⟩⟩ . n ∈ int}
  from A1 A2 have r ∼ s
      using int1.Int_ZF_2_5_L1 AlmostHoms_def Real_ZF_1_1_L5 by simp
  with A1 have m ∈ int k ∈ int
      (r,s) ∈ QuotientGroupRel(AlmostHoms(int, IntegerAddition),
      AlHomOp1(int, IntegerAddition),FinRangeFunctions(int, int))
      using Slopes_def SlopeOp1_def BoundedIntMaps_def by auto
  then show m=k by (rule int1.Int_ZF_2_5_L6)
qed

The embedding of integers preserves the order.

lemma (in real1) Real_ZF_1_4_L5: assumes A1: m ≤ k
shows m^R ≤ k^R
proof -
  let r = {⟨n, m·n⟩ . n ∈ int}
  let s = {⟨n, k·n⟩ . n ∈ int}
  from A1 have r ∈ S s ∈ S
      using Int_ZF_2_L1A int1.Int_ZF_2_5_L1 by auto
  moreover from A1 have r ∼ s ∨ s + (-r) ∈ S+_1
      using Slopes_def SlopeOp1_def BoundedIntMaps_def SlopeEquivalenceRel_def
      PositiveIntegers_def PositiveSlopes_def
      int1.Int_ZF_2_5_L4 by simp
  ultimately show m^R ≤ k^R using Real_ZF_1_2_L12
      by simp
qed

The embedding of integers preserves the strict order.

lemma (in real1) Real_ZF_1_4_L5A: assumes A1: m ≤ k m ≠ k
shows m^R < k^R
proof -
  from A1 have m^R ≤ k^R using Real_ZF_1_4_L5
      by simp
  moreover
  from A1 have T: m ∈ int k ∈ int
      using Int_ZF_2_L1A by auto
  with A1 have m^R ≠ k^R using Real_ZF_1_4_L4
    by (rule int1.Int_ZF_2_5_L6)
For any real number there is a positive integer whose real version is (strictly)
greater. This is Lemma 14 i) in [2].

lemma (in real1) Arthan_Lemma14i: assumes A1: a∈R
  shows ∃n∈\mathbb{Z}^+. a < n^R
proof -
  from A1 obtain m where I: m∈\mathbb{Z} and II: a ≤ m^R
  using Real_ZF_1_4_L2 by auto
  let n = GreaterOf(IntegerOrder,1_Z,m) + 1_Z
  from I have T: n ∈ \mathbb{Z}^+ and m ≤ n m̸=n
  using int0.Int_ZF_1_5_L7B by auto
  then have III: m^R < n^R
  using Real_ZF_1_4_L5A by simp
  with II have a < n^R by (rule real_strict_ord_transit)
  with T show thesis by auto
qed

If one embedding is less or equal than another, then the integers are also
less or equal.

lemma (in real1) Real_ZF_1_4_L6:
  assumes A1: k ∈ \mathbb{Z} and m ∈ \mathbb{Z}
  and A2: m^R ≤ k^R
  shows m ≤ k
proof -
{ assume A3: ⟨m,k⟩ /∈ IntegerOrder
  with A1 have ⟨k,m⟩ ∈ IntegerOrder
    by (rule int0.Int_ZF_2_L19)
  then have k^R ≤ m^R using Real_ZF_1_4_L5
    by simp
  with A2 have m^R = k^R by (rule real_ord_antisym)
  with A1 have k = m using Real_ZF_1_4_L4
    by auto
  moreover from A1 A3 have k̸=m by (rule int0.Int_ZF_2_L19)
  ultimately have False by simp
} then show m ≤ k by auto
qed

The floor function is well defined and has expected properties.

lemma (in real1) Real_ZF_1_4_L7: assumes A1: a∈R
  shows IsBoundedAbove({m ∈ \mathbb{Z}. m^R ≤ a},IntegerOrder)
  {m ∈ \mathbb{Z}. m^R ≤ a} ≠ 0
  \lfloor a \rfloor ∈ \mathbb{Z}
  \lfloor a \rfloor^R ≤ a
proof -
  let A = {m ∈ \mathbb{Z}. m^R ≤ a}
Every integer whose embedding is less or equal a real number $a$ is less or equal than the floor of $a$.

**lemma (in real1) Real_ZF_1_4_L8:**

assumes $A1: m \in \text{int}$ and $A2: m^R \leq a$

shows $m \leq \lfloor a \rfloor$

**proof** -

1. let $A = \{m \in \text{int}. m^R \leq a\}$
2. from $A2$ have IsBoundedAbove($A, \text{IntegerOrder}$) and $A \neq 0$
   using Real_ZF_1_2_L15 Real_ZF_1_4_L7 by auto
3. then have $\forall x \in A. \langle x, \text{Maximum}(\text{IntegerOrder},A) \rangle \in \text{IntegerOrder}$
   by (rule int0.int_bounded_above_has_max)
4. with $A1 A2$ show $m \leq \lfloor a \rfloor$ by simp

**qed**

Integer zero and one embed as real zero and one.

**lemma (in real1) int_0_1_are_real_zero_one:**

shows $0^R_{\text{Z}} = 0$ $1^R_{\text{Z}} = 1$

**proof** -

1. have $2^R_{\text{Z}} = 1^R_{\text{Z}} + 1^R_{\text{Z}}$
   using int0.int_zero_one_are_int Real_ZF_1_4_L1A by simp
2. also have $... = 2$ using int_0_1_are_real_zero_one by simp

**Integer two embeds as the real two.**

**lemma (in real1) int_two_is_real_two:**

shows $2^R_{\text{Z}} = 2$
finally show \(2^Z = 2\) by simp

\text{qed}

A positive integer embeds as a positive (hence nonnegative) real.

\text{lemma (in real1) int_pos_is_real_pos: assumes A1: } p \in \mathbb{Z}_+ \text{ shows } \\
\quad \begin{align*}
& p^R \in \mathbb{R} \\
& 0 \leq p^R \\
& p^R \in \mathbb{R}_+
\end{align*}

\text{proof - }
\begin{align*}
\text{from A1 have I: } p \in \mathbb{Z} & \leq p & 0_\mathbb{Z} \neq p \\
\quad \text{using PositiveSet_def by auto } \\
\text{then have } p^R \in \mathbb{R} & 0_\mathbb{Z}^R \leq p^R \\
\quad \quad \text{using real_int_is_real Real_ZF_1_4_L5 by auto } \\
\text{then show } p^R \in \mathbb{R} & 0 \leq p^R \\
\quad \quad \text{using int_0_1_are_real_zero_one by auto } \\
\text{moreover have } 0 \neq p^R & \\
\text{proof - } \\
\quad \{ \quad \text{assume } 0 = p^R \\
\quad \quad \text{with I have False using int_0_1_are_real_zero_one } \\
\quad \quad \text{int0.int_zero_one_are_int Real_ZF_1_4_L4 by auto } \\
\quad \} \quad \text{then show } 0 \neq p^R \text{ by auto } \\
\text{qed}
\end{align*}

\text{ultimately show } p^R \in \mathbb{R}_+ \text{ using PositiveSet_def by simp }

\text{qed}

The ordered field of reals we are constructing is archimedean, i.e., if \(x, y\) are its elements with \(y\) positive, then there is a positive integer \(M\) such that \(x\) is smaller than \(M^R y\). This is Lemma 14 ii) in [2].

\text{lemma (in real1) Arthan_Lemma14ii: assumes A1: } x \in \mathbb{R} & y \in \mathbb{R}_+ \\
\text{shows } \exists M \in \mathbb{Z}_+. \ x < M^R y

\text{proof - }
\begin{align*}
\text{from A1 have } & \exists C \in \mathbb{Z}_+. \ x < C^R \quad \text{and } \exists D \in \mathbb{Z}_+. \ y^{-1} < D^R \\
\quad \text{using Real_ZF_1_3_L1 Arthan_Lemma14i by auto } \\
\text{then obtain } C & \text{ and } D \\
\text{where } & \text{ I: } C \in \mathbb{Z}_+ \text{ and II: } x < C^R \\
\quad & \text{III: } D \in \mathbb{Z}_+ \text{ and IV: } y^{-1} < D^R \\
\quad \text{by auto } \\
\text{let } M & = C \cdot D \\
\text{from I III have } \\
\text{from I III have } T: M \in \mathbb{Z}_+ & \quad C^R \in \mathbb{R} \quad D^R \in \mathbb{R} \\
\quad \quad \text{using int0.pos_int_closed_mul_unfold PositiveSet_def real_int_is_real } \\
\quad \quad \text{by auto } \\
\quad \quad \text{with A1 I III have } C^R \cdot (D^R \cdot y) = M^R y \\
\quad \quad \text{using PositiveSet_def Real_ZF_1_L6A Real_ZF_1_4_L1C } \\
\quad \quad \text{by simp } \\
\text{moreover from A1 I II IV have }
\end{align*}
\[ x < c^R \cdot (D^R \cdot y) \]
using \texttt{int_pos_is_real_pos RealZF_1_3_L2 RealZF_1_2_L25}
by auto
ultimately have \( x < M^R \cdot y \)
by auto
with \( T \) show \( \text{thesis} \) by auto
qed

Taking the floor function preserves the order.

\textbf{Lemma} (in real1) \texttt{RealZF_1_4_L9}: \textit{assumes} \( A1: \ a \leq b \)
\textit{shows} \( \lfloor a \rfloor \leq \lfloor b \rfloor \)

\textbf{proof} -
from \( A1 \) have \( T: a \in \mathbb{R} \) using \texttt{RealZF_1_2_L15}
by simp
with \( A1 \) have \( \lfloor a \rfloor \leq a \) and \( a \leq b \)
using \texttt{RealZF_1_4_L7} by auto
then have \( \lfloor a \rfloor \leq b \) by \( \text{rule real_ord_transitive} \)
moreover from \( T \) have \( \lfloor a \rfloor \in \mathbb{I} \)
using \texttt{RealZF_1_4_L7} by simp
ultimately show \( \lfloor a \rfloor \leq \lfloor b \rfloor \) using \texttt{RealZF_1_4_L8}
by simp
qed

If \( S \) is bounded above and \( p \) is a positive integer, then \( \Gamma(S, p) \) is well defined.

\textbf{Lemma} (in real1) \texttt{RealZF_1_4_L10}:
\textit{assumes} \( A1: \ \text{IsBoundedAbove}(S, \text{OrderOnReals}) \ S \neq 0 \) and \( A2: \ p \in \mathbb{Z}_+ \)
\textit{shows} \( \text{IsBoundedAbove}\{\lfloor p^R \cdot x \rfloor. \ x \in S\}, \text{IntegerOrder} \)
\( \Gamma(S, p) \in \{\lfloor p^R \cdot x \rfloor. \ x \in S\} \)
\( \Gamma(S, p) \in \mathbb{I} \)

\textbf{proof} -
let \( A = \{\lfloor p^R \cdot x \rfloor. \ x \in S\} \)
from \( A1 \) obtain \( X \) where \( I: \ \forall x \in S. \ x \leq X \)
using \texttt{IsBoundedAbove_def} by auto
\{ fix \( m \) assume \( m \in A \)
then obtain \( x \) where \( x \in S \) and \( II: \ m = \lfloor p^R \cdot x \rfloor \)
by auto
with \( I \) have \( x \leq X \) by simp
moreover from \( A2 \) have \( 0 \leq p^R \) using \texttt{int_pos_is_real_pos}
by simp
ultimately have \( p^R \cdot x \leq p^R \cdot X \) using \texttt{RealZF_1_2_L14}
by simp
with \( II \) have \( m \leq \lfloor p^R \cdot X \rfloor \) using \texttt{RealZF_1_4_L9}
by simp
\} then have \( \forall m \in A. \ (m, \lfloor p^R \cdot X \rfloor) \in \text{IntegerOrder} \)
by auto
then show \( II: \ \text{IsBoundedAbove}(A, \text{IntegerOrder}) \)
by \( \text{rule real_ZF_3_L10} \)
moreover from A1 have III: $A \neq 0$ by simp
ultimately have $\text{Maximum}(\text{IntegerOrder},A) \in A$
by (rule int0.int_bounded_above_has_max)
moreover from II III have $\text{Maximum}(\text{IntegerOrder},A) \in \text{int}$
by (rule int0.int_bounded_above_has_max)
ultimately show $\Gamma(S,p) \in \{\lfloor p^R \cdot x \rfloor. \ x \in S\}$ and $\Gamma(S,p) \in \text{int}$
by auto
qed

If $p$ is a positive integer, then for all $s \in S$ the floor of $p \cdot x$ is not greater
that $\Gamma(S,p)$.

lemma (in real1) Real_ZF_1_4_L11:
assumes A1: $\text{IsBoundedAbove}(S,\text{OrderOnReals})$ and
A2: $x \in S$ and A3: $p \in \mathbb{Z}^+$
shows $\lfloor p^R \cdot x \rfloor \leq \Gamma(S,p)$
proof -
  let $A = \{\lfloor p^R \cdot x \rfloor. \ x \in S\}$
  from A2 have $S \neq 0$ by auto
  with A1 A3 have $\text{IsBoundedAbove}(A,\text{IntegerOrder})$ $A \neq 0$
    using Real_ZF_1_4_L10 by auto
  then have $\forall x \in A. \ (x,\text{Maximum}(\text{IntegerOrder},A)) \in \text{IntegerOrder}$
    by (rule int0.int_bounded_above_has_max)
  with A2 show $\lfloor p^R \cdot x \rfloor \leq \Gamma(S,p)$ by simp
qed

The candidate for supremum is an integer mapping with values given by $\Gamma$.

lemma (in real1) Real_ZF_1_4_L12:
assumes A1: $\text{IsBoundedAbove}(S,\text{OrderOnReals})$ $S \neq 0$ and
A2: $g = \{\langle p,\Gamma(S,p) \rangle. \ p \in \mathbb{Z}^+\}$
shows $g : \mathbb{Z}^+ \rightarrow \text{int}$
$\forall n \in \mathbb{Z}^+. \ g(n) = \Gamma(S,n)$
proof -
  from A1 have $\forall n \in \mathbb{Z}^+. \ \Gamma(S,n) \in \text{int}$ using Real_ZF_1_4_L10
    by simp
  with A2 show I: $g : \mathbb{Z}^+ \rightarrow \text{int}$ using ZF_fun_from_total by simp
  { fix $n$ assume $n \in \mathbb{Z}^+$
    with A2 I have $g(n) = \Gamma(S,n)$ using ZF_fun_from_tot_val
    by simp
  }
  then show $\forall n \in \mathbb{Z}^+. \ g(n) = \Gamma(S,n)$ by simp
qed

Every integer is equal to the floor of its embedding.

lemma (in real1) Real_ZF_1_4_L14: assumes A1: $m \in \text{int}$
shows $\lfloor m^R \rfloor = m$
proof -
  let $A = \{n \in \text{int}. \ n^R \leq m^R \}$
  have antisym($\text{IntegerOrder}$) using int0.Int_ZF_2_L4
    by simp
  moreover from A1 have $m \in A$

Floor of (real) zero is (integer) zero.

lemma (in real1) floor_01_is_zero_one: shows ⌊0⌋ = 0

proof -
  have ⌊(0Z)R⌋ = 0Z and ⌊(1Z)R⌋ = 1Z
  using int0.int_zero_one_are_int Real_ZF_1_4_L14
  by simp
  then show ⌊0⌋ = 0Z and ⌊1⌋ = 1Z
  using int_0_1_are_real_zero_one
  by simp

qed

Floor of (real) two is (integer) two.

lemma (in real1) floor_2_is_two: shows ⌊2⌋ = 2

proof -
  have ⌊(2Z)R⌋ = 2Z
  using int0.int_two_three_are_int Real_ZF_1_4_L14
  by simp
  then show ⌊2⌋ = 2Z using int_two_is_real_two
  by simp

qed

Floor of a product of embeddings of integers is equal to the product of integers.

lemma (in real1) Real_ZF_1_4_L14A: assumes A1: m ∈ int k ∈ int
  shows ⌊mR·kR⌋ = m·k

proof -
  from A1 have T: m·k ∈ int
  using int0.Int_ZF_1_1_L5 by simp
  from A1 have ⌊mR·kR⌋ = ⌊(m·k)R⌋ using Real_ZF_1_4_L14C
  by simp
  with T show ⌊mR·kR⌋ = m·k using Real_ZF_1_4_L14
  by simp

qed

Floor of the sum of a number and the embedding of an integer is the floor of the number plus the integer.

lemma (in real1) Real_ZF_1_4_L15: assumes A1: x ∈ R and A2: p ∈ int
  shows ⌊x + pR⌋ = ⌊x⌋ + p

proof -
  let A = {n ∈ int. nR ≤ x + pR}

765
have antisym(IntegerOrder) using int0.Int_ZF_2_L4 by simp
moreover have \(|x| + p \in A\)
proof -
  from A1 A2 have \(|x|^R \leq x\) and \(p^R \in \mathbb{R}\)
    using Real_ZF_1_4_L7 real_int_is_real by auto
  then have \(|x|^R + p^R \leq x + p^R\)
    using add_num_to_ineq by simp
  moreover from A1 A2 have \((|x| + p)^R = |x|^R + p^R\)
    using Real_ZF_1_4_L7 Real_ZF_1_4_L1A by simp
  ultimately have \((|x| + p)^R \leq x + p^R\)
    by simp
  moreover from A1 A2 have \(|x| + p \in \text{int}\)
    using Real_ZF_1_4_L7 int0.Int_ZF_1_1_L5 by simp
  ultimately show \(|x| + p \in A\) by auto
qed
moreover have \(\forall n \in A.\ n \leq |x| + p\)
proof
  fix n assume n\in A
  then have I: \(n \in \text{int}\) and \(n^R \leq x + p^R\)
    by auto
  with A1 A2 have \(n^R - p^R \leq x\)
    using real_int_is_real Real_ZF_1_2_L19 by simp
  with A2 I have \(|(n-p)^R| \leq |x|\)
    using Real_ZF_1_4_L1B Real_ZF_1_4_L9 by simp
  moreover from A2 I have \(n-p \in \text{int}\)
    using int0.Int_ZF_1_1_L5 by simp
  then have \(|(n-p)^R| = n-p\)
    using Real_ZF_1_4_L14 by simp
  ultimately have \(n-p \leq |x|\)
    by simp
  with A2 I show \(n \leq |x| + p\)
    using int0.Int_ZF_2_L9C by simp
  qed
ultimately show \(|x + p^R| = |x| + p\)
  using Order_ZF_4_L14 by auto
qed

Floor of the difference of a number and the embedding of an integer is the
floor of the number minus the integer.

lemma (in real1) Real_ZF_1_4_L16: assumes A1: \(x \in \mathbb{R}\) and A2: \(p \in \text{int}\)
shows \(|x - p^R| = |x| - p\)
proof -
  from A2 have \(|x - p^R| = |x + (-p)|^R\)
    using Real_ZF_1_4_L1 by simp
  with A1 A2 show \(|x - p^R| = |x| - p\)
The floor of sum of embeddings is the sum of the integers.

**Lemma** (in real1) RealZF_1_4_L17: assumes \( m \in \mathbb{Z} \) \( n \in \mathbb{Z} \)
shows \( \lfloor (mR + nR) \rfloor = m + n \)
using assms real_int_is_real RealZF_1_4_L15 RealZF_1_4_L14
by simp

A lemma about adding one to floor.

**Lemma** (in real1) RealZF_1_4_L17A: assumes \( a \in \mathbb{R} \)
shows \( 1 + \lfloor a \rfloor \leq 1 \)
proof -
have \( 1 + |a|^R = 1Z^R + |a|^R \)
  using int0.int_zero_one_are_real_zero_one by simp
with A1 show \( 1 + |a|^R = (1Z + |a|)^R \)
  using int0.int_zero_one_are_int RealZF_1_4_L7 RealZF_1_4_L1A
by simp
qed

The difference between the a number and the embedding of its floor is
(strictly) less than one.

**Lemma** (in real1) RealZF_1_4_L17B: assumes \( a \in \mathbb{R} \)
shows \( a - \lfloor a \rfloor < 1 \)
proof -
from A1 have T1: \( |a| \in \mathbb{Z} \) \( |a|^R \in \mathbb{R} \) and
T2: \( 1 \in \mathbb{R} \) \( a - |a|^R \in \mathbb{R} \)
using RealZF_1_4_L7 real_int_is_real RealZF_1_4_L6 RealZF_1_4_L4
by auto
{ assume \( 1 \leq a - |a|^R \)
  with A1 T1 have \( |a|^R + |a|^R \leq |a| \)
    using RealZF_1_2_L21 RealZF_1_4_L9 int0.int_zero_one_are_real_zero_one
    by simp
  with T1 have False
    using int0.int_zero_one_are_int RealZF_1_4_L17
int0.IntZF_1_2_L3AA by simp
} then have I: \( \neg(1 \leq a - |a|^R) \) by auto
with T2 show II: \( a - |a|^R < 1 \)
  by (rule RealZF_1_2_L20)
with A1 T1 I II have
\( a < 1 + |a|^R \)
  using RealZF_1_2_L26 by simp
with A1 show \( a < (1Z + |a|)^R \)
  using RealZF_1_4_L17A by simp
qed

The next lemma corresponds to Lemma 14 iii) in [2]. It says that we can
find a rational number between any two different real numbers.

**lemma (in real1) Arthan_Lemma14iii:** assumes $A1: x<y$
shows $\exists M \in \mathbb{Z}^+. \ x \cdot N_R < M \land M < y \cdot N_R$

**proof** -
from $A1$ have $(y-x)^{-1} \in \mathbb{R}_+$ using Real_ZF_1_3_L3
by simp
then have $\exists N \in \mathbb{Z}_+. \ (y-x)^{-1} < N_R$
using Arthan_Lemma14i PositiveSet_def by simp
then obtain $N$ where I: $N \in \mathbb{Z}_+$ and II: $(y-x)^{-1} < N_R$
by auto
let $M = 1_{\mathbb{Z}} + \lfloor x \cdot N \rfloor$
from $A1$ I have $T1: x \in \mathbb{R}^+$
using Real_ZF_1_4_L7 Real_ZF_1_2_L16 Real_ZF_1_2_L17
by simp
moreover from $A1$ II have $M \leq \lfloor 1 + x \cdot N \rfloor$
using Real_ZF_1_4_L17A by simp
ultimately have $M < y \cdot N_R$
by (rule real_strict_ord_transit)
with I T2 III show thesis by auto
qed

Some estimates for the homomorphism difference of the floor function.

**lemma (in real1) Real_ZF_1_4_L18:** assumes $A1: x \in \mathbb{R} \ y \in \mathbb{R}$
shows $\text{abs}(|x+y| - |x| - |y|) \leq 2_{\mathbb{Z}}$

**proof** -
from $A1$ have T:
$|x|^R \in \mathbb{R} \ |y|^R \in \mathbb{R}$
$x+y - (|x|^R) \in \mathbb{R}$
using Real_ZF_1_4_L7 real_int_is_real Real_ZF_1_1_L6
by auto
from $A1$ have
$0 \leq x - (|x|^R) + (y - (|y|^R))$
$x - (|x|^R) + (y - (|y|^R)) \leq 2$
using Real_ZF_1_4_L7 Real_ZF_1_2_L16 Real_ZF_1_2_L17
Real_ZF_1_4_L17B Real_ZF_1_2_L18 by auto
moreover from $A1$ T have
\[ x - (|x|^R) + (y - (|y|^R)) = x+y - (|x|^R) - (|y|^R) \]

using `Real_ZF_1_L7A` by simp

ultimately have

\[ 0 \leq x+y - (|x|^R) - (|y|^R) \]

by auto

then have

\[ \lfloor 0 \rfloor \leq |x+y - (|x|^R) - (|y|^R)| \]

\[ |x+y - (|x|^R) - (|y|^R)| \leq 2 \]

using `Real_ZF_1_4_L9` by auto

then have

\[ 0 \leq |x+y - (|x|^R) - (|y|^R)| \]

\[ |x+y - (|x|^R) - (|y|^R)| \leq 2 \]

by auto

then show \( |x+y - |x| - |y|| \leq 2 \)

using `floor_01_is_zero_one floor_2_is_two` by auto

moreover from `A1` have

\[ |x+y - (|x|^R) - (|y|^R)| = |x+y| - |x| - |y| \]

using `Real_ZF_1_L6 Real_ZF_1_4_L7 real_int_is_real Real_ZF_1_4_L16` by simp

ultimately have

\[ 0 \leq |x+y - |x| - |y|| \]

\[ |x+y - |x| - |y|| \leq 2 \]

by auto

then show \( |x+y - |x| - |y|| \leq 2 \)

using `int0.Int_ZF_2_L16` by simp

qed

Suppose \( S \neq \emptyset \) is bounded above and \( \Gamma(S, m) = |m R \cdot x| \) for some positive integer \( m \) and \( x \in S \). Then if \( y \in S, x \leq y \) we also have \( \Gamma(S, m) = |m R \cdot y| \).

**lemma (in real1) Real_ZF_1_4_L20:**

assumes `A1: IsBoundedAbove(S,OrderOnReals)` \( S \neq \emptyset \) and

`A2: n\in\mathbb{Z}. \ x\in S` and

`A3: \Gamma(S,n) = |n R \cdot x|` and

`A4: y\in S \ x\leq y` shows \( \Gamma(S, n) = |n R \cdot y| \)

**proof -**

from `A2 A4` have \( |n R \cdot x| \leq |(n R) y| \)

using `int_pos_is_real_pos Real_ZF_1_2_L14 Real_ZF_1_4_L9` by simp

with `A3` have \( \langle \Gamma(S,n), |(n R) y| \rangle \in \text{IntegerOrder} \)

by simp

moreover from `A1 A2 A4` have \( \langle |n R \cdot y|, \Gamma(S, n) \rangle \in \text{IntegerOrder} \)

using `Real_ZF_1_4_L11` by simp

ultimately show \( \Gamma(S, n) = |n R \cdot y| \)

by `(rule int0.Int_ZF_2_L3)`

qed

The homomorphism difference of \( n \mapsto \Gamma(S, n) \) is bounded by 2 on positive integers.

**lemma (in real1) Real_ZF_1_4_L21:**
assumes \( A1: \text{IsBoundedAbove}(S,\text{OrderOnReals}) \quad S \neq 0 \) and
\( A2: m \in \mathbb{Z}_+. \quad n \in \mathbb{Z}_+ \)
shows \( \left| \Gamma(S,m+n) - \Gamma(S,m) - \Gamma(S,n) \right| \leq 2Z \)

**proof** -
from \( A2 \) have \( m+n \in \mathbb{Z}_+ \) using \texttt{int0.pos_int_closed_add_unfolded}
by simp
with \( A1 \) \( A2 \) have
\( \Gamma(S,m) \in \{ [m^R.x] . x \in S \} \) and
\( \Gamma(S,n) \in \{ [n^R.x] . x \in S \} \) and
\( \Gamma(S,m+n) \in \{ [(m+n)^R.x] . x \in S \} \)
using \texttt{RealZF_1_4_L10} by auto
then obtain a b c where I: \( a \in S \quad b \in S \quad c \in S \)
and II:
\( \Gamma(S,m) = [m^R.a] \)
\( \Gamma(S,n) = [n^R.b] \)
\( \Gamma(S,m+n) = [(m+n)^R.c] \)
by auto
let \( d = \text{Maximum}(\text{OrderOnReals},\{a,b,c\}) \)
from \( A1 \) have \( a \in R \quad b \in R \quad c \in R \)
using \texttt{RealZF_1_2_L23} by auto
then have IV:
\( d \in \{a,b,c\} \)
\( d \in R \)
\( a \leq d \)
\( b \leq d \)
\( c \leq d \)
using \texttt{RealZF_1_2_L24} by auto
with I have V: \( d \in S \) by auto
from \( A1 \) \( T \) I II IV V have \( \Gamma(S,m+n) = [(m+n)^R.d] \)
using \texttt{RealZF_1_4_L20} by blast
also from \( A2 \) have \( ... = [(m^R)+(n^R)] . d \)
using \texttt{RealZF_1_4_L1A PositiveSet_def} by simp
also from \( A2 \) IV have \( ... = [(m^R).d + (n^R).d] \)
using \texttt{PositiveSet_def real_int_is_real RealZF_1_L7} by simp
finally have \( \Gamma(S,m+n) = [(m^R).d + (n^R).d] \)
by simp
moreover from \( A1 \) \( A2 \) I II IV V have \( \Gamma(S,m) = [m^R.d] \)
using \texttt{RealZF_1_4_L20} by blast
moreover from \( A1 \) \( A2 \) I II IV V have \( \Gamma(S,n) = [n^R.d] \)
using \texttt{RealZF_1_4_L20} by blast
moreover from \( A1 \) \( T \) I II IV V have \( \Gamma(S,m+n) = [(m+n)^R.d] \)
using \texttt{RealZF_1_4_L20} by blast
ultimately have \( \left| \Gamma(S,m+n) - \Gamma(S,m) - \Gamma(S,n) \right| \leq \left| [(m^R).d + (n^R).d] - [m^R.d] - [n^R.d] \right| \)
by simp
with \( A2 \) IV show
\( \left| \Gamma(S,m+n) - \Gamma(S,m) - \Gamma(S,n) \right| \leq 2Z \)
using \texttt{PositiveSet_def real_int_is_real RealZF_1_L6}

770
The next lemma provides sufficient condition for an odd function to be an almost homomorphism. It says for odd functions we only need to check that the homomorphism difference (denoted $\delta$ in the real1 context) is bounded on positive integers. This is really proven in Int_ZF_2.thy, but we restate it here for convenience. Recall from Group_ZF_3.thy that OddExtension of a function defined on the set of positive elements (of an ordered group) is the only odd function that is equal to the given one when restricted to positive elements.

**Lemma (in real1) Real_ZF_1_4_L21A:**

assumes A1: $f : \mathbb{Z}_+ \rightarrow \mathbb{Z}$

$\forall a \in \mathbb{Z}_+. \forall b \in \mathbb{Z}_+. \text{abs}(\delta(f,a,b)) \leq L$

shows $\text{OddExtension}(\mathbb{Z},\text{IntegerAddition},\text{IntegerOrder},f) \in S$

using A1 int1.Int_ZF_2_1_L24 by auto

**Theorem (in real1) Real_ZF_1_4_L22:**

assumes A1: IsBoundedAbove($S$,OrderOnReals) $S \neq 0$ and

A2: $g = \{\langle p,\Gamma(S,p)\rangle. p \in \mathbb{Z}_+\}$

shows $\text{OddExtension}(\mathbb{Z},\text{IntegerAddition},\text{IntegerOrder},g) \in S$

proof -

from A1 A2 have $g : \mathbb{Z}_+ \rightarrow \mathbb{Z}$ by (rule Real_ZF_1_4_L12)

moreover have $\forall m \in \mathbb{Z}_+. \forall n \in \mathbb{Z}_+. \text{abs}(\delta(g,m,n)) \leq 2Z$

proof -

{ fix $m$ $n$
  assume A3: $m \in \mathbb{Z}_+ \ n \in \mathbb{Z}_+$
  then have $m+n \in \mathbb{Z}_+ \ m \in \mathbb{Z}_+ \ n \in \mathbb{Z}_+$
  using int0.pos_int_closed_add_unfolded by auto

moreover from A1 A2 have $\forall n \in \mathbb{Z}_+. g(n) = \Gamma(S,n)$
  by (rule Real_ZF_1_4_L12)

ultimately have $\delta(g,m,n) = \Gamma(S,m+n) - \Gamma(S,m) - \Gamma(S,n)$
  by simp

moreover from A1 A3 have

$\text{abs}(\Gamma(S,m+n) - \Gamma(S,m) - \Gamma(S,n)) \leq 2Z$

by (rule Real_ZF_1_4_L21)

ultimately have $\text{abs}(\delta(g,m,n)) \leq 2Z$

by simp

} then show $\forall m \in \mathbb{Z}_+. \forall n \in \mathbb{Z}_+. \text{abs}(\delta(g,m,n)) \leq 2Z$

by simp

qed

ultimately show thesis by (rule Real_ZF_1_4_L21A)

qed

A technical lemma used in the proof that all elements of $S$ are less or equal than the candidate for supremum of $S$.

**Lemma (in real1) Real_ZF_1_4_L23:**
assumes A1: \( f \in S \) and A2: \( N \in \text{int} \) \( M \in \text{int} \) and
A3: \( \forall n \in \mathbb{Z}_+. M \cdot n \leq f(N \cdot n) \)
shows \( M^R \leq [f] \cdot (N^R) \)

proof -
let \( M_S = \{ (n, M \cdot n) \cdot n \in \text{int} \} \)
let \( N_S = \{ (n, N \cdot n) \cdot n \in \text{int} \} \)
from A1 A2 have T: \( M_S \in S \) \( N_S \in S \) \( f \circ N_S \in S \)
using int1.Int_ZF_2_5_L1 int1.Int_ZF_2_1_L11 SlopeOp2_def
by auto
moreover from A1 A2 A3 have \( f \circ N_S \sim M_S \lor M_S + (- (f \circ N_S)) \in S_+ \)
using int1.Int_ZF_2_5_L8 SlopeOp2_def SlopeOp1_def Slopes_def
BoundedIntMaps_def SlopeEquivalenceRel_def PositiveIntegers_def
PositiveSlopes_def by simp
ultimately have \( [M_S] \leq [f \circ N_S] \) using Real_ZF_1_2_L12
by simp
with A1 T show \( M^R \leq [f] \cdot (N^R) \) using Real_ZF_1_1_L4
by simp
qed

A technical lemma aimed used in the proof the candidate for supremum of \( S \) is less or equal than any upper bound for \( S \).

lemma (in real1) Real_ZF_1_4_L23A:
assumes A1: \( f \in S \) and A2: \( N \in \text{int} \) \( M \in \text{int} \) and
A3: \( \forall n \in \mathbb{Z}_+. M \cdot n \leq f(N \cdot n) \)
shows \( [f] \cdot (N^R) \leq M^R \)

proof -
let \( M_S = \{ (n, M \cdot n) \cdot n \in \text{int} \} \)
let \( N_S = \{ (n, N \cdot n) \cdot n \in \text{int} \} \)
from A1 A2 have T: \( M_S \in S \) \( N_S \in S \) \( f \circ N_S \in S \)
using int1.Int_ZF_2_5_L1 int1.Int_ZF_2_1_L11 SlopeOp2_def
by auto
moreover from A1 A2 A3 have \( f \circ N_S \sim M_S \lor M_S + (- (f \circ N_S)) \in S_+ \)
using int1.Int_ZF_2_5_L9 SlopeOp2_def SlopeOp1_def Slopes_def
BoundedIntMaps_def SlopeEquivalenceRel_def PositiveIntegers_def
PositiveSlopes_def by simp
ultimately have \( [f \circ N_S] \leq [M_S] \) using Real_ZF_1_2_L12
by simp
with A1 T show \( [f] \cdot (N^R) \leq M^R \) using Real_ZF_1_1_L4
by simp
qed

The essential condition to claim that the candidate for supremum of \( S \) is greater or equal than all elements of \( S \).

lemma (in real1) Real_ZF_1_4_L24:
assumes A1: \( \text{IsBoundedAbove}(S, \text{OrderOnReals}) \) and
A2: \( x < y \quad y \in S \) and
A4: \( N \in \mathbb{Z}_+ \) \( M \in \text{int} \) and
A5: \( M^R < y \cdot N^R \) and A6: \( p \in \mathbb{Z}_+ \)
shows \( p \cdot M \leq \Gamma(S,p \cdot N) \)

proof -

from A2 A4 A6 have T1:

\[ N^R \in \mathbb{R}_+ \quad y \in \mathbb{R} \quad p^R \in \mathbb{R}_+ \]

\[ pN \in \mathbb{Z}_+ \quad (p \cdot N)^R \in \mathbb{R}_+ \]

using int_pos_is_real_pos Real_ZF_1_2_L15

int0.pos_int_closed_mul_unfold by auto

with A4 A6 have T2:

\[ p \in \mathbb{Z} \quad p^R \in \mathbb{R} \quad N^R \in \mathbb{R} \quad N^R \neq 0 \quad M^R \in \mathbb{R} \]

using real_int_is_real PositiveSet_def by auto

from T1 A5 have \[ \lfloor (p \cdot N)^R \cdot (M^R \cdot (N^R)^{-1}) \rfloor \leq \lfloor (p \cdot N)^R \cdot y \rfloor \]

using Real_ZF_1_3_L4A Real_ZF_1_3_L7 Real_ZF_1_4_L9 by simp

moreover from A1 A2 T1 have \[ \lfloor (p \cdot N)^R \cdot y \rfloor \leq \Gamma(S,p \cdot N) \]

using Real_ZF_1_4_L11 by simp

ultimately have I:

\[ \lfloor (p \cdot N)^R \cdot (M^R \cdot (N^R)^{-1}) \rfloor \leq \Gamma(S,p \cdot N) \]

by (rule int_order_transitive)

from A4 A6 have \[ (p \cdot N)^R \cdot (M^R \cdot (N^R)^{-1}) = pN \cdot N^R \cdot (M^R \cdot (N^R)^{-1}) \]

using PositiveSet_def Real_ZF_1_4_L1C by simp

with A4 T2 have \[ \lfloor (p \cdot N)^R \cdot (M^R \cdot (N^R)^{-1}) \rfloor = p \cdot M \]

using Real_ZF_1_3_L8 Real_ZF_1_4_L14A by simp

with I show \[ p \cdot M \leq \Gamma(S,p \cdot N) \]

by simp

qed

An obvious fact about odd extension of a function \( p \mapsto \Gamma(s,p) \) that is used
a couple of times in proofs.

lemma (in real1) Real_ZF_1_4_L24A:

assumes A1: IsBoundedAbove(S,OrderOnReals) \( S \neq 0 \) and A2: \( p \in \mathbb{Z}_+ \)

and A3:

\( h = \text{OddExtension}(\mathbb{Z},\mathbb{Z},\text{IntegerOrder},\{\langle p,\Gamma(S,p) \rangle. p \in \mathbb{Z}_+\}) \)

shows \( h(p) = \Gamma(S,p) \)

proof -

let \( g = \{\langle p,\Gamma(S,p) \rangle. p \in \mathbb{Z}_+\} \)

from A1 have I: \( g : \mathbb{Z}_+ \rightarrow \mathbb{Z} \)

using Real_ZF_1_4_L12

by blast

with A2 A3 show \( h(p) = \Gamma(S,p) \)

using int0.Int_ZF_1_5_L11 ZF_fun_from_tot_val by simp

qed

The candidate for the supremum of \( S \) is not smaller than any element of \( S \).

lemma (in real1) Real_ZF_1_4_L25:

assumes A1: IsBoundedAbove(S,OrderOnReals) and

A2: \( \neg\text{HasAmaximum(OrderOnReals,S)} \) and

A3: \( x \in S \) and A4:

\( h = \text{OddExtension}(\mathbb{Z},\mathbb{Z},\text{IntegerOrder},\{\langle p,\Gamma(S,p) \rangle. p \in \mathbb{Z}_+\}) \)

shows \( x \leq [h] \)

proof -

from A1 A2 A3 have

\[ x \leq [h] \]

773
S ⊆ ℝ  ¬HasAmaximum(OrderOnReals,S)  \( x \in S \)
using Real_ZF_1_2_L23 by auto

then have \( \exists y \in S. \ x < y \) by (rule Real_ZF_1_2_L27)

then obtain \( y \) where I: \( y \in S \) and II: \( x < y \)
by auto

from II have
\( \exists M \in \text{int}. \ \exists N \in \mathbb{Z}_+. \ x \cdot N^R < M^R \wedge M^R < y \cdot N^R \)
using Arthan_Lemma14iii by simp

then obtain \( M \) \( N \) where III: \( M \in \text{int} \) \( N \in \mathbb{Z}_+ \) and
IV: \( x \cdot N^R < M^R \) \( M^R < y \cdot N^R \)
by auto

from II III IV have \( V: x \leq \frac{M^R}{N^R} \)
using int_pos_is_real_pos Real_ZF_1_2_L15 Real_ZF_1_3_L4
by auto

from A3 have VI: \( S \neq 0 \) by auto

with A1 A4 have T1: \( h \in S \) using Real_ZF_1_4_L22
by simp

moreover from III have \( \forall n \in \mathbb{Z}_+. \ N \cdot n \leq h(N \cdot n) \)
proof
let \( g = \{(p,Γ(S,p)): p \in \mathbb{Z}_+\} \)
fix \( n \) assume A5: \( n \in \mathbb{Z}_+ \)
with III have T2: \( n \cdot N \in \mathbb{Z}_+ \)
using int0.pos_int_closed_mul_unfold by simp

from III A5 have
\( N \cdot n = n \cdot N \) and \( n \cdot M = M \cdot n \)
using PositiveSet_def int0.Int_ZF_1_1_L5 by auto

moreover from A1 I II III IV A5 have
IsBoundedAbove(S,OrderOnReals)
\( \forall y \in S \)
\( N \in \mathbb{Z}_+ \) \( M \in \text{int} \)
\( M^R < y \cdot N^R \) \( n \in \mathbb{Z}_+ \)
by auto

then have n:M ≤ Γ(S,n:n) by (rule Real_ZF_1_4_L24)

moreover from A1 A4 VI T2 have h(N:n) = Γ(S,N:n)
using Real_ZF_1_4_L24A by simp

ultimately show M:n ≤ h(N:n) by auto
qed

ultimately have \( M^R \leq \frac{[h] \cdot N^R}{[h]} \) using Real_ZF_1_4_L23
by simp

with III T1 have \( M^R \cdot (N^R)^{-1} \leq \frac{[h]}{(h)} \)
using int_pos_is_real_pos Real_ZF_1_2_L13 Real_ZF_1_3_L4B
by simp

with V show \( x \leq \frac{[h]}{(h)} \) by (rule real_ord_transitive)
qed

The essential condition to claim that the candidate for supremum of \( S \) is
less or equal than any upper bound of \( S \).

lemma (in real1) Real_ZF_1_4_L26:
  assumes A1: IsBoundedAbove(\( S \),OrderOnReals) and
  A2: \( x \leq y \) \( x \in S \) and
  A4: \( N \in \mathbb{Z}_+ \) \( M \in \text{int} \) and
  A5: \( y \cdot N^R < M^R \) and A6: \( p \in \mathbb{Z}_+ \)
  shows \( \lfloor (N \cdot p)^R \cdot x \rfloor \leq M \cdot p \)
proof -
  from A2 A4 A6 have T:
  \( p \cdot N \in \mathbb{Z}_+ \) \( p \in \text{int} \) \( N \in \text{int} \)
  \( p^R \in \mathbb{R}_+ \) \( p^R \in \mathbb{R} \) \( N^R \in \mathbb{R} \) \( x \in \mathbb{R} \) \( y \in \mathbb{R} \)
  using int0.pos_int_closed_mul_unfold PositiveSet_def
  real_int_is_real Real_ZF_1_2_L15 int_pos_is_real_pos
  by auto
  with A2 have \( (p \cdot N)^R \cdot x \leq (p \cdot N)^R \cdot y \)
  using int_pos_is_real_pos Real_ZF_1_2_L14A
  by simp
  moreover from A4 T have I:
  \( (p \cdot N)^R = p^R \cdot N^R \)
  \( (p \cdot M)^R = p^R \cdot M^R \)
  using Real_ZF_1_4_L1C by auto
  ultimately have \( (p \cdot N)^R \cdot x \leq p^R \cdot N^R \cdot y \)
  by simp
  moreover
  from A5 T I have \( p^R \cdot (y \cdot N^R) < (p \cdot M)^R \)
  using Real_ZF_1_3.L7 by simp
  with T have \( p^R \cdot N^R \cdot y < (p \cdot M)^R \) using Real_ZF_1_1.L9
  by simp
  ultimately have \( (p \cdot N)^R \cdot x < (p \cdot M)^R \)
  by (rule real_strict_ord_transit)
  then have \( \lfloor (p \cdot N)^R \cdot x \rfloor \leq \lfloor (p \cdot M)^R \rfloor \)
  using Real_ZF_1_4.L9 by simp
  moreover
  from A4 T have \( p \cdot M \in \text{int} \) using Int_ZF_1_1_L5
  by simp
  then have \( \lfloor (p \cdot M)^R \rfloor = p \cdot M \) using Real_ZF_1_4.L14
  by simp
  moreover from A4 A6 have \( p \cdot N = N \cdot p \) and \( p \cdot M = M \cdot p \)
  using PositiveSet_def int0.Int_ZF_1_1_L5 by auto
  ultimately show \( \lfloor (N \cdot p)^R \cdot x \rfloor \leq M \cdot p \) by simp
qed

A piece of the proof of the fact that the candidate for the supremum of \( S \)
is not greater than any upper bound of \( S \), done separately for clarity (of mind).

lemma (in real1) Real_ZF_1_4_L27:
  assumes IsBoundedAbove(\( S \),OrderOnReals) \( S \neq 0 \) and
  \( h = \text{OddExtension}(\text{int},\text{IntegerAddition},\text{IntegerOrder},\{p,\Gamma(S,p). p \in \mathbb{Z}_+\}) \)
  and \( p \in \mathbb{Z}_+ \)

775
shows $\exists x \in S. h(p) = \lfloor p^R \cdot x \rfloor$
using assms Real_ZF_1_4_L10 Real_ZF_1_4_L24A by auto

The candidate for the supremum of $S$ is not greater than any upper bound of $S$.

lemma (in real1) Real_ZF_1_4_L28:
assumes A1: IsBoundedAbove($S$,OrderOnReals) $S \neq 0$
and A2: $\forall x \in S. x \leq y$ and A3:
h = OddExtension(int,IntegerAddition,IntegerOrder,$\{\langle p, \Gamma(S,p) \rangle . p \in \mathbb{Z}_+\}$)
shows $[h] \leq y$
proof -
from A1 obtain a where $a \in S$ by auto
with A1 A2 A3 have $T$: $y \in \mathcal{R} \cdot h \in S$ $[h] \in \mathcal{R}$
by auto
{ assume $\neg ([h] \leq y)$
with $T$ have $y < [h]$ using Real_ZF_1_2_L28
by blast
then have $\exists M \in \text{int} \cdot \exists N \in \mathbb{Z}_+ . \ y^R < M^R \land M^R < [h] \cdot N^R$
using Arthan_Lemma14iii by simp
then obtain M N where $\text{I: } M \in \text{int}$ $N \in \mathbb{Z}_+$ and $\text{II: } y^R < M^R \land M^R < [h] \cdot N^R$
by auto
from I have III: $N^R \in \mathcal{R}_+$ using int_pos_is_real_pos
by simp
have $\forall p \in \mathbb{Z}_+. \ h(N \cdot p) \leq M \cdot p$
proof
fix $p$ assume A4: $p \in \mathbb{Z}_+$
with A1 A3 I have $\exists x \in S. h(p) = \lfloor (N \cdot p)^R \cdot x \rfloor$
using int0.pos_int_closed_mul_unfold Real_ZF_1_4_L27
by simp
with A1 A2 I II A4 show $h(N \cdot p) \leq M \cdot p$
using Real_ZF_1_4_L26 by auto
qed
with $T$ I have $[h] \cdot N^R \leq M^R$
using PositiveSet_def Real_ZF_1_4_L23A
by simp
with $T$ III have $[h] \leq M^R \cdot (N^R)^{-1}$
using Real_ZF_1_3_L4C by simp
moreover from $T$ II III have $M^R \cdot (N^R)^{-1} < [h]$
using Real_ZF_1_3_L4A by simp
ultimately have False using Real_ZF_1_2_L29 by blast
} then show $[h] \leq y$ by auto
qed

Now we can prove that every nonempty subset of reals that is bounded above has a supremum. Proof by considering two cases: when the set has a maximum and when it does not.

lemma (in real1) real_order_complete:
assumes A1: IsBoundedAbove(S,OrderOnReals) S≠0
shows HasAminimum(OrderOnReals,⋂a∈S. OrderOnReals{a})

proof -
{ assume HasAmaximum(OrderOnReals,S)
  with A1 have HasAminimum(OrderOnReals,⋂a∈S. OrderOnReals{a})
    using Real_ZF_1_2_L10 IsAnOrdGroup_def IsPartOrder_def
Order_ZF_S_L6 by simp }
moreover
{ assume A2: ¬HasAmaximum(OrderOnReals,S)
  let h = OddExtension(int,IntegerAddition,IntegerOrder,{⟨p,Γ(S,p)⟩. p∈ınt})
  let r = OrderOnReals
  from A1 have antisym(OrderOnReals) S≠0
    using Real_ZF_1_4_L25 by simp
moreover from A1 A2 have ∀x∈S. ⟨x,⟨h⟩⟩ ∈ r
  using Real_ZF_1_4_L28 by simp
moreover from A1 have ∀y. (∀x∈S. ⟨x,y⟩ ∈ r) −→ ⟨⟨h⟩,y⟩ ∈ r
  using Real_ZF_1_4_L28 by simp
ultimately have HasAminimum(OrderOnReals,⋂a∈S. OrderOnReals{a})
    by (rule Order_ZF_5_L5)
}
ultimately show thesis by blast

qed

Finally, we are ready to formulate the main result: that the construction of real numbers from the additive group of integers results in a complete ordered field. This theorem completes the construction. It was fun.

theorem eudoxus_reals_are_reals: shows IsAmodelOfReals(RealNumbers,RealAddition,RealMultiplication,OrderOnReals)
using real1.reals_are_ord_field real1.real_order_complete
IsComplete_def IsAmodelOfReals_def by simp

end

60 Topology - introduction

theory Topology_ZF imports ZF1 Finite_ZF Fol1

begin

This theory file provides basic definitions and properties of topology, open and closed sets, closure and boundary.

60.1 Basic definitions and properties

A typical textbook defines a topology on a set $X$ as a collection $T$ of subsets of $X$ such that $X \in T$, $\emptyset \in T$ and $T$ is closed with respect to arbitrary unions and intersection of two sets. One can notice here that since we always
have $\bigcup T = X$, the set on which the topology is defined (the "carrier" of the topology) can always be constructed from the topology itself and is superfluous in the definition. Moreover, as Marnix Klooster pointed out to me, the fact that the empty set is open can also be proven from other axioms. Hence, we define a topology as a collection of sets that is closed under arbitrary unions and intersections of two sets, without any mention of the set on which the topology is defined. Recall that $\text{Pow}(T)$ is the powerset of $T$, so that if $M \in \text{Pow}(T)$ then $M$ is a subset of $T$. The sets that belong to a topology $T$ will be sometimes called "open in" $T$ or just "open" if the topology is clear from the context.

Topology is a collection of sets that is closed under arbitrary unions and intersections of two sets.

definition
  IsATopology (_ {is a topology} [90] 91) where
  T {is a topology} $\equiv$ ( $\forall M \in \text{Pow}(T)$. $\bigcup M \in T$ ) $\land$
  ( $\forall U \in T$. $\forall V \in T$. $U \cap V \in T$)

We define interior of a set $A$ as the union of all open sets contained in $A$. We use $\text{Interior}(A,T)$ to denote the interior of $A$.

definition
  Interior(A,T) $\equiv$ $\bigcup \{ U \in T. U \subseteq A \}$

A set is closed if it is contained in the carrier of topology and its complement is open.

definition
  IsClosed (infixl {is closed in} 90) where
  D {is closed in} T $\equiv$ (D $\subseteq \bigcup T \land \bigcup T - D \in T$)

To prove various properties of closure we will often use the collection of closed sets that contain a given set $A$. Such collection does not have a separate name in informal math. We will call it $\text{ClosedCovers}(A,T)$.

definition
  ClosedCovers(A,T) $\equiv$ \{ D $\in \text{Pow}(\bigcup T)$. D {is closed in} T $\land$ A$\subseteq$D \}

The closure of a set $A$ is defined as the intersection of the collection of closed sets that contain $A$.

definition
  Closure(A,T) $\equiv$ $\bigcap \text{ClosedCovers}(A,T)$

We also define boundary of a set as the intersection of its closure with the closure of the complement (with respect to the carrier).

definition
  Boundary(A,T) $\equiv$ $\text{Closure}(A,T) \cap \text{Closure}(\bigcup T - A,T)$
A set $K$ is compact if for every collection of open sets that covers $K$ we can choose a finite one that still covers the set. Recall that $\text{FinPow}(M)$ is the collection of finite subsets of $M$ (finite powerset of $M$), defined in IsarMathLib’s Finite_ZF theory.

**definition**

$\text{IsCompact (infixl \{is compact in\} 90)}$ where

$K \text{ is compact in } T \equiv (K \subseteq \bigcup T \land \forall M \in \text{Pow}(T). K \subseteq \bigcup M \longrightarrow (\exists N \in \text{FinPow}(M). K \subseteq \bigcup N)))$  

A basic example of a topology: the powerset of any set is a topology.

**lemma** $\text{Pow_is_top: shows Pow(X) is a topology}$

**proof**

- have $\forall A \in \text{Pow(Pow(X)). } \bigcup A \in \text{Pow(X)}$ by fast
- moreover have $\forall U \in \text{Pow(X). } \forall V \in \text{Pow(X). } U \cap V \in \text{Pow(X)}$ by fast
- ultimately show $\text{Pow(X) is a topology}$ using $\text{IsATopology_def}$ by auto

**qed**

Empty set is open.

**lemma** $\text{empty_open:}$

- assumes $T \text{ is a topology}$
- shows $0 \in T$

**proof**

- have $0 \in \text{Pow}(T)$ by simp
- with assms have $\bigcup 0 \in T$ using $\text{IsATopology_def}$ by blast
- thus $0 \in T$ by simp

**qed**

The carrier is open.

**lemma** $\text{carr_open:}$

- assumes $T \text{ is a topology}$
- shows $(\bigcup T) \in T$

**using** $\text{assms IsATopology_def}$ by auto

Union of a collection of open sets is open.

**lemma** $\text{union_open:}$

- assumes $T \text{ is a topology}$ and $\forall A \in A. A \in T$
- shows $(\bigcup A) \in T$

**using** $\text{assms IsATopology_def}$ by auto

Union of a indexed family of open sets is open.

**lemma** $\text{union_indexed_open:}$

- assumes $A1: T \text{ is a topology}$ and $A2: \forall i \in I. P(i) \in T$
- shows $(\bigcup i \in I. P(i)) \in T$

**using** $\text{assms union_open by simp}$

The intersection of any nonempty collection of topologies on a set $X$ is a topology.

**lemma** $\text{Inter_tops_is_top:}$

- assumes $A1: M \neq 0 \text{ and } A2: \forall T \in M. T \text{ is a topology}$
- shows $(\bigcap M) \text{ is a topology}$

**proof**

- { fix $A$ assume $A \in \text{Pow}(\bigcap M)$}

779
with $A_1$ have $\forall T \in \mathcal{M}. \ A \in \text{Pow}(T)$ by auto
with $A_1 \ A_2$ have $\bigcup A \in \bigcap \mathcal{M}$ using IsATopology_def by auto
} then have $\forall A. \ A \in \text{Pow}(\bigcap \mathcal{M}) \longrightarrow \bigcup A \in \bigcap \mathcal{M}$ by simp
hence $\forall A \in \text{Pow}(\bigcap \mathcal{M}). \ \bigcup A \in \bigcap \mathcal{M}$ by auto
moreover
{ fix $U$ $V$ assume $U \in \bigcap \mathcal{M}$ and $V \in \bigcap \mathcal{M}$
then have $\forall T \in \mathcal{M}. \ U \in T \land V \in T$ by auto
with $A_1 \ A_2$ have $\forall T \in \mathcal{M}. \ U \cap V \in T$ using IsATopology_def by simp
} then have $\forall U \in \bigcap \mathcal{M}. \ \forall V \in \bigcap \mathcal{M}. \ U \cap V \in \bigcap \mathcal{M}$ by auto
ultimately show $(\bigcap \mathcal{M}) \text{ (is a topology)}$
using IsATopology_def by simp
qed

Singletons are compact. Interestingly we do not have to assume that $T$ is a topology for this. Note singletons do not have to be closed, we need the the space to be $T_1$ for that (see Topology_ZF_1).

lemma singl_compact:
assumes $x \in \bigcup T$
shows $\{x\} \text{ (is compact in)} \ T$
using assms singleton_in_finpow unfolding IsCompact_def by auto

We will now introduce some notation. In Isar, this is done by defining a "locale". Locale is kind of a context that holds some assumptions and notation used in all theorems proven in it. In the locale (context) below called topology0 we assume that $T$ is a topology. The interior of the set $A$ (with respect to the topology in the context) is denoted $\text{int}(A)$. The closure of a set $A \subseteq \bigcup T$ is denoted $\text{cl}(A)$ and the boundary is $\partial A$.

locale topology0 =
  fixes $T$
  assumes topSpaceAssum: $T$ (is a topology)
  fixes $\text{int}$
  defines $\text{int}\_\text{def}$ [simp]: $\text{int}(A) \equiv \text{Interior}(A,T)$
  fixes $\text{cl}$
  defines $\text{cl}\_\text{def}$ [simp]: $\text{cl}(A) \equiv \text{Closure}(A,T)$
  fixes boundary ($\partial_\_\ [91] 92$)
  defines boundary\_\text{def} [simp]: $\partial A \equiv \text{Boundary}(A,T)$

Intersection of a finite nonempty collection of open sets is open.

lemma (in topology0) fin_inter_open_open: assumes $N \neq 0 \ N \in \text{FinPow}(T)$ shows $\bigcap N \in T$
using topSpaceAssum assms IsATopology_def inter_two_inter_fin by simp

780
Having a topology $T$ and a set $X$ we can define the induced topology as the one consisting of the intersections of $X$ with sets from $T$. The notion of a collection restricted to a set is defined in ZF1.thy.

**lemma (in topology0) Top_1_L4:**

shows $(T \{\text{restricted to}\} X) \{\text{is a topology}\}$

**proof**

let $S = T \{\text{restricted to}\} X$

have $\forall A \in \text{Pow}(S). \bigcup A \in S$

**proof**

fix $A$ assume $A1: A \in \text{Pow}(S)$

have $\forall V \in A. \bigcup \{U \in T. V = U \cap X\} \in T$

**proof**

fix $V$

let $M = \{U \in T. V = U \cap X\}$

have $M \in \text{Pow}(T)$ by auto

with topSpaceAssum have $(\bigcup V \in A. \bigcup \{U \in T. V = U \cap X\}) \cap X \in S$

using RestrictedTo_def by auto

moreover from $A1$ have $\forall V \in A. \exists U \in T. V = U \cap X$

using RestrictedTo_def by auto

hence $(\bigcup V \in A. \bigcup \{U \in T. V = U \cap X\}) \cap X = \bigcup A$ by blast

ultimately show $\bigcup A \in S$ by simp

**qed**

moreover have $\forall U \in S. \forall V \in S. U \cap V \in S$

**proof**

fix $U V$ assume $U \in S. V \in S$

then obtain $U_1 V_1$ where

$U_1 \in T \land U = U_1 \cap X$ and $V_1 \in T \land V = V_1 \cap X$

using RestrictedTo_def by auto

with topSpaceAssum have $U_1 \cap V_1 \in T$ and $U \cap V = (U_1 \cap V_1) \cap X$

using IsATopology_def by auto

then have $U \cap V \in S$ using RestrictedTo_def by auto

thus $\forall U \in S. \forall V \in S. U \cap V \in S$ by simp

**qed**

ultimately show $S \{\text{is a topology}\}$ using IsATopology_def by simp

**qed**

60.2 Interior of a set

In this section we show basic properties of the interior of a set.
Interior of a set $A$ is contained in $A$.

**lemma (in topology0) Top_2_L1:** shows $\text{int}(A) \subseteq A$

using $\text{Interior_def}$ by auto

Interior is open.

**lemma (in topology0) Top_2_L2:** shows $\text{int}(A) \in T$

proof -

  have $\{U \in T. \: U \subseteq A\} \in \text{Pow}(T)$ by auto

  with $\text{topSpaceAssum}$ show $\text{int}(A) \in T$

  using $\text{IsATopology_def}$ $\text{Interior_def}$ by auto

qed

A set is open iff it is equal to its interior.

**lemma (in topology0) Top_2_L3:** shows $U \subseteq T \iff \text{int}(U) = U$

proof

  assume $U \subseteq T$ then show $\text{int}(U) = U$

  using $\text{Interior_def}$ by auto

next assume $A1: \text{int}(U) = U$

  have $\text{int}(U) \in T$ using $\text{Top_2_L2}$ by simp

  with $A1$ show $U \subseteq T$ by simp

qed

Interior of the interior is the interior.

**lemma (in topology0) Top_2_L4:** shows $\text{int}(\text{int}(A)) = \text{int}(A)$

proof -

  let $U = \text{int}(A)$

  from $\text{topSpaceAssum}$ have $U \subseteq T$ using $\text{Top_2_L2}$ by simp

  then show $\text{int}(\text{int}(A)) = \text{int}(A)$ using $\text{Top_2_L3}$ by simp

qed

Interior of a bigger set is bigger.

**lemma (in topology0) interior_mono:**

assumes $A1: A \subseteq B$

shows $\text{int}(A) \subseteq \text{int}(B)$

proof -

  from $A1$ have $\forall \: U \subseteq T. \: (U \subseteq A \Rightarrow U \subseteq B)$ by auto

  then show $\text{int}(A) \subseteq \text{int}(B)$ using $\text{Interior_def}$ by auto

qed

An open subset of any set is a subset of the interior of that set.

**lemma (in topology0) Top_2_L5:** assumes $U \subseteq A$ and $U \in T$

shows $U \subseteq \text{int}(A)$

using assms $\text{Interior_def}$ by auto

If a point of a set has an open neighbourhood contained in the set, then the point belongs to the interior of the set.

**lemma (in topology0) Top_2_L6:** assumes $\exists U \in T. \: (x \in U \land U \subseteq A)$

shows $x \in \text{int}(A)$
A set is open iff its every point has an open neighbourhood contained in the set. We will formulate this statement as two lemmas (implication one way and the other way). The lemma below shows that if a set is open then every point has an open neighbourhood contained in the set.

**Lemma (in topology0) open_open_neigh:**

**assumes** \( A1: V \in T \)**

**shows** \( \forall x \in V. \exists U \in T. (x \in U \land U \subseteq V) \)**

**proof**

- from \( A1 \) have \( \forall x \in V. V \in T \land x \in V \land V \subseteq V \) by simp
- thus thesis by auto

**qed**

If every point of a set has an open neighbourhood contained in the set then the set is open.

**Lemma (in topology0) open_neigh_open:**

**assumes** \( A1: \forall x \in V. \exists U \in T. (x \in U \land U \subseteq V) \)**

**shows** \( V \in T \)

**proof**

- from \( A1 \) have \( V = \text{int}(V) \) using Top_2_L1 Top_2_L6
- by blast
- then show \( V \in T \) using Top_2_L3 by simp

**qed**

The intersection of interiors is equal to the interior of intersections.

**Lemma (in topology0) int_inter_int:**

**shows** \( \text{int}(A) \cap \text{int}(B) = \text{int}(A \cap B) \)

**proof**

- have \( \text{int}(A) \cap \text{int}(B) \subseteq A \cap B \) using Top_2_L1 by auto
- moreover have \( \text{int}(A) \cap \text{int}(B) \in T \) using Top_2_L2 IsATopology_def topSpaceAssum

  by auto

- ultimately show \( \text{int}(A) \cap \text{int}(B) \subseteq \text{int}(A \cap B) \) using Top_2_L5 by simp
- have \( A \cap B \subseteq A \) and \( A \cap B \subseteq B \) by auto
- then have \( \text{int}(A \cap B) \subseteq \text{int}(A) \) and \( \text{int}(A \cap B) \subseteq \text{int}(B) \) using interior_mono

  by auto

- thus \( \text{int}(A \cap B) \subseteq \text{int}(A) \cap \text{int}(B) \) by auto

**qed**

### 60.3 Closed sets, closure, boundary.

This section is devoted to closed sets and properties of the closure and boundary operators.

The carrier of the space is closed.

**Lemma (in topology0) Top_3_L1:**

**shows** \( (\bigcup T) \) (is closed in) \( T \)

**proof**

- have \( \bigcup T - \bigcup T = 0 \) by auto
with topSpaceAssum have $\bigcup T - \bigcup T \in T$ using IsATopology_def by auto
then show thesis using IsClosed_def by simp
qed

Empty set is closed.

lemma (in topology0) Top_3_L2: shows 0 \{is closed in\} T
using topSpaceAssum IsATopology_def IsClosed_def by simp

The collection of closed covers of a subset of the carrier of topology is never empty. This is good to know, as we want to intersect this collection to get the closure.

lemma (in topology0) Top_3_L3:
  assumes A1: A $\subseteq$ $\bigcup T$ shows ClosedCovers(A,T) $\neq$ 0
proof -
  from A1 have $\bigcup T \in$ ClosedCovers(A,T) using ClosedCovers_def Top_3_L1
  by auto
  thus thesis by auto
qed

Intersection of a nonempty family of closed sets is closed.

lemma (in topology0) Top_3_L4: assumes A1: K $\neq$ 0 and
  A2: $\forall D \in K. \ D$ \{is closed in\} T
shows ($\bigcap K$) \{is closed in\} T
proof -
  from A2 have I: $\forall D \in K. \ (D \subseteq \bigcup T \land (\bigcup T - D) \in T)$
  using IsClosed_def by simp
  then have $\{\bigcup T - D. \ D \in K\} \subseteq T$ by auto
  with topSpaceAssum have (\bigcup {\bigcup T - D. \ D \in K}) $\in$ T
  using IsATopology_def by auto
  moreover from A1 have $\bigcup \{\bigcup T - D. \ D \in K\} = \bigcup T - \bigcap K$ by fast
  moreover from A1 I $\bigcap K \subseteq \bigcup T$ by blast
  ultimately show ($\bigcap K$) \{is closed in\} T using IsClosed_def
  by simp
qed

The union and intersection of two closed sets are closed.

lemma (in topology0) Top_3_L5:
  assumes A1: D_1 \{is closed in\} T \ D_2 \{is closed in\} T
shows (D_1\cap D_2) \{is closed in\} T
(D_1\cup D_2) \{is closed in\} T
proof -
  have \{D_1, D_2\} $\neq$ 0 by simp
  with A1 have ($\bigcap \{D_1, D_2\}$) \{is closed in\} T using Top_3_L4
  by fast
  thus (D_1\cap D_2) \{is closed in\} T by simp
  from topSpaceAssum A1 have (\bigcup T - D_1) \cap (\bigcup T - D_2) \in T
  using IsClosed_def IsATopology_def by simp

784
moreover have \((\bigcup T - D_1) \cap (\bigcup T - D_2) = \bigcup T - (D_1 \cup D_2)\)
by auto
moreover from A1 have \(D_1 \cup D_2 \subseteq \bigcup T\) using IsClosed_def
by auto
ultimately show \((D_1 \cup D_2) \{\text{is closed in}\} T\) using IsClosed_def
by simp
qed

Finite union of closed sets is closed. To understand the proof recall that
\(D \in \text{Pow}(\bigcup T)\) means that \(D\) is a subset of the carrier of the topology.

lemma (in topology0) fin_union_cl_is_cl:
assumes
A1: \(N \in \text{FinPow}(\{D \in \text{Pow}(\bigcup T). D \{\text{is closed in}\} T\})\)
shows \((\bigcup N) \{\text{is closed in}\} T\)
proof -
let \(C = \{D \in \text{Pow}(\bigcup T). D \{\text{is closed in}\} T\}\)
have \(0 \in C\) using Top_3_L2 by simp
moreover have \(\forall A \in C. \forall B \in C. A \cup B \in C\)
using Top_3_L5 by auto
moreover note A1
ultimately have \(\bigcup N \in C\) by (rule union_two_union_fin)
thus \((\bigcup N) \{\text{is closed in}\} T\) by simp
qed

Closure of a set is closed, hence the complement of the closure is open.

lemma (in topology0) cl_is_closed:
assumes
A1: \(A \subseteq \bigcup T\)
shows \(\text{cl}(A) \{\text{is closed in}\} T\) and \(\bigcup T - \text{cl}(A) \in T\)
using asms Top_3_L3 Top_3_L4 Closure_def ClosedCovers_def IsClosed_def
by auto

Closure of a bigger sets is bigger.

lemma (in topology0) top_closure_mono:
assumes A1: \(B \subseteq \bigcup T\) and A2: \(A \subseteq B\)
shows \(\text{cl}(A) \subseteq \text{cl}(B)\)
proof -
from A2 have ClosedCovers(B,T) \(\subseteq\) ClosedCovers(A,T)
using ClosedCovers_def by auto
with A1 show thesis using Top_3_L3 Closure_def by auto
qed

Boundary of a set is closed.

lemma (in topology0) boundary_closed:
assumes A1: \(A \subseteq \bigcup T\) shows \(\partial A \{\text{is closed in}\} T\)
proof -
from A1 have \(\bigcup T - A \subseteq \bigcup T\) by fast
with A1 show \(\partial A \{\text{is closed in}\} T\)
using cl_is_closed Top_3_L5 Boundary_def by auto
qed

785
A set is closed iff it is equal to its closure.

**Lemma (in topology0) Top_3_L8**: assumes \( A \subseteq \bigcup T \)

shows \( A \) {is closed in} \( T \) \( \iff \) \( \text{cl}(A) = A \)

**Proof**

assume \( A \) {is closed in} \( T \)

with \( A1 \) show \( \text{cl}(A) = A \)

using \( \text{Closure_def} \) \( \text{ClosedCovers_def} \) by auto

next assume \( \text{cl}(A) = A \)

then have \( \bigcup T - A = \bigcup T - \text{cl}(A) \) by simp

with \( A1 \) show \( A \) {is closed in} \( T \) using \( \text{cl_is_closed} \) \( \text{IsClosed_def} \) by simp

**Qed**

Complement of an open set is closed.

**Lemma (in topology0) Top_3_L9**: assumes \( A \in T \)

shows \( (\bigcup T - A) \) {is closed in} \( T \)

**Proof**

from \( \text{topSpaceAssum} \ A1 \) have \( \bigcup T - (\bigcup T - A) = A \) and \( \bigcup T - A \subseteq \bigcup T \)

using \( \text{IsATopology_def} \) by auto

with \( A1 \) show \( (\bigcup T - A) \) {is closed in} \( T \) using \( \text{IsClosed_def} \) by simp

**Qed**

A set is contained in its closure.

**Lemma (in topology0) cl_contains_set**: assumes \( A \subseteq \bigcup T \)

shows \( A \subseteq \text{cl}(A) \)

using \( \text{assms} \) \( \text{Top_3_L1} \) \( \text{ClosedCovers_def} \) \( \text{Top_3_L3} \) \( \text{Closure_def} \) by auto

Closure of a subset of the carrier is a subset of the carrier and closure of the complement is the complement of the interior.

**Lemma (in topology0) Top_3_L11**: assumes \( A \subseteq \bigcup T \)

shows

\( \text{cl}(A) \subseteq \bigcup T \)

\( \text{cl}(\bigcup T - A) = \bigcup T - \text{int}(A) \)

**Proof**

from \( A1 \) show \( \text{cl}(A) \subseteq \bigcup T \) using \( \text{Top_3_L1} \) \( \text{Closure_def} \) \( \text{ClosedCovers_def} \) by auto

from \( A1 \) have \( \bigcup T - A \subseteq \bigcup T - \text{int}(A) \) using \( \text{Top_2_L1} \) by auto

moreover have \( \text{I}: \bigcup T - \text{int}(A) \subseteq \bigcup T \)

ultimately have \( \text{cl}(\bigcup T - A) \subseteq \text{cl}(\bigcup T - \text{int}(A)) \)

using \( \text{top_closure_mono} \) by simp

moreover from \( \text{I} \) have \( (\bigcup T - \text{int}(A)) \) {is closed in} \( T \)

using \( \text{Top_2_L2} \) \( \text{Top_3_L9} \) by simp

with \( \text{I} \) have \( \text{cl}((\bigcup T) - \text{int}(A)) = \bigcup T - \text{int}(A) \)

using \( \text{Top_3_L8} \) by simp

ultimately have \( \text{cl}(\bigcup T - A) \subseteq \bigcup T - \text{int}(A) \) by simp

moreover from \( \text{I} \) have \( \bigcup T - A \subseteq \text{cl}(\bigcup T - A) \) using \( \text{cl_contains_set} \) by simp
hence $\bigcup T - \text{cl}(\bigcup T - A) \subseteq A$ and $\bigcup T - A \subseteq \bigcup T$ by auto
then have $\bigcup T - \text{cl}(\bigcup T - A) \subseteq \text{int}(A)$
  using cl_is_closed IsClosed_def Top_2_L5 by simp
hence $\bigcup T - \text{int}(A) \subseteq \text{cl}(\bigcup T - A)$ by auto
ultimately show $\text{cl}(\bigcup T - A) = \bigcup T - \text{int}(A)$ by auto
qed

Boundary of a set is the closure of the set minus the interior of the set.

lemma (in topology0) Top_3_L12: assumes A1: $A \subseteq \bigcup T$
shows $\partial A = \text{cl}(A) - \text{int}(A)$
proof -
  from A1 have $\partial A = \text{cl}(A) \cap (\bigcup T - \text{int}(A))$
    using Boundary_def Top_3_L11 by simp
moreover from A1 have $\text{cl}(A) \cap (\bigcup T - \text{int}(A)) = \text{cl}(A) - \text{int}(A)$
  using Top_3_L11 by blast
ultimately show $\partial A = \text{cl}(A) - \text{int}(A)$ by simp
qed

If a set $A$ is contained in a closed set $B$, then the closure of $A$ is contained in $B$.

lemma (in topology0) Top_3_L13: assumes A1: $B \text{ is closed in } T$ $A \subseteq B$
shows $\text{cl}(A) \subseteq B$
proof -
  from A1 have $B \subseteq \bigcup T$ using IsClosed_def by simp
  with A1 show $\text{cl}(A) \subseteq B$ using ClosedCovers_def Closure_def by auto
qed

If a set is disjoint with an open set, then we can close it and it will still be disjoint.

lemma (in topology0) disj_open_cl_disj: assumes A1: $A \subseteq \bigcup T$ $V \in T$ and $A \cap V = 0$
shows $\text{cl}(A) \cap V = 0$
proof -
  from assms have $A \subseteq \bigcup T - V$ by auto
  moreover from A1 have $(\bigcup T - V) \text{ is closed in } T$ using Top_3_L9 by simp
  ultimately have $\text{cl}(A) - (\bigcup T - V) = 0$
    using Top_3_L13 by blast
  moreover from A1 have $\text{cl}(A) \subseteq \bigcup T$ using cl_is_closed IsClosed_def by simp
  then have $\text{cl}(A) - (\bigcup T - V) = \text{cl}(A) \cap V$ by auto
  ultimately show thesis by simp
qed

A reformulation of disj_open_cl_disj: If a point belongs to the closure of a set, then we can find a point from the set in any open neighborhood of the point.
lemma (in topology0) cl_inter_neigh:
  assumes A ⊆ ∪T and U∈T and x ∈ cl(A) ∩ U
  shows A∩U ≠ 0 using assms disj_open_cl_disj by auto

A reverse of cl_inter_neigh: if every open neighboorhood of a point has a
nonempty intersection with a set, then that point belongs to the closure of
the set.

lemma (in topology0) inter_neigh_cl:
  assumes A1: A ⊆ ∪T and A2: x∈∪T and A3: ∀U∈T. x∈U → U∩A ≠ 0
  shows x ∈ cl(A)
proof -
  { assume x /∈ cl(A)
    with A1 obtain D where D {is closed in} T and A⊆D and x∉D
      using Top_3_L3 Closure_def ClosedCovers_def by auto
    let U = (∪T) - D
    from A2 ‹D {is closed in} T› ‹x∉D› ‹A⊆D› have U∈T x∈U and U∩A = 0
      unfolding IsClosed_def by auto
    with A3 have False by auto
  }
  thus thesis by auto
qed

end

61 Topology 1

theory Topology_ZF_1 imports Topology_ZF
begin

In this theory file we study separation axioms and the notion of base and
subbase. Using the products of open sets as a subbase we define a natural
topology on a product of two topological spaces.

61.1 Separation axioms

Topological spaces can be classified according to certain properties called
"separation axioms". In this section we define what it means that a topo-
logical space is $T_0$, $T_1$ or $T_2$.

A topology on $X$ is $T_0$ if for every pair of distinct points of $X$ there is an
open set that contains only one of them.

definition
  isT0 (_ {is T_0} [90] 91) where
  T {is T_0} ≡ ∀ x y. ((x ∈ ∪T ∧ y ∈ ∪T ∧ x≠y) →
  (∃U∈T. (x∈U ∧ y∉U) ∨ (y∈U ∧ x∉U)))
A topology is $T_1$ if for every such pair there exist an open set that contains the first point but not the second.

definition

isT1 (_ {is T1}) (90) 91 where
T {is T1} ≡ ∀ x y. ((x ∈ ∪T ∧ y ∈ ∪T ∧ x ≠ y) → (∃U ∈ T. (x ∈ U ∧ y ∉ U)))

$T_1$ topological spaces are exactly those in which all singletons are closed.

lemma (in topology0) t1_def_alt:
shows T {is T1} ←→ (∀ x ∈ ∪T. {x} {is closed in} T)
proof
let X = ∪T
assume T1: T {is T1}
{ fix x assume x ∈ X
let U = X-{x}
have U ∈ T
proof -
  let W = ∪y ∈ U. y ∈ X ∧ x /∈ V
  { fix y assume y ∈ U
    with topSpaceAssum have (∪y ∈ X. y ∈ X ∧ x /∈ V) ∈ T
      unfolding IsATopology_def by blast
  } hence ∀y ∈ U. (∪y ∈ X. y ∈ X ∧ x /∈ V) ∈ T by blast
  with topSpaceAssum have W ∈ T by (rule union_indexed_open)
  have U = W
  proof
    show W ⊆ U by auto
      { fix y assume y ∈ U
        hence y ∈ X and y ≠ x by auto
          with unfolding isT1_def by blast
        then have ∃U ∈ T. (x ∈ U ∧ y /∈ U)
          unfolding IsClosed_def by auto
      } thus U ⊆ W by blast
  qed
  with W ∈ T show U ∈ T by simp
  qed
  with ∪y ∈ X. have (X-U) {is closed in} T and X-U = {x}
    using Top_3_L9 by auto
  hence {x} {is closed in} T by simp
  thus ∀x ∈ X. {x} {is closed in} T by blast
next
let X = ∪T
assume scl: ∀x ∈ ∪T. {x} {is closed in} T
{ fix x y assume x ∈ X y ∈ X x ≠ y
let U = X-{y}
from scl ∪x ∈ X. y ∈ X. x ≠ y have U ∈ T x ∈ U ∧ y ∉ U
  unfolding IsClosed_def by auto
  then have ∃U ∈ T. (x ∈ U ∧ y ∉ U) by (rule witness_exists)
} then show T {is T1} unfolding isT1_def by blast
qed
A topology is $T_2$ (Hausdorff) if for every pair of points there exist a pair of disjoint open sets each containing one of the points. This is an important class of topological spaces. In particular, metric spaces are Hausdorff.

**Definition**

$\text{isT}_2 \equiv \forall x \ y. ((x \in \bigcup T \land y \in \bigcup T \land x \neq y) \rightarrow (\exists U \in T. \exists V \in T. x \in U \land y \in V \land U \cap V = 0))$

A topology is regular if every closed set can be separated from a point in its complement by (disjoint) opens sets.

**Definition**

$\text{IsRegular} \equiv \forall D. D \subseteq T \rightarrow (\forall x \in T - D. \exists U \in T. \exists V \in T. D \subseteq U \land x \in V \land U \cap V = 0)$

Some sources (e.g. Metamath) use a different definition of regularity: any open neighborhood has a closed subneighborhood. The next lemma shows the equivalence of this with our definition.

**Lemma** $\text{is_regular_def_alt}$

Assumes $T \text{ is a topology}$

shows $T \text{ is regular} \iff (\forall W \in T. \forall x \in W. \exists V \in T. x \in V \land \text{Closure}(V,T) \subseteq W)$

Proof

Let $X = \bigcup T$

From assms(1) have $\text{cntx: topology0}(T)$ unfolding topology0_def by simp

Assume $T \text{ is regular}$

{ fix $W \ x$ assume $W \in T \ x \in W$ have $\exists V \in T. x \in V \land \text{Closure}(V,T) \subseteq W$

proof -

let $D = X - W$

from $\text{cntx} \langle W \in T \rangle \langle T \text{ is regular} \rangle \langle x \in W \rangle$

have $\exists U \in T. \exists V \in T. D \subseteq U \land x \in V \land U \cap V = 0$

using topology0.Top_3_L9 unfolding IsRegular_def by auto

then obtain $U \ V$ where $U \in T \ V \in T \ D \subseteq U \land x \in V \land U \cap V = 0$

by blast

from $\text{cntx} \langle V \in T \rangle$ have $\text{Closure}(V,T) \subseteq X$

using topology0.Top_3_L11(1) by blast

from $\text{cntx} \langle V \in T \rangle \langle U \in T \rangle \langle V \cup U = 0 \rangle \langle D \subseteq U \rangle$

have $\text{Closure}(V,T) \cap D = 0$

using topology0.disj_open_cl_disj by blast

with $\langle \text{Closure}(V,T) \subseteq X \rangle \langle V \in T \rangle \langle x \in V \rangle$ show thesis

by blast

qed

} thus $\forall W \in T. \forall x \in W. \exists V \in T. x \in V \land \text{Closure}(V,T) \subseteq W$

by simp

next

Let $X = \bigcup T$

From assms(1) have $\text{cntx: topology0}(T)$ unfolding topology0_def by simp
assume \( \forall W \in T. \forall x \in W. \exists V \in T. x \in V \land \text{Closure}(V,T) \subseteq W \)

\{
  \text{fix A assume A \{is closed in\} T}
  \text{have \( \forall x \in \text{X-A}. \exists U \in T. \exists V \in T. A \subseteq U \land x \in V \land U \cap V = 0 \)}
\}

\text{proof -}
\{
  \text{let \( W = X-A \)}}

\text{from \(<A \{is closed in\} T>\) have \( A \subseteq X \) and \( W \in T \)}

\text{unfolding IsClosed_def by auto}

\text{fix x assume \( x \in W \)}

\text{with \( \text{regAlt} \ <W \in T>\) have \( \exists V \in T. x \in V \land \text{Closure}(V,T) \subseteq W \)}

\text{by simp}

\text{then obtain \( V \) where \( V \in T \land x \in V \land \text{Closure}(V,T) \subseteq W \)}

\text{by auto}

\text{let \( U = X-\text{Closure}(V,T) \)}

\text{from \( \text{cntx} <V \in T>\) have \( V \subseteq X \land V \subseteq \text{Closure}(V,T) \)}

\text{using topology0.cl_contains_set by auto}

\text{with \( \text{cntx} <A \subseteq X, \text{Closure}(V,T) \subseteq W>\) have \( U \subseteq A \subseteq U \cup W = 0 \)}

\text{by blast}

\} \text{ thus thesis by blast}

\text{qed}

\text{then show T \{is regular\} unfolding IsRegular_def by blast}

\text{qed}

If a topology is \( T_1 \) then it is \( T_0 \). We don’t really assume here that \( T \) is a topology on \( X \). Instead, we prove the relation between \( \text{isT0} \) condition and \( \text{isT1} \).

\text{lemma T1_is_T0: assumes A1: T \{is T1\} shows T \{is T0\}}

\text{proof -}
\{
  \text{from A1 have \( \forall x y. x \in \bigcup T \land y \in \bigcup T \land x \neq y \rightarrow \)}

\( (\exists U \in T. x \in U \land y \notin U) \)

\text{using isT1_def by simp}

\text{then have \( \forall x y. x \in \bigcup T \land y \in \bigcup T \land x \neq y \rightarrow \)}

\( (\exists U \in T. x \in U \land y \notin U \lor y \in U \land x \notin U) \)

\text{by auto}

\text{then show T \{is T0\} using isT0_def by simp}

\text{qed}

If a topology is \( T_2 \) then it is \( T_1 \).

\text{lemma T2_is_T1: assumes A1: T \{is T2\} shows T \{is T1\}}

\text{proof -}
\{
  \text{fix x y assume x \in \bigcup T \land y \in \bigcup T \land x \neq y}

\text{with A1 have \( \exists U \in T. \exists V \in T. x \in U \land y \in V \land U \cap V = 0 \)}

\text{using isT2_def by auto}

\text{then have \( \exists U \in T. x \in U \land y \notin U \) by auto}

\text{} \text{then have \( \forall x y. x \in \bigcup T \land y \in \bigcup T \land x \neq y \rightarrow \)}

\( (\exists U \in T. x \in U \land y \notin U) \) \text{ by simp}

791
then show \( T \{\text{is } T_1}\) using \( \text{isT1_def}\) by simp
qed

In a \( T_0 \) space two points that can not be separated by an open set are equal.
Proof by contradiction.

lemma Top_1_1_L1: assumes A1: \( T \{\text{is } T_0}\) and A2: \( x \in \bigcup T \ y \in \bigcup T \)
and A3: \( \forall U \in T. (x \in U \longleftrightarrow y \in U) \)
shows \( x = y \)
proof -
{ assume \( x \neq y \)
  with A1 A2 have \( \exists U \in T. x \in U \land y \notin U \lor y \in U \land x \notin U \)
  using \( \text{isT0_def}\) by simp
  with A3 have False by auto
} then show \( x = y \) by auto
qed

61.2 Bases and subbases

Sometimes it is convenient to talk about topologies in terms of their bases
and subbases. These are certain collections of open sets that define the
whole topology.

A base of topology is a collection of open sets such that every open set is a
union of the sets from the base.

definition
IsAbaseFor (infixl \{is a base for\} 65) where
\( B \{\text{is a base for}\} T \equiv B \subseteq T \land T = \{\bigcup A. A \in \text{Pow}(B)\} \)

A subbase is a collection of open sets such that finite intersection of those
sets form a base.

definition
IsAsubBaseFor (infixl \{is a subbase for\} 65) where
\( B \{\text{is a subbase for}\} T \equiv \)
\( B \subseteq T \land \{\bigcap A. A \in \text{FinPow}(B)\} \{\text{is a base for}\} T \)

Below we formulate a condition that we will prove to be necessary and
sufficient for a collection \( B \) of open sets to form a base. It says that for any
two sets \( U, V \) from the collection \( B \) we can find a point \( x \in U \cap V \) with a
neighborhood from \( B \) contained in \( U \cap V \).

definition
SatisfiesBaseCondition (_ {satisfies the base condition} [50] 50) where
\( B \{\text{satisfies the base condition}\} \equiv \)
\( \forall U V. ((U \in B \land V \in B) \longrightarrow (\forall x \in U \cap V. \exists W \in B. x \in W \land W \subseteq U \cap V)) \)

A collection that is closed with respect to intersection satisfies the base
condition.
lemma inter_closed_base: assumes $\forall U \in B. (\forall V \in B. U \cap V \in B)$
shows $B$ {satisfies the base condition}
proof -
{ fix $U \ V \ x$ assume $U \in B$ and $V \in B$ and $x \in U \cap V$
with assms have $\exists W \in B. x \in W \land W \subseteq U \cap V$ by blast
} then show thesis using SatisfiesBaseCondition_def by simp
qed

Each open set is a union of some sets from the base.

lemma Top_1_2_L1: assumes $B$ {is a base for} $T$ and $U \in T$
shows $\exists A \in \text{Pow}(B). U = \bigcup A$
using assms IsAbaseFor_def by simp

Elements of base are open.

lemma base_sets_open:
assumes $B$ {is a base for} $T$ and $U \in B$
shows $U \in T$
using assms IsAbaseFor_def by auto

A base defines topology uniquely.

lemma same_base_same_top:
assumes $B$ {is a base for} $T$ and $B$ {is a base for} $S$
shows $T = S$
using assms IsAbaseFor_def by simp

Every point from an open set has a neighborhood from the base that is contained in the set.

lemma point_open_base_neigh:
assumes $A1$: $B$ {is a base for} $T$ and $A2$: $U \in T$ and $A3$: $x \in U$
shows $\exists V \in B. V \subseteq U \land x \in V$
proof -
from $A1 \ A2$ obtain $A$ where $A \in \text{Pow}(B)$ and $U = \bigcup A$
using assms Top_1_2_L1 by blast
with $A3$ obtain $V$ where $V \in A$ and $x \in V$ by auto
with $\langle A \in \text{Pow}(B) \rangle \langle U = \bigcup A \rangle$ show thesis by auto
qed

A criterion for a collection to be a base for a topology that is a slight reformulation of the definition. The only thing different that in the definition is that we assume only that every open set is a union of some sets from the base. The definition requires also the opposite inclusion that every union of the sets from the base is open, but that we can prove if we assume that $T$ is a topology.

lemma is_a_base_criterion: assumes $A1$: $T$ {is a topology}
and $A2$: $B \subseteq T$ and $A3$: $\forall V \in T. \exists A \in \text{Pow}(B). V = \bigcup A$
shows $B$ {is a base for} $T$
proof -
from A3 have $T \subseteq \{ \bigcup A. A \in \text{Pow}(B) \}$ by auto
moreover have $\{ \bigcup A. A \in \text{Pow}(B) \} \subseteq T$
proof
fix $U$ assume $U \in \{ \bigcup A. A \in \text{Pow}(B) \}$
then obtain $A$ where $A \in \text{Pow}(B)$ and $U = \bigcup A$
by auto
with $B \subseteq T$ have $A \in \text{Pow}(T)$ by auto
with A1 $\langle U = \bigcup A \rangle$ show $U \in T$
  unfolding IsATopology_def by simp
qed
ultimately have $T = \{ \bigcup A. A \in \text{Pow}(B) \}$ by auto
with A2 show $B$ {is a base for} $T$
  unfolding IsAbaseFor_def by simp
qed

A necessary condition for a collection of sets to be a base for some topology:
: every point in the intersection of two sets in the base has a neighborhood
from the base contained in the intersection.

lemma Top_1_2_L2:
  assumes A1: $\forall x \in V \cap W. \exists U \in \text{Pow}(B). x \in U \land U \subseteq V \cap W$
shows $V \cap W \in \{ \bigcup A. A \in \text{Pow}(B) \}$
proof -
  from A1 obtain $T$ where
    D1: $T$ {is a topology} $B$ {is a base for} $T$
    by auto
  then have $B \subseteq T$ using IsAbaseFor_def by auto
  with A2 have $V \in T$ and $W \in T$ using IsAbaseFor_def by auto
  with D1 have $\exists A \in \text{Pow}(B). V \cap W = \bigcup A$ using IsATopology_def Top_1_2_L1
    by auto
  then obtain $A$ where $A \subseteq B$ and $V \cap W = \bigcup A$ by auto
  then show $\forall x \in V \cap W. \exists U \in \text{Pow}(B). (x \in U \land U \subseteq V \cap W)$ by auto
qed

We will construct a topology as the collection of unions of (would-be) base.
First we prove that if the collection of sets satisfies the condition we want
to show to be sufficient, the the intersection belongs to what we will define
as topology (am I clear here?). Having this fact ready simplifies the proof
of the next lemma. There is not much topology here, just some set theory.

lemma Top_1_2_L3:
  assumes A1: $\forall x \in V \cap W. \exists U \in \text{Pow}(B). x \in U \land U \subseteq V \cap W$
shows $V \cap W \in \{ \bigcup A. A \in \text{Pow}(B) \}$
proof
  let $A = \bigcup x \in V \cap W. \{ U \in \text{Pow}(B). x \in U \land U \subseteq V \cap W \}$
  show $A \in \text{Pow}(B)$ by auto
  from A1 show $V \cap W = \bigcup A$ by blast
qed

The next lemma is needed when proving that the would-be topology is closed
with respect to taking intersections. We show here that intersection of two sets from this (would-be) topology can be written as union of sets from the topology.

**Lemma** Top.1_2_L4:

assumes A1: \( U_1 \in \{ \bigcup A. A \in \text{Pow}(B) \} \) \( U_2 \in \{ \bigcup A. A \in \text{Pow}(B) \} \)
and A2: B \{satisfies the base condition\}
shows \( \exists C. C \subseteq \{ \bigcup A. A \in \text{Pow}(B) \} \land U_1 \cap U_2 = \bigcup C \)

**Proof** -

from A1 A2 obtain \( A_1 A_2 \) where
D1: \( A_1 \in \text{Pow}(B) \) \( U_1 = \bigcup A_1 \) \( A_2 \in \text{Pow}(B) \) \( U_2 = \bigcup A_2 \)
between A2 have \( C = \bigcup \{ U \in A_1. \{ \bigcup \{ U \cap V. V \in A_2 \} \} \}

moreover from D1 have \( U_1 \cap U_2 = \bigcup C \) by auto
ultimately show thesis by auto

**QED**

If \( B \) satisfies the base condition, then the collection of unions of sets from \( B \) is a topology and \( B \) is a base for this topology.

**Theorem** Top.1_2_T1:

assumes A1: \( B \{satisfies the base condition\} \)
and A2: \( T = \{ \bigcup A. A \in \text{Pow}(B) \} \)
shows \( T \{is a topology\} \) \( B \{is a base for\} T \)

**Proof** -

show \( T \{is a topology\} \) and \( B \{is a base for\} T \)

have I: \( \forall C \in \text{Pow}(T). \bigcup C \in T \)

**Proof** -

\{ fix \( C \) assume A3: \( C \in \text{Pow}(T) \)
let \( Q = \bigcup \{ \bigcup \{ A \in \text{Pow}(B). U = \bigcup A \}. U \subseteq C \} \)
from A2 A3 have \( \forall U \subseteq C. \exists A \in \text{Pow}(B). U = \bigcup A \) by auto
then have \( \bigcup Q = \bigcup C \) using ZF1_1_L10 by simp
moreover from A2 have \( \bigcup Q \in T \) by auto
ultimately have \( \bigcup C \in T \) by simp \}
thus \( \forall C \in \text{Pow}(T). \bigcup C \in T \) by auto

**QED**

moreover have \( \forall U \subseteq T. \forall V \subseteq T. U \cap V \subseteq T \)

**Proof** -

\{ fix \( U V \) assume \( U \subseteq T \) \( V \subseteq T \)
with A1 A2 have \( \exists C. \{ C \subseteq T \land U \cap V = \bigcup C \} \)
using Top.1_2_L4 by simp
then obtain \( C \) where \( C \subseteq T \) \( U \cap V = \bigcup C \)
by auto
with I have \( U \cap V \in T \) by simp
\}
then show \( \forall U \subseteq T. \forall V \subseteq T. U \cap V \subseteq T \) by simp

**QED**

ultimately show \( T \{is a topology\} \) using IsATopology_def
by simp 
qed 
from A2 have B ⊆ T by auto 
with A2 show B {is a base for} T using IsAbaseFor_def 
by simp 
qed 

The carrier of the base and topology are the same.

**lemma Top_1_2_L5:** assumes B {is a base for} T 
shows ∪ T = ∪ B 
using assms IsAbaseFor_def by auto 

If B is a base for T, then T is the smallest topology containing B.

**lemma base_smallest_top:** 
assumes A1: B {is a base for} T and A2: S {is a topology} and A3: B ⊆ S 
shows T ⊆ S 
proof 
fix U assume U ∈ T 
with A1 obtain B_U where B_U ⊆ B and U = ∪ B_U using IsAbaseFor_def 
by auto 
with A3 have B_U ⊆ S by auto 
with A2 ∪ B_U show U ∈ S using IsATopology_def by simp 
qed 

If B is a base for T and B is a topology, then B = T.

**lemma base_topology:** 
assumes B {is a topology} and B {is a base for} T 
shows B = T using assms base_sets_open base_smallest_top by blast 

**61.3 Product topology**

In this section we consider a topology defined on a product of two sets.

Given two topological spaces we can define a topology on the product of the carriers such that the cartesian products of the sets of the topologies are a base for the product topology. Recall that for two collections S, T of sets the product collection is defined (in ZF1.thy) as the collections of cartesian products A × B, where A ∈ S, B ∈ T. The T × T notation is defined as an alternative to the verbose ProductTopology(T,S)).

**definition ProductTopology (infixl ×t 65) where** 
T ×t S ≡ {∪ W. W ∈ Pow(ProductCollection(T,S))} 

The product collection satisfies the base condition.

**lemma Top_1_4_L1:** 
assumes A1: T {is a topology} S {is a topology} 
and A2: A ∈ ProductCollection(T,S) B ∈ ProductCollection(T,S)
shows $\forall x \in (A \cap B). \exists W \in \text{ProductCollection}(T,S). (x \in W \land W \subseteq A \cap B)$

proof

fix $x$ assume $A3: x \in A \cap B$

from $A2$ obtain $U_1 \ V_1 \ U_2 \ V_2$ where

$D1: U_1 \in T \ V_1 \in S \ A = U_1 \times V_1 \ U_2 \in T \ V_2 \in S \ B = U_2 \times V_2$

using ProductCollection_def by auto

let $W = (U_1 \cap U_2) \times (V_1 \cap V_2)$

from $A1$ $D1$ have $U_1 \cap U_2 \in T$ and $V_1 \cap V_2 \in S$

using IsATopology_def by auto

then have $W \in \text{ProductCollection}(T,S)$ using ProductCollection_def by auto

moreover from $A3$ $D1$ have $x \in W$ and $W \subseteq A \cap B$

ultimately have $\exists W. (W \in \text{ProductCollection}(T,S) \land x \in W \land W \subseteq A \cap B)$

by auto

thus $\exists W \in \text{ProductCollection}(T,S). (x \in W \land W \subseteq A \cap B)$ by auto

qed

The product topology is indeed a topology on the product.

**Theorem Top_1_4_T1:** assumes $A1: T$ {is a topology} $S$ {is a topology}

shows

$(T \times S)$ {is a topology} $\text{ProductCollection}(T,S)$ {is a base for} $(T \times S)$

$\bigcup (T \times S) = \bigcup T \times \bigcup S$

**proof**

from $A1$ show

$(T \times S)$ {is a topology} $\text{ProductCollection}(T,S)$ {is a base for} $(T \times S)$

using Top_1_4_L1 ProductCollection_def

SatisfiesBaseCondition_def ProductTopology_def Top_1_2_T1 by auto

then show $\bigcup (T \times S) = \bigcup T \times \bigcup S$

using Top_1_2_L5 ZF1_1_L6 by simp

qed

Each point of a set open in the product topology has a neighborhood which is a cartesian product of open sets.

**Lemma prod_top_point_neighb:**

assumes $A1: T$ {is a topology} $S$ {is a topology} and

$A2: U \in \text{ProductTopology}(T,S)$ and $A3: x \in U$

shows $\exists V \ W. \ V \in T \land W \in S \land V \times W \subseteq U \land x \in V \times W$

**proof**

from $A1$ have

$\text{ProductCollection}(T,S)$ {is a base for} $\text{ProductTopology}(T,S)$

using Top_1_4_T1 by simp

with $A2$ $A3$ obtain $Z$ where

$Z \in \text{ProductCollection}(T,S)$ and $Z \subseteq U \land x \in Z$

using point_open_base_neigh by blast

then obtain $V \ W$ where $V \in T$ and $W \in S$ and $V \times W \subseteq U \land x \in V \times W$

using ProductCollection_def by auto

797
thus thesis by auto
qed

Products of open sets are open in the product topology.

**lemma prod_open_open_prod:**
assumes A1: T {is a topology} S {is a topology} and
A2: U ∈ T V ∈ S
shows U×V ∈ ProductTopology(T,S)

**proof** -
from A1 have
  ProductCollection(T,S) {is a base for} ProductTopology(T,S)
  using Top_1_4_T1 by simp
moreover from A2 have U×V ∈ ProductCollection(T,S)
  unfolding ProductCollection_def by auto
ultimately show U×V ∈ ProductTopology(T,S)
  using base_sets_open by simp
qed

Sets that are open in the product topology are contained in the product of the carrier.

**lemma prod_open_type:**
assumes A1: T {is a topology} S {is a topology} and
A2: V ∈ ProductTopology(T,S)
shows V ⊆ \bigcup T × \bigcup S

**proof** -
from A2 have V ⊆ \bigcup ProductTopology(T,S) by auto
with A1 show thesis using Top_1_4_T1 by simp
qed

A reverse of **prod_top_point_neighb:** if each point of set has an neighborhood in the set that is a cartesian product of open sets, then the set is open.

**lemma point_neighb_prod_top:**
assumes T {is a topology} S {is a topology} and
∀ p∈V. ∃ U∈T. ∃ W∈S. p∈U×W ∧ U×W ⊆ V
shows V ∈ ProductTopology(T,S)

**proof** -
from assms(1,2) have I: topology0(ProductTopology(T,S))
  using Top_1_4_T1(1) topology0_def by simp
moreover
{ fix p assume p∈V
  with assms(3) obtain U W where U∈T W∈S p∈U×W U×W ⊆ V
  by auto
  with assms(1,2) have ∃ N∈ProductTopology(T,S). p∈N ∧ N⊆V
    using prod_open_open_prod by auto
} hence ∀ p∈V. ∃ N∈ProductTopology(T,S). p∈N ∧ N⊆V by blast
ultimately show thesis using topology0.open_neigh_open by simp
qed

Suppose we have subsets \( A \subseteq X, B \subseteq Y \), where \( X, Y \) are topological spaces

798
with topologies $T, S$. We can consider relative topologies on $T_A, S_B$ on sets $A, B$ and the collection of cartesian products of sets open in $T_A, S_B$, namely $\{ U \times V : U \in T_A, V \in S_B \}$. The next lemma states that this collection is a base of the product topology on $X \times Y$ restricted to the product $A \times B$.

**lemma prod_restr_base_restr:**
assumes A1: $T$ {is a topology} $S$ {is a topology}
sows
ProductCollection($T$ {restricted to} $A$, $S$ {restricted to} $B$)
{is a base for} $(ProductTopology(T,S) {restricted to} A \times B)$
proof -
let $B = ProductCollection(T {restricted to} A, S {restricted to} B)$
let $\tau = ProductTopology(T,S)$
from A1 have $(\tau {restricted to} A \times B) {is a topology}$
  using Top_1_4_T1 topology0_def topology0.Top_1_L4
by simp
moreover have $B \subseteq (\tau {restricted to} A \times B)$
proof
fix $U$ assume $U \in B$
then obtain $U_A, U_B$ where $U_A \in (T {restricted to} A)$ and $U_B \in (S {restricted to} B)$
using ProductCollection_def by auto
then obtain $W_A, W_B$ where $W_A \in T$ $U_A = W_A \cap A$ and $W_B \in S$ $U_B = W_B \cap B$
using RestrictedTo_def by auto
with $U = U_A \times U_B$ have $U = W_A \times W_B \cap (A \times B)$ by auto
moreover from A1 $<W_A \in T>$ and $<W_B \in S>$ have $W_A \times W_B \in \tau$
using prod_open_open_prod by simp
ultimately show $U \in (\tau {restricted to} A \times B)$
using RestrictedTo_def by auto
qed
moreover have $\forall U \in (\tau {restricted to} A \times B)$.
exists $C \in Pow(B)$. $U = \bigcup C$
proof
fix $U$ assume $U \in (\tau {restricted to} A \times B)$
then obtain $W$ where $W \in \tau$ and $U = W \cap (A \times B)$
using RestrictedTo_def by auto
from A1 $<W \in \tau>$ obtain $A_W$ where
$A_W \in Pow(ProductCollection(T,S))$ and $W = \bigcup A_W$
using Top_1_4_T1 IsAbaseFor_def by auto
let $C = \{ V \cap A \times B. \ V \in A_W \}$
have $C \in Pow(B)$ and $U = \bigcup C$
proof -
{ fix $R$ assume $R \in C$
then obtain $V$ where $V \in A_W$ and $R = V \cap A \times B$
by auto
with $<A_W \in Pow(ProductCollection(T,S))>$ obtain $V_T, V_S$ where $V_T \in T$ and $V_S \in S$ and $V = V_T \times V_S$

799
using ProductCollection_def by auto

with \( R = V \cap A \times B \) have \( R \in B \)
using ProductCollection_def RestrictedTo_def
by auto
\}
then show \( C \in \text{Pow}(B) \) by auto
from \( \langle U = W \cap (A \times B) \rangle \) and \( \langle W = \bigcup A_W \rangle \)
show \( U = \bigcup C \) by auto
qed
thus \( \exists C \in \text{Pow}(B) \). \( U = \bigcup C \) by blast
qed

ultimately show thesis by (rule is_a_base_criterion)
qed

We can commute taking restriction (relative topology) and product topology.
The reason the two topologies are the same is that they have the same base.

**lemma prod_top_restr_comm:**
assumes \( A1: T \) \{is a topology\} \( S \) \{is a topology\}
shows
\( \text{ProductTopology}(T \{\text{restricted to}\} A, S \{\text{restricted to}\} B) = \text{ProductTopology}(T, S) \{\text{restricted to}\} (A \times B) \)
proof -
let \( B = \text{ProductCollection}(T \{\text{restricted to}\} A, S \{\text{restricted to}\} B) \)
from \( A1 \) have
\( B \) \{is a base for\} \( \text{ProductTopology}(T \{\text{restricted to}\} A, S \{\text{restricted to}\} B) \)
using topology0_def topology0.Top_1_L4 Top_1_4_T1 by simp
moreover have \( \forall y \in A. \exists W \subseteq S. \langle x, y \rangle \in U \times W \)
proof
fix \( y \) assume \( y \in A \)
then have \( \langle x, y \rangle \in V \) by simp
with \( A1 \) \( A2 \) have \( \langle x, y \rangle \in \bigcup T \times \bigcup S \) using prod_open_type by blast
hence \( x \in \bigcup T \) and \( y \in \bigcup S \) by auto
from \( A1 \) \( A2 \) have \( \exists U W. U \subseteq V \) and \( W \subseteq S \) and \( U \times W = \bigcup \{x, y\} \in U \times W \)
    by (rule prod_top_point_neighb)
then obtain \( U \ W \) where \( U \in T \ W \in S \ U \times W \subseteq V \ (x,y) \in U \times W \)
by auto
with A1 A2 show \( \exists W \in S. \ (y \in W \wedge W \subseteq A) \) using prod_open_type section_proj
by auto
qed
ultimately show thesis by (rule topology0.open_neigh_open)
qed

Projection of a section of an open set is open. This is dual of prod_sec_open1
with a very similar proof.

lemma prod_sec_open2: assumes A1: T \{is a topology\} S \{is a topology\}
and A2: \( V \in \text{ProductTopology}(T,S) \) and A3: \( y \in \bigcup S \)
shows \( \{x \in \bigcup T. \ (x,y) \in V\} \in T \)
proof
- let \( A = \{x \in \bigcup T. \ (x,y) \in V\} \)
from A1 have topology0(T) using topology0_def
by simp
moreover have \( \forall x \in A. \exists W \in T. \ (x \in W \wedge W \subseteq A) \)
proof
fix \( x \) assume \( x \in A \)
then have \( (x,y) \in V \) by simp
with A1 A2 have \( (x,y) \in \bigcup T \times \bigcup S \wedge (x,y) \in V \)
by blast
hence \( x \in \bigcup T \wedge y \in \bigcup S \)
by auto
from A1 A2 \( (x,y) \in V \) have \( \exists U W. \ U \in T \wedge W \in S \wedge U \times W \subseteq V \wedge (x,y) \in U \times W \)
by (rule prod_top_point_neighb)
then obtain \( U W \) where \( U \in T \ W \in S \ U \times W \subseteq V \ (x,y) \in U \times W \)
by auto
with A1 A2 show \( \exists W \in T. \ (x \in W \wedge W \subseteq A) \) using prod_open_type section_proj
by auto
qed
ultimately show thesis by (rule topology0.open_neigh_open)
qed

61.4 Hausdorff spaces

In this section we study properties of Hausdorff spaces (sometimes called
separated spaces) These are topological spaces that are \( T_2 \) as defined above.

A space is Hausdorff if and only if the diagonal \( \Delta = \{(x,x) : x \in X\} \) is
closed in the product topology on \( X \times X \).

theorem t2_iff_diag_closed: assumes T \{is a topology\}
shows T \{is \( T_2 \)\} \iff \{\{x,x\} : x \in \bigcup T\} \{is closed in\} \text{ProductTopology}(T,T)
proof
let \( X = \bigcup T \)
from assms(1) have I: topology0(\text{ProductTopology}(T,T))
using Top_1_4_T1(1) topology0_def by simp
assume T \{is \( T_2 \)\} show \( \{\{x,x\} : x \in X\} \{is closed in\} \text{ProductTopology}(T,T) \)
proof
let \( D_c = X \times X - \{ \langle x,x \rangle \mid x \in X \} \)
have \( \forall p \in D_c. \exists U \in T. \exists V \in T. p \in U \times V \wedge U \times V \subseteq D_c \)
proof
{ fix \( p \) assume \( p \in D_c \)
  then obtain \( x, y \) where \( p = \langle x, y \rangle \) \( x \in X, y \in X, x \neq y \) by auto
  unfolding \text{isT2_def} by blast
  with \text{assms} \( p = \langle x, y \rangle \) have \( \exists U \in T. \exists V \in T. p \in U \times V \wedge U \times V \subseteq D_c \) by auto
} hence \( \forall p. \ p \in D_c \longrightarrow (\exists U \in T. \exists V \in T. p \in U \times V \wedge U \times V \subseteq D_c) \) by simp
then show thesis by \text{(rule exists_in_set)}
qed
with \text{assms} show thesis using \text{Top_1_4_T1(3) point_neighb_prod_top}

unfolding \text{IsClosed_def} by auto
qed
next
let \( X = \bigcup T \)
assume \( A: \{ \langle x,x \rangle \mid x \in X \} \) \( \text{is closed in} \) \text{ProductTopology}(T, T) show \( T \) \( \text{is} \ T_2 \)
proof
{ let \( D_c = X \times X - \{ \langle x,x \rangle \mid x \in X \} \)
  fix \( x, y \) assume \( x \in X, y \in X, x \neq y \)
  with \text{assms} \( A \) have \( D_c \in \text{ProductTopology}(T, T) \) and \( \langle x, y \rangle \in D_c \)
  unfolding \text{IsClosed_def} by auto
  with \text{assms} obtain \( U, V \) where \( U \in T, V \in T, U \times V \subseteq D_c, \langle x, y \rangle \in U \times V \)
  using \text{prod_top_point_neighb} by blast
  moreover from \( \langle U \times V \subseteq D_c \rangle \) have \( U \cap V = 0 \) by auto
  ultimately have \( \exists U \in T. \exists V \in T. x \in U \wedge y \in V \wedge U \cap V = 0 \) by auto
} then show \( T \) \( \text{is} \ T_2 \) unfolding \text{isT2_def} by simp
qed
qed
end

62 Metric spaces

theory MetricSpace_ZF imports Topology_ZF_1 OrderedLoop_ZF Lattice_ZF begin

A metric space is a set on which a distance between points is defined as a function \( d : X \times X \to [0, \infty) \). With this definition each metric space is a topological space which is paracompact and Hausdorff \( (T_2) \), hence normal (in fact even perfectly normal).
62.1 Pseudometric - definition and basic properties

A metric on $X$ is usually defined as a function $d : X \times X \to [0, \infty)$ that satisfies the conditions $d(x, x) = 0$, $d(x, y) = 0 \Rightarrow x = y$ (identity of indiscernibles), $d(x, y) = d(y, x)$ (symmetry) and $d(x, y) \leq d(x, z) + d(z, y)$ (triangle inequality) for all $x, y \in X$. Here we are going to be a bit more general and define metric and pseudo-metric as a function valued in an ordered loop.

First we define a pseudo-metric, which has the axioms of a metric, but without the second part of the identity of indiscernibles. In our definition $\text{IsApseudoMetric}$ is a predicate on five sets: the function $d$, the set $X$ on which the metric is defined, the loop carrier $G$, the loop operation $A$ and the order $r$ on $G$.

\[ \text{definition} \quad \text{IsApseudoMetric}(d, X, G, A, r) \equiv d : X \times X \to \text{Nonnegative}(G, A, r) \]
\[ \land (\forall x \in X. \ d(x, x) = \text{TheNeutralElement}(G, A)) \]
\[ \land (\forall x \in X. \forall y \in X. \ d(x, y) = d(y, x)) \]
\[ \land (\forall x \in X. \forall y \in X. \forall z \in X. \ (d(x, z), A(d(x, y), d(y, z))) \in r) \]

We add the full axiom of identity of indiscernibles to the definition of a pseudometric to get the definition of metric.

\[ \text{definition} \quad \text{IsAmetric}(d, X, G, A, r) \equiv \text{IsApseudoMetric}(d, X, G, A, r) \land (\forall x \in X. \forall y \in X. \ d(x, y) = \text{TheNeutralElement}(G, A) \longrightarrow x = y) \]

A disk is defined as set of points located less than the radius from the center.

\[ \text{definition} \quad \text{Disk}(X, d, r, c, R) \equiv \{x \in X. \ (d(c, x), R) \in \text{StrictVersion}(r)\} \]

Next we define notation for metric spaces. We will reuse the additive notation defined in the $\text{loop1}$ locale adding only the assumption about $d$ being a pseudometric and notation for a disk centered at $c$ with radius $R$. Since for many theorems it is sufficient to assume the pseudometric axioms we will assume in this context that the sets $d, X, L, A, r$ form a pseudometric rather than a metric.

\[ \text{locale} \quad \text{pmetric_space} = \text{loop1} + \]
\[ \text{fixes} \quad d \text{ and } X \]
\[ \text{assumes} \quad \text{pmetricAssum: IsApseudoMetric}(d, X, L, A, r) \]
\[ \text{fixes} \quad \text{disk} \]
\[ \text{defines} \quad \text{disk_def [simp]: disk}(c, R) \equiv \text{Disk}(X, d, r, c, R) \]

The next lemma shows the definition of the pseudometric in the notation used in the $\text{metric_space}$ context.

\[ \text{lemma} \quad \text{(in pmetric_space) pmetric_properties: shows} \]
\[ d : X \times X \to L^+ \]

803
\( \forall x \in X. \text{d}(x,x) = 0 \)
\( \forall x \in X. \forall y \in X. \text{d}(x,y) = \text{d}(y,x) \)
\( \forall x \in X. \forall y \in X. \forall z \in X. \text{d}(x,z) \leq \text{d}(x,y) + \text{d}(y,z) \)

using \text{pmetricAssum unfolding IsApseudoMetric_def by auto}

The values of the metric are in the loop.

**Lemma (in pmetric_space) pmetric_loop_valued:** Assumes \( x \in X \ y \in X \)

**shows** \( \text{d}(x,y) \in L^+ \)

**proof**

- from assms show \( \text{d}(x,y) \in L^+ \) using \text{pmetric_properties(1)} apply_funtype
  by simp
- then show \( \text{d}(x,y) \in L \) using \text{Nonnegative_def} by auto

**qed**

The definition of the disk in the notation used in the \text{pmetric_space} context:

**Lemma (in pmetric_space) disk_definition:** Shows \( \text{disk}(c,R) = \{ x \in X. \text{d}(c,x) < R \} \)

**proof**

- have \( \text{disk}(c,R) = \text{Disk}(X,\text{d},r,c,R) \) by simp
- then have \( \text{disk}(c,R) = \{ x \in X. (\text{d}(c,x),R) \in \text{StrictVersion}(r) \} \) unfolding
  \text{Disk_def} by simp
- moreover have \( \forall x \in X. (\text{d}(c,x),R) \in \text{StrictVersion}(r) \iff \text{d}(c,x) < R \)
  using \text{def_of_strict_ver} by simp
- ultimately show thesis by auto

**qed**

If the radius is positive then the center is in disk.

**Lemma (in pmetric_space) center_in_disk:** Assumes \( c \in X \) and \( R \in L^+ \)

**shows** \( c \in \text{disk}(c,R) \)

**proof**

- using \text{pmetricAssum assms IsApseudoMetric_def PositiveSet_def disk_definition by simp}

A technical lemma that allows us to shorten some proofs:

**Lemma (in pmetric_space) radius_in_loop:** Assumes \( c \in X \) and \( x \in \text{disk}(c,R) \)

**shows** \( R \in L \ 	ext{and} \ R \in L^+ \ 	ext{and} \ (-\text{d}(c,x) + R) \in L^+ \)

**proof**

- from assms(2) have \( x \in X \) and \( \text{d}(c,x) < R \) using \text{disk_definition by auto}
  with assms(1) show \( 0 < R \) using \text{pmetric_properties(1)} apply_funtype
  \text{nonneg_definition loop_strict_ord_trans by blast}
- then show \( R \in L \) and \( R \in L^+ \) using \text{posset_definition PositiveSet_def by auto}
- from \( \text{d}(c,x) < R \) show \( (-\text{d}(c,x) + R) \in L^+ \)
  using \text{ls_other_side(2)} by simp

**qed**

If a point \( x \) is inside a disk \( B \) and \( m \leq R - \text{d}(c,x) \)

then the disk centered at the point \( x \) and with radius \( m \) is contained in the disk \( B \).

**Lemma (in pmetric_space) disk_in_disk:**
assumes $c \in X$ and $x \in \text{disk}(c,R)$ and $m \leq (-d(c,x) + R)$
shows $\text{disk}(x,m) \subseteq \text{disk}(c,R)$
proof
fix $y$ assume $y \in \text{disk}(x,m)$
then have $d(x,y) < m$ using \text{disk_definition} by simp
from \text{assms}(1,2) $\langle y \in \text{disk}(x,m) \rangle$ have $R \in L$ $x \in X$ $y \in X$
using \text{radius_in_loop}(1) \text{disk_definition} by auto
with \text{assms}(1) have $d(c,y) \leq d(c,x) + d(x,y)$ using \text{pmetric_properties}(4)
by simp
from \text{assms}(1) $\langle x \in X \rangle$ have $d(c,x) \in L$
using \text{pmetric_properties}(1) \text{apply_functype} \text{nonneg_subset} by auto
with $\langle d(x,y) < m \rangle$ \text{assms}(3) have $d(c,x) + d(x,y) < d(c,x) + (-d(c,x) + R)$
using \text{loop_strict_ord_trans1 strict_ord_trans_inv}(2) by blast
with $\langle d(c,x) \in L \rangle$ $\langle R \in L \rangle$ $\langle d(c,y) \leq d(c,x) + d(x,y) \rangle$ $\langle y \in X \rangle$ show $y \in \text{disk}(c,R)$
using \text{irdiv_props}(6) \text{loop_strict_ord_trans disk_definition} by simp
qed

If we assume that the order on the group makes the positive set a meet semi-lattice (i.e. every two-element subset of $G_+$ has a greatest lower bound) then the collection of disks centered at points of the space and with radii in the positive set of the group satisfies the base condition. The meet semi-lattice assumption can be weakened to "each two-element subset of $G_+$ has a lower bound in $G_+$", but we don't do that here.

\textbf{lemma (in pmetric_space) disks_form_base:}
assumes $r \{\text{down-directs}\} L_+$
defines $B \equiv \bigcup c \in X. \{\text{disk}(c,R), \ R \in L_+\}$
shows $B \{\text{satisfies the base condition}\}$
proof -
\{ fix $U \ V$ assume $U \in B$ $V \in B$
fix $x$ assume $x \in U \cap V$
have $\exists w \in B. \ x \in w \land w \subseteq U \cap V$
proof -
from \text{assms}(2) $\langle U \in B \rangle$ $\langle V \in B \rangle$ obtain $c_U$ $c_V$ $R_U$ $R_V$
where $c_U \in X$ $R_U \in L_+$ $c_V \in X$ $R_V \in L_+$ $U = \text{disk}(c_U,R_U)$ $V = \text{disk}(c_V,R_V)$
by auto
with $\langle x \in U \cap V \rangle$ have $x \in \text{disk}(c_U,R_U)$ and $x \in \text{disk}(c_V,R_V)$ by auto
then have $x \in X$ $d(c_U,x) < R_U$ $d(c_V,x) < R_V$ using \text{disk_definition} by auto
let $m_U = - d(c_U,x) + R_U$
let $m_V = - d(c_V,x) + R_V$
from $\langle c_U \in X \rangle$ $\langle x \in \text{disk}(c_U,R_U) \rangle$ $\langle c_V \in X \rangle$ $\langle x \in \text{disk}(c_V,R_V) \rangle$ have $m_U \in L_+$ and $m_V \in L_+$
using \text{radius_in_loop}(4) by auto
with \text{assms}(1) obtain $m$ where $m \in L_+$ $m \leq m_U$ $m \leq m_V$
unfolding \text{DownDirects_def} by auto
let $w = \text{disk}(x,m)$
from $\langle m \in L_+ \rangle$ $\langle m \leq m_U \rangle$ $\langle m \leq m_V \rangle$
\langle $c_U \in X \rangle$ $\langle x \in \text{disk}(c_U,R_U) \rangle$ $\langle c_V \in X \rangle$ $\langle x \in \text{disk}(c_V,R_V) \rangle$
Disks centered at points farther away than the sum of radii do not overlap.

**Lemma (in pmetric_space) far_disks:**
assumes \( x \in X \) \( y \in X \) \( r_x \leq d(x,y) \)
sorts \( \text{disk}(x,r_x) \cap \text{disk}(y,r_y) = 0 \)

**Proof** -
\[
\{ \text{assume } \text{disk}(x,r_x) \cap \text{disk}(y,r_y) \neq 0 \\
\text{then obtain } z \text{ where } z \in \text{disk}(x,r_x) \cap \text{disk}(y,r_y) \text{ by auto} \\
\text{then have } z \in X \text{ and } d(x,z) + d(y,z) < r_x + r_y \\
\text{using disk_definition add_ineq_strict by auto} \\
\text{moreover from asms(1,2) } <z \in X> \text{ have } d(x,y) \leq d(x,z) + d(y,z) \\
\text{using pmetric_properties(3,4) by auto} \\
\text{ultimately have } d(x,y) < r_x + r_y \text{ using loop_strict_ord_trans by simp} \\
\text{with asms(3) have False using loop_strict_ord_trans by auto} \\
\} \text{ thus thesis by auto} \]

**QED**

If we have a loop element that is smaller than the distance between two points, then we can separate these points with disks.

**Lemma (in pmetric_space) disjoint_disks:**
assumes \( x \in X \) \( y \in X \) \( r_x < d(x,y) \)
sorts \( \text{disk}(x,r_x) \cap \text{disk}(y,-r_x + d(x,y)) = 0 \)

**Proof** -
\[
\text{from asms(3) show } (-r_x + (d(x,y))) \in L_+ \\
\text{using is_other_side posset_definition1 by simp} \\
\text{from asms(1,2,3) have } r_x \in L \text{ and } d(x,y) \in L \\
\text{using less_members(1) pmetric_loop_valued(2) by auto} \\
\text{then have } r_x + (-r_x + (d(x,y))) = d(x,y) \text{ using lrdiv_props(6) by simp} \\
\text{with asms(1,2) } <d(x,y) \in L> \text{ show } \text{disk}(x,r_x) \cap \text{disk}(y,-r_x + (d(x,y))) = 0 \\
\text{using loop_ord_refl far_disks by simp} \]

**QED**

Unions of disks form a topology, hence (pseudo)metric spaces are topological spaces.

**Theorem (in pmetric_space) pmetric_is_top:**
assumes \( r \) \{down-directs\} \( L_+ \)
defines \( B \equiv \bigcup_{c \in X} \{ \text{disk}(c,R). R \in L_+ \} \)
defines \( T \equiv \{ \bigcup A. A \in \text{Pow}(B) \} \)
shows \( T \) is a topology \( B \) is a base for \( T \) \( \bigcup T = X \)

proof -
from assms show \( T \) is a topology \( B \) is a base for \( T \)
using disks_form_base Top_1_2_T1 by auto
then have \( \bigcup T = \bigcup B \) using Top_1_2_L5 by simp
moreover have \( \bigcup B = X \)
proof
from assms(2) show \( \bigcup B \subseteq X \)
using disk_definition by auto
\{ fix \( x \) assume \( x \in X \)
from assms(1) obtain \( R \) where \( R \in L^+ \)
unfolding DownDirects_def by blast
with assms(2) \( \langle x \in X \rangle \) have \( x \in \bigcup B \)
\} thus \( X \subseteq \bigcup B \)
by auto
qed
ultimately show \( \bigcup T = X \)
by simp
qed

To define the metric_space locale we take the pmetric_space and add the assumption of identity of indiscernibles.

locale metric_space = pmetric_space +
assumes ident_indisc: \( \forall x \in X. \forall y \in X. d \langle x,y \rangle = 0 \rightarrow x=y \)

In the metric_space locale \( d \) is a metric.

lemma (in metric_space) d_metric: shows IsAmetric(d,X,L,A,r)
using pmetricAssum ident_indisc unfolding IsAmetric_def by simp

Distance of different points is greater than zero.

lemma (in metric_space) dist_pos: assumes \( x \in X \) \( y \in X \) \( x \neq y \)
shows \( 0 < d \langle x,y \rangle \)
\( d \langle x,y \rangle \in L^+ \)

proof -
from assms(1,2) have \( d \langle x,y \rangle \in L^+ \)
using pmetric_properties(1) apply_funtype by simp
then have \( 0 \leq d \langle x,y \rangle \)
using Nonnegative_def by auto
with assms show \( d \langle x,y \rangle \in L^+ \) and \( 0 < d \langle x,y \rangle \)
using ident_indisc posset_definition posset_definition1 by auto
qed

An ordered loop valued metric space is \( T_2 \) (i.e. Hausdorff).

theorem (in metric_space) metric_space_T2:
assumes \( r \) \{down-directs\} \( L^+ \)
defines \( B \equiv \bigcup c \in X. \{ \text{disk}(c,R). R \in L^+ \} \)
defines \( T \equiv \{ \bigcup A. A \in \text{Pow}(B) \} \)
shows \( T \) is \( T_2 \)

proof -
\{ fix \( x \) \( y \) assume \( x \in \bigcup T \) \( y \in \bigcup T \) \( x \neq y \)
from assms have \( B \subseteq T \)
using pmetric_is_top(2) base_sets_open by auto
moreover have \( \exists U \in B. \exists V \in B. x \in U \) \( y \in V \) \( U \cap V = 0 \)
\}
proof -

let R = d⟨x,y⟩

from assms have \( \bigcup T = X \) using pmetric_is_top(3) by simp

with \( \langle x \in \bigcup T \rangle \langle y \in \bigcup T \rangle \) have \( x \in X \) y \( y \in X \) by auto

with \( x \neq y \) have \( R \in L_+ \) using dist_pos by simp

with assms(2) \( \langle x \in X \rangle \langle y \in X \rangle \) have disk\( (x,R) \in B \) and disk\( (y,R) \in B \)

by auto

\{ assume disk\( (x,R) \cap disk(y,R) = 0 \)

moreover from assms(2) \( \langle x \in X \rangle \langle y \in X \rangle \langle R \in L_+ \rangle \) have

\( disk(x,R) \in B \) disk\( (y,R) \in B \) x \( x \in disk(x,R) \) y \( y \in disk(y,R) \)

using center_in_disk by auto

ultimately have \( \exists U \in B. \exists V \in B. \ x \in U \land y \in V \land U \cap V = 0 \) by auto

\}

moreover

\{ assume disk\( (x,R) \cap disk(y,R) \neq 0 \)

then obtain z where \( z \in disk(x,R) \) and \( z \in disk(y,R) \)

by auto

then have \( d\langle x,z \rangle < R \) using disk_definition by simp

then have \( 0 < -d\langle x,z \rangle + R \) using ls_other_side(1) by simp

let \( r = -d\langle x,z \rangle + R \)

have \( r < R \)

proof -

from \( \langle z \in disk(y,R) \rangle \langle x \in X \rangle \langle y \in X \rangle \) have \( z \in X \) x \( x \neq z \)

using disk_definition pmetric_properties(3) by auto

with \( \langle x \in X \rangle \langle y \in X \rangle \langle z \in X \rangle \) show thesis

using pmetric_loop_valued dist_pos(1) subtract_pos(2) by simp

qed

with \( \langle x \in X \rangle \langle y \in X \rangle \) have disk\( (x,r) \cap disk(y,-r+R) = 0 \)

by (rule disjoint_disks)

moreover

from \( \langle 0<\langle r \rangle \rangle \langle r \in R \rangle \) have \( r \in L_+ \) \( -r+R \in L_+ \)

using ls_other_side posset_definition1 by auto

with assms(2) \( \langle x \in X \rangle \langle y \in X \rangle \) have

\( disk(x,r) \in B \) disk\( (y,-r+(d\langle x,y \rangle)) \in B \) and

\( x \in disk(x,r) \) y \( y \in disk(y,-r+(d\langle x,y \rangle)) \)

using center_in_disk by auto

ultimately have \( \exists U \in B. \exists V \in B. \ x \in U \land y \in V \land U \cap V = 0 \) by auto

\}

ultimately show thesis by auto

qed

ultimately have \( \exists U \in T. \exists V \in T. \ x \in U \land y \in V \land U \cap V = 0 \) by auto

\} then show thesis unfolding isT2_def by simp

qed

end

808
63 Basic properties of real numbers

theory Real_ZF_2 imports OrderedField_ZF MetricSpace_ZF begin

Isabelle/ZF and IsarMathLib do not have a set of real numbers built-in. The Real_ZF and Real_ZF_1 theories provide a construction but here we do not use it in any way and we just assume that we have a model of real numbers (i.e. a completely ordered field) as defined in the Ordered_Field theory. The construction only assures us that objects with the desired properties exist in the ZF world.

63.1 Basic notation for real numbers

In this section we define notation that we will use whenever real numbers play a role, i.e. most of mathematics.

The next locale sets up notation for contexts where real numbers are used.

locale reals = 
  fixes Reals\(\mathbb{R}\) and Add and Mul and ROrd
  assumes R_are_reals: IsAmodelOfReals(\(\mathbb{R}\),Add,Mul, ROrd)

  fixes zero \(0\)
  defines zero_def[simp]: \(0 \equiv \text{TheNeutralElement}(\mathbb{R},\text{Add})\)

  fixes one \(1\)
  defines one_def[simp]: \(1 \equiv \text{TheNeutralElement}(\mathbb{R},\text{Mul})\)

  fixesrealmul (infixl \(\cdot\) 71)
  defines realmul_def[simp]: \(x \cdot y \equiv \text{Mul}(x,y)\)

  fixes realadd (infixl + 69)
  defines realadd_def[simp]: \(x + y \equiv \text{Add}(x,y)\)

  fixes realminus(- _ 89)
  defines realminus_def[simp]: \((-x) \equiv \text{GroupInv}(\mathbb{R},\text{Add})(x)\)

  fixes realsub (infixl - 90)
  defines realsub_def [simp]: \(x-y \equiv x+(-y)\)

  fixes lesseq (infix \(\le\) 68)
  defines lesseq_def [simp]: \(x \leq y \equiv (x,y) \in \text{ROrd}\)

  fixes sless (infix \(<\) 68)
  defines sless_def [simp]: \(x < y \equiv x \leq y \land x 
eq y\)

  fixes nonnegative (\(\mathbb{R}^+\))
  defines nonnegative_def[simp]: \(\mathbb{R}^+ \equiv \text{Nonnegative}(\mathbb{R},\text{Add}, \text{ROrd})\)

809
fixes positiveset \((R_+)\)
defines positiveset_def\[simp\]: \(R_+ \equiv \text{PositiveSet}(\text{Add}, \text{ROrd})\)

fixes setinv \((- _ 72)\)
defines setinv_def\[simp\]: \(-A \equiv \text{GroupInv}(\text{Add})(A)\)

fixes non_zero \((R_0)\)
defines non_zero_def\[simp\]: \(R_0 \equiv R\{0\}\)

fixes abs \((| _ |)\)
defines abs_def\[simp\]: \(|x| \equiv \text{AbsoluteValue}(\text{Add}, \text{ROrd})(x)\)

fixes dist
defines dist_def\[simp\]: dist \(\equiv \{(p,|\text{fst}(p) - \text{snd}(p)|) : p \in R\times R\}\)

fixes two \((2)\)
defines two_def\[simp\]: \(2 \equiv 1 + 1\)

fixes inv \((_^{-1} [96] 97)\)
defines inv_def\[simp\]: \(x^{-1} \equiv \text{GroupInv}(R_0, \text{restrict} (\text{Mul}, R_0\times R_0))(x)\)

fixes realsq \((^2 [96] 97)\)
defines realsq_def\[simp\]: \(x^2 \equiv x\cdot x\)

fixes oddext \((^\circ)\)
defines oddext_def\[simp\]: \(f^\circ \equiv \text{OddExtension}(\text{Add}, \text{ROrd}, f)\)

fixes disk
defines disk_def\[simp\]: disk(c,r) \(\equiv \text{Disk}(R, \text{dist}, \text{ROrd}, c, r)\)

The assumptions of the field1 locale (that sets the context for ordered fields) hold in the reals locale.

**lemma** (in reals) field1_is_valid: shows field1(R, Add, Mul, ROrd)

**proof**
from R_are_reals show IsAring(R, Add, Mul) and Mul {is commutative on} R and ROrd \(\subseteq R \times R\) and IsLinOrder(R, ROrd) and \(\forall x\ y. \forall z\in R. \langle x, y\rangle \in ROrd \longrightarrow (\langle \text{Add}(x, z), \text{Add}(y, z)\rangle \in ROrd\) and Nonnegative(R, Add, ROrd) {is closed under} Mul and TheNeutralElement(R, Add) \(\neq\) TheNeutralElement(R, Mul) and \(\forall x\in R. x \neq \text{TheNeutralElement}(R, \text{Add}) \longrightarrow (\exists y\in R. \text{Mul}(x, y) = \text{TheNeutralElement}(R, \text{Mul}))\) using IsAmodelOfReals_def IsAnOrdField_def IsAnOrdRing_def by auto

**qed**

We can use theorems proven in the field1 locale in the reals locale. Note that since the the field1 locale is an extension of the ring1 locale, which is an extension of ring0 locale, this makes available also the theorems proven
in the ring1 and ring0 locales.

sublocale reals < field1 Reals Add Mul realadd realminus realsub realmul
  using field1_is_valid by auto

The group3 locale from the OrderedGroup_ZF theory defines context for theorems about ordered groups. We can use theorems proven in there in the reals locale as real numbers with addition form an ordered group.

sublocale reals < group3 Reals Add ROrd zero realadd realminus lesseq
  unfolding group3_def using OrdRing_ZF_1_L4 by auto

Since real numbers with addition form a group we can use the theorems proven in the group0 locale defined in the Group_ZF theory in the reals locale.

sublocale reals < group0 Reals Add zero realadd realminus
  unfolding group3_def using OrderedGroup_ZF_1_L1 by auto

Let’s recall basic properties of the real line.

lemma (in reals) basic_props: shows ROrd {is total on}
  and Add {is commutative on}
  using OrdRing_ZF_1_L4(2,3) by auto

The distance function dist defined in the reals locale is a metric.

lemma (in reals) dist_is_metric: shows dist : \R x y \rightarrow \R
  ∀x∈\R.∀y∈\R. dist(x,y) = |x - y|  
  ∀x∈\R. dist(x,x) = 0
  ∀x∈\R.∀y∈\R. dist(x,y) = dist(y,x)
  ∀x∈\R.∀y∈\R.∀z∈\R. |x - z| ≤ |x - y| + |y - z|
  ∀x∈\R.∀y∈\R.∀z∈\R. dist(x,z) ≤ dist(x, y) + dist(y,z)
  ∀x∈\R.∀y∈\R. dist(x,y) = 0 \implies x=y
  IsAmetric(dist,\R,\R,Add,ROrd)
proof
  show I: dist : \R x y \rightarrow \R+ using add_group.group_op_closed add_group.inverse_in_group
    OrdRing_ZF_1_L4
    then show II:∀x∈\R.∀y∈\R. dist(x,y) = |x-y| using ZF_fun_from_tot_val0
      by auto
    then show III: ∀x∈\R. dist(x,x) = 0 using add_group.group0_2_L6 OrderedGroup_ZF_3_L2A
      by simp
      { fix x y
        assume x∈\R y∈\R
        then have (-(x-y)) = y-x using add_group.group0_2_L12 by simp
        moreover from (x∈\R) (y∈\R) have |-(x-y)| =|x-y|
          using add_group.group_op_closed add_group.inverse_in_group basic_props(1)
      }
    }
OrderedGroup_ZF_3_L2B
by simp 
ultimately have \(|y-x| = |x-y|\) by simp 
with \(<x \in \mathbb{R}, y \in \mathbb{R}>\) II have \(\text{dist}(x,y) = \text{dist}(y,x)\) by simp 
\}
thus IV: \(\forall x \in \mathbb{R}, \forall y \in \mathbb{R}. \text{dist}(x,y) = \text{dist}(y,x)\) by simp 
\{ fix x y 
assume \(x \in \mathbb{R}, y \in \mathbb{R}\) \(\text{dist}(x,y) = 0\) by simp 
with \(<x \in \mathbb{R}, y \in \mathbb{R}>\) have \(|x-y| = 0\) by simp 
using add_group.group_op_closed add_group.inverse_in_group OrderedGroup_ZF_3_L3D by auto 
with \(<x \in \mathbb{R}, y \in \mathbb{R}>\) have \(|x-y| = 0\) by simp 
using add_group.group0_2_L11A by simp 
\}
thus V: \(\forall x \in \mathbb{R}. \forall y \in \mathbb{R}. \forall z \in \mathbb{R}. \text{dist}(x,z) \leq \text{dist}(x,y) + \text{dist}(y,z)\) by simp 
with I III IV V show IsApseudoMetric(\(\text{dist}, \mathbb{R}, \text{Add}, \text{ROrd}\)) and IsAmetric(\(\text{dist}, \mathbb{R}, \text{Add}, \text{ROrd}\)) unfolding IsApseudoMetric_def IsAmetric_def by auto 
qed 

Real numbers form an ordered loop.

lemma (in reals) reals_loop: shows IsAnOrdLoop(\(\mathbb{R}, \text{Add}, \text{ROrd}\)) 
proof - 
  have IsAloop(\(\mathbb{R}, \text{Add}\)) using add_group.group_is_loop by simp 
  moreover from R_are_reals have \(\text{ROrd} \subseteq \mathbb{R} \times \mathbb{R}\) and IsPartOrder(\(\mathbb{R}, \text{ROrd}\)) 
    using IsAmodelOfReals_def IsAnOrdField_def IsAnOrdRing_def Order_ZF_1_L2 
by auto 
moreover 
\{ fix x y z assume A: \(x \in \mathbb{R}, y \in \mathbb{R}, z \in \mathbb{R}\) 
  then have \(x \leq y \longleftrightarrow x+z \leq y+z\) 
    using ord_transl_inv ineq_cancel_right by blast 
  moreover from A have \(x \leq y \longleftrightarrow z+x \leq z+y\) 
    using ord_transl_inv OrderedGroup_ZF_1_L5AE by blast 
  ultimately have \((x \leq y \longleftrightarrow x+z \leq y+z) \land (x \leq y \longleftrightarrow z+x \leq z+y)\) 
    by simp 
\}
ultimately show IsAnOrdLoop(\(\mathbb{R}, \text{Add}, \text{ROrd}\)) unfolding IsAnOrdLoop_def by auto 
qed 

The assumptions of the \(\text{pmetric_space}\) locale hold in the \(\text{reals}\) locale.
lemma (in reals) pmetric_space_valid: shows pmetric_space(R,Add, ROrd,dist,R)

unfolding pmetric_space_def pmetric_space_axioms_def loop1_def
using reals_loop dist_is_metric(8)
by blast

The assumptions of the metric_space locale hold in the reals locale.

lemma (in reals) metric_space_valid: shows metric_space(R,Add, ROrd,dist,R)
proof -
  have ∀x∈R. ∀y∈R. dist⟨x,y⟩=0 −→ x=y
    using dist_is_metric(9) unfolding IsAmetric_def
    by auto
  then show thesis unfolding metric_space_def metric_space_axioms_def
    using pmetric_space_valid
    by simp
qed

Some properties of the order relation on reals:

lemma (in reals) pos_is_lattice: shows IsLinOrder(R,ROrd)
  IsLinOrder(R⁺,ROrd ∩ R⁺×R⁺)
(R̅Ord ∩ R⁺×R⁺) {is a lattice on} R⁺
proof -
  show IsLinOrder(R,ROrd) using OrdRing_ZF_1_L1 unfolding IsAnOrdRing_def
  by simp
  moreover have R⁺ ⊆ R using pos_set_in_gr by simp
  ultimately show IsLinOrder(R⁺,ROrd ∩ R⁺×R⁺) using ord_linear_subset(2)
  by simp
  moreover have (R̅Ord ∩ R⁺×R⁺) ⊆ R⁺×R⁺ by auto
  ultimately show (R̅Ord ∩ R⁺×R⁺) {is a lattice on} R⁺ using lin_is_latt
  by simp
qed

Of course the set of positive real numbers is nonempty as one is there.

lemma (in reals) pos_non_empty: shows R⁺≠0
using R_are_reals ordring_one_is_pos
unfolding IsAmodelOfReals_def IsAnOrdField_def by auto

We say that a relation r down-directs a set R if every two-element subset of R has a lower bound. The next lemma states that the natural order relation on real numbers down-directs the set of positive reals.

lemma (in reals) rord_down_directs: shows ROrd {down-directs} R⁺
using pos_is_lattice(3) pos_non_empty meet_down_directs down_dir_mono
unfolding IsAlatrace_def by blast

We define the topology on reals as one consisting of the unions of open disks.

definition (in reals) RealTopology (τR)
where τR ≡ {∪A. A ∈ Pow(∪c∈R.{disk(c,r) . r ∈ R⁺})}

813
Real numbers form a Hausdorff topological space with topology generated by open disks.

**theorem (in reals) reals_is_top:**

shows \( \tau_R \) is a topology \( \bigcup \tau_R = \tau_R \) is T\(_2\)

using rord_down_directs metric_space_valid pmetric_space_valid
  pmetric_space.pmetric_is_top metric_space.metric_space_T2
unfolding RealTopology_def
  by simp_all

end

64 Complex numbers

theory Complex_ZF imports func_ZF_1 OrderedField_ZF
begin

The goal of this theory is to define complex numbers and prove that the Metamath complex numbers axioms hold.

64.1 From complete ordered fields to complex numbers

This section consists mostly of definitions and a proof context for talking about complex numbers. Suppose we have a set \( R \) with binary operations \( A \) and \( M \) and a relation \( r \) such that the quadruple \((R, A, M, r)\) forms a complete ordered field. The next definitions take \((R, A, M, r)\) and construct the sets that represent the structure of complex numbers: the carrier \((C = R \times R)\), binary operations of addition and multiplication of complex numbers and the order relation on \( R = R \times 0 \). The \( \text{ImCxAdd} \), \( \text{ReCxAdd} \), \( \text{ImCxMul} \), \( \text{ReCxMul} \) are helper meta-functions representing the imaginary part of a sum of complex numbers, the real part of a sum of real numbers, the imaginary part of a product of complex numbers and the real part of a product of real numbers, respectively. The actual operations (subsets of \((R \times R) \times R\) are named \( \text{CplxAdd} \) and \( \text{CplxMul} \).

When \( R \) is an ordered field, it comes with an order relation. This induces a natural strict order relation on \( \{(x, 0) : x \in R\} \subseteq R \times R \). We call the set \( \{(x, 0) : x \in R\} \) \( \text{ComplexReals}(R, A) \) and the strict order relation \( \text{CplxROrder}(R, A, r) \). The order on the real axis of complex numbers is defined as the relation induced on it by the canonical projection on the first coordinate and the order we have on the real numbers. OK, let's repeat this slower. We start with the order relation \( r \) on a (model of) real numbers \( R \). We want to define an order relation on a subset of complex numbers, namely on \( R \times \{0\} \). To do that we use the notion of a relation induced by a mapping. The mapping here is \( f : R \times \{0\} \to R, f(x, 0) = x \) which is defined under a name of \( \text{SliceProjection} \) in \text{func_ZF.thy}. This defines a relation \( r_1 \) (called...
InducedRelation(f,r), see func_ZF on $R \times \{0\}$ such that $\langle x, 0, y, 0 \rangle \in r_1$ iff $\langle x, y \rangle \in r$. This way we get what we call CplxROrder(R,A,r). However, this is not the end of the story, because Metamath uses strict inequalities in its axioms, rather than weak ones like IsarMathLib (mostly). So we need to take the strict version of this order relation. This is done in the syntax definition of $<_R$ in the definition of complex0 context. Since Metamath proves a lot of theorems about the real numbers extended with $+\infty$ and $-\infty$, we define the notation for inequalities on the extended real line as well.

A helper expression representing the real part of the sum of two complex numbers.

**definition**

\[
\text{ReCxAdd}(R,A,a,b) \equiv A(fst(a),fst(b))
\]

An expression representing the imaginary part of the sum of two complex numbers.

**definition**

\[
\text{ImCxAdd}(R,A,a,b) \equiv A(snd(a),snd(b))
\]

The set (function) that is the binary operation that adds complex numbers.

**definition**

\[
\text{CplxAdd}(R,A) \equiv \{ (\langle p, \langle \text{ReCxAdd}(R,A,fst(p),snd(p)), \text{ImCxAdd}(R,A,fst(p),snd(p)) \rangle \rangle . p \in (R \times R) \times (R \times R) \}
\]

The expression representing the imaginary part of the product of complex numbers.

**definition**

\[
\text{ImCxMul}(R,A,M,a,b) \equiv A(M(fst(a),snd(b)), M(snd(a),fst(b)) \}
\]

The expression representing the real part of the product of complex numbers.

**definition**

\[
\text{ReCxMul}(R,A,M,a,b) \equiv A(M(fst(a),fst(b)), \text{GroupInv}(R,A)(M(snd(a),snd(b))))
\]

The function (set) that represents the binary operation of multiplication of complex numbers.

**definition**

\[
\text{CplxMul}(R,A,M) \equiv \{ (\langle p, \langle \text{ReCxMul}(R,A,M,fst(p),snd(p)), \text{ImCxMul}(R,A,M,fst(p),snd(p)) \rangle \rangle . p \in (R \times R) \times (R \times R) \}
\]

The definition real numbers embedded in the complex plane.

**definition**

\[
\text{ComplexReals}(R,A) \equiv R \times \{ \text{TheNeutralElement}(R,A) \}
\]
Definition of order relation on the real line.

**Definition**

CplxROrder(R,A,r) ≡ 
\[ \text{InducedRelation}(\text{SliceProjection}(\text{ComplexReals}(R,A)),r) \]

The next locale defines proof context and notation that will be used for complex numbers.

**locale complex0 =**

**fixes R and A and M and r**

**assumes R_are_reals: IsAmodelOfReals(R,A,M,r)**

**fixes complex (C)**

**defines complex_def[simp]: C ≡ R×R**

**fixes rone (1_R)**

**defines rone_def[simp]: 1_R ≡ \text{TheNeutralElement}(R,M)**

**fixes rzero (0_R)**

**defines rzero_def[simp]: 0_R ≡ \text{TheNeutralElement}(R,A)**

**fixes one (1)**

**defines one_def[simp]: 1 ≡ \{1_R, 0_R\}**

**fixes zero (0)**

**defines zero_def[simp]: 0 ≡ \{0_R, 0_R\}**

**fixes iunit (i)**

**defines iunit_def[simp]: i ≡ \{0_R,1_R\}**

**fixes creal (R)**

**defines creal_def[simp]: R ≡ \{(r,0_R). r∈R\}**

**fixes rmul (infixl · 71)**

**defines rmul_def[simp]: a · b ≡ M\langle a,b \rangle**

**fixes radd (infixl + 69)**

**defines radd_def[simp]: a + b ≡ A\langle a,b \rangle**

**fixes rneg (- _ 70)**

**defines rneg_def[simp]: \(- a ≡ \text{GroupInv}(R,A)(a)\)**

**fixes ca (infixl + 69)**

**defines ca_def[simp]: a + b ≡ CplxAdd(R,A)\langle a,b \rangle**

**fixes cm (infixl · 71)**

**defines cm_def[simp]: a · b ≡ CplxMux(R,A,M)\langle a,b \rangle**

**fixes cdiv (infixl / 70)**

**defines cdiv_def[simp]: a / b ≡ \bigcup \{ x ∈ C. b · x = a \}**
fixes \texttt{sub} (infixl - 69)
defines \texttt{sub\_def}[simp]: a - b ≡ \bigcup \{ x \in \mathbb{C}. b + x = a \}

fixes \texttt{cneg} (-_ 95)
defines \texttt{cneg\_def}[simp]: - a ≡ 0 - a

fixes \texttt{lesser} (infix \texttt{<_} 68)
defines \texttt{lesser\_def}[simp]:
a \texttt{<_} b ≡ (a, b) ∈ \texttt{StrictVersion(CplxROrder(R,A,r))}

fixes \texttt{cpnf} (+\infty)
defines \texttt{cpnf\_def}[simp]: +\infty ≡ \mathbb{C}

fixes \texttt{cmnf} (-\infty)
defines \texttt{cmnf\_def}[simp]: -\infty ≡ \{\mathbb{C}\}

fixes \texttt{cxr} (R^+)
defines \texttt{cxr\_def}[simp]: R^+ ≡ R \cup \{+\infty, -\infty\}

fixes \texttt{cxn} (N)
defines \texttt{cxn\_def}[simp]:
N ≡ \bigcap \{ N ∈ \texttt{Pow}(R). 1 ∈ N \land (\forall n. n \in N \implies n + 1 \in N) \}

fixes \texttt{clttrrset} (<)
defines \texttt{clttrrset\_def}[simp]:
< ≡ \texttt{StrictVersion(CplxROrder(R,A,r))} \cap R \times R \cup \{(-\infty, +\infty)\} \cup (R \times \{+\infty\}) \cup (\{-\infty\} \times R)

fixes \texttt{cltrr} (infix < 68)
defines \texttt{cltrr\_def}[simp]: a < b ≡ (a, b) ∈ <

fixes \texttt{lsq} (infix \leq 68)
defines \texttt{lsq\_def}[simp]: a \leq b ≡ \neg (b < a)

fixes \texttt{two} (2)
defines \texttt{two\_def}[simp]: 2 ≡ 1 + 1

fixes \texttt{three} (3)
defines \texttt{three\_def}[simp]: 3 ≡ 2 + 1

fixes \texttt{four} (4)
defines \texttt{four\_def}[simp]: 4 ≡ 3 + 1

fixes \texttt{five} (5)
defines \texttt{five\_def}[simp]: 5 ≡ 4 + 1

fixes \texttt{six} (6)
defines \texttt{six\_def}[simp]: 6 ≡ 5 + 1
fixes seven (7)  
defines seven_def[simp]: \( 7 \equiv 6+1 \)

fixes eight (8)  
defines eight_def[simp]: \( 8 \equiv 7+1 \)

fixes nine (9)  
defines nine_def[simp]: \( 9 \equiv 8+1 \)

### 64.2 Axioms of complex numbers

In this section we will prove that all Metamath’s axioms of complex numbers hold in the `complex0` context.

The next lemma lists some contexts that are valid in the `complex0` context.

**lemma (in complex0) valid_cntxts: shows**

- `field1(R,A,M,r)`
- `field0(R,A,M)`
- `ring1(R,A,M,r)`
- `group3(R,A,r)`
- `ring0(R,A,M)`
- `M {is commutative on} R`
- `group0(R,A)`

**proof**

- from `R_are_reals` have `I: IsAnOrdField(R,A,M,r)` using `IsAmodelOfReals_def` by simp
- then show `field1(R,A,M,r)` using `OrdField_ZF_1_L2` by simp
- then show `ring1(R,A,M,r)` and `I: field0(R,A,M)` using `field1.axioms ring1_def field1.OrdField_ZF_1_L1B` by auto
- then show `group3(R,A,r)` using `ring1.OrdRing_ZF_1_L4` by simp
- from `I` have `IsAfield(R,A,M)` using `field0.Field_ZF_1_L1` by simp
- then have `IsAring(R,A,M)` and `M {is commutative on} R` using `IsAfield_def` by auto
- then show `ring0(R,A,M)` and `M {is commutative on} R` using `ring0_def` by auto
- then show `group0(R,A)` using `ring0.Ring_ZF_1_L1` by simp

**qed**

The next lemma shows the definition of real and imaginary part of complex sum and product in a more readable form using notation defined in `complex0` locale.

**lemma (in complex0) cplx_mul_add_defs: shows**

- `ReCxAdd(R,A,⟨a,b⟩,⟨c,d⟩) = a + c`
\( \text{ImCxAdd}(R,A,\langle a,b \rangle,\langle c,d \rangle) = b + d \)
\( \text{ImCxMul}(R,A,M,\langle a,b \rangle,\langle c,d \rangle) = a \cdot d + b \cdot c \)
\( \text{ReCxMul}(R,A,M,\langle a,b \rangle,\langle c,d \rangle) = a \cdot c + (-b \cdot d) \)

**proof**

- let \( z_1 = \langle a,b \rangle \)
- let \( z_2 = \langle c,d \rangle \)

\[ \text{have ReCxAdd}(R,A,z_1,z_2) \equiv A(fst(z_1),fst(z_2)) \]
by (rule ReCxAdd_def)
moreover have \( \text{ImCxAdd}(R,A,z_1,z_2) \equiv A(snd(z_1),snd(z_2)) \)
by (rule ImCxAdd_def)
moreover have
\[ \text{ImCxMul}(R,A,M,z_1,z_2) \equiv A(M<fst(z_1),snd(z_2)>,M<snd(z_1),fst(z_2)>) \]
by (rule ImCxMul_def)
moreover have
\[ \text{ReCxMul}(R,A,M,z_1,z_2) \equiv A(M<fst(z_1),fst(z_2)>,\text{GroupInv}(R,A)(M<snd(z_1),snd(z_2)>) \]
by (rule ReCxMul_def)
ultimately show
\[ \text{ReCxAdd}(R,A,z_1,z_2) = a + c \]
\[ \text{ImCxAdd}(R,A,\langle a,b \rangle,\langle c,d \rangle) = b + d \]
\[ \text{ImCxMul}(R,A,M,\langle a,b \rangle,\langle c,d \rangle) = a \cdot d + b \cdot c \]
\[ \text{ReCxMul}(R,A,M,\langle a,b \rangle,\langle c,d \rangle) = a \cdot c + (-b \cdot d) \]
by auto

qed

Real and imaginary parts of sums and products of complex numbers are real.

**lemma** (in complex0) cplx_mul_add_types:

- assumes \( A1: z_1 \in \mathbb{C} \quad z_2 \in \mathbb{C} \)

shows
\[ \text{ReCxAdd}(R,A,z_1,z_2) \in R \]
\[ \text{ImCxAdd}(R,A,z_1,z_2) \in R \]
\[ \text{ImCxMul}(R,A,M,z_1,z_2) \in R \]
\[ \text{ReCxMul}(R,A,M,z_1,z_2) \in R \]

**proof**

- let \( a = \text{fst}(z_1) \)
- let \( b = \text{snd}(z_1) \)
- let \( c = \text{fst}(z_2) \)
- let \( d = \text{snd}(z_2) \)

from \( A1 \) have \( a \in R \quad b \in R \quad c \in R \quad d \in R \)
by auto
then have
\[ a + c \in R \]
\[ b + d \in R \]
\[ a \cdot d + b \cdot c \in R \]
\[ a \cdot c + (-b \cdot d) \in R \]
using valid_cntxts ring0.Ring_ZF_1_L4 by auto
with \( A1 \) show
\[ \text{ReCxAdd}(R,A,z_1,z_2) \in R \]
ImCxAdd(R,A,z₁,z₂) ∈ R
ImCxMul(R,A,M,z₁,z₂) ∈ R
ReCxMul(R,A,M,z₁,z₂) ∈ R
using cplx_mul_add_defs by auto
qed

Complex reals are complex. Recall the definition of R in the complex0 locale.

lemma (in complex0) axresscn: shows R ⊆ C
  using valid_cntxts group0.group0_2_L2 by auto

Complex 1 is not complex 0.
lemma (in complex0) ax1ne0: shows 1 ≠ 0
proof -
  have IsAfield(R,A,M) using valid_cntxts field0.Field_ZF_1_L1
    by simp
  then show 1 ≠ 0 using IsAfield_def by auto
qed

Complex addition is a complex valued binary operation on complex numbers.

lemma (in complex0) axaddopr: shows CplxAdd(R,A): C × C → C
proof -
  have ∀ p ∈ C×C.
    ⟨ReCxAdd(R,A,fst(p),snd(p)),ImCxAdd(R,A,fst(p),snd(p))⟩ ∈ C
    using cplx_mul_add_types by simp
  then have
    {{⟨p,⟨ReCxAdd(R,A,fst(p),snd(p)),ImCxAdd(R,A,fst(p),snd(p))⟩⟩: p ∈ C×C}: C×C → C
    by (rule ZF_fun_from_total)
  then show CplxAdd(R,A): C × C → C using CplxAdd_def by simp
qed

Complex multiplication is a complex valued binary operation on complex numbers.

lemma (in complex0) axmulopr: shows CplxMul(R,A,M): C × C → C
proof -
  have ∀ p ∈ C×C.
    ⟨ReCxMul(R,A,M,fst(p),snd(p)),ImCxMul(R,A,M,fst(p),snd(p))⟩ ∈ C
    using cplx_mul_add_types by simp
  then have
    {{⟨p,⟨ReCxMul(R,A,M,fst(p),snd(p)),ImCxMul(R,A,M,fst(p),snd(p))⟩⟩: p ∈ C×C}: C×C → C
    by (rule ZF_fun_from_total)
  then show CplxMul(R,A,M): C × C → C using CplxMul_def by simp
qed

What are the values of complex addition and multiplication in terms of their real and imaginary parts?

lemma (in complex0) cplx_mul_add_vals:
  assumes A1: a∈R b∈R c∈R d∈R
shows
\[(a,b) + (c,d) = (a + c, b + d)\]
\[(a,b) \cdot (c,d) = (a \cdot c + (-b \cdot d), a \cdot d + b \cdot c)\]

proof -
let \(S = \text{CplxAdd}(R,A)\)
let \(P = \text{CplxMul}(R,A,M)\)
let \(p = (\langle a,b \rangle, \langle c,d \rangle)\)
from A1 have \(S : C \times C \to C\) and \(p \in C \times C\)
  using axaddopr by auto
moreover have 
  \(S = \{(p, \langle \text{ReCxAdd}(R,A,\text{fst}(p),\text{snd}(p)),\text{ImCxAdd}(R,A,\text{fst}(p),\text{snd}(p))\rangle)\}\).
ultimately have \(S(p) = \langle \text{ReCxAdd}(R,A,\text{fst}(p),\text{snd}(p)),\text{ImCxAdd}(R,A,\text{fst}(p),\text{snd}(p))\rangle\)
  by (rule ZF_fun_from_tot_val)
then show \((a,b) + (c,d) = (a + c, b + d)\)
  using cplx_mul_add_defs by simp
from A1 have \(P : C \times C \to C\) and \(p \in C \times C\)
  using axmulopr by auto
moreover have 
  \(P = \{(p, \langle \text{ReCxMul}(R,A,M,\text{fst}(p),\text{snd}(p)),\text{ImCxMul}(R,A,M,\text{fst}(p),\text{snd}(p))\rangle)\}\).
ultimately have \(P(p) = \langle \text{ReCxMul}(R,A,M,\text{fst}(p),\text{snd}(p)),\text{ImCxMul}(R,A,M,\text{fst}(p),\text{snd}(p))\rangle\)
  by (rule ZF_fun_from_tot_val)
then show \((a,b) \cdot (c,d) = (a \cdot c + (-b \cdot d), a \cdot d + b \cdot c)\)
  using cplx_mul_add_defs by simp
qed

Complex multiplication is commutative.

lemma (in complex0) axmulcom: assumes A1: \(a \in C\) \(b \in C\)
shows \(a \cdot b = b \cdot a\)
using assms cplx_mul_add_vals valid_cntxts ring0.Ring_ZF_1_L4
  field0.field_mult_comm by auto

A sum of complex numbers is complex.

lemma (in complex0) axaddcl: assumes a \(\in C\) \(b \in C\)
shows \(a + b \in C\)
using assms axaddopr apply_funtype by simp

A product of complex numbers is complex.

lemma (in complex0) axmulcl: assumes a \(\in C\) \(b \in C\)
shows \(a \cdot b \in C\)
using assms axmulopr apply_funtype by simp

Multiplication is distributive with respect to addition.
lemma (in complex0) axdistr:
  assumes A1: a ∈ C  b ∈ C  c ∈ C
  shows a·(b + c) = a·b + a·c
proof -
  let a_r = fst(a)
  let a_i = snd(a)
  let b_r = fst(b)
  let b_i = snd(b)
  let c_r = fst(c)
  let c_i = snd(c)
  from A1 have T:
    a_r ∈ R  a_i ∈ R  b_r ∈ R  b_i ∈ R  c_r ∈ R  c_i ∈ R
    b_r + c_r ∈ R  b_i + c_i ∈ R
    a_r·b_r + (-a_i·b_i) ∈ R
    a_r·b_i + a_i·b_r ∈ R
    a_r·c_r + a_i·c_i ∈ R
    using valid_cntxts ring0.Ring_ZF_1_L4 by auto
  with A1 have a·(b + c) =
    ⟨a_r·(b_r + c_r) + (-a_i·(b_i + c_i)), a_r·(b_r + c_r) + a_i·(b_r + c_r)⟩
    using cplx_mul_add_vals by auto
  moreover from T have
    a_r·(b_r + c_r) + (-a_i·(b_i + c_i)) =
    a_r·b_r + (-a_i·b_i) + (a_r·c_r + (-a_i·c_i))
    and
    a_r·b_i + a_i·b_r + (a_r·c_i + a_i·c_r)
    using valid_cntxts ring0.Ring_ZF_2_L6 by auto
  moreover from A1 T have
    ⟨a_r·b_r + (-a_i·b_i) + (a_r·c_r + (-a_i·c_i)),
    a_r·b_i + a_i·b_r + (a_r·c_i + a_i·c_r)⟩ =
    a·b + a·c
    using cplx_mul_add_vals by auto
  ultimately show a·(b + c) = a·b + a·c
    by simp
qed

Complex addition is commutative.

lemma (in complex0) axaddcom: assumes a ∈ C  b ∈ C
  shows a+b = b+a
  using assms cplx_mul_add_vals valid_cntxts ring0.Ring_ZF_1_L4
  by auto

Complex addition is associative.

lemma (in complex0) axaddass: assumes A1: a ∈ C  b ∈ C  c ∈ C
  shows a + b + c = a + (b + c)
proof -
  let a_r = fst(a)
  let a_i = snd(a)

822
let br = fst(b)
let bi = snd(b)
let cr = fst(c)
let ci = snd(c)

from A1 have T:
  ar ∈ R ar ∈ R b_r ∈ R bi ∈ R cr ∈ R ci ∈ R
b_r + bi ∈ R br + c_r ∈ R
using valid_ctxts ring0.Ring_ZF_1_L4 by auto
with A1 have a + b + c = ⟨ar + br + c_r, ai + bi + ci⟩
  using cplx_mul_add_vals by auto
also from A1 T have ...
  using valid_ctxts ring0.Ring_ZF_1_L11 cplx_mul_add_vals
  by auto
finally show a + b + c = a + (b + c)
  by simp
qed

Complex multiplication is associative.

lemma (in complex0) axmulass: assumes A1: a ∈ C b ∈ C c ∈ C
  shows a · b · c = a · (b · c)
proof -
  let ar = fst(a)
  let ai = snd(a)
  let b_r = fst(b)
  let bi = snd(b)
  let cr = fst(c)
  let ci = snd(c)
from A1 have T:
  ar ∈ R ar ∈ R b_r ∈ R bi ∈ R cr ∈ R ci ∈ R
  ar + bi ∈ R ar + bi ∈ R
cr + ci ∈ R cr + ci ∈ R
using valid_ctxts ring0.Ring_ZF_1_L4 by auto
with A1 have a · b · c =
  ⟨(ar · br + (-ar · bi)) · c_r + -(ar · bi + ai · br) · c_i⟩,
  ⟨(ar · br + (-ar · bi)) · c_i + (ar · bi + ai · br) · c_r⟩
  using cplx_mul_add_vals by auto
moreover from A1 T have
  (ar · (b_r · c_r + (-b_r · ci))) + (-a_r · (b_r · c_i + b_r · c_r)) =
  a · (b · c)
  using cplx_mul_add_vals by auto
moreover from T have
  a_r · (b_r · c_r + (-b_r · ci)) + (-a_r · (b_r · c_i + b_r · c_r)) =
  (ar · br + (-ar · bi)) · c_r + -(ar · bi + ai · br) · c_i
and
  a_r · (b_r · c_i + b_r · c_r) + ai · (br · c_r + (-bi · c_i)) =
(a_r · b_r + (-a_i · b_i)) · c_i + (a_r · b_i + a_i · b_r) · c_r

ultimately show a · b · c = a · (b · c)

by auto

qed

Complex 1 is real. This really means that the pair \(1, 0\) is on the real axis.

lemma (in complex0) axire: shows 1 ∈ R

using valid_cntxts ring0.Ring_ZF_1_L2 by simp

The imaginary unit is a "square root" of \(-1\) (that is, \(i^2 + 1 = 0\)).

lemma (in complex0) axi2m1: shows i · i + 1 = 0

using valid_cntxts ring0.Ring_ZF_1_L2 ring0.Ring_ZF_1_L3
cplx_mul_add_vals ring0.Ring_ZF_1_L6 group0.group0_2_L6

by simp

0 is the neutral element of complex addition.

lemma (in complex0) ax0id: assumes a ∈ C

shows a + 0 = a

using assms cplx_mul_add_vals valid_cntxts

ring0.Ring_ZF_1_L2 ring0.Ring_ZF_1_L3

by auto

The imaginary unit is a complex number.

lemma (in complex0) axicn: shows i ∈ C

using valid_cntxts ring0.Ring_ZF_1_L2 by auto

All complex numbers have additive inverses.

lemma (in complex0) axnegex: assumes A1: a ∈ C

shows ∃x ∈ C. a + x = 0

proof -

let a_r = fst(a)

let a_i = snd(a)

let x = (−a_r, −a_i)

from A1 have T:

\(a_r \in \mathbb{R}\) \(a_i \in \mathbb{R}\) \((-a_r) \in \mathbb{R}\) \((-a_i) \in \mathbb{R}\)

using valid_cntxts ring0.Ring_ZF_1_L3 by auto

then have x ∈ C using valid_cntxts ring0.Ring_ZF_1_L3

by auto

moreover from A1 T have a + x = 0

using cplx_mul_add_vals valid_cntxts ring0.Ring_ZF_1_L3

by auto

ultimately show ∃x ∈ C. a + x = 0

by auto

qed

A non-zero complex number has a multiplicative inverse.

lemma (in complex0) axrecex: assumes A1: a ∈ C and A2: a ≠ 0

824
shows \( \exists x \in C. \ a \cdot x = 1 \)

**proof** -

  - let \( a_r = \text{fst}(a) \)
  - let \( a_i = \text{snd}(a) \)
  - let \( m = a_r \cdot a_r + a_i \cdot a_i \)
  - from \( A1 \) have \( T1: a_r \in R \ a_i \in R \) by auto
  - moreover from \( A1 \ A2 \) have \( a_r \neq 0_R \lor a_i \neq 0_R \)
    by auto
  - ultimately have \( \exists c \in R. \ m \cdot c = 1_R \)
    by auto
  - then obtain \( c \) where \( I: c \in R \) and \( II: m \cdot c = 1_R \)
    by auto
  - let \( x = (a_r \cdot c, -a_i \cdot c) \)
  - from \( T1 \ I \) have \( T2: a_r \cdot c \in R \ (-a_i \cdot c) \in R \)
    by auto
  - moreover from \( A1 \ T1 \ T2 \ I \ II \) have \( a \cdot x = 1 \)
    by auto
  - ultimately show \( \exists x \in C. \ a \cdot x = 1 \) by auto

**qed**

Complex 1 is a right neutral element for multiplication.

**lemma (in complex0) axid:** \( \text{assumes} \ A1: a \in C \)

**shows** \( a \cdot 1 = a \)

**using** \( \text{assms} \ \text{valid_cntxts} \ \text{ring0.RingZF_1_L2} \ \text{cplx_mul_add_vals} \ \text{ring0.RingZF_1_L6} \) by auto

A formula for sum of (complex) real numbers.

**lemma (in complex0) sum_of_reals:** \( \text{assumes} \ a \in R \ b \in R \)

**shows** \( a + b = (\text{fst}(a) + \text{fst}(b),0_R) \)

**using** \( \text{assms} \ \text{valid_cntxts} \ \text{ring0.RingZF_1_L2} \ \text{cplx_mul_add_vals} \ \text{ring0.RingZF_1_L3} \) by auto

The sum of real numbers is real.

**lemma (in complex0) axaddrcl:** \( \text{assumes} \ A1: a \in R \ b \in R \)

**shows** \( a + b \in R \)

**using** \( \text{assms} \ \text{sum_of_reals} \ \text{valid_cntxts} \ \text{ring0.RingZF_1_L4} \) by auto

The formula for the product of (complex) real numbers.

**lemma (in complex0) prod_of_reals:** \( \text{assumes} \ A1: a \in R \ b \in R \)

**shows** \( a \cdot b = (\text{fst}(a) \cdot \text{fst}(b),0_R) \)

**proof** -

  - let \( a_r = \text{fst}(a) \)

825
let \( b_r = \text{fst}(b) \)

from A1 have T:
\[
\begin{align*}
a_r & \in R \quad b_r \in R \\
0_R & \in R \\
\langle a_r \cdot b_r, 0_R \rangle & \in R
\end{align*}
\]
using valid_cntxts ring0.Ring_ZF_1_L2 ring0.Ring_ZF_1_L4 by auto

with A1 show \( a \cdot b = \langle a_r \cdot b_r, 0_R \rangle \)
using cplx_mul_add_vals valid_cntxts ring0.Ring_ZF_1_L2 ring0.Ring_ZF_1_L6 ring0.Ring_ZF_1_L3 by auto

qed

The product of (complex) real numbers is real.

lemma (in complex0) axmulrcl: assumes \( a \in R \quad b \in R \)
shows \( a \cdot b \in R \)
using assms prod_of_reals valid_cntxts ring0.Ring_ZF_1_L4 by auto

The existence of a real negative of a real number.

lemma (in complex0) axrnegex: assumes A1: \( a \in R \)
shows \( \exists x \in R. \quad a + x = 0 \)
proof -
\[
\begin{align*}
\text{let } a_r &= \text{fst}(a) \\
\text{let } x &= \langle -a_r, 0_R \rangle \\
\text{from A1 have T:} \\
& \quad \langle y, 0_R \rangle \in R
\end{align*}
\]
using valid_cntxts field0.Field_ZF_1_L5 by auto

moreover from A1 T have \( a + x = 0 \)
using cplx_mul_add_vals valid_cntxts ring0.Ring_ZF_1_L3 by auto
ultimately show \( \exists x \in R. \quad a + x = 0 \) by auto

qed

Each nonzero real number has a real inverse

lemma (in complex0) axrrecex: assumes A1: \( a \in R \quad a \neq 0 \)
shows \( \exists x \in R. \quad a \cdot x = 1 \)
proof -
\[
\begin{align*}
\text{let } R_0 &= R - \{0_R\} \\
\text{let } a_r &= \text{fst}(a) \\
\text{let } y &= \text{GroupInv}(R_0, \text{restrict}(M, R_0 \times R_0))(a_r) \\
\text{from A1 have T:} \\
& \quad \langle y, 0_R \rangle \in R
\end{align*}
\]
using valid_cntxts field0.Field_ZF_1_L5 by auto

moreover from A1 T have \( a \cdot \langle y, 0_R \rangle = 1 \)
using prod_of_reals valid_cntxts field0.Field_ZF_1_L5 field0.Field_ZF_1_L6 by auto
ultimately show \( \exists x \in R. \quad a \cdot x = 1 \) by auto

qed

826
Our \( R \) symbol is the real axis on the complex plane.

**Lemma** (in complex0) \texttt{real_means_real_axis}: shows \( R = \text{ComplexReals}(R,A) \)

using \texttt{ComplexReals_def} by auto

The \texttt{CplxROrder} thing is a relation on the complex reals.

**Lemma** (in complex0) \texttt{cplx_ord_on_cplx_reals}:

shows \( \text{CplxROrder}(R,A,r) \subseteq R \times R \)

using \texttt{ComplexReals_def} slice_proj_bij \texttt{real_means_real_axis} \texttt{CplxROrder_def} \texttt{InducedRelation_def} by auto

The strict version of the complex relation is a relation on complex reals.

**Lemma** (in complex0) \texttt{cplx_strict_ord_on_cplx_reals}:

shows \( \text{StrictVersion(CplxROrder(R,A,r))} \subseteq R \times R \)

using \texttt{cplx_ord_on_cplx_reals} \texttt{strict_ver_rel} by simp

The \texttt{CplxROrder} thing is a relation on the complex reals. Here this is formulated as a statement that in \texttt{complex0} context \( a < b \) implies that \( a, b \) are complex reals.

**Lemma** (in complex0) \texttt{strict_cplx_ord_type}: assumes \( a <_R b \) shows \( a \in R \land b \in R \)

using \texttt{assms} \texttt{CplxROrder_def} \texttt{def_of_strict_ver} \texttt{InducedRelation_def} \texttt{slice_proj_bij} \texttt{ComplexReals_def} \texttt{real_means_real_axis} by auto

A more readable version of the definition of the strict order relation on the real axis. Recall that in the \texttt{complex0} context \( r \) denotes the (non-strict) order relation on the underlying model of real numbers.

**Lemma** (in complex0) \texttt{def_of_real_axis_order}:

\[ \langle x,0_R \rangle <_R \langle y,0_R \rangle \iff \langle x,y \rangle \in r \land x \neq y \]

**Proof**

let \( f = \text{SliceProjection(ComplexReals(R,A))} \)

assume \( A1: \langle x,0_R \rangle <_R \langle y,0_R \rangle \)

then have \( \langle f(x,0_R), f(y,0_R) \rangle \in r \land x \neq y \)

using \texttt{CplxROrder_def} \texttt{def_of_strict_ver} \texttt{def_of_ind_relA} \texttt{InducedRelation_def} by simp

moreover from \( A1 \) have \( \langle x,0_R \rangle \in R \land \langle y,0_R \rangle \in R \)

using \texttt{valid_cntxts} \texttt{ring1.OrdRing_ZF_1_L3} by auto

ultimately show \( \langle x,y \rangle \in r \land x \neq y \)

using \texttt{slice_proj_bij} \texttt{ComplexReals_def} by simp

next assume \( A1: \langle x,y \rangle \in r \land x \neq y \)

let \( f = \text{SliceProjection(ComplexReals(R,A))} \)

have \( f : R \rightarrow R \)

using \texttt{ComplexReals_def} \texttt{slice_proj_bij} \texttt{real_means_real_axis} by simp

moreover from \( A1 \) have \( T: \langle x,0_R \rangle \in R \land \langle y,0_R \rangle \in R \)

using \texttt{valid_cntxts} \texttt{ring1.OrdRing_ZF_1_L3} by auto

moreover from \( A1 \) \( T \) have \( \langle f(x,0_R), f(y,0_R) \rangle \in r \)
using slice proj bij ComplexReals_def by simp
ultimately have \((x,0_R), (y,0_R)\) ∈ InducedRelation(f,r)
using def_of_ind_relB by simp
with A1 show \(x,0_R \prec_R (y,0_R)\)
using CplxROrder_def def_of_strict_ver
by simp
qed

The (non strict) order on complex reals is antisymmetric, transitive and total.

lemma (in complex0) cplx_ord_antsym_trans_tot: shows
antisym(CplxROrder(R,A,r))
trans(CplxROrder(R,A,r))
CplxROrder(R,A,r) {is total on} R
proof -
  let f = SliceProjection(ComplexReals(R,A))
  have f ∈ ord_iso(R,CplxROrder(R,A,r),R,r)
    using ComplexReals_def slice_proj_bij real_means_real_axis
    bij_is_ord_iso CplxROrder_def by simp
  moreover have CplxROrder(R,A,r) ⊆ R×R
    using cplx_ord_on_cplx_reals by simp
  moreover have I:
    antisym(r) r {is total on} R trans(r)
    using valid_cntxts ring1.OrdRing_ZF_1_L1 IsAnOrdRing_def
    IsLinOrder_def by auto
  ultimately show
    antisym(CplxROrder(R,A,r))
    trans(CplxROrder(R,A,r))
    CplxROrder(R,A,r) {is total on} R
    using ord_iso_pres_antsym ord_iso_pres_tot ord_iso_pres_trans
    by auto
qed

The trichotomy law for the strict order on the complex reals.

lemma (in complex0) cplx_strict_ord_trich:
  assumes A1: a ∈ R b ∈ R
  shows Exactly_1_of_3_holds(a R b, a=b, b R a)
  using assms cplx_ord_antsym_trans_tot strict_ans_tot_trich
  by simp

The strict order on the complex reals is kind of antisymmetric.

lemma (in complex0) pre_axlttri: assumes A1: a ∈ R b ∈ R
  shows a R b ↔ ¬(a=b ∨ b R a)
proof -
  from A1 have Exactly_1_of_3_holds(a R b, a=b, b R a)
    by (rule cplx_strict_ord_trich)
  then show a R b ↔ ¬(a=b ∨ b R a)
    by (rule Fol1_L8A)
qed
The strict order on complex reals is transitive.

**lemma (in complex0) cplx_strict_ord_trans:**
shows trans(StrictVersion(CplxROrder(R,A,r)))
using cplx_ord_antsym_trans_tot strict_of_transB by simp

The strict order on complex reals is transitive - the explicit version of cplx_strict_ord_trans.

**lemma (in complex0) pre_axlttrn:**
assumes A1: a <\_\_\_\_ R b b <\_\_\_\_ R c
shows a <\_\_\_\_ R c
proof -
  let s = StrictVersion(CplxROrder(R,A,r))
  from A1 have trans(s) ⟨a,b⟩ ∈ s ∧ ⟨b,c⟩ ∈ s
    using cplx_strict_ord_trans by auto
  then have ⟨a,c⟩ ∈ s by (rule Fol1_L3)
  then show a <\_\_\_\_ R c by simp
qed

The strict order on complex reals is preserved by translations.

**lemma (in complex0) pre_axltadd:**
assumes A1: a <\_\_\_\_ R b and A2: c ∈ \_\_\_\_\_\_' shows c+a <\_\_\_\_ R c+b
proof -
  from A1 have T: a∈R b∈R using strict_cplx_ord_type
    by auto
  with A1 A2 show c+a <\_\_\_\_ R c+b
    using def_of_real_axis_order valid_cntxts
      group3.group_strict_ord_transl_inv sum_of_reals
    by auto
qed

The set of positive complex reals is closed with respect to multiplication.

**lemma (in complex0) pre_axmulgt0:** assumes A1: 0 <\_\_\_\_ R a 0 <\_\_\_\_ R b
shows 0 <\_\_\_\_ R a·b
proof -
  from A1 have T: a∈R b∈R using strict_cplx_ord_type
    by auto
  with A1 show 0 <\_\_\_\_ R a·b
    using def_of_real_axis_order valid_cntxts field1.pos_mul_closed
      def_of_real_axis_order prod_of_reals
    by auto
qed

The order on complex reals is linear and complete.

**lemma (in complex0) cmplx_reals_ord_lin_compl:** shows CplxROrder(R,A,r) {is complete}
IsLinOrder(R,CplxROrder(R,A,r))
proof -
  have \( \text{SliceProjection}(R) \in \text{bij}(R, R) \)
    using slice_proj_bij ComplexReals_def real_means_real_axis
    by simp
  moreover have \( r \subseteq R \times R \) using valid_cntxts ring1.OrdRing_ZF_1_L1
    IsAnOrdRing_def by simp
  moreover from \( R \) are reals have
    \( r \subseteq R \times R \)
    using valid_cntxts ring1.OrdRing_ZF_1_L1
    IsAnOrdRing_def by auto
  ultimately show
    \( CplxROrder(R, A, r) \) {is complete}
    IsLinOrder(\( R, (CplxROrder(R, A, r)) \))
    using CplxROrder_def real_means_real_axis ind_rel_pres_compl
    ind_rel_pres_lin by auto
qed

The property of the strict order on complex reals that corresponds to completeness.

lemma (in complex0) pre_axsup: assumes \( A1: X \subseteq R \quad X \neq 0 \) and
  \( A2: \exists x \in R. \forall y \in X. y <_R x \)
  shows
    \( \exists x \in R. (\forall y \in X. \neg(x <_R y)) \land (\forall y \in R. (y <_R x \rightarrow (\exists z \in X. y <_R z))) \)
proof -
  let \( s = \text{StrictVersion}(CplxROrder(R, A, r)) \)
  have \( CplxROrder(R, A, r) \subseteq R \times R \)
    IsLinOrder(\( R, (CplxROrder(R, A, r)) \))
    CplxROrder(R, A, r) {is complete}
    using cplx_ord_on_cplx_reals cplx_reals_ord_lin_compl
    by auto
  moreover note \( A1 \)
  moreover have \( s = \text{StrictVersion}(CplxROrder(R, A, r)) \)
    by simp
  moreover from \( A2 \) have \( \exists u \in R. \forall y \in X. (y, u) \in s \)
    by simp
  ultimately have
    \( \exists x \in R. (\forall y \in X. (x, y) \notin s ) \land \)
    \( (\forall y \in R. (y, x) \in s \rightarrow (\exists z \in X. (y, z) \in s)) \)
    by (rule strict_of_compl)
  then show \( (\exists x \in R. (\forall y \in X. \neg(x <_R y)) \land \)
    \( (\forall y \in R. (y <_R x \rightarrow (\exists z \in X. y <_R z))) \))
    by simp
qed

end
65 Rings - Zariski Topology

This file deals with the definition of the topology on the set of prime ideals.

It defines the topology, computes the closed sets and the closure and interior operators.

```plaintext
theory Ring_Zariski_ZF imports Ring_ZF_2 Topology_ZF

begin

The set where the topology is defined is in the spectrum of a ring; i.e. the set of all prime ideals.

definition (in ring0) Spec where
Spec ≡ {I∈I. I◁p R}

The basic set that defines the topology is given by the D operator.

definition (in ring0) openBasic (D) where
S⊆R ⇒ D(S) ≡ {I∈Spec. ¬(S⊆I)}

The D operator preserves subsets.

lemma (in ring0) D_operator_preserve_subset:
assumes S ⊆ T T ⊆ R
shows D(S) ⊆ D(T)

proof
from assms have S:S ⊆ R by auto
fix x assume x∈D(S)
then have x:x∈Spec ¬(S⊆x) using openBasic_def S by auto
with assms(1) have x∈Spec ¬(T⊆x) by auto
then show x:D(T) using openBasic_def assms(2) by auto
qed

The D operator values can be obtained by considering only ideals. This is useful as we have operations on ideals that we do not have on subsets.

lemma (in ring0) D_operator_only_ideals:
assumes T ⊆ R
shows D(T) = D(⟨T⟩)

proof
have T:T ⊆ (⟨T⟩), ⟨T⟩, ⊆ R using generated_ideal_contains_set assms
  generated_ideal_is_ideal[OF assms] ideal_dest_subset by auto
with D_operator_preserve_subset show D(T) ⊆ D(⟨T⟩)
  by auto
{
  fix t assume t∈D(⟨T⟩)
  with T(2) have t:t∈Spec ¬(⟨T⟩⊆t) using openBasic_def by auto
  {
    assume as:T ⊆ t
    from t(1) have t∈R unfolding Spec_def primeIdeal_def by auto
    with as have ⟨T⟩ ⊆ t using generated_ideal_small by auto
```

831
with \( t(2) \) have False by auto 

\}
then have \( \neg (T \subseteq t) \) by auto 
with \( t(1) \) have \( t \in \mathcal{D}(T) \) using openBasic_def assms by auto 

\}
then show \( \mathcal{D}(\langle T \rangle) \subseteq \mathcal{D}(T) \) by auto 

qed 

The intersection of to D-sets is the D-set on the product of ideals 

\[
\text{lemma (in ring0) intersection_open_basic:}
\begin{align*}
\text{assumes } & I \triangleleft R \\
\text{shows } & D(I) \cap D(J) = D(I \cdot J)
\end{align*}
\]

\textbf{proof} 

have \( S : I \cdot J \subseteq R \) using product_in_intersection(2) ideal_dest_subset assms by auto 

\{
  \fix K assume K:K\in\mathcal{D}(I)\cap\mathcal{D}(J) 
  then have \( K \triangleleft R \) \( \neg (I \subseteq K) \rightarrow \neg (J \subseteq K) \) 
  using assms ideal_dest_subset openBasic_def 
  unfolding Spec_def by auto 
  then have \( \neg (I \subseteq K) \rightarrow \neg (J \subseteq K) \) \( \forall I \in I, \forall J \in J. I \cdot J \subseteq K \rightarrow I \subseteq K \lor J \subseteq K \) 
  unfolding primeIdeal_def by auto 
  then have \( \neg (I \cdot J \subseteq K) \) using assms 

moreover note K 
ultimately have \( K \in \mathcal{D}(I \cdot J) \) using openBasic_def[of I] ideal_dest_subset[of J] by auto 

then show \( \mathcal{D}(I) \cap \mathcal{D}(J) \subseteq \mathcal{D}(I \cdot J) \) by auto 
\}

Then show \( \mathcal{D}(I) \cap \mathcal{D}(J) \subseteq \mathcal{D}(I \cdot J) \) by auto 

\{
  \fix K assume K:K\in\mathcal{D}(I \cdot J) 
  then have \( K \triangleleft R \) \( \neg (I \subseteq K) \) using openBasic_def[of K] unfolding Spec_def by auto 

by auto 

\{
  \assume I \subseteq K \lor J \subseteq K 
  then have \( I \cap J \subseteq K \) by auto 
  then have \( I \cdot J \subseteq K \) using product_in_intersection assms by auto 
  with \( K(2) \) have False by auto 

\}
then have \( \neg (I \subseteq K) \rightarrow \neg (J \subseteq K) \) by auto 

with Kass have \( K \in \mathcal{D}(I) \cap \mathcal{D}(J) \) using assms ideal_dest_subset 
openBasic_def[of I] openBasic_def[of J] 
unfolding openBasic_def[of K] Spec_def by auto 

\}
then show \( \mathcal{D}(I \cdot J) \subseteq \mathcal{D}(I) \cap \mathcal{D}(J) \) by auto 

qed 

The union of D-sets is the D-set of the sum of the ideals 

832
lemma (in ring0) union_open_basic:
assumes \( J \subseteq I \)
shows \( \bigcup \{D(I). \ I \in J\} = D(\bigoplus I J) \)
proof
  have \( S: (\bigoplus I J) \subseteq R \) using generated_ideal_is_ideal[of \( \bigcup J \)] assms
    unfolding sumArbitraryIdeals_def[of assms]
    using ideal_dest_subset by auto
  
  { fix \( t \) assume \( \bigcup J \subseteq t \) using assms openBasic_def unfolding Spec_def
    by auto
  
    assume \((\bigoplus I J) \subseteq t \)
    then have \( \bigcup J \subseteq t \) using generated_ideal_contains_set[of \( \bigcup J \)]
      assms unfolding sumArbitraryIdeals_def[of assms] by auto
    with \( K(1) \) have \( K \subseteq t \) by auto
    with \( t(2) \) have False by auto
  }
  then have \( \neg((\bigoplus I J) \subseteq t) \) by auto moreover
  note \( K S \) ultimately have \( t \in D(\bigoplus I J) \) using openBasic_def[of \( K \)] openBasic_def[of \( \bigoplus I J \)]
    assms by auto
  }
  then show \( \bigcup \{D(I). \ I \in J\} \subseteq D(\bigoplus I J) \) by auto
  { fix \( t \) assume \( t \in D(\bigoplus I J) \)
    then have \( t \in \text{Spec} \ \neg((\bigoplus I J) \subseteq t) \) unfolding openBasic_def[of \( S \)]
      by auto
  
    assume \( \bigcup J \subseteq t \)
    with \( t(1) \) have \( (\bigcup J) I \subseteq t \) using generated_ideal_small
      unfolding Spec_def primeIdeal_def
      by auto
    with \( t(2) \) have False unfolding sumArbitraryIdeals_def[of assms] by auto
  }
  then obtain \( J \) where \( J: \neg(J \subseteq t) \) \( J \in J \) by auto
  note \( J(1) \) moreover have \( J \subseteq R \) using \( J \in J \) assms by auto
  moreover note \( t(1) \) ultimately have \( t \in D(J) \) using openBasic_def[of \( J \)]
    by auto
  
  then have \( t: \bigcup \{D(I). \ I \in J\} \) using \( J(2) \) by auto
  }
  then show \( D(\bigoplus I J) \subseteq \bigcup \{D(I). \ I \in J\} \) by auto
qed

From the previous results on intersetion and union, we conclude that the D-sets we computed form a topology
corollary (in ring0) zariski_top:
  shows \( \{D(J) \mid J \in I\} \) is a topology
unfolding IsATopology_def
proof (safe)
  fix M assume M \subseteq \{D(J) \mid J \in I\}
  then have M = \( \bigcup \{D(J) \mid J \in I\} \) by auto
  then have \( \bigcup M = D(\bigoplus \{K \in I \mid D(K) \in M\}) \) using union_open_basic[of \( \{K \in I \mid D(K) \in M\} \)] by auto
  moreover have \( (\bigoplus \{K \in I \mid D(K) \in M\}) \triangleleft R \) using
    generated_ideal_is_ideal[of \( \bigcup \{K \in I \mid D(K) \in M\} \)]
    sumArbitraryIdeals_def[of \( \{K \in I \mid D(K) \in M\} \)]
    by force
  then have \( (\bigoplus \{K \in I \mid D(K) \in M\}) \in I \) using ideal_dest_subset by auto
  ultimately show \( \bigcup M : \{D(J) \mid J \in I\} \) by auto
next
  fix x xa assume as: \( x \triangleleft R \) \( xa \triangleleft R \)
  then have D(x) \cap D(xa) = D(x \cdot I xa) using intersection_open_basic
    by auto
  moreover have \( (x \cdot I xa) \triangleleft R \) using product_in_intersection(2)
    as by auto
  then have \( (x \cdot I xa) : I \) using ideal_dest_subset by auto
  ultimately show D(x) \cap D(xa) \in \{D(J) \mid J \in I\} by auto
qed

We include all the results of topology0 into ring0 under the namespace "zariski"

definition (in ring0) ZarInt (int) where
  int(U) \equiv Interior(U, \{D(J) \mid J \in I\})

definition (in ring0) ZarCl (cl) where
  cl(U) \equiv Closure(U, \{D(J) \mid J \in I\})

definition (in ring0) ZarBound (\partial_) where
  \partial U \equiv Boundary(U, \{D(J) \mid J \in I\})

sublocale ring0 < zariski:topology0 \{D(J) \mid J \in I\}
  ZarInt ZarCl ZarBound unfolding topology0_def
  ZarBound_def ZarInt_def ZarCl_def
  using zariski_top by auto

The interior of a proper subset is given by the D-set of the intersection of all the prime ideals not in that subset

lemma (in ring0) interior_zariski:
  assumes U \subseteq \text{Spec} U \neq \text{Spec}
  shows int(U) = D(\bigcap (\text{Spec}-U))
proof
  { fix t assume t: t \in \bigcap (\text{Spec}-U)

834
then have $\forall V \in \text{Spec-U. } t:V$ by auto
moreover from $t$ have $(\text{Spec-U}) \neq 0$ by auto
ultimately obtain $V$ where $V \in \text{Spec-U } \cap V$ by auto
then have $t \in \bigcup \text{Spec by auto}$
then have $t \in R$ unfolding $\text{Spec_def}$ by auto

\}
then have $S: \bigcap (\text{Spec-U}) \subseteq R$ by auto
\}
then have $t \in \text{D(\bigcap (\text{Spec-U}))}$ by auto
\}
then have $t: t: \text{Spec } \neg(\bigcap (\text{Spec-U}) \subseteq t)$ using $\text{openBasic_def[OF S]}$ by auto
\}
assume $t \notin U$
with $t(1)$ have $t \in \text{Spec-U}$ by auto
then have $\bigcap (\text{Spec-U}) \subseteq t$ by auto
then have False using $t(2)$ by auto
\}
then have $t \in U$ by auto
\}
then have $\text{D}(\bigcap (\text{Spec-U})) \subseteq U$ by auto moreover
\{
assume $\text{Spec-U} = 0$
with $\text{assms(1)}$ have $U=\text{Spec}$ by auto
with $\text{assms(2)}$ have False by auto
\}
then have $\text{P:Spec-U} \subseteq I$ $\text{Spec-U} \neq 0$ unfolding $\text{Spec_def}$ by auto
then have $(\bigcap (\text{Spec-U})) \in R$ using $\text{intersection_ideals}$ by auto
then have $(\bigcap (\text{Spec-U})) \in I$ $\{\text{D(J). J} \in I\}$ by auto
ultimately
show $\text{D}(\bigcap (\text{Spec-U})) \subseteq \text{int(U)}$ using $\text{zariski.Top_2_L5}$ by auto
\{
fix $V$ assume $V: V \in \text{D(J). J} \in I \} \ V \subseteq U$
from $V(1)$ obtain $J$ where $J: J: V=\text{D(J)}$ by auto
from $V(2)$ have $\text{SS:Spec-U} \subseteq \text{Spec-V}$ by auto
\{ 
fix $K$ assume $K: K \in \text{Spec-U}$
\{ 
assume $\neg(J \subseteq K)$
with $K$ have $K \in \text{D(J)}$ using $J(1)$ using $\text{openBasic_def}$ by auto
with $\text{SS K J(2)}$ have $\text{False}$ by auto
\}
then have $J \subseteq K$ by auto
\}
then have $J \subseteq \bigcap (\text{Spec-U})$ using $\text{P(2)}$ by auto
then have $\text{D(J)} \subseteq D(\bigcap (\text{Spec-U}))$ using $\text{D_operator_preserve_subset}$ by auto
with $J(2)$ have $V \subseteq D(\bigcap (\text{Spec-U}))$ by auto

835
then show \( \text{int}(U) \subseteq D(\bigcap (\text{Spec}-U)) \) using 
\( \text{zariski.Top}_2\_L1 \) \( \text{zariski.Top}_2\_L2 \) by auto 
qed 

The whole space is the D-set of the ring as an ideal of itself 

**Lemma (in ring0)** openBasic_total: 
shows \( D(R) = \text{Spec} \) 

**Proof** 
show \( D(R) \subseteq \text{Spec} \) using openBasic_def by auto 
\{ 
fix \( t \) assume \( t:\in \text{Spec} \) 
\{ 
assume \( R \subseteq t \)
then have False using \( t \) unfolding \( \text{Spec}_\text{def} \) primeIdeal_def 
using \( \text{ideal\_dest\_subset[of t]} \) by auto
\} 
with \( t \) have \( t \in D(R) \) using openBasic_def by auto 
\} 
then show \( \text{Spec} \subseteq D(R) \) by auto 
qed 

corollary (in ring0) total_spec: 
shows \( \bigcup \{D(J). \ J \in I\} = \text{Spec} \) 

**Proof** 
show \( \bigcup \{D(J). \ J \in I\} \subseteq \text{Spec} \) using openBasic_def by auto 
have \( D(R) \in \{D(J). \ J \in I\} \) using ring_self_ideal by auto 
then have \( D(R) \subseteq \bigcup \{D(J). \ J \in I\} \) by auto 
then show \( \text{Spec} \subseteq \bigcup \{D(J). \ J \in I\} \) using openBasic_total by auto 
qed 

The empty set is the D-set of the zero ideal 

**Lemma (in ring0)** openBasic_empty: 
shows \( D(\{0\}) = \emptyset \) 

**Proof** 
\{ 
fix \( t \) assume \( t: \in D(\{0\}) \) 
then have \( t \circ R - (\{0\} \subseteq t) \) using openBasic_def 
\( \text{Ring}_Z\_F\_1\_L2(1) \) unfolding \( \text{Spec}_\text{def} \) by auto 
then have False using \( \text{ideal\_dest\_zero} \) unfolding \( \text{primeIdeal}_\text{def} \) by auto 
\} 
then show \( D(\{0\}) = \emptyset \) by auto 
qed 

A closed set is a set of primes containing a given ideal 

**Lemma (in ring0)** closeBasic: 
assumes \( U(\text{is\ closed\ in})\ \{D(J). \ J \in I\} \) 
obtains \( J \) where \( J \in I \) and \( U = \{K \in \text{Spec}. \ J \subseteq K\} \)
proof-

assume rule: \( \forall J. J \in I \implies U = \{ K \in \text{Spec}. J \subseteq K \} \implies \text{thesis} \)

from assms have \( U: U \subseteq \text{Spec} \cup \{ D(J). J \in I \} \) unfolding IsClosed_def
using total_spec by auto
then obtain \( J \) where \( J: J \in I \implies U = \text{Spec} \cup \{ D(J) \} \) by auto
moreover from \( U(1) \) have \( \text{Spec} \cup \{ \text{Spec} \cup U \} = U \) by auto
ultimately have \( U = \text{Spec} \cup \{ K \in \text{Spec.} \neg (J \subseteq K) \} \) using openBasic_def
by auto
then have \( U = \{ K \in \text{Spec}. J \subseteq K \} \) by auto
with \( J \) show thesis using rule by auto
qed

We define the closed sets as V-sets

\textbf{definition} (in ring0) closeBasic \((V)\) where
\( S \subseteq R \implies V(S) = \{ K \in \text{Spec}. S \subseteq K \} \)

V-sets and D-sets are complementary

\textbf{lemma} (in ring0) V_is_closed:

assumes \( J \in I \)
shows \( \text{Spec} \cup V(J) = D(J) \) and \( V(J) \) {is closed in} \( \{ D(J). J \in I \} \)
unfolding IsClosed_def
proof(safe)

from assms have \( V(J) \subseteq \text{Spec} \) using closeBasic_def by auto
then show \( \forall x. x \in \text{Spec} \implies x \in \bigcup \text{RepFun}(I, D) \) using total_spec by auto

show \( \text{Spec} \cup V(J) = D(J) \) using assms
closeBasic_def openBasic_def by auto

with assms show \( \bigcup \text{RepFun}(I, D) - V(J) \in \text{RepFun}(I, D) \)
using total_spec by auto
qed

As with D-sets, by De Morgan’s Laws we get the same result for unions and intersections on V-sets

\textbf{lemma} (in ring0) V_union:

assumes \( J \in I \) \( K \in I \)
shows \( V(J) \cup V(K) = V(J \cdot I K) \)
proof-

{ fix \( t \) assume \( t \in V(J) \)
then have \( t \in \text{Spec} J \subseteq t \) using closeBasic_def
assms(1) by auto
moreover have \( J \cdot I K \subseteq J \) using product_in_intersection(1)[of J K]
assms by auto
ultimately have \( t \in \text{Spec} J \cdot I K \subseteq t \) by auto
then have \( t: V(J \cdot K) \) using closeBasic_def
product_in_intersection(2)[of J K] assms ideal_dest_subset
by auto
}
moreover
{ 
  fix t assume t∈V(K)
  then have t∈Spec K⊆t using closeBasic_def
    assms(2) by auto
  moreover have J·I K ⊆ K using product_in_intersection(1)[of J K]
    assms by auto
  ultimately have t∈Spec J·I K ⊆ t by auto
  then have t: V(J·I K) using closeBasic_def
    product_in_intersection(2)[of J K] assms ideal_dest_subset
    by auto
}
moreover
{ 
  fix t assume t∈V(J·I K)
  then have t:t∈Spec J·I K ⊆ t using closeBasic_def
    assms product_in_intersection(2)[of J K] ideal_dest_subset
    by auto
  from this(1) have ∀Ia∈I . ∀J∈I . Ia · J ⊆ t → Ia ⊆ t ∨ J ⊆ t
    unfolding Spec_def primeIdeal_def by blast
  with assms have J·I K ⊆ J ⊆ t ∨ K ⊆ t by auto
  with t have t:Spec J ⊆ t ∨ K ⊆ t by auto
  then have t∈V(J)∪V(K) using closeBasic_def
    assms by auto
}
ultimately have thesis by auto
qed

lemma (in ring0) V_intersect:
  assumes J ⊆ I J ≠0
  shows ∩{V(I). I∈J} = V(⊕I J)
proof-
  have Spec - (∩{V(I). I∈J}) = ∪{D(I). I∈J}
    proof
      { 
        fix t assume t:Spec - (∩{V(I). I∈J})
        then have t:t:Spec t∉∩{V(I). I∈J} by auto
        from t(2) obtain K where (t∉V(K) ∧ K∈J) ∨ J=0 by auto
        with assms(2) have t∉V(K) K∈J by auto
        with t(1) have t:Spec-V(K) K:J by auto moreover
        from `K:J` have Spec-V(K) = D(K) using assms(1) V_is_closed(1)
        by auto
        ultimately have t:D(K) K:J by auto
        then have t∈∪{D(I). I∈J} by auto
      }
    then show Spec - (∩{V(I). I∈J}) ⊆ ∪{D(I). I∈J} by auto
      { 
        fix t assume t∈∪{D(I). I∈J}
        then obtain K where K:K:J t:D(K) using assms(2)
          by auto
      }

838
from `K:J` have Spec-V(K) = D(K) using assms(1) V_is_closed(1)
by auto

with K(2) have t:Spec-V(K) by auto
with K(1) have t∈Spec∩{V(I). I∈J} by auto

then show ∪{D(I). I∈J} ⊆ Spec - (⋂{V(I). I∈J}) by auto
qed

then have Spec - (⋂{V(I). I∈J}) = D(J) using union_open_basic
assms by auto

then have Spec-(Spec - (⋂{V(I). I∈J})) = Spec-D(J) by auto
moreover
have JI:(⊕I,J) ∈ I using generated_ideal_is_ideal[of ∪J] assms
unfolding sumArbitraryIdeals_def[of assms(1)]
using ideal_dest_subset by auto

then have Spec-V(⊕I,J) = D(⊕I,J) using V_is_closed(1)[of ⊕I,J]
by auto

then have Spec-(Spec-V(⊕I,J)) = Spec-D(⊕I,J) by auto
ultimately have R:Spec-(Spec - (⋂{V(I). I∈J})) = Spec-(Spec-V(⊕I,J))
by auto

{ fix t assume t:V(⊕I,J)
with JI have t:Spec using closeBasic_def by auto
with t have t∈Spec-(Spec-V(⊕I,J)) by auto
with R have t∈Spec-(Spec-V(⊕I,J)) by auto
then have t∈V(⊕I,J) by auto
} moreover

{ fix t assume t:V(⊕I,J)
with JI have t:Spec using closeBasic_def by auto
with t have t∈Spec-(Spec-V(⊕I,J)) by auto
then have t∈Spec-(Spec-V(⊕I,J)) using R by auto
then have t∈⋂{V(I). I∈J} by auto
}
ultimately show thesis by force
qed

The closure of a set is the V-set of the intersection of all its points.

lemma (in ring0) closure_zariski:
  assumes U ⊆ Spec U≠0
  shows cl(U) = V(⋂U)
proof
  have U ⊆ I using assms(1) unfolding Spec_def by auto
  then have ideal:(⋂U)∈R using intersection_ideals[of U] assms(2) by auto
  { fix t assume t:V(⋂U)

839
\{ 
  fix q assume q : q ∈ \bigcap U 
  with q obtain M where M : U q : M using assms(2) by blast 
  with assms have q ∈ \bigcup \text{Spec} by auto 
  then have q : R unfolding Spec_def by auto 
\} 
then have \text{sub :} \bigcap U ⊆ R by auto 
with t have tt : t ∈ \text{Spec} \bigcap U ⊆ t using closeBasic_def by auto 
\{ 
  fix V V assume V V : V ∈ \{ D(J) . J ∈ I \} t ∈ V V 
  then obtain J where J : V V = D(J) J ∈ I by auto 
  from V V(2) J have jt : ¬ (J ⊆ t) using openBasic_def by auto 
  \{ 
    assume U ∩ D(J) = 0 
    then have ∀ x ∈ U. x \notin D(J) by auto 
    with J(2) have ∀ x ∈ U. J ⊆ x using openBasic_def[of J] 
    assms(1) by auto 
    then have J ⊆ \bigcap U ∨ U = 0 by auto 
    with tt(2) jt have False using assms(2) by auto 
  \} 
  then have U ∩ V V \neq 0 using J(1) by auto 
\} 
then have R : ∀ V V ∈ \{ D(J) . J ∈ I \}. t ∈ V V \rightarrow V V ∩ U \neq 0 by auto 
from tt(1) assms(1) have RR : t ∈ \bigcup \{ D(J) . J ∈ I \} U ⊆ \bigcup \{ D(J) . J ∈ I \} 
  using total_spec by auto 
have t ∈ cl(U) using zariski.inter_neigh_cl[OF RR(2,1) R]. 
\} 
then show V(\bigcap U) ⊆ cl(U) 
  apply (rule subsetI) by auto 
\{ 
  fix p assume p : p ∈ U 
  then have \bigcap U ⊆ p by auto 
  moreover 
  from p assms(1) have p ∈ Spec \bigcap U \subseteq \text{Spec} by auto 
  then have p ∈ Spec \bigcap U ⊆ R unfolding Spec_def by auto 
  ultimately have p ∈ V(\bigcap U) using closeBasic_def[of \bigcap U] 
  by auto 
\} 
then have A : U ⊆ V(\bigcap U) by auto 
have B : V(\bigcap U){is closed in}{D(J). J ∈ I} 
  using V_is_closed(2) ideal ideal_dest_subset by auto 
show cl(U) ⊆ V(\bigcap U) 
  apply (rule zariski.Top_3_L13[of V(\bigcap U) U]) 
  using A B by auto 
qed 
end
66 Rings - Zariski Topology - Properties

theory Ring_Zariski_ZF_2 imports Ring_Zariski_ZF Topology_ZF_1

begin

theorem (in ring0) zariski_t0:
  shows \{D(I). I \in I\} \text{is T}_0\text{def}
proof
  { 
    fix x y assume ass:x:Spec y:Spec x \neq y
    from ass(3) have \neg(x \subseteq y) \lor \neg(y \subseteq x) by auto
    then have x:D(x) \lor y:D(y) using ass(1,2)
      unfolding Spec_def using ass(1,2) openBasic_def by auto
    then have \neg(x \subseteq y) \lor \neg(y \subseteq x) by auto
    then have \exists U \in \{D(I). I \in I\}. (x \in U \land y \notin U) \lor (y \in U \land x \notin U)
      using ass(1,2) by auto
  }
  then show \forall x y. x \in \bigcup \text{RepFun}(I, D) \land y \in \bigcup \text{RepFun}(I, D) \land x \neq y \rightarrow 
    (\exists U \in \text{RepFun}(I, D). x \in U \land y \notin U \lor y \in U \land x \notin U)
    using total_spec by auto
qed

Noetherian rings have compact Zariski topology

theorem (in ring0) zariski_compact:
  assumes \forall I \in I. (I\{is finitely generated})
  shows \text{Spec} \{is compact in\} \{D(I). I \in I\}
proof (safe)
  show \forall x. x \in \text{Spec} \rightarrow x \in \bigcup \text{RepFun}(I, D) using total_spec by auto
  fix M assume M:M \subseteq \text{RepFun}(I, D) \land \text{Spec} \subseteq \bigcup M
  let J = \{J \in I. D(J) \in M\}
  have m:M = \text{RepFun}(J, D) using M(1) by auto
  then have mm:mm : \bigcup M = D(\oplus J) using union_open_basic[of J] by auto
  obtain T where T:T \in \text{FinPow}(J) \{\oplus J\} = \oplus T using
    sum_ideals_noetherian[of assms(1), of J] by blast
  from T(2) have D(\oplus J) = D(\oplus T) by auto
  moreover have T \subseteq I using T(1) unfolding FinPow_def by auto
  ultimately have D(\oplus J) = \bigcup \text{RepFun}(T, D) using union_open_basic[of T]
    by auto
  with mm have \bigcup M = \bigcup \text{RepFun}(T, D) by auto
  then have Spec \subseteq \bigcup \text{RepFun}(T, D) using M(2) by auto moreover
  from T(1) have \text{RepFun}(T, D) \subseteq \text{RepFun}(J, D) unfolding FinPow_def by auto
  with m have \text{RepFun}(T, D) \subseteq M by auto moreover
  from T(1) have Finite(\text{RepFun}(T, D)) unfolding FinPow_def
    using Finite_RepFun by auto

841
ultimately show $\exists N \in \text{FinPow}(M)$. Spec $\subseteq \bigcup N$ unfolding \text{FinPow\_def}

by auto

qed

end

67 Topology 1b

theory Topology_ZF_1b imports Topology_ZF_1

begin

One of the facts demonstrated in every class on General Topology is that in a $T_2$ (Hausdorff) topological space compact sets are closed. Formalizing the proof of this fact gave me an interesting insight into the role of the Axiom of Choice (AC) in many informal proofs.

A typical informal proof of this fact goes like this: we want to show that the complement of $K$ is open. To do this, choose an arbitrary point $y \in K^c$. Since $X$ is $T_2$, for every point $x \in K$ we can find an open set $U_x$ such that $y \not\in U_x$. Obviously $\{U_x\}_{x \in K}$ covers $K$, so select a finite subcollection that covers $K$, and so on. I had never realized that such reasoning requires the Axiom of Choice. Namely, suppose we have a lemma that states “In $T_2$ spaces, if $x \neq y$, then there is an open set $U$ such that $x \in U$ and $y \not\in \overline{U}$” (like our lemma T2_cl_open_sep below). This only states that the set of such open sets $U$ is not empty. To get the collection $\{U_x\}_{x \in K}$ in this proof we have to select one such set among many for every $x \in K$ and this is where we use the Axiom of Choice. Probably in 99/100 cases when an informal calculus proof states something like $\forall \exists \delta \cdots$ the proof uses AC. Most of the time the use of AC in such proofs can be avoided. This is also the case for the fact that in a $T_2$ space compact sets are closed.

67.1 Compact sets are closed - no need for AC

In this section we show that in a $T_2$ topological space compact sets are closed.

First we prove a lemma that in a $T_2$ space two points can be separated by the closure of an open set.

lemma (in topology0) T2_c1_open_sep:

assumes $T$ {is $T_2$} and $x \in \bigcup T$ $y \in \bigcup T$ $x \neq y$

shows $\exists U \in T$. ($x \in U$ $\land$ $y \not\in \overline{U}$)

proof -

from assms have $\exists U \in T$. $\exists V \in T$. $x \in U$ $\land$ $y \in V$ $\land$ $U \cap V = 0$

using isT2_def by simp

then obtain $U$ $V$ where $U \in T$ $V \in T$ $x \in U$ $y \in V$ $U \cap V = 0$
by auto
then have $U \in T \land x \in U \land y \in V \land \text{cl}(U) \cap V = 0$
  using disj_open_cl_disj by auto
thus $\exists U \in T. (x \in U \land y \notin \text{cl}(U))$ by auto
qed

AC-free proof that in a Hausdorff space compact sets are closed. To understand the notation recall that in Isabelle/ZF $\text{Pow}(A)$ is the powerset (the set of subsets) of $A$ and $\text{FinPow}(A)$ denotes the set of finite subsets of $A$ in IsarMathLib.

**Theorem (in topology0) in_t2_compact_is_cl:**
assumes A1: $T$ (is T2) and A2: $K$ (is compact in) $T$
shows $K$ (is closed in) $T$

proof -
let $X = \bigcup T$
have $\forall y \in X - K. \exists U \in T. y \in U \land U \subseteq X - K$

proof -
let $B = \bigcup \{ V \in T. x \in V \land y \notin \text{cl}(V) \}$
have $I: B \in \text{Pow}(T), \text{FinPow}(B) \subseteq \text{Pow}(B)$
using FinPow_def by auto
from $\langle K \text{ (is compact in)} T \rangle \langle \forall x \in K. \forall y \in X \land x \neq y \rangle$
using IsCompact_def by auto
with $\langle T \text{ (is T2)} \rangle \langle \forall x \in K. \forall V \in B. x \in V \land y \notin \text{cl}(V) \rangle \neq 0$
using T2_cl_open_sep by auto
hence $K \subseteq \bigcup B$ by blast
with $\langle K \text{ (is compact in)} T \rangle$ I have
$\exists N \in \text{FinPow}(B). K \subseteq \bigcup N$
using IsCompact_def by auto
then obtain $N$ where $N \in \text{FinPow}(B) \land K \subseteq \bigcup N$
by auto
with I have $N \subseteq B$ by auto
hence $\forall V \in N. V \subseteq B$ by auto
let $M = \{ \text{cl}(V) \mid V \in N \}$
let $C = \{ D \in \text{Pow}(X) \mid D \text{ (is closed in)} T \}$
from $\langle N \in \text{FinPow}(B) \rangle \langle \forall V \in B. \text{cl}(V) \in C \land N \in \text{FinPow}(B) \rangle$
using cl_is_closed IsClosed_def by auto
then have $M \in \text{FinPow}(C)$ by (rule fin_image_fin)
then have $X - \bigcup M \in T$ using fin_union_cl_is_cl IsClosed_def
by simp
moreover from $\langle \forall x \in X. y \notin \text{cl}(V) \rangle \langle \forall V \in N. V \subseteq B \rangle$
\begin{enumerate}
  \item $y \in X - \bigcup M$ by simp
\end{enumerate}
morerover have $X - \bigcup M \subseteq X - K$

proof -
from $\langle \forall V \in N. V \subseteq B \rangle$ have $\bigcup N \subseteq \bigcup M$ using cl_contains_set by auto
with $\langle K \subseteq \bigcup N \rangle$ show $X - \bigcup M \subseteq X - K$ by auto

843
ultimately have $\exists U. U \in T \land y \in U \land U \subseteq X - K$

by auto

thus $\exists U \in T. y \in U \land U \subseteq X - K$ by auto

qed

end

68 Topology 2

theory Topology_ZF_2 imports Topology_ZF_1 func1 Fol1

begin

This theory continues the series on general topology and covers the definition and basic properties of continuous functions. We also introduce the notion of homeomorphism and prove the pasting lemma.

68.1 Continuous functions.

In this section we define continuous functions and prove that certain conditions are equivalent to a function being continuous.

In standard math we say that a function is continuous with respect to two topologies $\tau_1, \tau_2$ if the inverse image of sets from topology $\tau_2$ are in $\tau_1$. Here we define a predicate that is supposed to reflect that definition, with a difference that we don’t require in the definition that $\tau_1, \tau_2$ are topologies. This means for example that when we define measurable functions, the definition will be the same.

The notation $f-(A)$ means the inverse image of (a set) $A$ with respect to (a function) $f$.

definition

IsContinuous($\tau_1, \tau_2, f$) $\equiv (\forall U \in \tau_2. f-(U) \in \tau_1)$

The space of continuous functions mapping $X = \bigcup \tau_1$ to $Y = \bigcup \tau_2$ will be denoted $\text{Cont}(\tau_1, \tau_2)$.

definition

$\text{Cont}(\tau_1, \tau_2)$ $\equiv \{ f \in (\bigcup \tau_1) \rightarrow (\bigcup \tau_2). \text{IsContinuous}(\tau_1, \tau_2, f) \}$

A trivial example of a continuous function - identity is continuous.
lemma id_cont: shows IsContinuous(τ, τ, id(⋃τ))
proof -
  { fix U assume U∈τ
    then have id(⋃τ)-(U) = U using vimage_id_same by auto
    with ⋃U∈τ have id(⋃τ)-(U) ∈ τ by simp
  } then show IsContinuous(τ, τ, id(⋃τ)) unfolding IsContinuous_def
  by simp
qed

Identity is in the space of continuous functions from ∪τ to itself.

lemma id_cont_sp: shows {⟨x,x⟩. x∈⋃τ} ∈ Cont(τ, τ)
proof -
  have id(⋃τ) : ⋃τ → ⋃τ and IsContinuous(τ, τ, id(⋃τ))
    using id_type id_cont by auto
  moreover have id(⋃τ) = {⟨x,x⟩. x∈⋃τ} by blast
  ultimately show thesis unfolding Cont_def by simp
qed

A constant function is continuous.

lemma const_cont: assumes T {is a topology}
  shows IsContinuous(T, τ, ConstantFunction(⋃T,c))
proof -
  let C = ConstantFunction(⋃T,c)
  { fix U assume U∈τ
    have C-(U) ∈ T proof -
      { assume c∈U
        with assms have C-(U) ∈ T using carr_open const_vimage_domain
          by simp
      }
      moreover
      { assume c∉U
        with assms have C-(U) ∈ T using empty_open const_vimage_empty
          by simp
      }
      ultimately show C-(U) ∈ T by auto
    qed
  } then show thesis unfolding IsContinuous_def
  by simp
qed

If c ∈ Y = ⋃S, then the constant function defined on X = ∪T that is
equal to c is in the the space of continuous functions from X to Y.

lemma const_cont_sp: assumes T {is a topology} c∈⋃S
  shows {(x,c). x∈⋃T} ∈ Cont(T, S)
proof using assms ZF_fun_from_total const_fun_def_alt const_cont
  unfolding Cont_def by simp
We will work with a pair of topological spaces. The following locale sets up our context that consists of two topologies $\tau_1, \tau_2$ and a continuous function $f : X_1 \rightarrow X_2$, where $X_i$ is defined as $\bigcup \tau_i$ for $i = 1, 2$. We also define notation $\text{cl}_1(A)$ and $\text{cl}_2(A)$ for closure of a set $A$ in topologies $\tau_1$ and $\tau_2$, respectively.

locale two_top_spaces0 =
  fixes $\tau_1$
  assumes tau1_is_top: $\tau_1$ {is a topology}
  fixes $\tau_2$
  assumes tau2_is_top: $\tau_2$ {is a topology}
  fixes $X_1$
  defines $X1\_def$ [simp]: $X_1 \equiv \bigcup \tau_1$
  fixes $X_2$
  defines $X2\_def$ [simp]: $X_2 \equiv \bigcup \tau_2$
  fixes $f$
  assumes fmapAssum: $f : X_1 \rightarrow X_2$
  fixes isContinuous (_ {is continuous} [50] 50)
  defines isContinuous\_def [simp]: $g$ {is continuous} $\equiv$ IsContinuous($\tau_1, \tau_2, g$)
  fixes $\text{cl}_1$
  defines $cl\_1\_def$ [simp]: $\text{cl}_1(A) \equiv \text{Closure}(A, \tau_1)$
  fixes $\text{cl}_2$
  defines $cl\_2\_def$ [simp]: $\text{cl}_2(A) \equiv \text{Closure}(A, \tau_2)$

First we show that theorems proven in locale topology0 are valid when applied to topologies $\tau_1$ and $\tau_2$.

lemma (in two_top_spaces0) topol\_cntxs\_valid:
  shows topology0($\tau_1$) and topology0($\tau_2$)
  using tau1\_is\_top tau2\_is\_top topology0\_def by auto

For continuous functions the inverse image of a closed set is closed.

lemma (in two_top_spaces0) TopZF\_2\_1\_L1:
  assumes A1: $f$ {is continuous} and A2: $D$ {is closed in} $\tau_2$
  shows $f\-(D)$ {is closed in} $\tau_1$
  proof -
  from fmapAssum have $f\-(D) \subseteq X_1$ using func1\_1\_L3 by simp
  moreover from fmapAssum have $f\-(X_2 - D) = X_1 - f\-(D)$
    using Pi\_iff function\_vimage\_Diff func1\_1\_L4 by auto
  ultimately have $X_1 - f\-(X_2 - D) = f\-(D)$ by auto
  moreover from A1 A2 have $(X_1 - f\-(X_2 - D))$ {is closed in} $\tau_1$
    using isClosed\_def IsContinuous\_def topol\_cntxs\_valid topology0\_Top\_3\_L9

846
by simp
ultimately show $f^{-}(D) \{\text{is closed in}\} \tau_1$ by simp
qed

If the inverse image of every closed set is closed, then the image of a closure is contained in the closure of the image.

**Lemma (in two_top_spaces0) Top_ZF_2_1_L2:**
assumes $A1: \forall D. (D \{\text{is closed in}\} \tau_2) \rightarrow f^{-}(D) \{\text{is closed in}\} \tau_1$
and $A2: A \subseteq X_1$
shows $f(cl_1(A)) \subseteq cl_2(f(A))$

**Proof:**
- from fmapAssum have $f(A) \subseteq cl_2(f(A))$
  - using func1_1_L6 topol_cntxs_valid topology0.cl_contains_set by simp
with fmapAssum have $f^{-}(f(A)) \subseteq f^{-}(cl_2(f(A)))$
  - by auto
moreover from fmapAssum A2 have $A \subseteq f^{-}(f(A))$
  - using func1_1_L9 by simp
ultimately have $A \subseteq f^{-}(cl_2(f(A)))$ by auto
with fmapAssum A1 have $f(cl_1(A)) \subseteq f^{-}(cl_2(f(A)))$
  - using func1_1_L6 func1_1_L8 IsClosed_def topol_cntxs_valid topology0.cl_is_closed topology0.Top_3_L13 by simp
moreover from fmapAssum have $f(f^{-}(cl_2(f(A)))) \subseteq cl_2(f(A))$
  - using fun_is_function function_image_vimage by simp
ultimately show $f(cl_1(A)) \subseteq cl_2(f(A))$
  - by auto
qed

If $f(A) \subseteq f(A)$ (the image of the closure is contained in the closure of the image), then $f^{-1}(B) \subseteq f^{-1}(B)$ (the inverse image of the closure contains the closure of the inverse image).

**Lemma (in two_top_spaces0) Top_ZF_2_1_L3:**
assumes $A1: \forall A. (A \subseteq X_1 \rightarrow f(cl_1(A)) \subseteq cl_2(f(A)))$
shows $\forall B. (B \subseteq X_2 \rightarrow cl_1(f^{-}(B)) \subseteq f^{-}(cl_2(B)))$

**Proof:**
- { fix $B$ assume $B \subseteq X_2$
  - from fmapAssum A1 have $f(cl_1(f^{-}(B))) \subseteq cl_2(f^{-}(B))$
    - using func1_1_L3 by simp
  moreover from fmapAssum have $f(cl_1(f^{-}(B))) \subseteq cl_2(f^{-}(B))$
    - using fun_is_function function_image_vimage func1_1_L6 topol_cntxs_valid topology0.top_closure_mono by simp
ultimately have $f^{-}(f(cl_1(f^{-}(B)))) \subseteq f^{-}(cl_2(B))$
    - using fmapAssum fun_is_function by auto
moreover from fmapAssum have $f(cl_1(f^{-}(B))) \subseteq f^{-}(f(cl_1(f^{-}(B))))$
    - using func1_1_L3 func1_1_L9 IsClosed_def topol_cntxs_valid topology0.cl_is_closed by simp

ultimately have $\text{cl}_1(f^{-1}(B)) \subseteq f^{-1}(\text{cl}_2(B))$ by auto
\}
then show thesis by simp
qed

If $f^{-1}(B) \subseteq f^{-1}(\overline{B})$ (the inverse image of a closure contains the closure of the inverse image), then the function is continuous. This lemma closes a series of implications in lemmas TopZF_2_1_L1, TopZF_2_1_L2 and TopZF_2_1_L3 showing equivalence of four definitions of continuity.

**Lemma (in two_top_spaces0) TopZF_2_1_L4:**

assumes $A1: \forall B. (B \subseteq X_2 \longrightarrow \text{cl}_1(f^{-1}(B)) \subseteq f^{-1}(\text{cl}_2(B)))$

shows $f$ {is continuous}

proof -
\{
fix $U$ assume $U \in \tau_2$
then have $(X_2 - U)$ {is closed in} $\tau_2$
  using topol_cntxs_valid topology0.Top_3_L9 by simp
moreover have $X_2 - U \subseteq \bigcup \tau_2$ by auto
ultimately have $\text{cl}_2(X_2 - U) = X_2 - U$
  using topol_cntxs_valid topology0.Top_3_L8 by simp
moreover from $A1$ have $\text{cl}_1((X_2 - U)) \subseteq f^{-1}(\text{cl}_2(X_2 - U))$
  by auto
ultimately have $\text{cl}_1(f^{-1}(X_2 - U)) \subseteq f^{-1}(X_2 - U)$ by simp
moreover from fmapAssum have $f^{-1}(X_2 - U) \subseteq \text{cl}_1(f^{-1}(X_2 - U))$
  using func1_1_L3 topol_cntxs_valid topology0.cl_contains_set by simp
ultimately have $f^{-1}(X_2 - U)$ {is closed in} $\tau_1$
  using fmapAssum func1_1_L3 topol_cntxs_valid topology0.Top_3_L8 by auto
with fmapAssum have $f(U) \in \tau_1$
  using fun_is_function function_vimage_Diff func1_1_L4 func1_1_L3 IsClosed_def double_complement by simp
\} then have $\forall U \in \tau_2. f(U) \in \tau_1$ by simp
then show thesis using IsContinuous_def by simp
qed

For continuous functions the closure of the inverse image is contained in the inverse image of the closure. This is a shortcut through a series of implications provided by TopZF_2_1_L1, TopZF_2_1_L2 and TopZF_2_1_L3.

**Corollary (in two_top_spaces0) im_cl_in_cl_im:**

assumes $f$ {is continuous} and $B \subseteq X_2$
shows $\text{cl}_1(f^{-1}(B)) \subseteq f^{-1}(\text{cl}_2(B))$
using assms TopZF_2_1_L1 TopZF_2_1_L2 TopZF_2_1_L3 by simp

Another condition for continuity: it is sufficient to check if the inverse image of every set in a base is open.

**Lemma (in two_top_spaces0) TopZF_2_1_L5:**

assumes $A1: B$ {is a base for} $\tau_2$ and $A2: \forall U \in B. f(U) \in \tau_1$
shows $f$ {is continuous}

proof -
We can strengthen the previous lemma: it is sufficient to check if the inverse image of every set in a subbase is open. The proof is rather awkward, as usual when we deal with general intersections. We have to keep track of the case when the collection is empty.

**Lemma (in two_top_spaces0) Top_ZF_2_1_L6:**

assumes A1: B \{is a subbase for\} \(\tau_2\) and A2: \(\forall U \in B. \ f-(U) \in \tau_1\)

shows \(f \{is continuous}\)

**proof -**

let \(C = \{\bigcap A. A \in \text{FinPow}(B)\}\)

from A1 have C \{is a base for\} \(\tau_2\)

using IsAsubBaseFor_def by simp

moreover have \(\forall U \in C. \ f-(U) \in \tau_1\)

proof

fix U assume U \in C

\{ assume \(f-(U) = 0\)

with tau1_is_top have \(f-(U) \in \tau_1\)

using empty_open by simp \}

moreover

\{ assume \(f-(U) \neq 0\)

then have U\#0 by (rule func1_1_L13)

moreover from \(U \in C\) obtain A where

\(A \in \text{FinPow}(B)\) and \(U = \bigcap A\)

by auto

ultimately have \(\bigcap A \neq 0\) by simp

then have A\#0 by (rule inter_nempty_nempty)

then have \(\{f-(W). W \in A\} \neq 0\) by simp

moreover from A2 \(A \in \text{FinPow}(B)\) have \(\{f-(W). W \in A\} \in \text{FinPow}(\tau_1)\)

by (rule fin_image_fin)

ultimately have \(\bigcap \{f-(W). W \in A\} \in \tau_1\)

using topol_cntxs_valid topology0.fin_inter_open_open by simp

moreover

from \(A \in \text{FinPow}(B)\) have A \subseteq B using FinPow_def by simp

with tau2_is_top A1 have A \subseteq Pow(X_2)

using IsAsubBaseFor_def IsATopology_def by auto

with fmapAssum \(A \neq 0\) \(U = \bigcap A\) have \(f-(U) = \bigcap \{f-(W). W \in A\}\)

using func1_1_L12 by simp

qed
ultimately have \( f(U) \in \tau_1 \) by simp 
ultimately show \( f(U) \in \tau_1 \) by blast
qed
ultimately show \( f \) \((\text{is continuous})\)
using Top_ZF_2_1_L5 by simp
qed

A dual of Top_ZF_2_1_L5: a function that maps base sets to open sets is open.

**Lemma (in two_top_spaces0) base_image_open:**
- **Assumptions:** A1: \( B \) \((\text{is a base for})\) \( \tau_1 \) and A2: \( \forall B \in B. \ f(B) \in \tau_2 \) and A3: \( U \in \tau_1 \)
- **Shows:** \( f(U) \in \tau_2 \)

**Proof -**
- From A1 A3 obtain \( E \) where \( E \in \text{Pow}(B) \) and \( U = \bigcup E \) using Top_1_2_L1 by blast
- With A1 have \( f(U) = \bigcup \{f(E). E \in E\} \) using Top_1_2_L5 fmapAssum image_of_Union by auto
- Moreover from A2 \( \forall E \in \text{Pow}(B). \ \text{have} \ \{f(E). E \in E\} \in \text{Pow}(\tau_2) \) by auto
- Then have \( \bigcup \{f(E). E \in E\} \in \tau_2 \) using tau2_is_top IsATopology_def by simp
- Ultimately show thesis using tau2_is_top IsATopology_def by auto
qed

A composition of two continuous functions is continuous.

**Lemma comp_cont:**
- Assumes IsContinuous(T,S,f) and IsContinuous(S,R,g)
- Shows IsContinuous(T,R,g O f)
- Using assms IsContinuous_def vimage_comp by simp

A composition of three continuous functions is continuous.

**Lemma comp_cont3:**
- Assumes IsContinuous(T,S,f) and IsContinuous(S,R,g) and IsContinuous(R,P,h)
- Shows IsContinuous(T,P,h O g O f)
- Using assms IsContinuous_def vimage_comp by simp

The graph of a continuous function into a Hausdorff topological space is closed in the product topology. Recall that in ZF a function is the same as its graph.

**Lemma (in two_top_spaces0) into_T2_graph_closed:**
- Assumes \( f \) \((\text{is continuous})\) \( \tau_2 \) \((\text{is T}_2)\)
- Shows \( f \) \((\text{is closed in})\) ProductTopology(\( \tau_1, \tau_2 \))

**Proof -**
- From fmapAssum have \( f = \{(x,f(x)). x \in X_1\} \) using fun_is_set_of_pairs by simp
- Let \( f_c = X_1 \times X_2 - f \)
- Have \( f_c \in \text{ProductTopology}(\tau_1,\tau_2) \)
- Proof -
  - \{ fix p assume p\(\in f_c \)
then have \( p \in X_1 \times X_2 \) and \( p \not\in f \) by auto
from \( \langle p \in X_1 \times X_2 \rangle \) obtain \( x \ y \) where \( x \in X_1 \ y \in X_2 \ p = \langle x, y \rangle \)
by auto
have \( y \neq f(x) \)
proof -
\[
\begin{align*}
\{ \text{ assume } & y = f(x) \\
& \text{ with } \langle x \in X_1 \rangle \ \langle p = \langle x, y \rangle \rangle \ \text{ have } p \in \{ \langle x, f(x) \rangle \dot\ x \in X_1 \} \ \text{ by auto} \\
& \text{ with } \langle f = \{ \langle x, f(x) \rangle \dot\ x \in X_1 \} \ \langle p \not\in f \rangle \ \text{ have False by auto} \\
\} \ \text{ thus } y \neq f(x) \ \text{ by auto}
\end{align*}
\]
qed
from \( \langle x \in X_1 \rangle \) have \( f(x) \in X_2 \) by (rule apply_funtype)
\[
\begin{align*}
\text{with } \langle y \in X_2 \rangle & \ \text{ have } y \in \bigcup \tau_2 f(x) \in \bigcup \tau_2 \ \text{ by auto} \\
& \text{ where } U \in \tau_2 \ V \in \tau_2 \ y \in U \ f(x) \in V \ U \cap V = 0
\end{align*}
\]
unfolding isT2_def by blast
let \( W = f^{-1}(V) \)
have \( W \subseteq X_1 \ W \subseteq X_2 \ x \in W \ p \in W \times U \ f(W) \subseteq V \)
proof -
\[
\begin{align*}
\text{from } \langle x \in X_1 \rangle \ \text{ have } & \text{IsContinuous}((\tau_1, \tau_2, f) \ \text{ by simp} \\
& \text{ with } \langle V \in \tau_2 \rangle \ \langle U \in \tau_2 \rangle \ \text{ show } W \in \tau_1 \ W \subseteq X_1 \ U \subseteq X_2 \\
& \text{ unfolding } \text{IsContinuous_def} \ \text{ by auto} \\
& \text{ from } \langle y \in U \rangle \ \langle y \in U \rangle \ \langle p = \langle x, y \rangle \rangle \ \text{ show } \ p \in W \times U \ \text{ by simp} \\
& \text{ from } \langle f(U) \subseteq V \rangle \ \text{ show } f(W) \subseteq V \\
& \text{ using } \text{fun_is_fun function_image_vimage} \ \text{ by simp} \\
f_c
\end{align*}
\]
qed
from \( \langle U \cap V = 0 \rangle \ \langle W \subseteq X_1 \rangle \ \langle U \subseteq X_2 \rangle \ \text{ have } W \times U \subseteq f_c \\
\text{ using } \text{vimage_prod_dis_graph} \ \text{ by blast} \\
\text{ with } \langle W \in \tau_1 \rangle \ \langle U \in \tau_2 \rangle \ \langle p \in W \times U \rangle \ \text{ have } \exists W \in \tau_1 \ \exists U \in \tau_2 \ p \in (W \times U \ 
\times U \subseteq f_c
\]
by blast
\[
\begin{align*}
& \text{ with } \text{tau1_is_top tau2_is_top show } f_c \in \text{ProductTopology}(\tau_1, \tau_2) \\
& \text{ using } \text{point_neighb_prod_top} \ \text{ by simp} \\
& \text{ with } \text{fmapAssum tau1_is_top tau2_is_top show thesis} \\
& \text{ using } \text{fun_subset_prod Top_1_4_T1(3)} \ \text{ unfolding } \text{IsClosed_def} \\
& \text{ by auto}
\end{align*}
\]
qed

68.2 Homeomorphisms

This section studies "homeomorphisms" - continuous bijections whose inverses are also continuous. Notions that are preserved by (commute with) homeomorphisms are called "topological invariants".

Homeomorphism is a bijection that preserves open sets.

\[
\text{definition } \text{IsAhomeomorphism}(T, S, f) \equiv
\]

851
(f ∈ bij(∪T, ∪S) ∧ IsContinuous(T, S, f) ∧ IsContinuous(S, T, converse(f))

Inverse (converse) of a homeomorphism is a homeomorphism.

**Lemma homeo_inv:** assumes IsAhomeomorphism(T, S, f)
shows IsAhomeomorphism(S, T, converse(f))
using assms IsAhomeomorphism_def bij_converse_bij bij_converse_converse
by auto

Homeomorphisms are open maps.

**Lemma homeo_open:** assumes IsAhomeomorphism(T, S, f) and U ∈ T
shows f(U) ∈ S
using assms image_converse IsAhomeomorphism_def IsContinuous_def by simp

A continuous bijection that is an open map is a homeomorphism.

**Lemma bij_cont_open_homeo:**
assumes f ∈ bij(∪T, ∪S) and IsContinuous(T, S, f) and ∀ U ∈ T. f(U) ∈ S
shows IsAhomeomorphism(T, S, f)
using assms image_converse IsAhomeomorphism_def IsContinuous_def by auto

A continuous bijection that maps base to open sets is a homeomorphism.

**Lemma (in two_top_spaces0) bij_base_open_homeo:**
assumes A1: f ∈ bij(X₁, X₂) and A2: B {is a base for} τ₁ and A3: C {is a base for} τ₂
A4: ∀ U ∈ C. f(U) ∈ τ₁ and A5: ∀ V ∈ B. f(V) ∈ τ₂
shows IsAhomeomorphism(τ₁, τ₂, f)
using assms tau2_is_top tau1_is_top bij_converse_bij bij_is_fun two_top_spaces0_def
image_converse two_top_spaces0.Top_ZF_2_1_L5 IsAhomeomorphism_def by simp

A bijection that maps base to base is a homeomorphism.

**Lemma (in two_top_spaces0) bij_base_homeo:**
assumes A1: f ∈ bij(X₁, X₂) and A2: B {is a base for} τ₁ and A3: {f(B). B ∈ B} {is a base for} τ₂
shows IsAhomeomorphism(τ₁, τ₂, f)
proof -
  note A1
  moreover have f {is continuous}
proof -
  { fix C assume C ∈ {f(B). B ∈ B}
    then obtain B where B ∈ B and I: C = f(B) by auto
    with A2 have B ⊆ X₁ using Top_1_2_L5 by auto
    with A1 A2 ⟨B ∈ B⟩ I have f⁻¹(C) ∈ τ₁
      using bij_def inj_vimage_image base_sets_open by auto
  } hence ∀ C ∈ {f(B). B ∈ B}. f⁻¹(C) ∈ τ₁ by auto

852
with A3 show thesis by (rule Top_ZF_2_1_L5)

qed

moreover

from A3 have \( \forall B \in \mathcal{B}. f(B) \in \tau_2 \) using base_sets_open by auto

with A2 have \( \forall U \in \tau_1. f(U) \in \tau_2 \) using base_image_open by simp

ultimately show thesis using bij_cont_open_homeo by simp

qed

Interior is a topological invariant.

theorem int_top_invariant: assumes A1: \( A \subseteq \bigcup T \) and A2: \( \text{IsAhomeomorphism}(T,S,f) \)

shows \( f(\text{Interior}(A,T)) = \text{Interior}(f(A),S) \)

proof -

let \( A = \{U \in T. U \subseteq A\} \)

have I: \( \{f(U). U \in A\} = \{V \in S. V \subseteq f(A)\} \)

proof

from A2 show \( \{f(U). U \in A\} \subseteq \{V \in S. V \subseteq f(A)\} \)

using homeo_open by auto

{ fix \( V \) assume \( V \in \{V \in S. V \subseteq f(A)\} \)

hence \( V \subseteq f(A) \) by auto

let \( U = f^{-1}(V) \)

from II have \( U \subseteq f^{-1}(f(A)) \) by auto

moreover from asms have \( f^{-1}(f(A)) = A \)

using IsAhomeomorphism_def bij_def inj_vimage_image by auto

moreover from A2 \( \langle V \in S \rangle \) have \( U \in T \)

using IsAhomeomorphism_def IsContinuous_def by simp

moreover

from \( \langle V \in S \rangle \) have \( V \subseteq \bigcup S \) by auto

with A2 have \( V = f(U) \)

using IsAhomeomorphism_def bij_def surj_image_vimage by auto

ultimately have \( V \in \{f(U). U \in A\} \) by auto

} thus \( \{V \in S. V \subseteq f(A)\} \subseteq \{f(U). U \in A\} \) by auto

qed

have \( f(\text{Interior}(A,T)) = f(\bigcup A) \) unfolding Interior_def by simp

also from A2 have \( \ldots = \bigcup \{f(U). U \in A\} \)

using IsAhomeomorphism_def bij_def inj_def_image_of_Union by auto

also from I have \( \ldots = \text{Interior}(f(A),S) \) unfolding Interior_def by simp

finally show thesis by simp

qed

68.3 Topologies induced by mappings

In this section we consider various ways a topology may be defined on a set that is the range (or the domain) of a function whose domain (or range) is a topological space.

A bijection from a topological space induces a topology on the range.

theorem bij_induced_top: assumes A1: \( T \) {is a topology} and A2: \( f \in \text{bij}(\bigcup T,Y) \)

shows \( \{f(U). U \in T\} \) {is a topology} and
\{ \{ f(x) \mid x \in U \} \mid U \in T \} \{ is a topology \} and 
\{ \bigcup \{ f(U) \mid U \in T \} \} = Y \ and 
IsAThomeomorphism(T, \{ f(U) \mid U \in T \}, f)

**proof** -
from A2 have \( f \in \text{inj}(\bigcup T, Y) \) using bij_def by simp
then have \( f: \bigcup T \to Y \) using inj_def by simp
let \( S = \{ f(U) \mid U \in T \} \)
have \( f: \bigcup T \to Y \) using inj_def by simp
let \( S = \{ f(U) \mid U \in T \} \)
let \( \mathcal{M} = \{ f(V) \mid V \in \mathcal{N} \} \)
by blast

with \( \bigcup \mathcal{N} \) show \( \bigcup \mathcal{N} \) by simp
with A1 have \( (\bigcup \mathcal{M}) \subseteq S \) by auto
moreover have \( f(\bigcup \mathcal{M}) = \bigcup S \)
by auto

with \( \bigcup \mathcal{N} \) show \( \bigcup \mathcal{N} \) by simp
with A2 have \( \bigcup \mathcal{M} = \bigcup S \)
by auto

ultimately show \( f(\bigcup \mathcal{M}) = \bigcup S \) by simp
ultimately have \( \bigcup S \subseteq S \)
because \( \bigcup S \subseteq S \)
by auto

68.4 Partial functions and continuity

Suppose we have two topologies \( \tau_1, \tau_2 \) on sets \( X_i = \bigcup \tau_i, i = 1, 2 \). Consider some function \( f : A \to X_2 \), where \( A \subseteq X_1 \) (we will call such function "partial"). In such situation we have two natural possibilities for the pairs of topologies with respect to which this function may be continuous. One is obviously the original \( \tau_1, \tau_2 \) and in the second one the first element of the pair is the topology relative to the domain of the function: \( \{ A \cap U | U \in \tau_1 \} \). These two possibilities are not exactly the same and the goal of this section is to explore the differences.

If a function is continuous, then its restriction is continuous in relative topology.
lemma (in two_top_spaces0) restr_cont:
assumes A1: A ⊆ X₁ and A2: f {is continuous}
shows IsContinuous(τ₁ {restricted to} A, τ₂, restrict(f,A))
proof -
  let g = restrict(f,A)
  { fix U assume U ∈ τ₂
    with A2 have f⁻¹(U) ∈ τ₁
    moreover from A1 have g⁻¹(U) = f⁻¹(U) ∩ A
      using fmapAssum func1_2_L1 by simp
    ultimately have g⁻¹(U) ∈ (τ₁ {restricted to} A)
      using RestrictedTo_def by auto
  } then show thesis using IsContinuous_def by simp
qed

If a function is continuous, then it is continuous when we restrict the topology on the range to the image of the domain.

lemma (in two_top_spaces0) restr_image_cont:
assumes A1: f {is continuous}
shows IsContinuous(τ₁, τ₂ {restricted to} f(X₁),f)
proof -
  have ∀ U ∈ τ₂ {restricted to} f(X₁). f⁻¹(U) ∈ τ₁
    proof
      fix U assume U ∈ τ₂ {restricted to} f(X₁)
      then obtain V where V ∈ τ₂ and U = V ∩ f(X₁)
      using RestrictedTo_def by auto
      with A1 show f⁻¹(U) ∈ τ₁
        using fmapAssum inv_im_inter_im IsContinuous_def by simp
    qed
  then show thesis using IsContinuous_def by simp
qed

A combination of restr_cont and restr_image_cont.

lemma (in two_top_spaces0) restr_restr_image_cont:
assumes A1: A ⊆ X₁ and A2: f {is continuous} and
A3: g = restrict(f,A) and
A4: τ₃ = τ₁ {restricted to} A
shows IsContinuous(τ₃, τ₂ {restricted to} g(A),g)
proof -
  from A1 A4 have ⋃ τ₃ = A
    using union_restrict by auto
  have two_top_spaces0(τ₃, τ₂, g)
    proof -
      from A4 have τ₃ {is a topology} and τ₂ {is a topology}
        using tau1_is_top tau2_is_top
topology0_def topology0.Top_1_L4 by auto
      moreover from A1 A3 ⋃ τ₃ = A have g: ⋃ τ₃ → ⋃ τ₂
        using fmapAssum restrict_type2 by simp
ultimately show thesis using two_top_spaces0_def
by simp
qed
moreover from assms have IsContinuous(τ₃, τ₂, g)
    using restr_cont by simp
ultimately have IsContinuous(τ₃, τ₂ {restricted to} g(∪τ₃),g)
    by (rule two_top_spaces0.restr_image_cont)
moreover note (∪τ₃ = A)
ultimately show thesis by simp
qed

We need a context similar to two_top_spaces0 but without the global function $f : X₁ \to X₂$.
locale two_top_spaces1 =

  fixes τ₁
  assumes tau1_is_top: τ₁ {is a topology}

  fixes τ₂
  assumes tau2_is_top: τ₂ {is a topology}

  fixes X₁
defines X1_def [simp]: X₁ ≡ ∪τ₁

  fixes X₂
defines X2_def [simp]: X₂ ≡ ∪τ₂

If a partial function $g : X₁ \supseteq A \to X₂$ is continuous with respect to $(τ₁, τ₂)$, then $A$ is open (in $τ₁$) and the function is continuous in the relative topology.

lemma (in two_top_spaces1) partial_fun_cont:
  assumes A1: $g:A\to X₂$ and A2: IsContinuous(τ₁,τ₂,g)
  shows $A \in τ₁$ and IsContinuous(τ₁ {restricted to} A, τ₂, g)
proof -
    from A2 have $g⁻(X₂) \in τ₁$
        using tau2_is_top IsATopology_def IsContinuous_def by simp
    with A1 show $A \in τ₁$ using func1_1_L4 by simp
    { fix $V$ assume $V \in τ₂$
      with A2 have $g⁻(V) \in τ₁$ using IsContinuous_def by simp
      moreover
      from A1 have $g⁻(V) \subseteq A$ using func1_1_L3 by simp
      hence $g⁻(V) = A \cap g⁻(V)$ by auto
      ultimately have $g⁻(V) \in (τ₁ {restricted to} A)$
          using RestrictedTo_def by auto
    } then show IsContinuous(τ₁ {restricted to} A, τ₂, g)
        using IsContinuous_def by simp
qed

For partial function defined on open sets continuity in the whole and relative topologies are the same.
lemma (in two_top_spaces1) part_fun_on_open_cont:
  assumes A1: g:A→Xₐ and A2: A ∈ τ₁
  shows IsContinuous(τ₁,τ₂,g) ←→
  IsContinuous(τ₁ {restricted to} A, τ₂, g)
proof
  assume IsContinuous(τ₁,τ₂,g)
  with A1 show IsContinuous(τ₁ {restricted to} A, τ₂, g)
    using partial_fun_cont by simp
next
  assume I: IsContinuous(τ₁ {restricted to} A, τ₂, g)
  { fix V assume V ∈ τ₂
    with I have g-(V) ∈ (τ₁ {restricted to} A)
      using IsContinuous_def by simp
    then obtain W where W ∈ τ₁ and g-(V) = A∩W
      using RestrictedTo_def by auto
    with A2 have g-(V) ∈ τ₁ using tau1_is_top IsATopology_def
      by simp
  } then show IsContinuous(τ₁,τ₂,g) using IsContinuous_def
    by simp
qed

68.5 Product topology and continuity

We start with three topological spaces (τ₁,X₁),(τ₂,X₂) and (τ₃,X₃) and a
function f : X₁ × X₂ → X₃. We will study the properties of f with respect
to the product topology τ₁ × τ₂ and τ₃. This situation is similar as in locale
two_top_spaces0 but the first topological space is assumed to be a product
of two topological spaces.

First we define a locale with three topological spaces.
locale prod_top_spaces0 =
  fixes τ₁
  assumes tau1_is_top: τ₁ {is a topology}
  fixes τ₂
  assumes tau2_is_top: τ₂ {is a topology}
  fixes τ₃
  assumes tau3_is_top: τ₃ {is a topology}
  fixes X₁
defines X1_def [simp]: X₁ ≡ ⋃τ₁
  fixes X₂
defines X2_def [simp]: X₂ ≡ ⋃τ₂
  fixes X₃
defines X3_def [simp]: X₃ ≡ ⋃τ₃
fixes η
defines eta_def [simp]: η ≡ ProductTopology(τ₁, τ₂)

Fixing the first variable in a two-variable continuous function results in a continuous function.

lemma (in prod_top_spaces0) fix_1st_var_cont:
  assumes f: X₁ × X₂ → X₃ and IsContinuous(η, τ₃, f)
  and x ∈ X₁
  shows IsContinuous(τ₂, τ₃, Fix1stVar(f, x))
  using asms fix_1st_var_vimage IsContinuous_def tau1_is_top tau2_is_top
  prod_sec_open1 by simp

Fixing the second variable in a two-variable continuous function results in a continuous function.

lemma (in prod_top_spaces0) fix_2nd_var_cont:
  assumes f: X₁ × X₂ → X₃ and IsContinuous(η, τ₃, f)
  and y ∈ X₂
  shows IsContinuous(τ₁, τ₃, Fix2ndVar(f, y))
  using asms fix_2nd_var_vimage IsContinuous_def tau1_is_top tau2_is_top
  prod_sec_open2 by simp

Having two continuous mappings we can construct a third one on the cartesian product of the domains.

lemma cart_prod_cont:
  assumes A1: τ₁ {is a topology} τ₂ {is a topology} and
  A2: η₁ {is a topology} η₂ {is a topology} and
  A3a: f₁: τ₁ → η₁ and A3b: f₂: τ₂ → η₂ and
  A4: IsContinuous(τ₁, η₁, f₁) IsContinuous(τ₂, η₂, f₂) and
  A5: g = {{p, (f₁(fst(p)), f₂(snd(p)))}. p ∈ τ₁ × τ₂}
  shows IsContinuous(ProductTopology(τ₁, τ₂), ProductTopology(η₁, η₂), g)
proof -
  let τ = ProductTopology(τ₁, τ₂)
  let η = ProductTopology(η₁, η₂)
  let X₁ = τ₁
  let X₂ = τ₂
  let Y₁ = η₁
  let Y₂ = η₂
  let B = ProductCollection(η₁, η₂)
from A1 A2 have τ {is a topology} and η {is a topology}
  using Top_1_4_T1 by auto
moreover have g: X₁ × X₂ → Y₁ × Y₂
proof -
  { fix p assume p ∈ X₁ × X₂
    hence fst(p) ∈ X₁ and snd(p) ∈ X₂ by auto
    from A3a ⟨fst(p) ∈ X₁⟩ have f₁(fst(p)) ∈ Y₁
      by (rule apply_funtype)
    moreover from A3b ⟨snd(p) ∈ X₂⟩ have f₂(snd(p)) ∈ Y₂
    moreover
by (rule apply_functtype)
ultimately have \( \langle f_1(\text{fst}(p)), f_2(\text{snd}(p)) \rangle \in \bigcup \eta_1 \times \bigcup \eta_2 \) by auto
hence \( \forall p \in X_1 \times X_2. \ (f_1(\text{fst}(p)), f_2(\text{snd}(p))) \in Y_1 \times Y_2 \)
by simp
with A5 show \( g : X_1 \times X_2 \rightarrow Y_1 \times Y_2 \) using ZF_fun_from_total
by simp
qed
moreover from A1 A2 have \( \bigcup \tau = X_1 \times X_2 \) and \( \bigcup \eta = Y_1 \times Y_2 \)
using Top_1_4_T1 by auto
ultimately have two_top_spaces0(\( \tau, \eta \),g) using two_top_spaces0_def
by simp
moreover from A2 have B {is a base for} \( \eta \) using Top_1_4_T1
by simp
moreover have \( \forall U \in B. \ g-(U) \in \tau \)
proof
fix U assume U\in B
then obtain V W where V \in \( \eta_1 \) W \in \( \eta_2 \) and U = V \times W
using ProductCollection_def by auto
with A3a A3b A5 have \( g-(U) = f_1-(V) \times f_2-(W) \)
using cart_prod_fun_vimage by simp
moreover from A1 A4 \( \langle V \in \eta_1, W \in \eta_2 \rangle \) have \( f_1-(V) \times f_2-(W) \in \tau \)
using IsContinuous_def prod_open_open_prod by simp
ultimately show \( g-(U) \in \tau \) by simp
qed
ultimately show thesis using two_top_spaces0.Top_ZF_2_1_L5
by simp
qed

A reformulation of the cart_prod_cont lemma above in slightly different notation.

theorem (in two_top_spaces0) product_cont_functions:
assumes f: \( X_1 \rightarrow X_2 \) g: \( \bigcup \tau_3 \rightarrow \bigcup \tau_4 \)
IsContinuous(\( \tau_1, \tau_2 \),f) IsContinuous(\( \tau_3, \tau_4 \),g)
\( \tau_4 \) {is a topology} \( \tau_3 \) {is a topology}
shows IsContinuous(ProductTopology(\( \tau_1, \tau_3 \)),ProductTopology(\( \tau_2, \tau_4 \)),\( \langle \langle x,y \rangle, \langle fx,gy \rangle \rangle \)).
\( \langle x,y \rangle \in X_1 \times \bigcup \tau_3 \))
proof -
have \( \{\langle x,y \rangle, \langle fx,gy \rangle \}. \ \langle x,y \rangle \in X_1 \times \bigcup \tau_3 \} = \{\langle p, \langle f(\text{fst}(p)), g(\text{snd}(p)) \rangle \rangle \}. \ p \in X_1 \times \bigcup \tau_3 \}
by force
with tau1_is_top tau2_is_top asms show thesis using cart_prod_cont
by simp
qed

A special case of cart_prod_cont when the function acting on the second axis is the identity.

lemma cart_prod_cont1:
assumes A1: \( \tau_1 \) {is a topology} and A1a: \( \tau_2 \) {is a topology} and
A2: \( \eta_1 \) {is a topology} and

860
A3: \( f_1: \bigcup \tau_1 \rightarrow \bigcup \eta_1 \) and A4: \( \text{IsContinuous}(\tau_1, \eta_1, f_1) \) and
A5: \( g = \{ (p, \langle f_1(\text{fst}(p)), \text{snd}(p) \rangle) : p \in \bigcup \tau_1 \times \bigcup \tau_2 \} \)
shows \( \text{IsContinuous}(\text{ProductTopology}(\tau_1, \tau_2), \text{ProductTopology}(\eta_1, \tau_2), g) \)
proof -
let \( f_2 = \text{id}(\bigcup \tau_2) \)
have \( \forall x \in \bigcup \tau_2. \ f_2(x) = x \) using \( \text{id}\_\text{conv} \) by blast
hence I: \( \forall p \in \bigcup \tau_1 \times \bigcup \tau_2. \ \text{snd}(p) = f_2(\text{snd}(p)) \) by simp
note A1 A1a A2 A1a A3
moreover have \( f_2: \bigcup \tau_2 \rightarrow \bigcup \tau_2 \) using \( \text{id}\_\text{type} \) by simp
moreover note A4
moreover have \( \text{IsContinuous}(\tau_2, \tau_2, f_2) \) using \( \text{id}\_\text{cont} \) by simp
moreover have \( g = \{ (p, \langle f_1(\text{fst}(p)), f_2(\text{snd}(p)) \rangle) : p \in \bigcup \tau_1 \times \bigcup \tau_2 \} \)
proof
  from A5 I show \( g \subseteq \{ (p, \langle f_1(\text{fst}(p)), f_2(\text{snd}(p)) \rangle) : p \in \bigcup \tau_1 \times \bigcup \tau_2 \} \)
    by auto
  from A5 I show \( \{ (p, \langle f_1(\text{fst}(p)), f_2(\text{snd}(p)) \rangle) : p \in \bigcup \tau_1 \times \bigcup \tau_2 \} \subseteq g \)
    by auto
qed
ultimately show thesis by (rule cart_prod_cont)

Having two continuous mappings \( f, g \) we can construct a third one with values in the cartesian product of the codomains of \( f, g \), defined by \( x \mapsto (f(x), g(x)) \).

lemma (in prod_top_spaces0) cont funcs prod:
  assumes \( f: X_1 \rightarrow X_2 \) \( g: X_1 \rightarrow X_3 \) \( \text{IsContinuous}(\tau_1, \tau_2, f) \) \( \text{IsContinuous}(\tau_1, \tau_3, g) \)
  defines \( h \equiv \{ (x, \langle f(x), g(x) \rangle) : x \in X_1 \} \)
  shows \( \text{IsContinuous}(\tau_1, \text{ProductTopology}(\tau_2, \tau_3), h) \)
proof -
  let \( B = \text{ProductCollection}(\tau_2, \tau_3) \)
  have \( \text{two_top_spaces0}(\tau_1, \text{ProductTopology}(\tau_2, \tau_3), h) \)
  \( B \) \{is a base for\} \( \text{ProductTopology}(\tau_2, \tau_3) \)
  \( \forall W \in B. \ h^{-1}(W) \in \tau_1 \)
proof -
  from \( \text{tau1_is_top} \) \( \text{tau2_is_top} \) \( \text{tau3_is_top} \) \( \text{assms}(1, 2, 5) \)
  show \( \text{two_top_spaces0}(\tau_1, \text{ProductTopology}(\tau_2, \tau_3), h) \)
    using \( \text{vimage}\_\text{prod} \) \( \text{Top}\_1\_4\_T1(1, 3) \) unfolding \( \text{two_top_spaces0_def} \)
    by simp
  from \( \text{tau2_is_top} \) \( \text{tau3_is_top} \) \( \text{assms} \)
    show \( \text{two_top_spaces0}(\tau_1, \text{ProductTopology}(\tau_2, \tau_3), h) \)
    using \( \text{Top}\_1\_4\_T1(2) \) by simp
  from \( \text{tau1_is_top} \) \( \text{assms} \)
    show \( \forall W \in B. \ h^{-1}(W) \in \tau_1 \)
    unfolding \( \text{ProductCollection_def} \) \( \text{IsContinuous_def} \) \( \text{IsATopology_def} \)
    using \( \text{vimage}\_\text{prod} \) by simp
qed
then show thesis by (rule \( \text{two_top_spaces0.Top}\_1\_2\_1\_L5 \))
Having two continuous mappings \( f, g \) we can construct a third one with values in the cartesian product of the codomains of \( f, g \), defined by \( x \mapsto (f(x), g(x)) \). This is essentially the same as \text{cont\_funcs\_prod} but formulated in a way that is sometimes easier to apply. Recall that \( \tau_2 \times_\tau \tau_3 \) is a notation for the product topology of \( \tau_1 \) and \( \tau_2 \).

**Lemma cont\_funcs\_prod1:**

assumes \( \tau_1 \{\text{is a topology}\} \) \( \tau_2 \{\text{is a topology}\} \) \( \tau_3 \{\text{is a topology}\} \) and \( \langle x, p(x) \rangle. x \in \bigcup \tau_1 \in \text{Cont}(\tau_1, \tau_2) \) \( \langle x, q(x) \rangle. x \in \bigcup \tau_1 \in \text{Cont}(\tau_1, \tau_3) \)

shows \( \langle x, \langle p(x), q(x) \rangle \rangle. x \in \bigcup \tau_1 \in \text{Cont}(\tau_1, \tau_2 \times_\tau \tau_3) \)

**Proof -**

let \( X = \bigcup \tau_1 \)
let \( Y = \bigcup \tau_2 \)
let \( Z = \bigcup \tau_3 \)
let \( f = \{\langle x, p(x) \rangle. x \in X\} \)
let \( g = \{\langle x, q(x) \rangle. x \in X\} \)
let \( h = \{\langle x, \langle p(x), q(x) \rangle \rangle. x \in X\} \)

from assms(4,5) have \( f: X \to Y \) and \( g: X \to Z \)

using \text{prod\_fun\_val}(1) using \text{Top\_1\_4\_T1}(3) by simp

moreover have \text{IsContinuous}(\tau_1, \tau_2 \times_\tau \tau_3, h)

**Proof -**

let \( B = \text{ProductCollection}(\tau_2, \tau_3) \)
from assms(1,2,3) have \( h\text{Fun}: h: X \to \bigcup (\tau_2 \times_\tau \tau_3) \)

using \text{prod\_fun\_val}(1) using \text{Top\_1\_4\_T1}(3) by simp

moreover have \( \forall W \in B. h\text{-(W)} \in \tau_1 \)

**Proof -**

\{ fix \( W \) assume \( W \in B \)
then obtain \( U V \) where \( U \in \tau_2 \) \( V \in \tau_3 \) \( W = U \times V \)

unfolding \text{ProductCollection\_def} by auto
have \( \forall x \in X. \langle p(x), q(x) \rangle = \langle f(x), g(x) \rangle \)

**Proof -**

\{ fix \( x \) assume \( x \in X \)
then have \( \langle p(x), q(x) \rangle = \langle f(x), g(x) \rangle \)

using \text{ZF\_fun\_from\_tot\_val1} by simp
\} thus thesis by auto

qed

then have \( h = \{\langle x, \langle f(x), g(x) \rangle \rangle. x \in X\} \) by \text{rule\ set\_comp\_eq}

with assms(1,4,5) \( f: X \to Y \) \( g: X \to Z \) \( U \in \tau_2 \) \( V \in \tau_3 \) \( W = U \times V \)
have \( h\text{-(W)} \in \tau_1 \) using \text{vimage\_prod}(3)

unfolding \text{Cont\_def} \text{IsContinuous\_def} \text{IsATopology\_def} by auto

\} thus thesis by simp

qed

ultimately show thesis by \text{rule\ two\_top\_spaces0\_TopZF\_2\_1\_L5} by auto

qed
ultimately show thesis unfolding Cont_def by simp

qed

Two continuous functions into a Hausdorff space are equal on a closed set. Note that in the lemma below \( f \) is assumed to map \( X_1 \) to \( X_2 \) in the locale, we only need to add a similar assumption for the second function.

**lemma** (in two_top_spaces0) two_fun_eq_closed:
assumes \( g: X_1 \rightarrow X_2 \) \( f \) {is continuous} \( g \) {is continuous} \( \tau_2 \) {is \( T_2 \)}
shows \( \{ x \in X_1. \ f(x) = g(x) \} \) {is closed in} \( \tau_1 \)

**proof** -
let \( D = \{ x \in X_1. \ f(x) = g(x) \} \)
let \( h = \{ \langle x, \langle f(x), g(x) \rangle \rangle. x \in X_1 \} \)
have \( h - (\{ \langle y, y \rangle. y \in X_2 \}) \) {is closed in} \( \tau_1 \)

**proof** -
have two_top_spaces0(\( \tau_1 \),ProductTopology(\( \tau_2, \tau_2 \)),h)

**proof**
from tau1_is_top tau2_is_top show \( \tau_1 \) {is a topology} and ProductTopology(\( \tau_2, \tau_2 \)) {is a topology}
using Top_1_4_T1(1) by auto
from tau1_is_top tau2_is_top fmapAssum assms(1)
show h : \( \bigcup \tau_1 \rightarrow \bigcup \)ProductTopology(\( \tau_2, \tau_2 \))
using vimage_prod(1) Top_1_4_T1(3) by simp

qed
moreover have IsContinuous(\( \tau_1 \),ProductTopology(\( \tau_2, \tau_2 \)),h)

**proof** -
from tau1_is_top tau2_is_top have prod_top_spaces0(\( \tau_1, \tau_2, \tau_2 \))
unfolding prod_top_spaces0_def by simp
with fmapAssum assms(1,2,3) show thesis
using prod_top_spaces0.cont_functs_prod by simp

qed
moreover from tau2_is_top assms(4)
have \( \{ \langle y, y \rangle. y \in X_2 \} \) {is closed in} ProductTopology(\( \tau_2, \tau_2 \))
using t2_iff_diag_closed by simp
ultimately show thesis using two_top_spaces0.TopZF_2_1_L1
by simp

qed

Closure of an image of a singleton by a relation in \( X \times Y \) is contained in the image of this singleton by the closure of the relation (in the product topology). Compare the proof of Metamath’s theorem with the same name.

**lemma** imasncls:
assumes \( T \) {is a topology} \( S \) {is a topology} \( R \subseteq (\bigcup T) \times (\bigcup S) \ x \in \bigcup T \)
shows \( \text{Closure}(R \{ x \}, S) \subseteq \text{Closure}(R, T \times S) \{ x \} \)

**proof** -
let \( X = \bigcup T \)
let \( Y = \bigcup S \)
let \( f = \{ \langle y, \langle x, y \rangle \rangle. \ y \in Y \} \)

from axsms(3) have \( R\{x\} = f-(R) \) by blast
hence \( \text{Closure}(R\{x\}, S) = \text{Closure}(f-(R), S) \) by simp
also have \( \text{Closure}(f-(R), S) \subseteq f-(\text{Closure}(R, T \times S)) \)
proof -

from axsms(1,2,4) have \( f \in \text{Cont}(S, T \times S) \)
  using const_cont_sp id_cont_sp cont_funcs_prod1 by simp
with axsms(1,2,3) have
  \( \text{two_top_spaces0}(S, T \times S, f) \text{ IsContinuous}(S, T \times S, f) \ R \subseteq \bigcup (T \times S) \)
unfolding Cont_def two_top_spaces0_def using Top_1_4_T1(1,3) by auto
then show \( \text{Closure}(f-(R), S) \subseteq f-(\text{Closure}(R, T \times S)) \)
  using two_top_spaces0.im_cl_in_cl_im by simp
qed
also
have \( \text{Closure}(R, T \times S) \subseteq X \times Y \)
proof -

from axsms(1,2,3) have \( \text{topology0}(T \times S) \ R \subseteq \bigcup (T \times S) \)
unfolding topology0_def using Top_1_4_T1(1,3) by auto
then have \( \text{Closure}(R, T \times S) \subseteq \bigcup (T \times S) \) by (rule topology0.Top_3_L11)
with axsms(1,2) show thesis using Top_1_4_T1(3) by simp
qed
with axsms(4) have \( f-(\text{Closure}(R, T \times S)) = \text{Closure}(R, T \times S)\{x\} \) by blast
finally show \( \text{Closure}(R\{x\}, S) \subseteq \text{Closure}(R, T \times S)\{x\} \) by simp
qed

68.6 Pasting lemma

The classical pasting lemma states that if \( U_1, U_2 \) are both open (or closed) and a function is continuous when restricted to both \( U_1 \) and \( U_2 \) then it is continuous when restricted to \( U_1 \cup U_2 \). In this section we prove a generalization statement stating that the set \( \{ U \in \tau_1 | f|_U \text{ is continuous} \} \) is a topology.

A typical statement of the pasting lemma uses the notion of a function restricted to a set being continuous without specifying the topologies with respect to which this continuity holds. In \text{two_top_spaces0} context the notation \( g \{ \text{is continuous} \} \) means continuity with respect to topologies \( \tau_1, \tau_2 \).

The next lemma is a special case of \text{partial_fun_cont} and states that if for some set \( A \subseteq X_1 = \bigcup \tau_1 \) the function \( f|_A \) is continuous (with respect to \( (\tau_1, \tau_2) \)), then \( A \) has to be open. This clears up terminology and indicates why we need to pay attention to the issue of which topologies we talk about when we say that the restricted (to some closed set for example) function is continuos.

lemma (in \text{two_top_spaces0}) restriction_continuous1:
assumes \( A1: A \subseteq X_1 \) and \( A2: \text{restrict}(f,A) \) {is continuous}
shows \( A \in \tau_1 \)

proof -

from assms have two_top_spaces1(\( \tau_1, \tau_2 \)) and
\( \text{restrict}(f,A):A \rightarrow X_2 \) and \( \text{restrict}(f,A) \) {is continuous}
using tau1_is_top tau2_is_top two_top_spaces1_def fmapAssum restrict_fun
by auto

then show thesis using two_top_spaces1.partial_fun_cont by simp

qed

If a function is continuous on each set of a collection of open sets, then it is continuous on the union of them. We could use continuity with respect to the relative topology here, but we know that on open sets this is the same as the original topology.

lemma (in two_top_spaces0) pasting_lemma1:
assumes \( A1: M \subseteq \tau_1 \) and \( A2: \forall U \in M. \text{restrict}(f,U) \) {is continuous}
shows \( \text{restrict}(f,\bigcup M) \) {is continuous}

proof -

\{ fix \( V \) assume \( V \in \tau_2 \)
from \( A1 \) have \( \bigcup M \subseteq X_1 \) by auto
then have \( \text{restrict}(f,\bigcup M)-(V) = f-(V) \cap (\bigcup M) \)
using func1_2_L1 fmapAssum by simp
also have \( \ldots = \bigcup \{f-(V) \cap U. U \in M\} \) by auto
finally have \( \text{restrict}(f,\bigcup M)-(V) = \bigcup \{f-(V) \cap U. U \in M\} \) by simp
moreover have \( \{f-(V) \cap U. U \in M\} \in \text{Pow}(\tau_1) \)
proof -
\{ fix \( W \) assume \( W \in \{f-(V) \cap U. U \in M\} \)
then obtain \( U \) where \( U \in M \) and \( I: W = f-(V) \cap U \) by auto
with \( A2 \) have \( \text{restrict}(f,U) \) {is continuous} by simp
with \( \{V \in \tau_2\} \) have \( \text{restrict}(f,U)-(V) \in \tau_1 \)
using IsContinuous_def by simp
moreover from \( \{\bigcup M \subseteq X_1\} \) and \( \{U \in M\} \)
have \( \text{restrict}(f,\bigcup M)-(V) = f-(V) \cap U \)
using fmapAssum func1_2_L1 by blast
ultimately have \( f-(V) \cap U \in \tau_1 \) by simp
with \( I \) have \( W \in \tau_1 \) by simp
\} then show thesis by auto
qed

then have \( \bigcup \{f-(V) \cap U. U \in M\} \in \tau_1 \)
using tau1_is_top IsATopology_def by auto
ultimately have \( \text{restrict}(f,\bigcup M)-(V) \in \tau_1 \)
by simp
\} then show thesis using IsContinuous_def by simp
qed

If a function is continuous on two sets, then it is continuous on intersection.

lemma (in two_top_spaces0) cont_inter_cont:
assumes \( A1: A \subseteq X_1 \) \( B \subseteq X_1 \) and
A2: restrict(f,A) \{is continuous\} restrict(f,B) \{is continuous\} 
shows restrict(f,A \cap B) \{is continuous\}

proof -
{ fix V assume \( V \in \tau_2 \) with asms have
    restrict(f,A)-(V) = f-(V) \cap A  restrict(f,B)-(V) = f-(V) \cap B and
    restrict(f,A)-(V) \in \tau_1 \ and \ restrict(f,B)-(V) \in \tau_1
    using func1_2_L1 fmapAssum IsContinuous_def by auto
then have (restrict(f,A)-(V)) \cap (restrict(f,B)-(V)) = f-(V) \cap (A\cap B)
    by auto
moreover from A2 \( \langle V \in \tau_2 \rangle \) have
    restrict(f,A)-(V) \in \tau_1 \ and \ restrict(f,B)-(V) \in \tau_1
    using IsContinuous_def by auto
then have (restrict(f,A)-(V)) \cap (restrict(f,B)-(V)) \in \tau_1
    using tau1_is_top IsATopology_def by simp
moreover from A1 have \((A\cap B) \subseteq X_1\) by auto
then have restrict(f,A\cap B)-(V) = f-(V) \cap (A\cap B)
    using func1_2_L1 fmapAssum by simp
ultimately have restrict(f,A\cap B)-(V) \in \tau_1 by simp
} then show thesis using IsContinuous_def by auto
qed

The collection of open sets \( U \) such that \( f \) restricted to \( U \) is continuous, is a topology.

theorem (in two_top_spaces0) pasting_theorem:
shows \{U \in \tau_1. restrict(f,U) \{is continuous\}\} \{is a topology\}

proof -
let \( T = \{U \in \tau_1. restrict(f,U) \{is continuous\}\} \)
have \( \forall M \in \text{Pow}(T). \bigcup M \in T \)
proof
fix M assume \( M \in \text{Pow}(T) \)
then have restrict(f,\bigcup M) \{is continuous\}
    using pasting_lemma by auto
with \( \langle M \in \text{Pow}(T) \rangle \) show \( \bigcup M \in T \)
    using tau1_is_top IsATopology_def by auto
qed
moreover have \( \forall U \in T. \forall V \in T. U \cap V \in T \)
    using cont_inter_cont tau1_is_top IsATopology_def by auto
ultimately show thesis using IsATopology_def by simp
qed

0 is continuous.

corollary (in two_top_spaces0) zero_continuous:
shows 0 \{is continuous\}

proof -
let \( T = \{U \in \tau_1. restrict(f,U) \{is continuous\}\} \)
have \( T \{is a topology\} \) by (rule pasting_theorem)
then have 0 \in T by (rule empty_open)

hence restrict(f,0) {is continuous} by simp
moreover have restrict(f,0) = 0 by simp
ultimately show thesis by simp
qed

end

69 Rings - Zariski Topology - maps

theory Ring_Zariski_ZF_3 imports Ring_Zariski_ZF Ring_ZF_3 Topology_ZF_2

begin

lemma (in ring_homo) spectrum_surj:
defines g ≡ λu∈target_ring.Spec. f-u
assumes f∈surj(R,S)
shows g: target_ring.Spec → V(ker)
proof-

have g: target_ring.Spec → {f-u. u∈target_ring.Spec} using lam_funtype
  unfolding g_def by auto
moreover

fix t assume t∈{f-u. u∈target_ring.Spec}
then obtain u where u:u:target_ring.Spec t=f-u by auto
from u(1) have u2:u∈primeR, unfolding target_ring.Spec_def by auto
then have (f-u)∈primeR, using preimage_prime_ideal_surj
  assms(2) by auto
moreover

then have f-u ∈ origin_ring.ideals
  unfolding origin_ring.primeIdeal_def
  using origin_ring.ideal_dest_subset by auto
ultimately have f-u ∈ origin_ring.Spec unfolding origin_ring.Spec_def
  by auto
moreover from u2 have 0∈u unfolding
  target_ring.primeIdeal_def using target_ring.ideal_dest_zero
  by auto
then have {0} ⊆ u by auto
then have f-{0} ⊆ f-u by auto
moreover have f-u ⊆ R using func1_1_L15[OF surj_is_fun[OF assms(2)]] by auto
ultimately have f-u ∈ origin_ring.closeBasic(f-{0})
  using origin_ring.closeBasic[of f-{0}] by force
  with u(2) have t∈origin_ring.closeBasic(f-{0}) by auto
}
then have {f-u. u∈target_ring.Spec} ⊆ origin_ring.closeBasic(f-{0})
  by auto
ultimately show thesis using func1_1_L1B by auto
qed
lemma (in ring_homo) spectrum_surj_bij:
defines g ≡ λu∈target_ring.Spec. f-u
assumes f∈surj(R,S)
shows g∈bij(target_ring.Spec, V(ker))
proof-
{
  fix s t assume st:s∈target_ring.Spec t∈target_ring.Spec
gs = gt
then have f-s = f-t using beta unfolding g_def by auto
then have f(f-s) = f(f-t) by auto
moreover from st(1,2) have s ⊆ S t ⊆ S
  unfolding target_ring.Spec_def origin_ring.Spec_def
  by auto
moreover note assms(2) st(1,2)
ultimately have s=t
  using surj_image_vimage
  unfolding target_ring.Spec_def origin_ring.Spec_def
  by auto
}
then have g∈inj(target_ring.Spec, origin_ring.closeBasic(f-\{0\}S))
  unfolding inj_def using spectrum_surj assms(2)
  unfolding g_def by auto
moreover
{
  fix t assume t:t∈origin_ring.closeBasic(f-\{0\}S)
then have tt:t∈origin_ring.Spec f-\{0\}S ⊆ t
  using origin_ring.closeBasic_def func1_1_L6A[OF surj_is_fun]
  assms(2) by auto
{
  fix y assume y:y∈f-(ft)
then have y:y∈R fy∈ft using func1_1_L15
  surj_is_fun[OF assms(2)] by auto
from y(2) obtain x where x:x∈t fy = fx
  using func_imagedef[OF surj_is_fun]
  assms(2) tt(1) unfolding origin_ring.Spec_def by auto
from x(2) have (fy)-S(fx) = (fx)-S(fx) by auto
then have (fy)-S(fx) = 0S using target_ring.Ring_ZF_1_L3(7)
  apply_type[OF surj_is_fun] assms(2) x(1) tt(1)
  unfolding origin_ring.Spec_def by auto
then have f(y-Rx) = 0S using homomor_dest_subs
  x(1) tt(1) y(1) unfolding origin_ring.Spec_def by auto
moreover
from x(1) tt(1) have x∈R unfolding origin_ring.Spec_def by auto
with y(1) have y-Rx ∈R using origin_ring.Ring_ZF_1_L4(2) by auto
ultimately have y-Rx ∈ f-\{0\}S using func1_1_L15
  surj_is_fun[OF assms(2)] by auto
then have y-Rx ∈ t using tt(2) by auto moreover
have t∈R, using tt(1) unfolding origin_ring.Spec_def by auto
ultimately have x+R(y-Rx) ∈t using x(1)
  origin_ring.ideal_dest_sum by auto

868
then have \( y \in t \) using \( \text{origin\_ring.Ring\_ZF\_2\_L1A(5)} \)
\[
y(1) \prec x \in R
\]
by auto
}
then have \( f-(ft) = t \) using \( \text{func1\_1\_L9[of f R S t]} \)
\( \text{surj\_is\_fun[of assms(2)]} \) \( tt(1) \) unfolding \( \text{origin\_ring.Spec\_def} \)
by auto moreover
then have \( (ft) \subseteq R_t \) using \( \text{prime\_ideal\_quot\_3[of ft]} \)
\( \text{assms(2)} \) \( tt(1) \) unfolding \( \text{origin\_ring.Spec\_def} \)
using \( \text{image\_ideal\_surj[of t]} \) \( \text{origin\_ring.primeIdeal\_def[of t]} \)
by auto
then have \( (ft) \subseteq R_t \)
using \( \text{target\_ring.openBasic[of ft]} \)
by auto
then have \( (ft) : \text{target\_ring.Spec} \)
unfolding \( \text{target\_ring.Spec\_def} \)
using \( \text{target\_ring.ideal\_dest\_subset} \)
by auto morever
then have \( f-(ft) = g(ft) \) unfolding \( g\_def \)
using \( \text{beta} \) by auto
ultimately have \( g(ft) = t \) \( \text{(ft) : target\_ring.Spec} \)
by auto
ultimately show \( g : \text{bij(target\_ring.Spec, V(ker))} \)
unfolding \( \text{bij\_def surj\_def inj\_def} \) by auto
qed

definition (in ring_homo) top_origin \( (\tau_o) \) where
\( \text{top\_origin} \equiv \{ \text{origin\_ring.openBasic(J)} . J \in \text{origin\_ring.ideals} \} \)
definition (in ring_homo) top_target \( (\tau_t) \) where
\( \text{top\_target} \equiv \{ \text{target\_ring.openBasic(J)} . J \in \text{target\_ring.ideals} \} \)
definition (in ring_homo) spec_cont where
\( \text{spec\_cont(h)} \equiv \text{IsContinuous}(\tau_t, \tau_o, h) \)

lemma (in ring_homo) spectrum_surj_cont:
defines \( g \equiv \lambda u \in \text{target\_ring.Spec}. f-u \)
assumes \( f \in \text{surj(R,S)} \)
shows \( \text{IsContinuous}(\tau_f, \tau_o \text{ (restricted to)}\{V(ker)\}, g) \)
unfolding \( \text{IsContinuous\_def top\_target\_def RestrictedTo\_def top\_origin\_def} \)
proof(safe)
fix \( x \) assume \( \text{ass: x \in R, x \subseteq R} \)
have \( \text{origin\_ring.openBasic(x)} = \{ u \in \text{origin\_ring.Spec. \neg(x \subseteq u)} \} \)
unfolding \( \text{origin\_ring.openBasic\_def[of assms(2)]} \) by auto
have \( g-(\text{origin\_ring.openBasic(x)}) = \{ t \in \text{target\_ring.Spec. gt \in origin\_ring.openBasic(x)} \} \)
using \( \text{spectrum\_surj assms(2)} \) unfolding \( g\_def \)
using \( \text{func1\_1\_L15} \) by auto
then have \( G = \{ \text{origin\_ring.openBasic(x)} \} = \{ t \in \text{target\_ring.Spec. f-t \in origin\_ring.openBasic(x)} \} \)
using beta unfolding g_def by auto
have (fx)tR, using image_ideal_surj assms(2) ass(1) by auto
then have H:fx target_ring.ideals using
  target_ring.ideal_dest_subset by auto
then have F:target_ring.openBasic(fx) = \{t \in target_ring.Spec. \neg (fx \subseteq t)\}
  using target_ring.openBasic_def by auto

  \{
  fix s assume s \in \{t \in target_ring.Spec. f - t \in origin_ring.openBasic(x)\}
  then have s:s \in target_ring.Spec f - s \in origin_ring.openBasic(x)
    by auto
  from this(2) have E:f - s \in origin_ring.Spec \neg (x \subseteq f - s)
    using origin_ring.openBasic_def ass(1) origin_ring.ideal_dest_subset
    by auto
  \{
  assume fx \subseteq s
  then have f-(fx) \subseteq f - s by auto
  then have x \subseteq f - s using func1_1_L9[of surj_is_fun]
    assms(2) ass(1) origin_ring.ideal_dest_subset by force
  with E(2) have False by auto
  \}
  then have \neg (fx \subseteq s) by auto
  with s(1) have s \in \{t \in target_ring.Spec. \neg (fx \subseteq t)\}
    by auto
  \}
then have \{t \in target_ring.Spec . f - t \in origin_ring.openBasic(x)\}
  \subseteq \{t \in target_ring.Spec. \neg (fx \subseteq t)\}
  by auto moreover

  \{
  fix s assume s \in \{t \in target_ring.Spec. \neg (fx \subseteq t)\}
  then have s:s \in target_ring.Spec \neg (fx \subseteq s) by auto
  have origin_ring.openBasic(x) = \{t \in origin_ring.Spec. \neg (x \subseteq t)\}
    using origin_ring.openBasic_def ass(1) origin_ring.ideal_dest_subset
    by auto
  \{
  assume x \subseteq f - s
  then have fx \subseteq f(f - s) by auto
  then have fx \subseteq s using surj_image_vimage
    assms(2) s(1) unfolding target_ring.Spec_def
    target_ring.primeIdeal_def target_ring.ideal_dest_subset
    by auto
  with s(2) have False by auto
  \}
  then have \neg (x \subseteq f - s) by auto
  moreover
  from s(1) have (f - s) \in \mu_R, unfolding target_ring.Spec_def
    using preimage_prime_ideal_surj assms(2) by auto
  then have (f - s) \in origin_ring.Spec unfolding origin_ring.Spec_def
    origin_ring.primeIdeal_def using origin_ring.ideal_dest_subset
  870
ultimately have \( f - s \in \text{origin_ring.openBasic}(x) \)
using \text{origin_ring.openBasic_def} \( \text{ass}(1) \)
\text{origin_ring.ideal_dest_subset} by auto
then have \( s \in \{ t \in \text{target_ring.Spec} : f - t \in \text{origin_ring.openBasic}(x) \} \)
using \( s(1) \) by auto
}
then have \( \{ t \in \text{target_ring.Spec} : \neg f \ x \subseteq t \} \subseteq \{ t \in \text{target_ring.Spec} : f - t \in \text{origin_ring.openBasic}(x) \} \)
by auto
ultimately have \( \{ t \in \text{target_ring.Spec} : \neg f \ x \subseteq t \} = \{ t \in \text{target_ring.Spec} : f - t \in \text{origin_ring.openBasic}(x) \} \)
by auto
with \( F \) have \text{target_ring.openBasic}(fx) = \{ t \in \text{target_ring.Spec} : f - t \in \text{origin_ring.openBasic}(x) \} 
by auto
with \( G \) have \( T : \text{target_ring.openBasic}(fx) = g - \text{origin_ring.openBasic}(x) \)
by auto
have \( g - (\text{origin_ring.closeBasic}(f - \{0_S\})) = \{ t \in \text{target_ring.Spec} : gt \in \text{origin_ring.closeBasic}(f - \{0_S\}) \} \)
using \text{spectrum_surj} \( \text{assms}(2) \) unfolding \( g \_ \text{def} \)
using func1_1_L15 by auto
then have \( g - (\text{origin_ring.closeBasic}(f - \{0_S\})) = \{ t \in \text{target_ring.Spec} : f - t \in \text{origin_ring.closeBasic}(f - \{0_S\}) \} \)
using beta unfolding \( g \_ \text{def} \) by auto
then have \( E : g - (\text{origin_ring.closeBasic}(f - \{0_S\})) = \{ t \in \text{target_ring.Spec} : f - t \in \{ q \in \text{origin_ring.Spec} : (f - \{0_S\}) \subseteq q \} \} \)
unfolding \text{origin_ring.closeBasic_def}[OF func1_1_L3][OF \text{surj_is_fun}[OF \text{assms}(2)]] by auto

\[
\begin{align*}
\text{fix} & \ s & \text{assume} & \ s \in \text{target_ring.openBasic}(fx) \\
\text{with} & \ E & \text{have} & \ s \in \text{target_ring.Spec} \ \neg (fx \subseteq s) \\
& & & \text{using} \text{target_ring.openBasic_def} \ \text{func1_1_L6(2)} \ \text{surj_is_fun}[OF \text{assms}(2)] \\
& & & \text{by auto} \\
& & & \text{from} \ this(1) & \text{have} & \ gs \in \text{origin_ring.closeBasic}(f - \{0_S\}) \ \text{using} \text{spectrum_surj}[OF \text{assms}(2)] \\
& & & & & \text{apply_type}[\text{of g target_ring.Spec} \ \lambda u. \ \text{origin_ring.closeBasic}(f - \{0_S\})] \ \text{unfolding} \ g \_ \text{def} \\
& & & & & \text{by auto} \\
& & & & & \text{with ss(1) have} & \ f - s \in \text{origin_ring.closeBasic}(f - \{0_S\}) \ \text{using} \ \text{beta} \\
& & & & & & \ \text{unfolding} \ g \_ \text{def} \ \text{by auto} \\
& & & & & & \ \text{moreover} \\
& & & & & & \text{from ss(1) have} & \ s \in R \ \text{unfolding} \ \text{target_ring.Spec_def} \\
& & & & & & \ \text{target_ring.primeIdeal_def} \ \text{by auto} \\
& & & & & & \ \text{then have} & 0_S \ \in \ s \ \text{using} \ \text{target_ring.ideal_dest_zero} \ \text{by auto} \\
& & & & & & \ \text{then have} & \{0_S\} \subseteq s \ \text{by auto} \\
& & & & & & \ \text{moreover} \\
& & & & & & \ \text{have} & f - \{0_S\} \subseteq R \ \text{using} \ \text{func1_1_L15}[OF} \\
\end{align*}
\]
ultimately have \( f - s \in \{ q \in \text{origin\_ring} . \text{Spec.} \ (f - \{0_S\}) \subseteq q \} \)
using \( \text{origin\_ring} . \text{closeBasic\_def} \) by auto
then have \( s \in \{ t \in \text{target\_ring} . \text{Spec.} \ f - t \in \{ q \in \text{origin\_ring} . \text{Spec.} \ (f - \{0_S\}) \subseteq q \} \}
using \( \text{ss}(1) \) by auto
then have \( s \in \text{g} - (\text{origin\_ring} . \text{closeBasic}(f - \{0_S\})) \)
using \( \text{E} \) by auto

ultimately have \( \text{target\_ring} . \text{openBasic}(fx) \subseteq \text{g} - (\text{origin\_ring} . \text{closeBasic}(f - \{0_S\})) \) by auto
then have \( \text{g} - (\text{origin\_ring} . \text{closeBasic}(f - \{0_S\})) \cap \text{target\_ring} . \text{openBasic}(fx)
= \text{target\_ring} . \text{openBasic}(fx) \)
by auto
with \( T \) have \( \text{g} - (\text{origin\_ring} . \text{closeBasic}(f - \{0_S\})) \cap \text{g} - \text{origin\_ring} . \text{openBasic}(x)
= \text{g} - (\text{origin\_ring} . \text{closeBasic}(f - \{0_S\})) \cap \text{origin\_ring} . \text{openBasic}(x) \)
using \( \text{invim\_inter\_inter\_invim}[\text{OF spectrum\_surj}[\text{OF assms}(2)]] \)
unfolding \( \text{g\_def} \) by auto
ultimately have \( \text{g} -
(\text{origin\_ring} . \text{closeBasic}(f - \{0_S\})) \cap \text{origin\_ring} . \text{openBasic}(x) = \text{target\_ring} . \text{openBasic}(fx) \)
by auto
with \( H \) show \( \text{g} -
(\text{origin\_ring} . \text{closeBasic}(f - \{0_S\})) \cap \text{origin\_ring} . \text{openBasic}(x) \subseteq
\text{RepFun}(\text{target\_ring} . \text{ideals}, \text{target\_ring} . \text{openBasic}) \)
by auto
qed

lemma (in ring_homo) spectrum\_surj\_open:
defines \( g \equiv \lambda u \in \text{target\_ring} . \text{Spec.} \ f - u \)
assumes \( f \in \text{surj}(R,S) \)
shows \( \forall U \in \tau_I . \ gU \in \tau_o \ {\text{restricted to} V(\ker)} \)
proof
fix \( U \) assume \( U : U \in \tau_I \)
then obtain \( I \) where \( I : I \subseteq R, I \subseteq S \)
\( U \in \text{target\_ring} . \text{openBasic}(I) \)
unfolding \( \text{top\_target\_def} \)
by auto
from \( I(3) \) have \( \text{sub}\_U \subseteq \text{target\_ring} . \text{Spec} \)
using \( \text{target\_ring} . \text{openBasic\_def}[\text{OF I}(2)] \) by auto
\{ 
fix \( t \) assume \( t : t \in gU \)
then obtain \( u \) where \( u : u \in U \ t = gu \)
using \( \text{func\_imagedef} \text{ spectrum\_surj}[\text{OF assms}(2)] \) \( \text{sub} \)
unfolding \( \text{g\_def} \) by auto
then have \( t : t = f - u \) using \( \text{beta} \text{ sub} \)
unfolding \( \text{g\_def} \) by auto

872
with \( \text{sub } u(1) \) have \( t \cap \mathbb{R}_0 \) unfolding \( \text{target_ring.Spec_def} \)
using \( \text{preimage_prime_ideal_surj[OF _ assms(2), of u]} \)
by auto

then have \( p : t \cap \text{origin_ring.Spec} \) unfolding \( \text{origin_ring.Spec_def} \)
unfolding \( \text{origin_ring.primeIdeal_def} \)
using \( \text{origin_ring.ideal_dest_subset} \) by auto

from \( u(1) \) \( I(2,3) \) have \( Iu : \neg (I \subseteq u) \) using \( \text{target_ring.openBasic_def} \)
by auto

\[
\begin{align*}
\text{assume } f-I & \subseteq t \\
\text{then have } f(f-I) & \subseteq ft \text{ by auto} \\
\text{then have } I & \subseteq ft \text{ using } \text{surj_image_vimage[OF assms(2)] I(2)} \text{ by auto} \\
\text{with } t & \text{ have } I \subseteq f(f-u) \text{ by auto} \\
\text{moreover from } u(1) \text{ sub have } u \subseteq S \\
\text{unfolding } \text{target_ring.Spec_def by auto} \\
\text{ultimately have } I & \subseteq f(f-u) \text{ using } \text{surj_image_vimage[OF assms(2)]} \\
\text{by auto} \\
\text{with } Iu \text{ have False by auto} \\
\end{align*}
\]

then have \( \neg(f-I \subseteq t) \) by auto

with \( p \) have \( t \in \text{origin_ring.openBasic}(f-I) \)
using \( \text{origin_ring.openBasic_def func1_1_L6A[OF surj_is_fun]} \)
assms(2) by auto

then have \( gU \subseteq \text{origin_ring.openBasic}(f-I) \) by auto

moreover
\[
\begin{align*}
\text{fix } t \text{ assume } t & : t \in \text{origin_ring.openBasic}(f-I) \text{ t} \in \mathbb{V}(\text{ker}) \\
\text{have } f-I & \subseteq R \text{ using } \text{func1_1_L6A[OF surj_is_fun]} \\
\text{assms(2)} \text{ by auto} \\
\text{with } t & \text{ have } p : t \in \text{origin_ring.Spec } \neg(f-I \subseteq t) \\
\text{using } \text{origin_ring.openBasic_def by auto} \\
\text{from } t(2) & \text{ have } kt : \text{ker} \subseteq t \text{ using } \text{origin_ring.closeBasic_def} \\
\text{func1_1_L3 f_is_fun by auto} \\
\end{align*}
\]

\[
\begin{align*}
\text{fix } x \text{ assume } x & \in f-(ft) \\
\text{then have } t : f(x) \in ft x \in R \text{ using } \text{func1_1_L15} \\
\text{surj_is_fun[OF assms(2)]} \text{ by auto} \\
\text{then obtain } s \text{ where } s : fx = fs s \in t \text{ using } \\
\text{func_imagedef[OF surj_is_fun[OF assms(2)]]} \\
p(1) \text{ unfolding } \text{origin_ring.Spec_def by auto} \\
\text{from } s(2) & \text{ have } ss : s \in R \text{ using } p(1) \\
\text{unfolding } \text{origin_ring.Spec_def by auto} \\
\text{from } s(1) & \text{ have } (fx) -_S (fs) = 0_S \text{ using } \\
\text{target_ring.Ring_ZF_1_L3(7)[OF apply_type[OF} \\
surj_is_fun[OF assms(2)]]} \\
t(2))] \text{ by auto} \\
\text{then have } f(x -_S hs) = 0_S \text{ using } \text{homomor_dest_subs} \\
t(2) ss \text{ by auto} \text{ moreover} \\
\text{from } t(2) & \text{ ss have } x -_S hs \in R \text{ using } \text{origin_ring.Ring_ZF_1_L4(2)} \text{ by}
\end{align*}
\]
ultimately have $x - R s \in f^{-\{0_R\}}$ using $\text{func1}_1\_L15$
$\text{surj_is_fun}[\text{OF assms}(2)]$ by auto
then have $x - R s \in t$ using $kt$ by auto
then have $s + R (x - R s) \in t$
  using $\text{origin_ring.ideal_dest_sum}$
$\text{s}(2) \text{ p}(1)$ unfolding $\text{origin_ring.Spec_def}$ by auto
then have $x - R s \in t$ using $\text{origin_ring.Ring_ZF}_2\_L1A(5)$
$\text{ss t}(2)$ by auto
}
then have $\text{eq:} f^{-\{f t\}} = t$
  using $\text{func1}_1\_L9[\text{OF surj_is_fun}[\text{OF assms}(2)]$ origin_ring.ideal_dest_subset[of t]] \text{ p}(1)$ unfolding origin_ring.Spec_def
origin_ring.primeIdeal_def by auto
then have $(f t) \triangleleft R t \implies (ft) \triangleleft p R t$
  using $\text{prime_ideal_quot_3}[\text{OF ft}] \text{ assms}(2)$
$\text{p}(1)$ unfolding $\text{origin_ring.Spec_def}$ by auto
then have $\text{id:} (ft) \triangleleft p R t (f t) \triangleleft R t$
  using $\text{image_ideal_surj}$
$\text{p}(1) \text{ assms}(2)$ unfolding $\text{origin_ring.Spec_def}$ by auto
{
  assume $I \subseteq ft$
  then have $f - I \subseteq f^{-\{ft\}}$ by auto
  with $\text{eq}$ have $f - I \subseteq t$ by auto
  with $\text{p}(2)$ have False by auto
}
then have $\neg (I \subseteq ft)$ by auto
then have $ft \in \text{target_ring.openBasic}(I)$
  using $\text{id}$ target_ring.ideal_dest_subset unfolding $\text{target_ring.openBasic_def}[\text{OF I}(2)]$
  target_ring.Spec_def by auto
then have $q : ft \in U$ using $I(3)$ by auto
then have $q : ft \in \text{target_ring.Spec}$ using $\text{sub}$ by auto
from $q$ have $g(ft) \in gU$ using $\text{func1}_1\_L15D[\text{OF bij_is_fun}$
  $\text{OF spectrum_surj_bij}[\text{OF assms}(2)]]$, of ft $U$
  unfolding $g$ def using $\text{sub}$ by auto
then have $f^{-\{ft\}} \in gU$ using $\text{beta}[\text{OF ft}$
target_ring.Spec $\lambda o. f - o]$ $q2$
  unfolding $g$ def by auto
with $\text{eq}$ have $t \in gU$ by auto
}
then have $V(\ker) \cap D(f - I) \subseteq gU$ by auto
ultimately have $V(\ker) \cap D(f - I) = gU$
  using $\text{func1}_1\_L6(2) [\text{OF bij_is_fun}[\text{OF}$
spectrum_surj_bij$[\text{OF assms}(2)]]$, of $U$
  unfolding $g$ def by blast
moreover
from $I(1)$ have $(f^{-\{I\}}) \subseteq R$ and $(f^{-\{I\}}) \subseteq R$
  using $\text{preimage_ideal}(2)$ origin_ring.ideal_dest_subset by simp_all

874
then have \( V(\ker) \cap D(f^- (I)) \in \{V(\ker) \cap A . A \in \tau_o\} \)

unfolding top_origin_def by auto

ultimately show \( gU : \tau_o \{\text{restricted to}\} V(\ker) \)

unfolding RestrictedTo_def by auto

qed

A quotient ring has a spectrum homeomorphic to a closed subspace of the spectrum of the base ring. Specifically, the closed subspace associated to the ideal by which we quotient.

corollary (in ring homo) surj_homeomorphism:
assumes \( f \in \text{surj}(R, S) \)
defines \( g \equiv \lambda u \in \text{target_ring.Spc.} f - u \)
shows \( \text{IsAhomeomorphism}(\tau_t, \tau_o \{\text{restricted to}\} V(\ker) , g) \)

proof-

have \( \bigcup (\tau_o \{\text{restricted to}\} V(\ker)) = \text{origin_ring.Spc} \cap V(\ker) \)
unfolding top_origin_def RestrictedTo_def using origin_ring.total_spec by auto

then have \( \bigcup (\tau_o \{\text{restricted to}\} V(\ker)) = V(\ker) \)
using origin_ring.closeBasic_def func1_1_L3 f_is_fun by auto moreover

have \( \bigcup \tau_t = \text{target_ring.Spc} \)
unfolding top_target_def using target_ring.total_spec by auto

ultimately show thesis using bij_cont_open_homeo[of \( g \), \( \tau_t \), \( \tau_o \{\text{restricted to}\} V(\ker) \) ]
spectrum_surj_bij[OF assms(1)] spectrum_surj_open[OF assms(1)]
spectrum_surj_cont[OF assms(1)]
unfolding g_def by auto

qed

end

70 Topology 3

theory Topology_ZF_3 imports Topology_ZF_2 FiniteSeq_ZF

begin

Topology_ZF_1 theory describes how we can define a topology on a product of two topological spaces. One way to generalize that is to construct topology for a cartesian product of \( n \) topological spaces. The cartesian product approach is somewhat inconvenient though. Another way to approach product topology on \( X^n \) is to model cartesian product as sets of sequences (of length \( n \)) of elements of \( X \). This means that having a topology on \( X \) we want to define a topology on the space \( n \to X \), where \( n \) is a natural number (recall that \( n = \{0, 1, \ldots, n-1\} \) in ZF). However, this in turn can be done more generally by defining a topology on any function space \( I \to X \), where \( I \) is any set of indices. This is what we do in this theory.
70.1 The base of the product topology

In this section we define the base of the product topology.

Suppose \( X = I \to \bigcup T \) is a space of functions from some index set \( I \) to the carrier of a topology \( T \). Then take a finite collection of open sets \( W : N \to T \) indexed by \( N \subseteq I \). We can define a subset of \( X \) that models the cartesian product of \( W \).

**definition**

\[ \text{FinProd}(X, W) \equiv \{ x \in X. \forall i \in \text{domain}(W). x(i) \in W(i) \} \]

Now we define the base of the product topology as the collection of all finite products (in the sense defined above) of open sets.

**definition**

\[ \text{ProductTopBase}(I, T) \equiv \bigcup_{N \in \text{FinPow}(I)} \{ \text{FinProd}(I \to \bigcup T, W). W \in N \to T \} \]

Finally, we define the product topology on sequences. We use the "Seq" prefix although the definition is good for any index sets, not only natural numbers.

**definition**

\[ \text{SeqProductTopology}(I, T) \equiv \{ \bigcup B. B \in \text{Pow}(\text{ProductTopBase}(I, T)) \} \]

Product topology base is closed with respect to intersections.

**lemma** prod_top_base_inter:

assumes \( A1: T \) is a topology \( \) and \( A2: U \in \text{ProductTopBase}(I, T) \) \( V \in \text{ProductTopBase}(I, T) \)

shows \( U \cap V \in \text{ProductTopBase}(I, T) \)

**proof** -

let \( X = I \to \bigcup T \)

from \( A2 \) obtain \( N_1, W_1, N_2, W_2 \) where

I: \( N_1 \in \text{FinPow}(I) \) \( W_1 : N_1 \to T \) \( U = \text{FinProd}(X, W_1) \) and

II: \( N_2 \in \text{FinPow}(I) \) \( W_2 : N_2 \to T \) \( V = \text{FinProd}(X, W_2) \)

using ProductTopBase_def by auto

let \( N_3 = N_1 \cup N_2 \)

let \( W_3 = \{ i, \text{if } i \in N_1 - N_2 \text{ then } W_1(i) \) \( \text{else if } i \in N_2 - N_1 \text{ then } W_2(i) \) \( \text{else } (W_1(i)) \cap (W_2(i)) \}. i \in N_3 \} \)

from \( A1 \) I II have \( \forall i \in N_1 \cap N_2. (W_1(i) \cap W_2(i)) \in T \)

using apply_funtype IsATopology_def by auto

moreover from I II have \( \forall i \in N_1 - N_2. W_1(i) \in T \) and \( \forall i \in N_2 - N_1. W_2(i) \in T \)

using apply_funtype by auto

ultimately have \( W_3 : N_3 \to T \) by (rule fun_union_overlap)

with I II have \( \text{FinProd}(X, W_3) \in \text{ProductTopBase}(I, T) \) using union_finpow

ProductTopBase_def

by auto

moreover have \( U \cap V = \text{FinProd}(X, W_3) \)

proof
proof seq_prod_top_is_top:
theorem The (sequence) product topology is indeed a topology on the space of se-
theorem satisfies the base condition. This is a condition, defined in
In the next theorem we show the collection of sets defined above as ProductTopBase(X,T)
satisfies the base condition. This is a condition, defined in Topology_ZF_1 that allows to claim that this collection is a base for some topology.

|{ fix i assume i ∈ N |

|{| fix x assume x ∈ U and x ∈ V |

|{| with ⟨U = FinProd(X,W₁)⟩ and ⟨V = FinProd(X,W₂)⟩ |

|{|⟨W₂ ∈ N₂→T⟩ |

|{| have x ∈ X and ∀ i ∈ N₁. x(i) ∈ W₁(i) and ∀ i ∈ N₂. x(i) ∈ W₂(i) |

|{| using func1_1_L1 FinProd_def by auto |

|{| with ⟨W₃: N₃→T⟩ ⟨x ∈ X⟩ have x ∈ FinProd(X,W₃) |

|{| using ZF_fun_from_tot_val func1_1_L1 FinProd_def by auto |

|{| thus U ∪ V ⊆ FinProd(X,W₃) by auto |

|{|⟨ W₃: N₃→T ⟩ |

|{| have x:I→∪T and III: ∀ i ∈ N₃. x(i) ∈ W₃(i) |

|{| using FinProd_def func1_1_L1 by auto |

|{|⟨ fix i assume i ∈ N₁ |

|{| with ⟨W₃: N₃→T⟩ have W₃(i) ⊆ W₁(i) using ZF_fun_from_tot_val by auto |

|{| with III ⟨i ∈ N₁⟩ have x(i) ∈ W₁(i) by auto |

|{|⟨ W₃: N₃→T ⟩ |

|{| have x ∈ U using func1_1_L1 FinProd_def by auto |

|{| moreover |

|{|⟨ fix i assume i ∈ N₂ |

|{| with ⟨W₃: N₃→T⟩ have W₃(i) ⊆ W₂(i) using ZF_fun_from_tot_val by auto |

|{| with III ⟨i ∈ N₂⟩ have x(i) ∈ W₂(i) by auto |

|{|⟨ W₃: N₃→T ⟩ |

|{| have x ∈ V using func1_1_L1 FinProd_def by auto |

|{| ultimately have x ∈ U ∪ V by simp |

|{| thus FinProd(X,W₃) ⊆ U ∪ V by auto |

|{ qed |

|{| ultimately show thesis by simp |

|{ qed |

In the next theorem we show the collection of sets defined above as ProductTopBase(X,T)
satisfies the base condition. This is a condition, defined in Topology_ZF_1 that allows to claim that this collection is a base for some topology.

|theorem prod_top_base_is_base: assumes T {is a topology} |

|shows ProductTopBase(I,T) {satisfies the base condition} |

|using assms prod_top_base_inter inter_closed_base by simp |

The (sequence) product topology is indeed a topology on the space of se-
quences. In the proof we are using the fact that (∅ → X) = {∅}.

|theorem seq_prod_top_is_top: assumes T {is a topology} |

|shows |

|SeqProductTopology(I,T) {is a topology} and |

|ProductTopBase(I,T) {is a base for} SeqProductTopology(I,T) and |

|∪ SeqProductTopology(I,T) = (I→∪T) |

|proof |

|from assms show SeqProductTopology(I,T) {is a topology} and |

|I: ProductTopBase(I,T) {is a base for} SeqProductTopology(I,T) |

|using prod_top_base_is_base SeqProductTopology_def Top_1_2_T1 |

|by auto |

|877 |
from I have \( \bigcup \text{SeqProductTopology}(I,T) = \bigcup \text{ProductTopBase}(I,T) \)
using Top_1_2_L5 by simp
also have \( \bigcup \text{ProductTopBase}(I,T) = (I \to \bigcup T) \)
proof
  have 0 \in FinPow(I) using empty_in_finpow by simp
  then have \( \{\text{FinProd}(I \to \bigcup T,W). W \in 0 \to T\} \subseteq (\bigcup N \in \text{FinPow}(I).\{\text{FinProd}(I \to \bigcup T,W)\} \)
    by blast
  then show \( (I \to \bigcup T) \subseteq \bigcup \text{ProductTopBase}(I,T) \) using ProductTopBase_def
proof
  have \( \{\text{FinProd}(I \to \bigcup T,W). W \in n \to T\} \subseteq \bigcup \text{ProductTopBase}(n,T) \)
    using ProductTopBase_def
  qed
finally show \( \bigcup \text{SeqProductTopology}(I,T) = (I \to \bigcup T) \) by simp
qed

70.2 Finite product of topologies

As a special case of the space of functions \( I \to X \) we can consider space of lists of elements of \( X \), i.e. space \( n \to X \), where \( n \) is a natural number (recall that in ZF set theory \( n = \{0,1,\ldots,n-1\} \)). Such spaces model finite cartesian products \( X^n \) but are easier to deal with in formalized way (than the said products). This section discusses natural topology defined on \( n \to X \) where \( X \) is a topological space.

When the index set is finite, the definition of \( \text{ProductTopBase}(I,T) \) can be simplified.

lemma fin_prod_def_nat: assumes A1: \( n \in \text{nat} \) and A2: \( T \) is a topology shows \( \text{ProductTopBase}(n,T) = \{\text{FinProd}(n \to \bigcup T,W). W \in n \to T\} \)
proof
  from A1 have \( n \in \text{FinPow}(n) \) using nat_finpow_nat fin_finpow_self by auto
  then show \( \{\text{FinProd}(n \to \bigcup T,W). W \in n \to T\} \subseteq \text{ProductTopBase}(n,T) \) using ProductTopBase_def
proof
  { fix B assume B \in \text{ProductTopBase}(n,T)
    then obtain N W where \( N \in \text{FinPow}(n) \) and \( W \in n \to T \) and \( B = \text{FinProd}(n \to \bigcup T,W) \)
      using ProductTopBase_def by auto
    let \( W_n = \{i,if i \in N then W(i) else \bigcup T\}. i \in n\}
    from A2 have \( \forall N \in \text{FinPow}(n). \forall W \in n \to T \) have \( \forall i \in n. (if i \in N then W(i)) \)
      by (rule ZF_fun_from_total)
    moreover have \( B = \text{FinProd}(n \to \bigcup T,W_n) \)
  proof
    { fix x assume \( x \in B \)
      with \( B = \text{FinProd}(n \to \bigcup T,W_n) \) have \( x \in n \to \bigcup T \) using FinProd_def
      qed
  qed

878
by simp

moreover have \( \forall i \in \text{domain}(W_n). x(i) \in W_n(i) \)

proof
  fix i assume i \in \text{domain}(W_n)
  with \( \langle W_n, n \mapsto T \rangle \) have i:n using func1_1_L1 by simp
  with \( \langle x : n \mapsto \bigcup T \rangle \) have x(i) \in \bigcup T using apply_funtype by blast
  with \( \langle x : n \mapsto \bigcup T \rangle \) have x(i) \in \bigcup T using apply_funtype by blast
  show x(i) \in W_n(i) using func1_1_L1 FinProd_def ZF_fun_from_tot_val
by simp
qed

ultimately have x \in \text{FinProd}(n \mapsto \bigcup T, W_n)
using FinProd_def by simp

next

{ fix x assume x \in \text{FinProd}(n \mapsto \bigcup T, W_n)
then have x \in n \mapsto \bigcup T and \( \forall i \in \text{domain}(W_n). x(i) \in W_n(i) \)
  using FinProd_def by auto
  with \( \langle W_n, n \mapsto T \rangle \) and \( \langle N \in \text{FinPow}(n) \rangle \) have \( \forall i \in N. W_n(i) = W(i) \)
  using ZF_fun_from_tot_val FinPow_def by auto
  moreover from \( \langle W_n, n \mapsto T \rangle \) and \( \langle N \in \text{FinPow}(n) \rangle \)
  have \( \forall i \in N. x(i) \in W(i) \)
  by simp
  with \( \langle W \in n \mapsto T \rangle \) and \( \langle B = \text{FinProd}(n \mapsto \bigcup T, W) \rangle \) have \( x \in B \)
  using func1_1_L1 FinProd_def by simp
}
thus FinProd(n \mapsto \bigcup T, W_n) \subseteq B by auto
qed

ultimately have B \in \{\text{FinProd}(n \mapsto \bigcup T, W). W \in n \mapsto T\} by auto
}
thus ProductTopBase(n, T) \subseteq \{\text{FinProd}(n \mapsto \bigcup T, W). W \in n \mapsto T\} by auto
qed

A technical lemma providing a formula for finite product on one topological space.

lemma single_top_prod: assumes A1: \( W : n \mapsto T \)
  shows \( \text{FinProd}(1 \mapsto \bigcup T, W) = \{ \{0,y\}. y \in W(0)\} \)
proof
  have 1 = \{0\} by auto
  from A1 have \( \text{domain}(W) = \{0\} \) using func1_1_L1 by auto
  then have \( \text{FinProd}(1 \mapsto \bigcup T, W) = \{ \{x \in 1 \mapsto T. x(0) \in W(0)\} \}
  using FinProd_def by simp
  also have \( \{x \in 1 \mapsto \bigcup T. x(0) \in W(0)\} \subseteq \{ \{0,y\}. y \in W(0)\} \)
  proof
    from \( x = \{0\} \) show \( \{x \in 1 \mapsto \bigcup T. x(0) \in W(0)\} \subseteq \{ \{0,y\}. y \in W(0)\} \)
    using func_singleton_pair by auto
    { fix x assume x \in \{ \{0,y\}. y \in W(0)\}
      then obtain y where x = \{\{0, y\}\} and II: y \in W(0) by auto
      with A1 have y \in \bigcup T using apply_funtype by auto
      with \( x = \{0,y\} \) have x:1 \mapsto \bigcup T using pair_func_singleton by auto
    }
  qed
  by simp
qed

879
with \( \langle x = \{\langle 0, y \rangle \} \rangle \) II have \( x \in \{ x \in 1 \rightarrow \bigcup \tau. x(0) \in W(0) \} \)
using pair_val by simp
\)
thus \( \{ \langle 0, y \rangle \}. y \in W(0) \} \subseteq \{ x \in 1 \rightarrow \bigcup \tau. x(0) \in W(0) \} \) by auto
qed
finally show thesis by simp
qed

Intuitively, the topological space of singleton lists valued in \( X \) is the same as \( X \). However, each element of this space is a list of length one, i.e. a set consisting of a pair \( \langle 0, x \rangle \) where \( x \) is an element of \( X \). The next lemma provides a formula for the product topology in the corner case when we have only one factor and shows that the product topology of one space is essentially the same as the space.

**lemma singleton_prod_top:** assumes \( A1: \tau \) is a topology shows
SeqProductTopology(1, \( \tau \)) = \( \{ \{ \langle 0, y \rangle \}. y \in U \}. U \in \tau \} \)
and IsAhomeomorphism(\( \tau \), SeqProductTopology(1, \( \tau \)), \( \{ \langle y, \{ \langle 0, y \rangle \} \}. y \in \bigcup \tau \})
proof
- have \( \{ 0 \} = 1 \) by auto
let \( b = \{ \langle y, \{ \langle 0, y \rangle \} \}. y \in \bigcup \tau \} \)
have \( b \in bij(\bigcup \tau, 1 \rightarrow \bigcup \tau) \) using list_singleton_bij by blast
with \( A1 \) have \( \{ b(U). U \in \tau \} \) is a topology and IsAhomeomorphism(\( \tau \), \( \{ b(U). U \in \tau \} \), \( b \)) using bij_induced_top by auto
moreover have \( \forall U \in \tau. b(U) = \{ \{ \langle 0, y \rangle \}. y \in U \} \)
proof
  fix \( V \) assume \( U \in \tau \)
  from \( \langle b \in bij(\bigcup \tau, 1 \rightarrow \bigcup \tau) \rangle \) have \( b:\bigcup \tau \rightarrow (1 \rightarrow \bigcup \tau) \) using bij_def inj_def
  by simp
  \{ fix \( y \) assume \( y \in \bigcup \tau \)
  with \( \langle b:\bigcup \tau \rightarrow (1 \rightarrow \bigcup \tau) \rangle \) have \( b(y) = \{ \langle 0, y \rangle \} \) using ZF_fun_from_tot_val
  by simp
  \} hence \( \forall y \in \bigcup \tau. b(y) = \{ \langle 0, y \rangle \} \) by auto
  with \( \langle U \in \tau \rangle \) \( \langle b:\bigcup \tau \rightarrow (1 \rightarrow \bigcup \tau) \rangle \) show \( b(U) = \{ \{ \langle 0, y \rangle \}. y \in U \} \)
  using func_imagedef by auto
qed
moreover have ProductTopBase(1, \( \tau \)) = \( \{ \{ \langle 0, y \rangle \}. y \in U \}. U \in \tau \} \)
proof
  \{ fix \( V \) assume \( V \in ProductTopBase(1, \tau) \)
with \( A1 \) obtain \( W \) where \( W:1 \rightarrow \tau \) and \( V = FinProd(1 \rightarrow \bigcup \tau, W) \)
using fin_prod_def_nat by auto
  then have \( V \in \{ \{ \langle 0, y \rangle \}. y \in U \}. U \in \tau \} \) using apply_funtype single_top_prod
  by auto
  \} thus ProductTopBase(1, \( \tau \)) \subseteq \( \{ \{ \langle 0, y \rangle \}. y \in U \}. U \in \tau \} \) by auto
  \{ fix \( V \) assume \( V \in \{ \{ \langle 0, y \rangle \}. y \in U \}. U \in \tau \} \)
then obtain \( U \in \tau \) and \( V = \{ \{ \langle 0, y \rangle \}. y \in U \} \) by auto
let \( W = \{ \langle 0, U \} \)
from \( \langle U \in \tau \rangle \) have \( W:0 \rightarrow \tau \) using pair_func_singleton by simp
with \( \langle \{ 0 \} = 1 \rangle \) have \( W:1 \rightarrow \tau \) and \( W(0) = U \) using pair_val by auto
with $V = \{ \{(0,y)\} : y \in U \}$ have $V = \text{FinProd}(1 \to \bigcup \tau, W)$ using single_top_prod by simp

with A1 $\langle W : 1 \to \tau \rangle$ have $V \in \text{ProductTopBase}(1, \tau)$ using fin_prod_def_nat by auto

} thus $\{ \{(0,y)\} : y \in U \} \subseteq \text{ProductTopBase}(1, \tau)$ by auto

qed ultimately have I: $\text{ProductTopBase}(1, \tau)$ {is a topology} and II: $\text{IsAhomeomorphism}(\tau, \text{ProductTopBase}(1, \tau), b)$ by auto

from A1 have $\text{ProductTopBase}(1, \tau)$ {is a base for} $\text{SeqProductTopology}(1, \tau)$ using seq_prod_top_is_top by simp

with {‹ProductTopBase(1, \tau) = \{ \{(0,y)\} : y \in U \} \subseteq \text{ProductTopBase}(1, \tau)›} II show SeqProductTopology(1, \tau) = $\{ \{(0,y)\} : y \in U \}$ and IsAhomeomorphism($\tau, \text{SeqProductTopology}(1, \tau), \langle y, \{(0,y)\} : y \in \bigcup \tau \rangle$) by auto

qed

A special corner case of finite_top_prod_homeo: a space $X$ is homeomorphic to the space of one element lists of $X$.

theorem singleton_prod_top1: assumes A1: $\tau$ {is a topology} shows $\text{IsAhomeomorphism}(\text{SeqProductTopology}(1, \tau), \tau, \langle x, x(0) \rangle. x \in 1 \to \bigcup \tau)$

proof -

have $\langle x, x(0) \rangle. x \in 1 \to \bigcup \tau = \text{converse}(\langle y, \{(0,y)\} : y \in \bigcup \tau \rangle)$

using list_singleton_bij by blast

with A1 show thesis using singleton_prod_top homeo_inv by simp

qed

A technical lemma describing the carrier of a (cartesian) product topology of the (sequence) product topology of $n$ copies of topology $\tau$ and another copy of $\tau$.

lemma finite_prod_top: assumes $\tau$ {is a topology} and $T = \text{SeqProductTopology}(n, \tau)$ shows $(\bigcup \text{ProductTopology}(T, \tau)) = (n \to \bigcup \tau) \times (\bigcup \tau)$

using assms Top_1_4_T1 seq_prod_top_is_top by simp

If $U$ is a set from the base of $X^n$ and $V$ is open in $X$, then $U \times V$ is in the base of $X^{n+1}$. The next lemma is an analogue of this fact for the function space approach.

lemma finite_prod_succ_base: assumes A1: $\tau$ {is a topology} and A2: $n \in \text{nat}$ and A3: $U \in \text{ProductTopBase}(n, \tau)$ and A4: $V \in \tau$

shows $\{ x \in \text{succ}(n) \to \bigcup \tau. \text{Init}(x) \in U \wedge x(n) \in V \} \in \text{ProductTopBase}(\text{succ}(n), \tau)$

proof -

let $B = \{ x \in \text{succ}(n) \to \bigcup \tau. \text{Init}(x) \in U \wedge x(n) \in V \}$

from A1 A2 have $\text{ProductTopBase}(n, \tau) = \{ \text{FinProd}(n \to \bigcup \tau, W) : W \in n \to \tau \}$

using fin_prod_def_nat by simp

with A3 obtain $W$ where $W : n \to \tau$ and $U = \text{FinProd}(n \to \bigcup \tau, W)$ by auto

881
let $W = \text{Append}(W_U, V)$
from A4 and $\langle W_U : n \rightarrow \tau \rangle$ have $W : \text{succ}(n) \rightarrow \tau$ using append_props by simp
moreover have $B = \text{FinProd}(\text{succ}(n) \rightarrow \bigcup \tau, W)$
proof
{ fix $x$ assume $x \in B$
   with $\langle W : \text{succ}(n) \rightarrow \tau \rangle$ have $x \in \text{succ}(n) \rightarrow \bigcup \tau$ and domain($W$) = succ($n$)
   using func1_1_L1
   by auto
moreover from A2 A4 $\langle x \in \text{succ}(n) \rightarrow \tau \rangle$
   have $\forall i \in \text{succ}(n). \ x(i) \in W(i)$ using func1_1_L1 FinProd_def init_props
append_props
by simp
ultimately have $x \in \text{FinProd}(\text{succ}(n) \rightarrow \bigcup \tau, W)$ using FinProd_def
by simp}
thus $B \subseteq \text{FinProd}(\text{succ}(n) \rightarrow \bigcup \tau, W)$ by auto
next
{ fix $x$ assume $x \in \text{FinProd}(\text{succ}(n) \rightarrow \bigcup \tau, W)$
then have $x : \text{succ}(n) \rightarrow \bigcup \tau$ and I: $\forall i \in \text{domain}(W). \ x(i) \in W(i)$
   using FinProd_def by auto
moreover have $\text{Init}(x) \in U$
proof -
from A2 and $\langle x : \text{succ}(n) \rightarrow \bigcup \tau \rangle$ have $\text{Init}(x) : n \rightarrow \bigcup \tau$ using init_props
moreover have $\forall i \in \text{domain}(W_U). \ \text{Init}(x)(i) \in W_U(i)$
proof -
from A2 $\langle x \in \text{FinProd}(\text{succ}(n) \rightarrow \bigcup \tau, W) \rangle$ $\langle W : \text{succ}(n) \rightarrow \tau \rangle$ have
$\forall i \in n. \ x(i) \in W(i)$
   using FinProd_def func1_1_L1 by simp
moreover from A2 $\langle x: \text{succ}(n) \rightarrow \bigcup \tau \rangle$ have $\forall i \in n. \ \text{Init}(x)(i) = x(i)$
   using init_props by simp
moreover from A4 and $\langle W_U : n \rightarrow \tau \rangle$ have $\forall i \in n. \ W(i) = W_U(i)$
   using append_props by simp
ultimately have $\forall i \in n. \ \text{Init}(x)(i) \in W_U(i)$ by simp
with $\langle W_U : n \rightarrow \tau \rangle$ show thesis using func1_1_L1 by simp
qed
ultimately have $\text{Init}(x) \in \text{FinProd}(n \rightarrow \bigcup \tau, W_U)$ using FinProd_def
by simp
with $\langle U = \text{FinProd}(n \rightarrow \bigcup \tau, W_U) \rangle$ show thesis by simp
qed
moreover have $x(n) \in V$
proof -
from $\langle W : \text{succ}(n) \rightarrow \tau \rangle$ I have $x(n) \in W(n)$ using func1_1_L1 by simp
moreover from A4 $\langle W_U : n \rightarrow \tau \rangle$ have $W(n) = V$ using append_props
by simp
ultimately show thesis by simp
qed

882
ultimately have $x \in B$ by simp
} thus $\text{FinProd}(\text{succ}(n) \to \bigcup \tau, W) \subseteq B$ by auto
qed

moreover from A1 A2 have
ProductTopBase(\text{succ}(n), \tau) = \{\text{FinProd}(\text{succ}(n) \to \bigcup \tau, W). W \in \text{succ}(n) \to \tau\}
using fin_prod_def_nat by simp
ultimately show thesis by auto
qed

If $U$ is open in $X^n$ and $V$ is open in $X$, then $U \times V$ is open in $X^{n+1}$. The next lemma is an analogue of this fact for the function space approach.

lemma finite_prod_succ: assumes A1: \tau \text{ is a topology} \text{ and } A2: n \in \text{nat}
and
A3: $U \in \text{SeqProductTopology}(n, \tau)$ \text{ and } A4: $V \in \tau$
shows \{\begin{align*}
&\text{x : succ(n) -> } \bigcup \tau. \text{ Init(x) } \in U \land x(n) \in V \} \in \text{SeqProductTopology(succ(n), } \tau)\\
\end{align*}
proof -
from A1 have ProductTopBase(succ(n), \tau) \{is a base for} SeqProductTopology(n, \tau)
and
I: ProductTopBase(succ(n), \tau) \{is a base for} SeqProductTopology(succ(n), \tau)
and
II: SeqProductTopology(succ(n), \tau) \{is a topology\}
using seq_prod_top_is_top by auto
with A3 have $\exists B \in \text{Pow(ProductTopBase(n, } \tau))$. $U = \bigcup B$ using Top_1_2_L1
by simp
then obtain $B$ where $B \subseteq \text{ProductTopBase(n, } \tau)$ \text{ and } $U = \bigcup B$ by auto
then have
\{\begin{align*}
&\text{x : succ(n) -> } \bigcup \tau. \text{ Init(x) } \in U \land x(n) \in V \} = (\bigcup_{B \in B}. \{\text{x : succ(n) -> } \bigcup \tau. \text{ Init(x) } \in B \land x(n) \in V \})\\
\end{align*}
by auto
moreover from A1 A2 A4 \langle B : \subseteq \text{ProductTopBase(n, } \tau)\rangle have
\forall B \in B. (\{\text{x : succ(n) -> } \bigcup \tau. \text{ Init(x) } \in B \land x(n) \in V \} \in \text{ProductTopBase(succ(n), } \tau))
using finite_prod_succ_base by auto
with I II have
(\bigcup_{B \in B}. \{\text{x : succ(n) -> } \bigcup \tau. \text{ Init(x) } \in B \land x(n) \in V \}) \in \text{SeqProductTopology(succ(n), } \tau)
using base_sets_open union_indexed_open by auto
ultimately show thesis by simp
qed

In the Topology_ZF_2 theory we define product topology of two topological spaces. The next lemma explains in what sense the topology on finite lists of length $n$ of elements of topological space $X$ can be thought as a model of the product topology on the cartesian product of $n$ copies of that space. Namely, we show that the space of lists of length $n+1$ of elements of $X$ is homeomorphic to the product topology (as defined in Topology_ZF_2) of two spaces: the space of lists of length $n$ and $X$. Recall that if $B$ is a base (i.e. satisfies the base condition), then the collection $\{\bigcup B | B \in \text{Pow}(B)\}$ is a topology (generated by $B$).

theorem finite_top_prod_homeo: assumes A1: \tau \text{ is a topology} \text{ and } A2:
\( n \in \text{nat} \) and

\[ A_3: f = \langle x, \langle \text{Init}(x), x(n) \rangle \rangle. x \in \text{succ}(n) \rightarrow \bigcup \tau \] and

\[ A_4: T = \text{SeqProductTopology}(n, \tau) \] and

\[ A_5: S = \text{SeqProductTopology}(\text{succ}(n), \tau) \]

shows \( \text{IsAhomeomorphism}(S, \text{ProductTopology}(T, \tau), f) \)

\[ \text{proof} - \]

- let \( C = \text{ProductCollection}(T, \tau) \)
  - let \( B = \text{ProductTopBase}(\text{succ}(n), \tau) \)
  - from \( A_1 \ A_4 \)
    - have \( T \) \{is a topology\} using \( \text{seq_prod_top_is_top} \) by simp
  - with \( A_1 \ A_5 \)
    - have \( S \) \{is a topology\} and \( \text{ProductTopology}(T, \tau) \) \{is a topology\} using \( \text{seq_prod_top_is_top} \ \text{Top_1_4_T1} \) by auto
  - moreover from \( \text{assms} \)
    - have \( f \in \text{bij}(\bigcup S, \bigcup \text{ProductTopology}(T, \tau)) \) using \( \text{lists_cart_prod} \ \text{seq_prod_top_is_top} \ \text{Top_1_4_T1} \) by simp
  - then have \( f: \bigcup S \rightarrow \bigcup \text{ProductTopology}(T, \tau) \) using \( \text{bij_is_fun} \) by simp
  - ultimately have \( \text{two_top_spaces0}(S, \text{ProductTopology}(T, \tau), f) \) using \( \text{two_top_spaces0_def} \) by simp

moreover note \( f \in \text{bij}(\bigcup S, \bigcup \text{ProductTopology}(T, \tau)) \)
- moreover from \( A_1 \ A_5 \)
  - have \( B \) \{is a base for\} \( S \) using \( \text{seq_prod_top_is_top} \) by simp
  - moreover from \( A_1 \ \langle T \) \{is a topology\}\rangle have \( C \) \{is a base for\} \( \text{ProductTopology}(T, \tau) \)

using \( \text{Top_1_4_T1} \) by auto
- moreover have \( \forall W \in C. f^{-1}(W) \in S \)
  - proof
    - fix \( W \) assume \( W \in C \)
    - then obtain \( U \ V \) where \( U \in T \ V \in \tau \) and \( W = U \times V \) using \( \text{ProductCollection_def} \) by auto
    - from \( A_1 \ A_5 \)
      - have \( \langle f, \bigcup S \rightarrow \bigcup \text{ProductTopology}(T, \tau) \rangle \) have \( f: (\text{succ}(n) \rightarrow \bigcup \tau) \rightarrow \bigcup \text{ProductTopology}(T, \tau) \) using \( \text{seq_prod_top_is_top} \) by simp
      - with \( \text{assms} \)
        - have \( \langle W = U \times V \rangle \ \langle U \in T \rangle \ \langle V \in \tau \rangle \) show \( f^{-1}(W) \in S \)
          - using \( \text{ZF_fun_from_tot_val} \ \text{func1_1_L15} \ \text{finite_prod_succ} \) by simp

qed
- moreover have \( \forall V \in B. f(V) \in \text{ProductTopology}(T, \tau) \)
  - proof
    - fix \( V \) assume \( V \in B \)
    - with \( A_1 \ A_2 \)
      - obtain \( W \) where \( W \in \text{succ}(n) \rightarrow \tau \) and \( V = \text{FinProd}(\text{succ}(n) \rightarrow \bigcup \tau, W) \)
        - using \( \text{fin_prod_def_nat} \) by auto
      - let \( U = \text{FinProd}(n \rightarrow \bigcup \tau, \text{Init}(W)) \)
      - let \( W = W(n) \)
      - have \( U \in T \)
        - proof
          - from \( A_1 \ A_2 \)
            - have \( W \in \text{succ}(n) \rightarrow \tau \) have \( U \in \text{ProductTopBase}(n, \tau) \)
              - using \( \text{fin_prod_def_nat} \ \text{init_props} \) by auto
            - with \( A_1 \ A_4 \) show thesis using \( \text{seq_prod_top_is_top} \ \text{base_sets_open} \)
              by blast

884
qed

from A1 \( \forall V \in \text{succ}(n) \rightarrow \tau \) \( \langle T \text{ is a topology} \rangle \) \( \langle U \in T \rangle \) have \( U \times W \in \text{ProductTopology}(T, \tau) \)

using apply_funtype prod_open_open_prod by simp

moreover have \( f(V) = U \times W \)

proof -
  from A2 \( \forall W \in \text{succ}(n) \rightarrow \tau \) have Init(\( W \)) : \( \tau \rightarrow n \) and III: \( \forall k \in n. \text{Init}(W)(k) = W(k) \)
  using init_props by auto
  then have domain(Init(\( W \))) = n using func1_1_L1 by simp
  have \( f(V) = \{ \langle \text{Init}(x), x(n) \rangle. x \in V \} \)
  proof -
    have \( f(V) = \{ f(x). x \in V \} \)
    proof -
      from A1 A5 have B \( \{ \text{is a base for} \} S \) using seq_prod_top_is_top by simp
      with \( V \subseteq \bigcup S \) have \( V \subseteq \bigcup \text{ProductTopology}(T, \tau) \) show thesis using func_imagedef by simp
    qed
  qed
  moreover have \( \forall x \in V. f(x) = \{ \langle \text{Init}(x), x(n) \rangle \} \)
  proof -
    from A1 A3 A5 \( \forall V = \text{FinProd}(\text{succ}(n) \rightarrow \bigcup \tau, W) \) have \( V \subseteq \bigcup S \) and
    \( fdef: f = \{ \langle x, \langle \text{Init}(x), x(n) \rangle \rangle. x \in \bigcup S \} \) using seq_prod_top_is_top by auto
    with \( f: \bigcup S \rightarrow \bigcup \text{ProductTopology}(T, \tau) \) show thesis using func_imagedef by auto
  qed
  qed
  also have \( \{ \langle \text{Init}(x), x(n) \rangle. x \in V \} = U \times W \)
  proof
    \{ fix \( y \) assume \( y \in \{ \langle \text{Init}(x), x(n) \rangle. x \in V \} \) then obtain \( x \) where I: \( y = \langle \text{Init}(x), x(n) \rangle \) and \( x \in V \) by auto
    with \( V = \text{FinProd}(\text{succ}(n) \rightarrow \bigcup \tau, W) \) have \( x: \text{succ}(n) \rightarrow \bigcup \tau \) and II: \( \forall k \in \text{domain}(W). x(k) \in W(k) \)
    unfolding FinProd_def by auto
    with A2 \( \forall W \in \text{succ}(n) \rightarrow \tau \) have IV: \( \forall k \in n. \text{Init}(x)(k) = x(k) \)
    using init_props by simp
    have Init(x) \in U using init_props by simp
    proof -
      from A2 \( \forall x: \text{succ}(n) \rightarrow \bigcup \tau \) have Init(x): \( n \rightarrow \bigcup \tau \) using init_props
    moreover have \( \forall k \in \text{domain}(\text{Init}(W)). \text{Init}(x)(k) \in \text{Init}(W)(k) \)
    qed

885
proof -
from A2 \langle W_v : \text{succ}(n) \rightarrow \tau \rangle have Init(W_v) : n \rightarrow \tau using init_props
by simp

then have domain(Init(W_v)) = n using func1_1_L1 by simp
note III IV \langle domain(Init(W_v)) = n \rangle
moreover from II \langle W_v \in \text{succ}(n) \rightarrow \tau \rangle have \forall k \in n. x(k) \in W_v(k)

\text{using func1_1_L1}
ultimately show thesis by simp
qed

ultimately show Init(x) \in U using FinProd_def by simp
qed

moreover from \langle W_v : \text{succ}(n) \rightarrow \tau \rangle II have x(n) \in \mathbb{W} using func1_1_L1
by simp
ultimately have \langle Init(x),x(n) \rangle \in U \times \mathbb{W}

proof -
from \langle \text{fst}(y) : n \rightarrow \bigcup \tau \rangle \langle \text{snd}(y) \in \bigcup \tau \rangle have x : \text{succ}(n) \rightarrow \bigcup \tau
using append_props by simp
moreover have \forall i \in \text{domain}(W_v). x(i) \in W_v(i)
proof -
from \langle \text{fst}(y) : n \rightarrow \bigcup \tau \rangle \langle \text{snd}(y) \in \bigcup \tau \rangle have \forall k \in n. x(k) = \text{fst}(y)(k) and x(n) = \text{snd}(y)
\text{using append Props by auto}
moreover from III V have \forall k \in n. x(k) \in W_v(k)
ultimately note \langle \text{snd}(y) \in \mathbb{W} \rangle
ultimately have \forall i \in \text{succ}(n). x(i) \in W_v(i) by simp
with \langle W_v \in \text{succ}(n) \rightarrow \tau \rangle show thesis using func1_1_L1 by simp

qed
ultimately have x \in \text{FinProd(}\text{succ}(n) \rightarrow \bigcup \tau,W_v) using FinProd_def
by simp

with \langle V = \text{FinProd(}\text{succ}(n) \rightarrow \bigcup \tau,W_v) \rangle show x \in V by simp
qed

moreover from A2 \langle y \in U \times \mathbb{W} \rangle \langle \text{fst}(y) : n \rightarrow \bigcup \tau \rangle \langle \text{snd}(y) \in \bigcup \tau \rangle
have y = \langle \text{Init}(x),x(n) \rangle
\text{using init_append append Props by auto}
ultimately have y \in \{ \langle \text{Init}(x),x(n) \rangle. x \in V \} by auto
\[
\{ (\text{Init}(x), x(n)) \mid x \in V \} \text{ by auto}
\]

\text{qed}

\text{finally show } f(V) = U \times W \text{ by simp}

\text{qed}

\text{ultimately show } f(V) \in \text{ProductTopology}(T, \tau) \text{ by simp}

\text{qed}

\text{ultimately show thesis using two_top_spaces0.bij_base_open_homeo by simp}

\text{qed}

\text{end}

71 Topology 4

theory Topology_ZF_4 imports Topology_ZF_1 Order_ZF func1 NatOrder_ZF begin

This theory deals with convergence in topological spaces. Contributed by Daniel de la Concepcion.

71.1 Nets

Nets are a generalization of sequences. It is known that sequences do not determine the behavior of the topological spaces that are not first countable; i.e., have a countable neighborhood base for each point. To solve this problem, nets were defined so that the behavior of any topological space can be thought in terms of convergence of nets.

We say that a relation \( r \) directs a set \( X \) if the relation is reflexive, transitive on \( X \) and for every two elements \( x, y \) of \( X \) there is some element \( z \) such that both \( x \) and \( y \) are in the relation with \( z \). Note that this naming is a bit inconsistent with what is defined in Order_ZF where we define what it means that \( r \) up-directs \( X \) (the third condition in the definition below) or \( r \) down-directs \( X \). This naming inconsistency will be fixed in the future (maybe).

definition

\text{IsDirectedSet \(_\text{(_directs_90)}\) where \text{r directs X} \equiv \text{refl}(X, r) \land \text{trans}(r) \land (\forall x \in X. \forall y \in X. \exists z \in X. (x, z) \in r \land (y, z) \in r)}

Any linear order is a directed set; in particular \((\mathbb{N}, \leq)\).

\text{lemma linorder_imp_directed:}

\text{assumes IsLinOrder(X, r)}

\text{shows r directs X}

\text{proof-}
from assms have trans(r) using IsLinOrder_def by auto
 moreover
from assms have r:refl(X,r) using IsLinOrder_def total_is_refl by auto
 moreover
{ fix x y
  assume R: x∈X y∈X
  with assms have ⟨x,y⟩∈r ∨ ⟨y,x⟩∈r using IsLinOrder_def IsTotal_def
  by auto
  with r have ⟨(x,y)∈r ∧ (y,y)∈r⟩ ∨ ⟨(y,x)∈r ∧ (x,x)∈r⟩ using R refl_def
  by auto
  then have ∃z∈X. ⟨x,z⟩∈r ∧ ⟨y,z⟩∈r using R by auto
 } ultimately show thesis using IsDirectedSet_def function_def by auto
 qed

Natural numbers are a directed set.

corollary Le_directs_nat: shows IsLinOrder(nat,Le) Le directs nat
  proof -
  show IsLinOrder(nat,Le) by (rule NatOrder_ZF_1_L2)
  then show Le directs nat using linorder_imp_directed by auto
 qed

We are able to define the concept of net, now that we know what a directed
set is.

definition (in topology0) IsNet (_ {is a net on} _ 90)
  where N {is a net on} X ≡ fst(N):domain(fst(N))→X ∧ (snd(N) directs
domain(fst(N))) ∧ domain(fst(N))≠0

Provided a topology and a net directed on its underlying set, we can talk
about convergence of the net in the topology.

definition (in topology0) NetConverges (_ → N _ 90)
  where N {is a net on} ∪T ⇒ N →N x ≡
  (x∈∪T) ∧ (∀U∈Pow(∪T). (x∈int(U) ⇒ (∃t∈domain(fst(N)). ∀m∈domain(fst(N)).
  (⟨t,m⟩∈snd(N) ⇒ fst(N)m∈U))))

One of the most important directed sets, is the neighborhoods of a point.

theorem (in topology0) directedset_neighborhoods:
  assumes x∈∪T
  defines Neigh≡{U∈Pow(∪T). x∈int(U)}
  defines r≡{(U,V)∈(Neigh × Neigh). V⊆U}
  shows r directs Neigh
  proof-
  {
fix \( U \)
assume \( U \in \text{Neigh} \)
then have \( \langle U, U \rangle \in r \) using \( r \_\text{def} \) by auto
\}
then have \( \text{refl}(\text{Neigh}, r) \) using \( \text{refl} \_\text{def} \) by auto
moreover
\{
fix \( U, V, W \)
assume \( \langle U, V \rangle \in r \langle V, W \rangle \in r \)
then have \( U \in \text{Neigh} \ W \in \text{Neigh} W \subseteq U \) using \( r \_\text{def} \) by auto
then have \( \langle U, W \rangle \in r \) using \( r \_\text{def} \) by auto
\}
then have \( \text{trans}(r) \) using \( \text{trans} \_\text{def} \) by blast
moreover
\{
fix \( A, B \)
assume \( \text{p: } A \in \text{Neigh} B \in \text{Neigh} \)
have \( A \cap B \in \text{Neigh} \) proof
- from \( \text{p} \) have \( A \cap B \in \text{Pow}(\bigcup T) \) using \( \text{Neigh} \_\text{def} \) by auto
moreover
\{
from \( \text{p} \) have \( x \in \text{int}(A) x \in \text{int}(B) \) using \( \text{Neigh} \_\text{def} \) by auto
then have \( x \in \text{int}(A) \cap \text{int}(B) \) by auto
moreover
\{
have \( \text{int}(A) \cap \text{int}(B) \subseteq A \cap B \) using \( \text{Top} \_2 \_\text{L1} \) by auto
moreover have \( \text{int}(A) \cap \text{int}(B) \subseteq T \)
using \( \text{Top} \_2 \_\text{L2} \text{Top} \_2 \_\text{L2} \text{topSpaceAssum} \text{IsATopology} \_\text{def} \) by blast
ultimately have \( \text{int}(A) \cap \text{int}(B) \subseteq \text{int}(A \cap B) \)
using \( \text{Top} \_2 \_\text{L5} \) by auto
\}
ultimately have \( x \in \text{int}(A \cap B) \) by auto
\}
ultimately show \( \text{thesis} \) using \( \text{Neigh} \_\text{def} \) by auto
qed
moreover from \( A \cap B \in \text{Neigh} \) have \( \langle A, A \cap B \rangle \in r \wedge \langle B, A \cap B \rangle \in r \)
using \( r \_\text{def} \ p \) by auto
ultimately have \( \exists z \in \text{Neigh}. \langle A, z \rangle \in r \wedge \langle B, z \rangle \in r \) by auto
\}
ultimately show \( \text{thesis} \) using \( \text{IsDirectedSet} \_\text{def} \) by auto
qed

There can be nets directed by the neighborhoods that converge to the point; if there is a choice function.

\textbf{theorem (in \text{topology}0)} \text{net_direct_neigh_converg}:
\begin{align*}
\text{assumes } & x \in \bigcup T \\
\text{defines } & \text{Neigh} = \{ U \in \text{Pow}(\bigcup T). x \in \text{int}(U) \} \\
\text{defines } & r = \{ \langle U, V \rangle \in (\text{Neigh} \times \text{Neigh}). V \subseteq U \} \\
\text{assumes } & f: \text{Neigh} \rightarrow \bigcup T \ \forall U \in \text{Neigh}. \ f(U) \in U
\end{align*}

889
shows \((f,r) \rightarrow_N x\)

proof -
from assms(4) have dom_def: Neigh = domain(f) using Pi_def by auto
moreover have \(\bigcup T \in T\) using topSpaceAssum IsATopology_def by auto
with assms(1) have \(\bigcup T \in \text{Neigh}\) using Neigh_def by auto
then have \(\bigcup T \in \text{domain}(\text{fst}((f,r)))\) using dom_def by auto
moreover from assms(4) dom_def have \(\text{fst}((f,r)) : \text{domain}(\text{fst}((f,r))) \rightarrow \bigcup T\)
by auto
moreover from assms(1,2,3) dom_def have \(\text{snd}((f,r)) \text{ directs \ domain(\text{fst}((f,r)))}\)
using directedset_neighborhoods by simp
ultimately have Net:\((f,r) \{is a net on\} \bigcup T\) unfolding IsNet_def by auto

\{ 
  fix \(U\) 
  assume \(U \in \text{Pow}(\bigcup T)\) \(x \in \text{int}(U)\) 
  then have \(U \in \text{Neigh}\) using Neigh_def by auto 
  then have \(t: U \in \text{domain}(f)\) using dom_def by auto 
  \{ 
    fix \(W\) 
    assume \((U,W) \in r\) 
    then have \(W \in \text{Neigh}\) using dom_def by auto 
    with assms(5) have \(fW \in W\) by auto 
    with A(2) r_def have \(fW \in U\) by auto 
  } 
  then have \(\forall W \in \text{domain}(f). (\langle U,W \rangle \in r \rightarrow fW \in U)\) by auto 
  with \(t\) have \(\exists V \in \text{domain}(f). \forall W \in \text{domain}(f). (\langle V,W \rangle \in r \rightarrow f(W) \in U))\) 
by auto 
with assms(1) Net show thesis using NetConverges_def by auto 
qed 

71.2 Filters

Nets are a generalization of sequences that can make us see that not all topological spaces can be described by sequences. Nevertheless, nets are not always the tool used to deal with convergence. The reason is that they make use of directed sets which are completely unrelated with the topology.

The topological tools to deal with convergence are what is called filters.

definition
IsFilter (_ {is a filter on} _ 90) 
where \(\mathcal{F} \{is a filter on\} X \equiv (\emptyset \in \mathcal{F}) \land (X \in \mathcal{F}) \land \mathcal{F} \subseteq \text{Pow}(X) \land 
(\forall A \in \mathcal{F}. \forall B \in \mathcal{F}. A \cup B \in \mathcal{F}) \land (\forall B \in \mathcal{F}. \forall C \subseteq \text{Pow}(X). B \subseteq C \rightarrow C \in \mathcal{F})\)
The next lemma splits the definition of a filter into four conditions to make it easier to reference each one separately in proofs.

**lemma is_filter_def_split:** assumes $F$ {is a filter on} $X$

shows $0 \notin F$

$X \in F$

$F \subseteq \mathcal{P}(X)$

$\forall A \in F. \forall B \in F. A \cap B \in F$

$\forall B \in F. \forall C \in \mathcal{P}(X). B \subseteq C \rightarrow C \in F$

**using assms unfolding IsFilter_def by auto**

Not all the sets of a filter are needed to be consider at all times; as it happens with a topology we can consider bases.

**definition**

IsBaseFilter (_ {is a base filter} _ 90)

where $C$ {is a base filter} $F \equiv C \subseteq F$ and $F = \{ A \in \mathcal{P}(\bigcup F). (\exists D \subseteq C. D \subseteq A) \}$

Not every set is a base for a filter, as it happens with topologies, there is a condition to be satisfied.

**definition**

SatisfiesFilterBase (_ {satisfies the filter base condition} 90)

where $C$ {satisfies the filter base condition} $\equiv (\forall A \subseteq C. \forall B \subseteq C. \exists D \subseteq C. D \subseteq A \cap B) \land C \neq 0 \land 0 \notin C$

Every set of a filter contains a set from the filter's base.

**lemma basic_element_filter:**

assumes $A \in F$ and $C$ {is a base filter} $F$

shows $\exists D \subseteq C. D \subseteq A$

**proof-**

- from assms(2) have $t: F = \{ A \in \mathcal{P}(\bigcup \mathcal{F}). (\exists D \subseteq C. D \subseteq A) \}$ using IsBaseFilter_def by auto

- with assms(1) have $A \subseteq \{ A \in \mathcal{P}(\bigcup \mathcal{F}). (\exists D \subseteq C. D \subseteq A) \}$ by auto

- then have $A \subseteq \mathcal{P}(\bigcup \mathcal{F}) \exists D \subseteq C. D \subseteq A$ by auto

- then show thesis by auto

qed

The following two results state that the filter base condition is necessary and sufficient for the filter generated by a base, to be an actual filter. The third result, rewrites the previous two.

**theorem basic_filter_1:**

assumes $C$ {is a base filter} $F$ and $C$ {satisfies the filter base condition}

shows $F$ {is a filter on} $\bigcup F$

**proof-**

- fix $A$ $B$

- assume $A \in F$ and $B \in F$

- with assms(1) have $\exists D \subseteq C. D \subseteq A$ using basic_element_filter by simp

- then obtain $D$ where perA: $D \subseteq C$ and subA: $D \subseteq A$ by auto

- from $B$ assms have $\exists D \subseteq C. D \subseteq B$ using basic_element_filter by simp

- then obtain $D$ where perB: $D \subseteq C$ and subB: $D \subseteq B$ by auto

- from assms(2) perA perB have $\exists D \subseteq C. D \subseteq DA \land DB$

891
unfolding SatisfiesFilterBase_def by auto
then obtain D where D ∈ C D ⊆ A ∩ B by auto
with sub A B AF BF have A ∩ B ∈ Pow(∪ F) . ∃ D ∈ C. D ⊆ A by auto
with assms(1) have A ∩ B ∈ F unfolding IsBaseFilter_def by auto

moreover 
{ 
fix A B 
assume AF: A ∈ F and BS: B ∈ Pow(∪ F) and sub: A ⊆ B
from assms(1) AF have ∃ D ∈ C. D ⊆ A using basic_element_filter by auto
then obtain D where D ∈ C by auto
with sub BS have B ∈ {A ∈ Pow(∪ F). ∃ D ∈ C. D ⊆ A} by auto
with assms(1) have B ∈ F unfolding IsBaseFilter_def by auto
}
moreover
from assms(2) have F ≠ 0 using SatisfiesFilterBase_def by auto
then obtain D where D ∈ C by auto
with assms(1) have D ∈ {A ∈ Pow(∪ F). ∃ D ∈ C. D ⊆ A} by auto
with assms(1) have D ∈ F unfolding IsBaseFilter_def by auto
moreover 
{ 
assume 0 ∈ F
with assms(1) have ∃ D ∈ C. D ⊆ 0 using basic_element_filter by simp
then obtain D where D ∈ C by auto
with assms(2) have False using SatisfiesFilterBase_def by auto
}
then have 0 /∈ F by auto
ultimately show thesis using IsFilter_def by auto
qed

A base filter satisfies the filter base condition.

theorem basic_filter_2: 
assumes C {is a base filter} F and F {is a filter on} ∪ F
shows C {satisfies the filter base condition}
proof-
{ 
fix A B 
assume AF: A ∈ C and BF: B ∈ C
then have A ∈ F and B ∈ F using assms(1) IsBaseFilter_def by auto
then have A ∩ B ∈ F using assms(2) IsFilter_def by auto
then have ∃ D ∈ C. D ⊆ A ∩ B using assms(1) basic_element_filter by blast
}
then have ∀ A ∈ C. ∀ B ∈ C. ∃ D ∈ C. D ⊆ A ∩ B by auto
moreover 
{ 
assume 0 ∈ C

892
then have $0 \in \mathcal{F}$ using assms(1) IsBaseFilter_def by auto
then have False using assms(2) IsFilter_def by auto

} then have $0 \not\in C$ by auto
moreover
{ assume $C=0$
then have $\mathcal{F}=0$ using assms(1) IsBaseFilter_def by auto
then have False using assms(2) IsFilter_def by auto
}
then have $C \neq 0$ by auto
ultimately show thesis using SatisfiesFilterBase_def by auto
qed

A base filter for a collection satisfies the filter base condition iff that collection is in fact a filter.

**Theorem basic_filter:**
assumes C {is a base filter}
shows (C {satisfies the filter base condition}) $\iff$ (C {is a filter on} $\bigcup C$)
using assms basic_filter_1 basic_filter_2 by auto

A base for a filter determines a filter up to the underlying set.

**Theorem base_unique_filter:**
assumes C {is a base filter} $\mathcal{F}$1 and C {is a base filter} $\mathcal{F}$2
shows $\mathcal{F}$1=$\mathcal{F}$2 $\iff$ $\bigcup C_1=\bigcup C_2$
using assms unfolding IsBaseFilter_def by auto

Suppose that we take any nonempty collection $C$ of subsets of some set $X$.
Then this collection is a base filter for the collection of all supersets (in $X$) of sets from $C$.

**Theorem base_unique_filter_set1:**
assumes $C \subseteq \text{Pow}(X)$ and $C \neq 0$
such that $\forall A \in \text{Pow}(X) \exists D \in C. D \subseteq A$ and $\bigcup \{A \in \text{Pow}(X) \exists D \in C. D \subseteq A\}=X$
proof-
from assms(1) have $C \subseteq \{A \in \text{Pow}(X) \exists D \in C. D \subseteq A\}$ by auto
moreover
from assms(2) obtain $D$ where $D \in C$ by auto
then have $D \subseteq X$ using assms(1) by auto
with $D \in C$ have $X \in \{A \in \text{Pow}(X) \exists D \in C. D \subseteq A\}$ by auto
then show $\bigcup \{A \in \text{Pow}(X) \exists D \in C. D \subseteq A\}=X$ by auto
ultimately
show C {is a base filter} $\{A \in \text{Pow}(X) \exists D \in C. D \subseteq A\}$ using IsBaseFilter_def by auto
qed

A collection $C$ that satisfies the filter base condition is a base filter for some other collection $\mathcal{F}$ iff $\mathcal{F}$ is the collection of supersets of $C$.
theorem base_unique_filter_set2:
  assumes C ⊆ Pow(X) and C {satisfies the filter base condition}
  shows ((C {is a base filter} F) ∧ ∪F=X) ←→ F={A∈Pow(X). ∃D∈C. D⊆A}
  using assms IsBaseFilter_def SatisfiesFilterBase_def base_unique_filter_set1
  by auto

A simple corollary from the previous lemma.

corollary base_unique_filter_set3:
  assumes C ⊆ Pow(X) and C {satisfies the filter base condition}
  shows C {is a base filter} {A∈Pow(X). ∃D∈C. D⊆A} and ∪{A∈Pow(X).}
proof -
  let F = {A∈Pow(X). ∃D∈C. D⊆A}
  from assms have (C {is a base filter} F) ∧ ∪F=X
  using base_unique_filter_set2 by simp
  thus C {is a base filter} F and ∪F = X
  by auto
qed

The convergence for filters is much easier concept to write. Given a topology and a filter on the same underlying set, we can define convergence as containing all the neighborhoods of the point.

definition (in topology0)
  FilterConverges (_ ⟷ 50) where
  F{is a filter on} T =⇒ F→F x ≡
  x∈T ∧ {U∈Pow(T). x∈int(U)} ⊆ F

The neighborhoods of a point form a filter that converges to that point.

lemma (in topology0) neigh_filter:
  assumes x∈T
  defines Neigh≡{U∈Pow(T). x∈int(U)}
  shows Neigh {is a filter on} T and Neigh→T x
proof -
  { fix A B
    assume p:A∈Neigh B∈Neigh
    have A∩B∈Neigh
    proof -
      from p have A∩B∈Pow(T) using Neigh_def by auto
      moreover
      { from p have x∈int(A) x∈int(B) using Neigh_def by auto
        then have x∈int(A∩int(B)) by auto
        moreover
        { have int(A)∩int(B)⊆A∩B using Top_2_L1 by auto
          moreover have int(A)∩int(B)∈T
            using Top_2_L2 topSpaceAssum IsATopology_def by blast
          ultimately have int(A∩int(B))⊆int(A∩B) using Top_2_L5 by auto
          ultimately have x∈int(A∩B) by auto
        }
      }
    }
  }

894
ultimately show thesis using Neigh_def by auto
qed

moreover
{
fix A B
assume A: A∈Neigh and B: B∈Pow(∪T) and sub: A⊆B
from sub have int(A)∈T int(A)⊆B using Top_2_L2 Top_2_L1
by auto
then have int(A)⊆int(B) using Top_2_L5 by auto
with A have x∈int(B) using Neigh_def by auto
with B have B∈Neigh using Neigh_def by auto
}
moreover
{
assume 0∈Neigh
then have x∈Interior(0,T) using Neigh_def by auto
then have x∈0 using Top_2_L1 by auto
then have False by auto
}
then have 0∉Neigh by auto
moreover
have ∪T∈T using topSpaceAssum IsATopology_def by auto
then have Interior(∪T,T)=∪T using Top_2_L3 by auto
with assms(1) have ab: ∪T∈Neigh unfolding Neigh_def by auto
moreover have Neigh⊆Pow(∪T) using Neigh_def by auto
ultimately show Neigh {is a filter on} ∪T using IsFilter_def
by auto
moreover from ab have ∪Neigh=∪T unfolding Neigh_def by auto
ultimately show Neigh →_F x using FilterConverges_def assms(1) Neigh_def
by auto
qed

Note that with the net we built in a previous result, it wasn’t clear that we
could construct an actual net that converged to the given point without the
axiom of choice. With filters, there is no problem.

Another positive point of filters is due to the existence of filter basis. If
we have a basis for a filter, then the filter converges to a point iff every
neighborhood of that point contains a basic filter element.

theorem (in topology0) convergence_filter_base1:

assumes ⪯ {is a filter on} ∪T and C {is a base filter} ⪯ and ⪯ →_F x
shows ∀U∈Pow(∪T). x∈int(U) → (∃D∈C. D⊆U) and x∈∪T

proof -
{
fix U
assume U⊆ (∪T) and x∈int(U)
with assms(1,3) have U∈⋯ using FilterConverges_def by auto
with assms(2) have \( \exists D \in C. D \subseteq U \) using basic_element_filter by blast
} thus \( \forall U \in \text{Pow}(U). x \in \text{int}(U) \rightarrow (\exists D \in C. D \subseteq U) \) by auto
from assms(1,3) show \( x \in \bigcup T \) using FilterConverges_def by auto qed

A sufficient condition for a filter to converge to a point.

theorem (in topology0) convergence_filter_base2:
assumes \( \mathcal{F} \) {is a filter on} \( \bigcup T \) and \( C \) {is a base filter} \( \mathcal{F} \)
and \( \forall U \in \text{Pow}(U). x \in \text{int}(U) \rightarrow (\exists D \in C. D \subseteq U) \) and \( x \in \bigcup T \)
shows \( \mathcal{F} \rightarrow_F x \)
proof
{ fix \( U \)
  assume AS: \( U \in \text{Pow}(U) \) \( x \in \text{int}(U) \)
  then obtain \( D \) where \( pD:D \in C \) and \( s:D \subseteq U \) using assms(3) by blast
  with assms(2) AS have \( D \in F \) and \( D \subseteq U \) and \( U \in \text{Pow}(U) \)
  using IsBaseFilter_def by auto
  with assms(1) have \( U \in \mathcal{F} \) using IsFilter_def by auto
} with assms(1,4) show thesis using FilterConverges_def by auto qed

A necessary and sufficient condition for a filter to converge to a point.

theorem (in topology0) convergence_filter_base_eq:
assumes \( \mathcal{F} \) {is a filter on} \( \bigcup T \) and \( C \) {is a base filter} \( \mathcal{F} \)
shows \( (\mathcal{F} \rightarrow_F x) \iff ((\forall U \in \text{Pow}(U). x \in \text{int}(U) \rightarrow (\exists D \in C. D \subseteq U)) \land x \in \bigcup T) \)
proof
assume \( \mathcal{F} \rightarrow_F x \)
with assms show \( ((\forall U \in \text{Pow}(U). x \in \text{int}(U) \rightarrow (\exists D \in C. D \subseteq U)) \land x \in \bigcup T) \)
using convergence_filter_base1 by simp
next
assume \( (\forall U \in \text{Pow}(U). x \in \text{int}(U) \rightarrow (\exists D \in C. D \subseteq U)) \land x \in \bigcup T \)
with assms show \( \mathcal{F} \rightarrow_F x \) using convergence_filter_base2 by auto
qed

71.3 Relation between nets and filters

In this section we show that filters do not generalize nets, but still nets and filter are in a way equivalent as far as convergence is considered.

Let’s build now a net from a filter, such that both converge to the same points.

definition
NetOffFilter (Net(_)) 40 where
\( \mathcal{F} \) {is a filter on} \( \bigcup \mathcal{F} \) \( \Rightarrow \) Net(\( \mathcal{F} \)) \equiv
\{\{(A,\text{fst}(A)). A \in (\bigcup \mathcal{F}) \times \mathcal{F}. x \in \mathcal{F}\}\},\{(A,B)\in(\bigcup \mathcal{F}) \times \mathcal{F}. x \in \mathcal{F}\} \times \{(x,F)\in(\bigcup \mathcal{F}) \times \mathcal{F}. x \in \mathcal{F}\}. \text{snd}(B) \subseteq \text{snd}(A)\}

896
Net of a filter is indeed a net.

**Theorem net_of_filter_is_net:**

assumes \( \mathcal{F} \) {is a filter on} \( X \)

shows \( \text{Net}(\mathcal{F}) \) {is a net on} \( X \)

**Proof:**

- from assumptions have \( X \in \mathcal{F} \) \( \mathcal{F} \subseteq \text{Pow}(X) \) using IsFilter_def by auto

- let \( f = \{ (A, \text{fst}(A)) \mid A \in (\bigcup \mathcal{F}) \times \mathcal{F}, x \in F \} \)

- let \( r = \{ (A, B) \in (\bigcup \mathcal{F}) \times \mathcal{F} \mid x \in F \} \times \{ (x, F) \in (\bigcup \mathcal{F}) \times \mathcal{F} \mid x \in F \}. \text{snd}(B) \subseteq \text{snd}(A) \}

- have function(f) using function_def by auto

- moreover have relation(f) using relation_def by auto

- ultimately have \( f: \text{domain}(f) \rightarrow \text{range}(f) \) using function_imp_Pi by auto

- have \( \text{dom}(f) = (\bigcup \mathcal{F}) \times \mathcal{F} \) by auto with \( \langle f: \text{domain}(f) \rightarrow \text{range}(f) \rangle \) have \( f: \text{domain}(f) \rightarrow \bigcup \mathcal{F} \) using fun_weaken_type by auto

- moreover

  - { fix \( t \)
    assume \( \text{pp} : t \in \text{domain}(f) \)
    then have \( \text{snd}(t) \subseteq \text{snd}(t) \) by auto
    with \( \text{dom} \text{ pp} \) have \( \langle t, t \rangle \in r \) by auto
  }

  - then have refl(domain(f), r) using refl_def by auto

  - moreover

    - { fix \( t_1 t_2 t_3 \)
      assume \( \langle t_1, t_2 \rangle \in r \langle t_2, t_3 \rangle \in r \)
      then have \( \text{snd}(t_3) \subseteq \text{snd}(t_1) t_1 \in \text{domain}(f) t_3 \in \text{domain}(f) \) using \( \text{dom} \) by auto
      then have \( \langle t_1, t_3 \rangle \in r \) by auto
    }

    - then have trans(r) using trans_def by auto

    - moreover

      - { fix \( x y \)
        assume \( \text{as} : x \in \text{domain}(f) y \in \text{domain}(f) \)
        then have \( \text{snd}(x) \in \mathcal{F} \text{snd}(y) \in \mathcal{F} \) by auto
        then have \( \text{p: snd}(x) \cap \text{snd}(y) \in \mathcal{F} \) using assumptions IsFilter_def by auto
        { assume \( \text{snd}(x) \cap \text{snd}(y) = 0 \)
          with \( \text{p} \) have \( 0 \in \mathcal{F} \) by auto
          then have \( \text{False} \) using assumptions IsFilter_def by auto
        }
      }

      - then have \( \text{snd}(x) \cap \text{snd}(y) \neq 0 \) by auto

      - then obtain \( xy \) where \( xy \in \text{snd}(x) \cap \text{snd}(y) \) by auto

      - then have \( xy \in \text{snd}(x) \cap \text{snd}(y) (xy, \text{snd}(x) \cap \text{snd}(y)) \in (\bigcup \mathcal{F}) \times \mathcal{F} \) using \( \text{p} \)
by auto
then have \( \langle xy, \text{snd}(x) \cap \text{snd}(y) \rangle \in \{ \langle x, F \rangle \in (\bigcup F) \times F. \ x \in F\} \) by auto
with dom have d: \( \langle xy, \text{snd}(x) \cap \text{snd}(y) \rangle \in \text{domain}(f) \) by auto
with as have \( \langle x, \langle xy, \text{snd}(x) \cap \text{snd}(y) \rangle \rangle \in r \) \( \land \langle y, \langle xy, \text{snd}(x) \cap \text{snd}(y) \rangle \rangle \in r \) by auto
with d have \( \exists z \in \text{domain}(f). \langle x, z \rangle \in r \) \( \land \langle y, z \rangle \in r \) by auto
ultimately have \( r \text{ directs } \text{domain}(f) \) using IsDirectedSet_def by blast
moreover
\{ have p\:X \in F and 0 \notin F using assms IsFilter_def by auto
then have X \neq 0 by auto
then obtain q where q \in X by auto
with p dom have \( \langle q, X \rangle \in \text{domain}(f) \) by auto
then have \( \text{domain}(f) \neq 0 \) by blast
\}
ultimately have \( \langle f, r \rangle \) \( \{ \text{is a net on} \} \bigcup F \) using IsNet_def by auto
then show \( (\text{Net}(\bigcup F)) \) \( \{ \text{is a net on} \} X \) using NetOfFilter_def assms uu by auto
qed

If a filter converges to some point then its net converges to the same point.

theorem (in topology0) filter_conver_net_of_filter_conver:
assumes \( \bigcup T \in F \) and \( F \to F \) \( x \)
shows \( (\text{Net}(\bigcup F)) \) \( \to_N x \)
proof-
from assms have \( \bigcup T \in F \) \( \subseteq \text{Pow}(\bigcup T) \) using IsFilter_def by auto
then have uu: \( \bigcup T = \bigcup T \) by blast
from assms(1) have func: \( \text{fst}(\text{Net}(\bigcup F)) = \{\langle A, \text{fst}(A) \rangle. \ A \in \{\langle x, F \rangle \in (\bigcup F) \times F. \ x \in F\}\} \)
and dir: \( \text{snd}(\text{Net}(\bigcup F)) = \{\langle A, B \rangle \in \{\langle x, F \rangle \in (\bigcup F) \times F. \ x \in F\} \times \{\langle x, F \rangle \in (\bigcup F) \times F. \ x \in F\}. \ x \in F\}. \ x \in F\} \)
using NetOfFilter_def uu by auto
then have dom_def: \( \text{domain}(\text{fst}(\text{Net}(\bigcup F))) = \{\langle x, F \rangle \in (\bigcup F) \times F. \ x \in F\} \) by auto
from func have fun: \( \text{fst}(\text{Net}(\bigcup F)) = \{\langle x, F \rangle \in (\bigcup F) \times F. \ x \in F\} \to (\bigcup F) \)
using ZF_fun_from_total by simp
from assms(1) have NN: \( (\text{Net}(\bigcup F)) \) \( \{ \text{is a net on} \} \bigcup T \) using net_of_filter_is_net
by auto
moreover from assms have \( x \in \bigcup T \) using FilterConverges_def by auto
moreover
\{ fix U
assume AS: \( U \subseteq \text{Pow}(\bigcup T) \) \( x \in \text{int}(U) \)
with assms have \( U \subseteq F \) \( x \in U \) using Top_2_L1 FilterConverges_def by auto

898

then have pp: \((x, U) \in \text{domain}(\text{fst}(\text{Net}(\emptyset)))\) using dom_def by auto
{
  fix \(m\)
  assume ASS: \(m \in \text{domain}(\text{fst}(\text{Net}(\emptyset)))\)
  from ASS(1) fun func have \(\text{fst}(\text{Net}(\emptyset))(m) = \text{fst}(m)\)
  using func1_1_L1 ZF_fun_from_tot_val by simp
  with dir ASS have \(\text{fst}(\text{Net}(\emptyset))(m) \in U\) using dom_def by auto
}
then have \(\forall m \in \text{domain}(\text{fst}(\text{Net}(\emptyset))). (\langle x, U \rangle, m) \in \text{snd}(\text{Net}(\emptyset)) \rightarrow \text{fst}(\text{Net}(\emptyset))m \in U\)
by auto
with pp have \(\exists t \in \text{domain}(\text{fst}(\text{Net}(\emptyset))). \forall m \in \text{domain}(\text{fst}(\text{Net}(\emptyset))). (\langle t, m \rangle \in \text{snd}(\text{Net}(\emptyset))) \rightarrow \text{fst}(\text{Net}(\emptyset))m \in U\)
by auto
ultimately show thesis using NetConverges_def by auto
qed

If a net converges to a point, then a filter also converges to a point.

**Theorem (in topology0)** net_of_filter_conver_filter_conver:
assumes \(\emptyset\) {is a filter on} \(\bigcup T\) and \((\text{Net}(\emptyset)) \rightarrow_N x\)
shows \(\emptyset \rightarrow_F x\)
**Proof**
from assms have \(\bigcup T \in \emptyset \subseteq \text{Pow}(\bigcup T)\) using IsFilter_def by auto
then have uu: \(\bigcup T = \bigcup T\) by blast
have \(x \in \bigcup T\) using assms NetConverges_def net_of_filter_is_net by auto
moreover
\{|fix U assume U \in \text{Pow}(\bigcup T) x \in \text{int}(U) then obtain t where t: t \in \text{domain}(\text{fst}(\text{Net}(\emptyset))) and reg: \forall m \in \text{domain}(\text{fst}(\text{Net}(\emptyset))). (t, m) \in \text{snd}(\text{Net}(\emptyset)) \rightarrow \text{fst}(\text{Net}(\emptyset))m \in U|
using assms net_of_filter_is_net NetConverges_def by blast
with assms(1) uu obtain ti t2 where t_def: t = (ti,t2) and ti \in t2 and tFF: t2 \in \emptyset
using NetOfFilter_def by auto
\{|fix s assume s \in t2 then have \(\langle s, t2 \rangle \in \bigcup \emptyset \times \emptyset\). q1 \in q2\) using tFF by auto
moreover
from assms(1) uu have domain(\text{fst}(\text{Net}(\emptyset))) = \bigcup \emptyset \times \emptyset. q1 \in q2\}
using NetOfFilter_def by auto
ultimately have \(\langle s, t2 \rangle \in \text{domain}(\text{fst}(\text{Net}(\emptyset)))\) by auto

999
moreover
from assms(1) uu t t_def tt have \( \langle t1, t2 \rangle, \langle s, t2 \rangle \rangle \in \text{snd}(\text{Net}(\mathcal{F})) \) using NetOfFilter_def
  by auto
ultimately
have \( \text{fst}(\text{Net}(\mathcal{F})) \langle s, t2 \rangle \in U \) using reg t_def by auto
moreover
from assms(1) uu have function(fst(\text{Net}(\mathcal{F}))) using NetOfFilter_def
function_def
  by auto
moreover
from tt assms(1) uu have \( \langle \langle s, t2 \rangle, s \rangle \rangle \in \text{fst}(\text{Net}(\mathcal{F})) \) using NetOfFilter_def
by auto
ultimately
have \( s \in U \) using NetOfFilter_def function_apply_equality by auto
\}
then have \( t2 \subseteq U \) by auto
with tFF assms(1) \( \langle U \subseteq \bigcup T \rangle \) have \( U \subseteq \mathcal{F} \) using IsFilter_def by auto
ultimately
show thesis using FilterConverges_def assms(1) by auto
qed

A filter converges to a point if and only if its net converges to the point.

\textbf{Theorem (in topology0) filter_conver_iff_net_of_filter_conver:}
assumes \( \mathcal{F} \) \{is a filter on\} \( \bigcup T \)
shows \( (\mathcal{F} \to x) \iff (\text{Net}(\mathcal{F}) \to x) \)
  using filter_conver_net_of_filter_conver net_of_filter_conver_filter_conver_assms
by auto

The previous result states that, when considering convergence, the filters do not generalize nets. When considering a filter, there is always a net that converges to the same points of the original filter.

Now we see that with nets, results come naturally applying the axiom of choice; but with filters, the results come, may be less natural, but with no choice. The reason is that \( \text{Net}(\mathcal{F}) \) is a net that doesn’t come into our attention as a first choice; maybe because we restrict ourselves to the anti-symmetry property of orders without realizing that a directed set is not an order.

The following results will state that filters are not just a subclass of nets, but that nets and filters are equivalent on convergence: for every filter there is a net converging to the same points, and also, for every net there is a filter converging to the same points.

\textbf{Definition}
  \text{FilterOfNet (Filter \(_\_\_\_\_) 40 where}
\( (N \text{ is a net on } X) \implies \text{Filter } N .. X \equiv \{ A \in \text{Pow}(X). \exists D \subseteq \{ \{ \text{fst}(N) \text{snd}(s). s \in \text{domain}(\text{fst}(N)) \times \text{domain}(\text{fst}(N)). s \in \text{snd}(N) \land \text{fst}(s) = t0 \}. t0 \in \text{domain}(\text{fst}(N)) \}. D \subseteq A \} \)

Filter of a net is indeed a filter

**Theorem filter_of_net_is_filter:**

assumes \( N \text{ is a net on } X \)

shows \( \text{Filter } N .. X \text{ is a filter on } X \) and

\[ \{ \{ \text{fst}(N) \text{snd}(s). s \in \{ s \in \text{domain}(\text{fst}(N)) \times \text{domain}(\text{fst}(N)). s \in \text{snd}(N) \land \text{fst}(s) = t0 \}. t0 \in \text{domain}(\text{fst}(N)) \} \text{ is a base filter} \} \text{ (Filter } N .. X) \]

**Proof**

let \( C = \{ \{ \text{fst}(N) \text{snd}(s). s \in \{ s \in \text{domain}(\text{fst}(N)) \times \text{domain}(\text{fst}(N)). s \in \text{snd}(N) \land \text{fst}(s) = t0 \}. t0 \in \text{domain}(\text{fst}(N)) \} \)

have \( C \subseteq \text{Pow}(X) \)

proof

- fix \( t \)
  assume \( t \in C \)
  then obtain \( t1 \) where \( t1 \in \text{domain}(\text{fst}(N)) \) and
    \( t = \{ \text{fst}(N) \text{snd}(s). s \in \{ s \in \text{domain}(\text{fst}(N)) \times \text{domain}(\text{fst}(N)). s \in \text{snd}(N) \land \text{fst}(s) = t1 \} \}
    by auto
  
  \hspace{1cm} fix \( x \)
    assume \( x \in t \)
    with \( t \text{ Def: } t = \{ \text{fst}(N) \text{snd}(s). s \in \{ s \in \text{domain}(\text{fst}(N)) \times \text{domain}(\text{fst}(N)). s \in \text{snd}(N) \land \text{fst}(s) = t1 \} \}
    by auto
  
  \hspace{2cm} with \( s \text{ Def: } s \in \text{domain}(\text{fst}(N)) \times \text{domain}(\text{fst}(N)) \land \text{fst}(s) = t1 \) and
    \( x \text{ Def: } x = \text{fst}(N) \text{snd}(ss)\) by blast
    then have \( \text{snd}(ss) \in \text{domain}(\text{fst}(N)) \) by auto
  
  \hspace{3cm} from \( \text{assms have } \text{fst}(N) : \text{domain}(\text{fst}(N)) \to X \) unfolding \text{IsNet_def}

by simp

- with \( \langle \text{snd}(ss) \in \text{domain}(\text{fst}(N)) \rangle \) have \( x \in X \) using apply_funtype

x \_ def
  by auto

\hspace{1cm} hence \( t \subseteq X \) by auto

\hspace{2cm} thus \( \text{thesis} \) by blast

qed

have \( \text{sat: } C \text{ satisfies the filter base condition} \)

**Proof**

- from \( \text{assms obtain } t1 \) where \( t1 \in \text{domain}(\text{fst}(N)) \) using \text{IsNet_def} by blast

  \hspace{1cm} hence \( \{ \text{fst}(N) \text{snd}(s). s \in \{ s \in \text{domain}(\text{fst}(N)) \times \text{domain}(\text{fst}(N)). s \in \text{snd}(N) \land \text{fst}(s) = t1 \} \} \subseteq C \)

  \hspace{2cm} by \( \text{auto} \)

  \hspace{3cm} hence \( C \neq 0 \) by \( \text{auto} \)

  \hspace{3.5cm} moreover

  \hspace{4cm} 

fix U
assume U∈C
then obtain q where q_dom: q∈domain(fst(N)) and
U_def: U={fst(N)snd(s). s∈{s∈domain(fst(N))×domain(fst(N)). s∈snd(N)}
∧ fst(s)=q}
by blast
with assms have \(q,q\)∈snd(N) ∧ fst((q,q))=q unfolding IsNet_def
IsDirectedSet_def refl_def
by auto
with q_dom have \(q,q\)∈{s∈domain(fst(N))×domain(fst(N)). s∈snd(N)}
∧ fst(s)=q}
by auto
with U_def have fst(N)(snd((q,q))) ∈ U by blast
hence U≠0 by auto
\}
then have 0∉C by auto
moreover
have ∀A∈C. ∀B∈C. (∃D∈C. D⊆A∩B)
proof
fix A
assume pA: A∈C
show ∀B∈C. ∃D∈C. D⊆A∩B
proof
\{ 
fix B
assume B∈C
with pA obtain qA qB where per: qA∈domain(fst(N)) qB∈domain(fst(N))
and
A_def: A={fst(N)snd(s). s∈{s∈domain(fst(N))×domain(fst(N)). s∈snd(N)}
∧ fst(s)=qA}
B_def: B={fst(N)snd(s). s∈{s∈domain(fst(N))×domain(fst(N)). s∈snd(N)}
∧ fst(s)=qB}
by blast
have dir: snd(N) directs domain(fst(N)) using assms IsNet_def
by auto
with per obtain qD where ine: (qA,qD)∈snd(N) (qB,qD)∈snd(N)
and
perD: qD∈domain(fst(N)) unfolding IsDirectedSet_def
by blast
let D = {fst(N)snd(s). s∈{s∈domain(fst(N))×domain(fst(N)). s∈snd(N)}
∧ fst(s)=qD})
from perD have D∈C by auto
moreover
\{ 
fix d
assume d∈D
then obtain sd where sd∈{s∈domain(fst(N))×domain(fst(N)).
∧ fst(s)=qD} and

d_def: d=fst(N)snd(sd) by blast
then have \(sdN: sa \in \text{snd}(N)\) and \(qdd: \text{fst}(sd) = qD\) and \(sd \in \text{domain}(\text{fst}(N)) \times \text{domain}(\text{fst}(N))\)

by auto
then obtain \(qI\) aa where \(sd = \langle aa, qI \rangle\)

by auto
with \(sdN\) have \(\langle qD, qI \rangle \in \text{snd}(N)\)

by auto
then have \(\text{sd_def}: sd = \langle qD, qI \rangle\)

by auto
with \(qdd\) have \(\text{qIdom}: qI \in \text{domain}(\text{fst}(N))\)

by auto
with \(\text{dir}\) have \(\text{trans(snd}(N))\)

unfolding \(\text{IsDirectedSet_def}\)

by auto
then have \(\langle qA, qD \rangle, \langle qD, qI \rangle \in \text{snd}(N)\) \(\rightarrow \langle qA, qI \rangle \in \text{snd}(N)\)

and
\(\langle qB, qQ \rangle \in \text{snd}(N)\) \(\rightarrow \langle qD, qI \rangle \in \text{snd}(N)\)

using \(\text{trans_def}\) by auto
with \(\text{ine} \langle qD, qI \rangle \in \text{snd}(N)\) have \(\langle qA, qI \rangle \in \text{snd}(N)\)

by auto
with \(\text{sdN}\) have \(\langle qA, qD \rangle \in \text{snd}(N)\)

\(\times \text{domain}(\text{fst}(N))\)

by auto
then have \(\langle qB, qI \rangle \in \text{domain}(\text{fst}(N)) \times \text{domain}(\text{fst}(N))\).

\(s \in \text{snd}(N)\) \(\land \text{fst}(s) = qA\) \(\rightarrow \langle qB, qI \rangle \in \text{domain}(\text{fst}(N)) \times \text{domain}(\text{fst}(N))\).

\(s \in \text{snd}(N)\) \(\land \text{fst}(s) = qB\)

by auto
then have \(\text{fst}(N)(qI) \in A \cap B\) using \(A\_\text{def} B\_\text{def}\) by auto
then have \(\text{fst}(N)(\text{snd}(sd)) \in A \cap B\) using \(\text{sd_def}\) by auto
then have \(d \in A \cap B\) using \(d\_\text{def}\) by auto
}

ultimately show \(\exists D \subseteq C. D \subseteq A \cap B\) by blast

qed

ultimately
show \(\text{thesis}\) unfolding \(\text{SatisfiesFilterBase_def}\) by blast

qed

Base: \(C\) {is a base filter} \(\{A \in \text{Pow}(X). \exists D \subseteq C. D \subseteq A\}\) \(\cup \{A \in \text{Pow}(X). \exists D \subseteq C. D \subseteq A\} = X\)

proof -
from \(\langle C \subseteq \text{Pow}(X)\rangle\) sat show \(C\) {is a base filter} \(\{A \in \text{Pow}(X). \exists D \subseteq C. D \subseteq A\}\)

by (rule \(\text{base_unique_filter_set3}\))
from \(\langle C \subseteq \text{Pow}(X)\rangle\) sat show \(\bigcup \{A \in \text{Pow}(X). \exists D \subseteq C. D \subseteq A\} = X\)

by (rule \(\text{base_unique_filter_set3}\))

qed

with sat show \((\text{Filter } N..X)\) {is a filter on} \(X\)

using sat basic_filter FilterOfNet_def assms by auto
from \(\text{Base(1)}\) show \(C\) {is a base filter} \((\text{Filter } N..X)\)

using FilterOfNet_def assms by auto

qed
Convergence of a net implies the convergence of the corresponding filter.

**theorem** (in topology0) net_conver_filter_of_net_conver:
assumes N {is a net on} \( \bigcup T \) and N \( \to_N x \)
shows (Filter N..(\( \bigcup T \))) \( \to_F x \)

**proof** -
let C = \{\((\text{fst}(N)\text{snd}(s)). s \in \{s \in \text{domain}(\text{fst}(N)) \times \text{domain}(\text{fst}(N)). s \in \text{snd}(N) \wedge \text{fst}(s)=t\}\}. 
t \in \text{domain}(\text{fst}(N))
from assms(1) have (Filter N..(\( \bigcup T \))) \( \to_F \) \( \bigcup T \) and C \( \to_F \)
Filter N..(\( \bigcup T \))
using filter_of_net_is_filter by auto
moreover have \( \forall U \in \text{Pow}(\bigcup T). x \in \text{int}(U) \to (\exists D \in C. D \subseteq U) \)
proof -
{ fix U 
assume U \in \text{Pow}(\bigcup T) x \in \text{int}(U) 
with assms have \( \exists t \in \text{domain}(\text{fst}(N)). (\forall m \in \text{domain}(\text{fst}(N)). (\langle t,m \rangle \in \text{snd}(N) \to \text{fst}(N)m \in U) \)
using NetConverges_def by auto 
then obtain t where t \in \text{domain}(\text{fst}(N)) and reg: \( \forall m \in \text{domain}(\text{fst}(N)). (\langle t,m \rangle \in \text{snd}(N) \to \text{fst}(N)m \in U) \) by auto 
{ fix f 
assume f \in \{\text{fst}(N)\text{snd}(s). s \in \{s \in \text{domain}(\text{fst}(N)) \times \text{domain}(\text{fst}(N)). s \in \text{snd}(N) \wedge \text{fst}(s)=t\}. 
s \in \text{snd}(N) \wedge \text{fst}(s)=t\}
then obtain s where s \in \{s \in \text{domain}(\text{fst}(N)) \times \text{domain}(\text{fst}(N)). s \in \text{snd}(N) \wedge \text{fst}(s)=t\} and f_def: f=\text{fst}(N)\text{snd}(s) by blast 
hence s \in \text{domain}(\text{fst}(N)) \times \text{domain}(\text{fst}(N)) and s \in \text{snd}(N) and \text{fst}(s)=t by auto 
by auto 
hence s=(t,\text{snd}(s)) and \text{snd}(s) \in \text{domain}(\text{fst}(N)) by auto 
with \langle s \in \text{snd}(N) \rangle reg have \text{fst}(N)\text{snd}(s) \in U by auto 
with f_def have f \in U by auto 
} 
{ fix U 
assume U \in \text{Pow}(\bigcup T) x \in \text{int}(U) 
with assms have \( \exists t \in \text{domain}(\text{fst}(N)). (\forall m \in \text{domain}(\text{fst}(N)). (\langle t,m \rangle \in \text{snd}(N) \to \text{fst}(N)m \in U) \)
using NetConverges_def by auto 
then obtain t where t \in \text{domain}(\text{fst}(N)) and reg: \( \forall m \in \text{domain}(\text{fst}(N)). (\langle t,m \rangle \in \text{snd}(N) \to \text{fst}(N)m \in U) \) by auto 
thus \( \forall U \in \text{Pow}(\bigcup T). x \in \text{int}(U) \to (\exists D \subseteq C. D \subseteq U) \) by auto 
} 
qued 
moreover from assms have x \in \bigcup T using NetConverges_def by auto 
ultimately show (Filter N..(\( \bigcup T \))) \( \to_F x \) by (rule convergence_filter_base2)
qued 

Convergence of a filter corresponding to a net implies convergence of the net.
theorem (in topology0) filter_of_net_conver_net_conver:
assumes N {is a net on} ⋃T and (Filter N..(⋃T)) →F x
shows N →N x
proof -
let C = \{fst(N)snd(s). s∈\{s∈domain(fst(N))×domain(fst(N)). s∈snd(N)
∧ fst(s)=t\}. t∈domain(fst(N))\}
from asm have I: (Filter N..(⋃T)) {is a filter on} (⋃T)
C {is a base filter} (Filter N..(⋃T)) (Filter N..(⋃T)) →F x
using filter_of_net_is_filter by auto
then have reg: ∀U∈Pow(⋃T). x∈int(U) −→ (∃D∈C. D⊆U)
by (rule convergence_filter_base1)
from I have x∈⋃T by (rule convergence_filter_base1)
major premise
moreover
{ fix U
assume U∈Pow(⋃T) x∈int(U)
with reg have ∃D∈C. D⊆U by auto
then obtain D where D∈C D⊆U
by auto
then obtain td where td∈domain(fst(N)) and
D_def: D={fst(N)snd(s). s∈\{s∈domain(fst(N))×domain(fst(N)). s∈snd(N)
∧ fst(s)=td\}. td∈domain(fst(N)) and
∧ fst(s)=td\})
by auto
{ fix m
assume m∈domain(fst(N)) ⟨td,m⟩∈snd(N)
with ⟨td∈domain(fst(N))⟩ have
⟨td,m⟩∈\{s∈domain(fst(N))×domain(fst(N)). s∈snd(N) ∧ fst(s)=td\}
by auto
with D_def have fst(N)m∈D by auto
with ⟨D∈U⟩ have fst(N)m∈U by auto
}
then have ∀m∈domain(fst(N)). ⟨td,m⟩∈snd(N) −→ fst(N)m∈U by auto
with ⟨td∈domain(fst(N))⟩ have
∃t∈domain(fst(N)). ∀m∈domain(fst(N)). ⟨t,m⟩∈snd(N) −→ fst(N)m∈U
by auto
}
then have ∀U∈Pow(⋃T). x∈int(U) −→
(∃t∈domain(fst(N)). ∀m∈domain(fst(N)). ⟨t,m⟩∈snd(N) −→ fst(N)m∈U)
by auto
ultimately show thesis using NetConverges_def asms(1) by auto
qed

Filter of net converges to a point x if and only the net converges to x.

theorem (in topology0) filter_of_net_conv_iff_net_conv:
assumes N {is a net on} ⋃T
shows ((Filter N..(⋃T)) →F x) −→ (N →N x)
We know now that filters and nets are the same thing, when working convergence of topological spaces. Sometimes, the nature of filters makes it easier to generalized them as follows.

Instead of considering all subsets of some set $X$, we can consider only open sets (we get an open filter) or closed sets (we get a closed filter). There are many more useful examples that characterize topological properties.

This type of generalization cannot be done with nets.

Also a filter can give us a topology in the following way:

**Theorem** `top_of_filter`:

```
assumes $F$ to be a filter on $\bigcup F$
shows $(\bigcup F \cup \{0\})$ is a topology
```

**Proof** -

```
\{ fix A B 
  assume A \in (\bigcup F \cup \{0\}) \land B \in (\bigcup F \cup \{0\})
  then have \((A \land B) \lor (A \land B = 0)\) by auto
  with assms have A \land B \in (\bigcup F \cup \{0\}) unfolding IsFilter_def
  by blast
\}
then have \( \forall A \in (\bigcup F \cup \{0\}). \forall B \in (\bigcup F \cup \{0\}). A \land B \in (\bigcup F \cup \{0\}) \) by auto
moreover 
\{ fix M 
  assume A : M \subseteq Pow(\bigcup F \cup \{0\})
  then have M = 0 \lor \forall \exists T \in M. T \in F \lor (M \subseteq F) by blast
  then have \( \bigcup M = 0 \lor \forall \exists T \in M. T \subseteq F \lor (M \subseteq F) \) by auto
  then obtain T where \( \bigcup M = 0 \lor \forall \exists T \in M. T \subseteq F \lor (M \subseteq F) \) by auto
  then have \( \bigcup M = 0 \lor \forall \exists T \in M. T \subseteq F \lor (M \subseteq F) \) by auto
  moreover from this A have \( \bigcup M \subseteq F \) by auto
  ultimately have \( \bigcup M \subseteq (\bigcup F \cup \{0\}) \) using IsFilter_def assms by auto
\}
then have \( \forall M \subseteq Pow(\bigcup F \cup \{0\}). \bigcup M \subseteq (\bigcup F \cup \{0\}) \) by auto
ultimately show thesis using IsATopology_def by auto
qed
```

We can use `topology0` locale with filters.

**Lemma** `topology0_filter`:

```
assumes $F$ to be a filter on $\bigcup F$
shows `topology0 $(\bigcup F \cup \{0\})`
```

**Using** `top_of_filter` `topology0_def` assms by auto

The next abbreviation introduces notation where we want to specify the space where the filter convergence takes place.
The next abbreviation introduces notation where we want to specify the space where the net convergence takes place.

**abbreviation** NetConvTop(_ →ₕ _ {in} _)

where \( N \rightarrowₕ N x \{\text{in}\} T \equiv \text{topology0.NetConverges}(T,N,x) \)

Each point of a the union of a filter is a limit of that filter.

**lemma** lim_filter_top_of_filter:

assumes \( \mathcal{F} \{\text{is a filter on}\} \bigcup \mathcal{F} \) and \( x \in \bigcup \mathcal{F} \)

shows \( \mathcal{F} \rightarrow F x \{\text{in}\} (\mathcal{F} \cup \{0\}) \)

**proof**

\[
\text{have } \bigcup \mathcal{F}=\bigcup (\mathcal{F} \cup \{0\}) \text{ by auto}
\]

with \( \text{assms}(1) \) have \( \text{assms1: } \mathcal{F} \{\text{is a filter on}\} \bigcup (\mathcal{F} \cup \{0\}) \text{ by auto} \)

\{
  \text{fix } U
  \text{assume } U\in\text{Pow}(\bigcup (\mathcal{F} \cup \{0\})) \text{ x}\in\text{Interior}(U,(\mathcal{F} \cup \{0\}))
  \text{with } \text{assms}(1) \text{ have } \text{Interior}(U,(\mathcal{F} \cup \{0\}))\subseteq\mathcal{F} \text{ using topology0_def top_of_filter}
  \text{topology0.Top_2_L2 by blast}
  \text{moreover}
  \text{from } \text{assms}(1) \text{ have } \text{Interior}(U,(\mathcal{F} \cup \{0\}))\subseteq U \text{ using topology0_def top_of_filter}
  \text{topology0.Top_2_L1 by auto}
  \text{moreover}
  \text{from } U\in\text{Pow}(\bigcup (\mathcal{F} \cup \{0\})) \text{ have } U\in\mathcal{F} \text{ using } \text{assms}(1) \text{ IsFilter_def by auto}
\}

with \( \text{assms } \text{assms1 } \text{show } \text{thesis using} \text{ topology0.FilterConverges_def top_of_filter}
\text{topology0_def by auto} \)

qed

end

**72 Topology and neighborhoods**

theory Topology_ZF_4a imports Topology_ZF_4 begin

This theory considers the relations between topology and systems of neighborhood filters.

**72.1 Neighborhood systems**

The standard way of defining a topological space is by specifying a collection of sets that we consider "open" (see the Topology_ZF theory). An alternative of this approach is to define a collection of neighborhoods for each point of the space.
We define a neighborhood system as a function that takes each point \( x \in X \) and assigns it a collection of subsets of \( X \) which is called the neighborhoods of \( x \). The neighborhoods of a point \( x \) form a filter that satisfies an additional axiom that for every neighborhood \( N \) of \( x \) we can find another one \( U \) such that \( N \) is a neighborhood of every point of \( U \).

**definition**

\[
\text{IsNeighSystem} \quad \text{(is a neighborhood system on)} \quad 90)
\]

where \( \mathcal{M} \) \{is a neighborhood system on\} \( X \equiv \mathcal{M} : X \to \text{Pow(Pow}(X)) \) \& \( (\forall x \in X. (\mathcal{M}(x) \text{ is a filter on}) X) \) \& \( (\forall N \in \mathcal{M}(x). x \in N \) \& \( (\exists U \in \mathcal{M}(x). \forall y \in U. (N \in \mathcal{M}(y)) \))

A neighborhood system on \( X \) consists of collections of subsets of \( X \).

**lemma neighborhood_subset:**

assumes \( \mathcal{M} \) \{is a neighborhood system on\} \( X \) and \( x \in X \) and \( N \in \mathcal{M}(x) \)

shows \( N \subseteq X \) and \( x \in N \)

**proof** -

from \( \mathcal{M} \) \{is a neighborhood system on\} \( X \) have \( \mathcal{M} : X \to \text{Pow(Pow}(X)) \) unfolding \text{IsNeighSystem_def} by simp

with \( x \in X \) have \( \mathcal{M}(x) \in \text{Pow(Pow}(X)) \) using apply_funtype by blast

with \( N \in \mathcal{M}(x) \) show \( N \subseteq X \) by blast

from assms show \( x \in N \) using \text{IsNeighSystem_def} by simp

qed

Some sources (like Wikipedia) use a bit different definition of neighborhood systems where the \( U \) is required to be contained in \( N \). The next lemma shows that this stronger version can be recovered from our definition.

**lemma neigh_def_stronger:**

assumes \( \mathcal{M} \) \{is a neighborhood system on\} \( X \) and \( x \in X \) and \( N \in \mathcal{M}(x) \)

shows \( \exists U \in \mathcal{M}(x). U \subseteq N \) \& \( (\forall y \in U. (N \in \mathcal{M}(y)) \))

**proof** -

from assms obtain \( W \) where \( W \in \mathcal{M}(x) \) and \( \text{areNeigh} \vdash (\forall y \in W. (N \in \mathcal{M}(y)) \)

using \text{IsNeighSystem_def} by blast

let \( U = N \cap W \)

from assms \( \mathcal{M}(x) \) have \( U \in \mathcal{M}(x) \)

unfolding \text{IsNeighSystem_def} \text{IsFilter_def} by blast

moreover have \( U \subseteq N \) by blast

moreover from \( \text{areNeigh} \) have \( \forall y \in U. (N \in \mathcal{M}(y)) \) by auto

ultimately show thesis by auto

qed

### 72.2 From a neighborhood system to topology

Given a neighborhood system \( \{\mathcal{M}_x\}_{x \in X} \) we can define a topology on \( X \). Namely, we consider a subset of \( X \) open if \( U \in \mathcal{M}_x \) for every element \( x \) of \( U \).

The collection of sets defined as above is indeed a topology.
theorem topology_from_neighs:
assumes \( M \) \{is a neighborhood system on\} \( X \)
defines \( T \equiv \{U \in \text{Pow}(X). \ \forall x \in U. \ U \in M(x)\} \)
shows \( T \) \{is a topology\} and \( \bigcup T = X \)
proof -
{ fix \( U \) assume \( U \in \text{Pow}(T) \)
  have \( \bigcup U \in T \)
  proof -
  from \( \langle U \in \text{Pow}(T) \rangle \) Tdef have \( \bigcup U \in \text{Pow}(X) \) by blast
  moreover
  { fix \( x \) assume \( x \in \bigcup U \)
    then obtain \( U \) where \( U \subseteq \bigcup U \) by blast
    with assms \( \langle U \in \text{Pow}(T) \rangle \)
    have \( U \in M(x) \) and \( U \subseteq \bigcup U \) and \( M(x) \) \{is a filter on\} \( X \)
    unfolding IsNeighSystem_def by auto
    with \( \langle \bigcup U \in \text{Pow}(X) \rangle \) have \( \bigcup U \in M(x) \) unfolding IsFilter_def
    by simp
  }
  ultimately show \( \bigcup U \in T \) using Tdef by blast
qed
}
moreover
{ fix \( U \) \( V \) assume \( U \in T \) and \( V \in T \)
  have \( U \cap V \in T \)
  proof -
  from Tdef \( \langle U \in T \rangle \) \( \langle V \in T \rangle \) have \( U \cap V \in \text{Pow}(X) \) by auto
  moreover
  { fix \( x \) assume \( x \in U \cap V \)
    with assms \( \langle U \in T \rangle \) \( \langle V \in T \rangle \) Tdef have \( U \in M(x) \) \( V \in M(x) \) and \( M(x) \) \{is a filter on\} \( X \)
    unfolding IsNeighSystem_def by auto
    then have \( U \cap V \in M(x) \) unfolding IsFilter_def by simp
  }
  ultimately show \( U \cap V \in T \) using Tdef by simp
qed
}
ultimately show \( T \) \{is a topology\} unfolding IsATopology_def by blast

from assms show \( \bigcup T = X \) unfolding IsNeighSystem_def IsFilter_def by blast
qed

Some sources (like Wikipedia) define the open sets generated by a neighborhood system "as those sets containing a neighborhood of each of their points". The next lemma shows that this definition is equivalent to the one we are using.

lemma topology_from_neighs1:
assumes \( M \) \{is a neighborhood system on\} \( X \)
shows \( \{\forall x \in U. \ U \in M(x)\} = \{\forall x \in U. \ \exists V \in M(x)\} \)

909
\[ V \subseteq U \] 
proof
let \( T = \{ U \in \text{Pow}(X). \forall x \in U. U \in M(x) \} \)
let \( S = \{ U \in \text{Pow}(X). \forall x \in U. \exists V \in M(x). V \subseteq U \} \)
show \( S \subseteq T \)
proof
- fix \( U \) assume \( U \in S \) then have \( U \in \text{Pow}(X) \) by simp
moreover from \( U \in S \) \( U \in \text{Pow}(X) \) have \( \forall x \in U. U \in M(x) \)
unfolding \text{IsNeighSystem_def IsFilter_def} by blast
ultimately have \( U \in T \) by auto
thus thesis by auto
qed
show \( T \subseteq S \) by auto
qed

72.3 From a topology to a neighborhood system

Once we have a topology \( T \) we can define a natural neighborhood system on \( X = \bigcup T \). In this section we define such neighborhood system and prove its basic properties.

For a topology \( T \) we define a neighborhood system of \( T \) as a function that takes an \( x \in X = \bigcup T \) and assigns it the collection of supersets of open sets containing \( x \). We call that the "neighborhood system of \( T \)"

definition
\text{NeighSystem} ((\text{neighborhood system of}) _ 91)
where \( \{ \text{neighborhood system of} \} \ T \equiv \{ \langle x, \{ N \in \text{Pow}(\bigcup T). \exists U \in T. (x \in U \land U \subseteq N) \} \rangle. x \in \bigcup T \} \)

The way we defined the neighborhood system of \( T \) means that it is a function on \( \bigcup T \).

lemma \text{neigh_fun}: shows ((\text{neighborhood system of}) \( T \)) \( : \bigcup T \to \text{Pow}(\text{Pow}(\bigcup T)) \)
proof
- let \( X = \bigcup T \)
have \( \forall x \in X. \{ N \in \text{Pow}(X). \exists U \in T. (x \in U \land U \subseteq N) \} \in \text{Pow}(\text{Pow}(X)) \)
by blast
then show thesis unfolding \text{NeighSystem_def} using \text{ZF_fun_from_tot_val1} unfolding \text{NeighSystem_def} by simp
qed

The value of the neighborhood system of \( T \) at \( x \in \bigcup T \) is the collection of supersets of open sets containing \( x \).

lemma \text{neigh_val}: assumes \( x \in \bigcup T \)
shows ((\text{neighborhood system of}) \( T \))(x) = \( \{ N \in \text{Pow}(\bigcup T). \exists U \in T. (x \in U \land U \subseteq N) \} \)
using \( \text{assms ZF_fun_from_tot_val1} \) unfolding \text{NeighSystem_def} by simp

910
The next lemma shows that open sets are members of (what we will prove later to be) the natural neighborhood system on $\bigcup T$.

**lemma open_are_neighs:**

assumes $U \in T$ and $x \in U$

shows $x \in \bigcup T$ and $U \in \{V \in \text{Pow}(\bigcup T) \mid \exists U \in T. (x \in U \land U \subseteq V)\}$

using assms by auto

Another fact we will need is that for every $x \in X = \bigcup T$ the neighborhoods of $x$ form a filter

**lemma neighs_is_filter:**

assumes $T$ {is a topology} and $x \in \bigcup T$

defines $M \equiv \{\text{neighborhood system of } T\}$

shows $M(x)$ {is a filter on} $(\bigcup T)$

proof -

let $X = \bigcup T$

let $F = \{V \in \text{Pow}(X) \mid \exists U \in T. (x \in U \land U \subseteq V)\}$

have $0 \notin F$ by blast

moreover have $X \in F$

proof -

from assms $\langle x \in X \rangle$ have $X \in \text{Pow}(X)$ $X \in T$ and $x \in X \land X \subseteq X$

using carr_open by auto

hence $\exists U \in T. (x \in U \land U \subseteq X)$ by auto

thus thesis by auto

qed

moreover have $\forall A \in F$. $\forall B \in F$. $A \cap B \in F$

proof -

{ fix $A$ $B$

  assume $A \in F$. $B \in F$

  then obtain $U_A$ $U_B$ where $U_A \in T$ and $x \in U_A \land U_A \subseteq A$ $U_B \in T$ and $x \in U_B \land U_B \subseteq B$

  by auto

  with $T$ {is a topology} $A \cap B \in \text{Pow}(X)$ and $U_A \cup U_B \in T$ and $x \in U_A \cup U_B$ $U_A \cup U_B \subseteq A \cap B$ using IsATopology_def

  by auto

  hence $A \cap B \in F$ by blast

  } thus thesis by blast

qed

moreover have $\forall B \in F$. $\forall C \subseteq \text{Pow}(X)$. $B \subseteq C \rightarrow C \in F$

proof -

{ fix $B$ $C$

  assume $B \in F$. $C \subseteq \text{Pow}(X)$ $B \subseteq C$

  then obtain $U$ where $U \in T$ and $x \in U \subseteq B$ by blast

  with $C \subseteq \text{Pow}(X)$ $B \subseteq C$ have $C \in F$ by blast

  } thus thesis by auto

qed

ultimately have $F$ {is a filter on} $X$ unfolding IsFilter_def by blast

with $M \equiv \langle x \in X \rangle$ show $M(x)$ {is a filter on} $X$ using ZF_fun_from_tot_val1

NeighSystem_def

by simp

qed
The next theorem states that the the natural neighborhood system on $X = \bigcup T$ indeed is a neighborhood system.

**Theorem neigh_from_topology:**

assumes $T$ {is a topology} 
shows $\{\text{neighborhood system of } T\}$ {is a neighborhood system on} $(\bigcup T)$

**Proof** -

let $X = \bigcup T$
let $M = \{\text{neighborhood system of } T\}$

have $M : X \to \Pow(\Pow(X))$

proof -

{ fix $x$ assume $x \in X$
  hence $\{V \in \Pow(\bigcup T). \exists U \in T. (x \in U \land U \subseteq V)\} \subseteq \Pow(\Pow(X))$ by auto
  } hence $\forall x \in X. \{V \in \Pow(\bigcup T). \exists U \in T. (x \in U \land U \subseteq V)\} \subseteq \Pow(\Pow(X))$ by auto

then show thesis using ZF_fun_from_total NeighSystem_def by simp

qed

moreover from assms have $\forall x \in X. (M(x) \text{ is a filter on } X)$

using neighs_is_filter NeighSystem_def by auto

moreover have $\forall x \in X. \forall N \in M(x). x \in N \land (\exists U \in M(x). \forall y \in U. (N \in M(y)))$

proof -

{ fix $x$ $N$ assume $x \in X$ $N \in M(x)$
  let $\mathfrak{F} = \{V \in \Pow(X). \exists U \in T. (x \in U \land U \subseteq V)\}$
  from $\langle x \in X \rangle$ have $M(x) = \mathfrak{F}$ using ZF_fun_from_tot_val1 NeighSystem_def
  by simp
  with $\langle N \in M(x) \rangle$ have $N \in \mathfrak{F}$ by simp
  hence $x \in N$ by blast
  from $\langle N \in \mathfrak{F} \rangle$ obtain $U$ where $U \in T$ $x \in U$ and $U \subseteq N$ by blast
  with $\langle N \in \mathfrak{F} \rangle$ $\langle M(x) = \mathfrak{F} \rangle$ have $U \in M(x)$ by auto
  moreover from assms $\langle U \in T \rangle$ $\langle U \subseteq N \rangle$ $\langle N \in \mathfrak{F} \rangle$ have $\forall y \in U. (N \in M(y))$
  using ZF_fun_from_tot_val1 open_are_neighs neighs_is_filter NeighSystem_def IsFilter_def by auto
  ultimately have $\exists U \in M(x). \forall y \in U. (N \in M(y))$ by blast
  with $\langle x \in N \rangle$ have $x \in N \land (\exists U \in M(x). \forall y \in U. (N \in M(y)))$ by simp
  } thus thesis by auto

qed

ultimately show thesis unfolding IsNeighSystem_def by blast

qed

Any neighborhood of an element of the closure of a subset intersects the subset.

**Lemma neigh_inter_nempty:**

assumes $T$ {is a topology} $A \subseteq \bigcup T$ $x \in \text{Closure}(A,T)$ and $N \in \{\text{neighborhood system of } T\}(x)$

shows $N \cap A \neq 0$

**Proof** -

let $X = \bigcup T$

from assms(1) have cntx: topology0(T)
  unfolding topology0_def by simp
  with assms(2,3) have $x \in X$

912
using topology0.Top_3_L11(1) by blast
with assms(4) obtain U where U∈T x∈U and U⊆N
using neigh_val by auto
from assms(2,3) cntx {U∈T} {x∈U} have A∩U ≠ 0
using topology0.cl_inter_neigh by simp
with {U⊆N} show N∩A ≠ 0 by blast
qed

72.4 Neighborhood systems are 1:1 with topologies

We can create a topology from a neighborhood system and neighborhood system from topology. The question is then if we start from a neighborhood system, create a topology from it then create the topology’s natural neighborhood system, do we get back the neighborhood system we started from? Similarly, if we start from a topology, create its neighborhood system and then create a topology from that, do we get the original topology? This section provides the affirmative answer (for now only for the first question).

This means that there is a one-to-one correspondence between the set of topologies on a set and the set of abstract neighborhood systems on the set.

Each abstract neighborhood of x contains an open neighborhood of x.

lemma open_nei_in_nei:
  assumes M {is a neighborhood system on} X x∈X N∈M(x)
defines Tdef: T ≡ {U∈Pow(X). ∀x∈U. U ∈ M(x)}
sows N∈Pow(X) and ∃U∈T. (x∈U ∧ U⊆N)
proof -
  from assms(1) have M:X→Pow(Pow(X)) unfolding IsNeighSystem_def
  by simp
  with assms(2,3) show N∈Pow(X) using apply_funtype by blast
  let U = {y∈X. N∈M(y)}
  have U∈T proof -
  have U ∈ Pow(X) by auto
  moreover have ∀y∈U. U∈M(y)
  proof -
  { fix y assume y∈U
    then have y∈X and N∈M(y) by auto
    with assms(1) obtain V where V∈M(y) and ∀z∈V. N∈M(z)
    unfolding IsNeighSystem_def by blast
    with assms(1) ⟨y∈X⟩ ⟨V∈M(y)⟩ have V⊆U
    using neighborhood_subset(1) by blast
    with assms(1) ⟨y∈X⟩ ⟨V∈M(y)⟩ ⟨U ∈ Pow(X)⟩ have U∈M(y)
    unfolding IsNeighSystem_def using in_filter_def_split(5) by blast
  } thus thesis by simp
  qed
  ultimately have U ∈ {U∈Pow(X). ∀x∈U. U ∈ M(x)} by simp
  with assms(4) show U⊆T by simp

913
moreover from assms(1,2) \( \forall \in M(x) \) have \( x \in U \land U \subseteq N \)
using neighborhood_subset(2) by auto
ultimately show \( \exists U \in T. (x \in U \land U \subseteq N) \) by (rule witness_exists)
qed

In the next theorem we show that if we start from a neighborhood system, create a topology from it, then create its natural neighborhood system, we get back the original neighborhood system.

**Theorem nei_top_nei_round_trip:**

assumes \( M \) {is a neighborhood system on} \( X \)
defines \( T \equiv \{ U \in \text{Pow}(X). \forall x \in U. U \in M(x) \} \)
shows \( \{ \text{neighborhood system of} \ T \} = M \)

proof -
  let \( M = \{ \text{neighborhood system of} \ T \} \)
  from assms have \( T \) {is a topology} and \( \bigcup T = X \) using topology_from_neighs
  by auto
then have \( M \) {is a neighborhood system on} \( X \) using neigh_from_topology
  by blast
with assms(1) have \( M : X \to \text{Pow}(\text{Pow}(X)) \) and \( M : X \to \text{Pow}(\text{Pow}(X)) \)
unfolding IsNeighSystem_def by auto
moreover
\{ fix \( x \) assume \( x \in X \)
from \( \bigcup T = X \) \( \forall x \in X \) have I: \( M(x) = \{ V \in \text{Pow}(X). \exists U \in T. (x \in U \land U \subseteq V) \} \)
unfolding NeighSystem_def using ZF_fun_from_tot_val1 by simp
have \( M(x) = M(x) \)
proof
  \{ fix \( V \) assume \( V \in M(x) \)
  with I obtain \( U \) where \( U \in T \) \( x \in U \subseteq V \) by auto
  from assms(2) \( \forall U \in T \) \( \forall x \in U \) have \( U \in M(x) \) by simp
  from assms(1) \( \forall x \in X \) have \( M(x) \) {is a filter on} \( X \)
  unfolding IsNeighSystem_def by simp
  with \( \forall \in M(x) \) \( \forall V \in M(x) \) I \( \forall U \subseteq V \) have \( V \in M(x) \)
  unfolding IsFilter_def by simp
  \} thus \( M(x) \subseteq M(x) \) by auto
\{ fix \( N \) assume \( N \in M(X) \)
  with assms \( \forall x \in X \) \( \forall U \in T \) \( x \in U \)
  have \( N \in M(x) \) using open_nei_in_nei
  by auto
  \} thus \( M(x) \subseteq M(x) \) by auto
qed
hence \( \forall x \in X. M(x) = M(x) \) by simp
ultimately show thesis by (rule func_eq)
qed

914
72.5 Set neighborhoods

Some sources (like Metamath) take a somewhat different approach where instead of defining the collection of neighborhoods of a point \( x \in X \) they define a collection of neighborhoods of a subset of \( X \) (where \( X \) is the carrier of a topology \( T \) (i.e. \( X = \bigcup T \)). In this approach a neighborhood system is a function whose domain is the powerset of \( X \), i.e. the set of subsets of \( X \). The two approaches are equivalent in a sense as having a neighborhood system we can create a set neighborhood system and vice versa.

We define a set neighborhood system as a function that takes a subset \( A \) of the carrier of the topology and assigns it the collection of supersets of all open sets that contain \( A \).

**definition**

\[
\text{SetNeighSystem} \text{ ( \{set neighborhood system of\} } T) = \text{\{set neighborhood system of\} } T = \{\langle A, \{N \in \text{Pow}(\bigcup T). \exists U \in T. (A \subseteq U \land U \subseteq N)\} \rangle. A \in \text{Pow}(\bigcup T)\}
\]

Given a set neighborhood system we can recover the (standard) neighborhood system by taking the values of the set neighborhood system at singletons \( x \) where \( x \in X = \bigcup T \).

**lemma** neigh_from_nei:

\[\text{assumes } x \in \bigcup T\]

\[\text{shows } (\{\text{neighborhood system of\} } T)(x) = (\{\text{set neighborhood system of\} } T)x\]

\[\text{using assms ZF_fun_from_tot_val1}\]

\[\text{unfolding NeighSystem_def SetNeighSystem_def}\]

\[\text{by simp}\]

The set neighborhood system of \( T \) is a function mapping subsets of \( \bigcup T \) to collections of subsets of \( \bigcup T \).

**lemma** nei_fun:

\[\text{shows } (\{\text{set neighborhood system of\} } T):\text{Pow}(\bigcup T) \rightarrow \text{Pow}(\text{Pow}(\bigcup T))\]

**proof**

\[\text{let } X = \bigcup T\]

\[\text{have } \forall A \in \text{Pow}(X). \{N \in \text{Pow}(X). \exists U \in T. (A \subseteq U \land U \subseteq N)\} \in \text{Pow}(\text{Pow}(X))\]

\[\text{by blast}\]

\[\text{then have}\]

\[\{\langle A, \{N \in \text{Pow}(X). \exists U \in T. (A \subseteq U \land U \subseteq N)\} \rangle. A \in \text{Pow}(X)\}:\text{Pow}(X) \rightarrow \text{Pow}(\text{Pow}(X))\]

\[\text{by (rule ZF_fun_from_total)}\]

\[\text{then show thesis unfolding SetNeighSystem_def by simp}\]

**qed**

The value of the set neighborhood system of \( T \) at subset \( A \) of \( \bigcup T \) is the collection of subsets \( N \) of \( \bigcup T \) for which exists an open subset \( U \subseteq N \) that contains \( A \).

**lemma** nei_val: \[\text{assumes } A \subseteq \bigcup T\]

\[\text{shows}\]
A member of the value of the set neighborhood system of \( T \) at \( A \) is a subset of \( \bigcup T \). The interesting part is that we can show it without any assumption on \( A \).

**Lemma nei_val_subset:**

- Assumes \( N \in (\{\text{set neighborhood system of}\ T)(A) \)
- Shows \( A \subseteq \bigcup T \) and \( N \subseteq \bigcup T \)

**Proof:**

- Let \( f = (\{\text{set neighborhood system of}\ T) \)
- Have \( f: \text{Pow}(\bigcup T) \rightarrow \text{Pow}(\text{Pow}(\bigcup T)) \)
  - Using \( \text{nei}_\text{fun} \) by simp
- With assms show \( A \subseteq \bigcup T \) using \( \text{arg}_\text{in_domain} \) by blast
- With assms show \( N \subseteq \bigcup T \) using nei_val by simp

qed

If \( T \) is a topology, then every subset of its carrier (i.e. \( \bigcup T \)) is a (set) neighborhood of the empty set.

**Lemma nei_empty:**

- Assumes \( T \{\text{is a topology}\} N \subseteq \bigcup T \)
- Shows \( N \in (\{\text{set neighborhood system of}\ T)(0) \)
  - Using assms empty_open nei_val by auto

If \( T \) is a topology, then the (set) neighborhoods of a nonempty subset of \( \bigcup T \) form a filter on \( X = \bigcup T \).

**Theorem nei_filter:**

- Assumes \( T \{\text{is a topology}\} D \subseteq (\bigcup T) D \neq 0 \)
  - Shows \( (\{\text{set neighborhood system of}\ T)(D) \{\text{is a filter on}\} (\bigcup T) \)

**Proof:**

- Let \( X = \bigcup T \)
- Let \( F = (\{\text{set neighborhood system of}\ T)(D) \)
- From assms(2) have \( I: F = \{N \in \text{Pow}(X). \exists U \in T. (D \subseteq U \land U \subseteq N)\} \)
  - Using nei_val by simp
- With assms(3) have \( 0 \notin F \) by auto
- Moreover from assms(1,2) I have \( X \in F \)
  - Using carr_open by auto
- Moreover from I have \( F \subseteq \text{Pow}(X) \) by auto
- Moreover have \( \forall A \in F. \forall B \subseteq F. A \cap B \in F \)

**Proof:**

- \{ fix \( A, B \) assume \( A \in F \) \( B \in F \)
  - With I obtain \( U_A \subseteq A \) where
    - \( U_A \subseteq U \subseteq U_A \subseteq A \) and \( U_B \subseteq U \subseteq U_B \subseteq B \)
    - By auto
  - Let \( U = U_A \cap U_B \)
    - From assms(1) \( U_A \subseteq U \subseteq U_B \subseteq \text{Pow}(X) \)
  - Have \( U \subseteq \bigcup T \subseteq X \subseteq A \cap B \)
    - Unfolding IsATopology_def by auto
  - With I \( A \in F \) \( B \in F \) have \( A \cap B \in F \) by auto
  - Thus thesis by simp

qed

916
moreover have $\forall B \in F. \forall C \in \text{Pow}(X). B \subseteq C \rightarrow C \in F$

proof -
\[
\{ \text{fix } B, C \text{ assume } B \in F, C \in \text{Pow}(X). B \subseteq C \}
\]
\[
\text{from } I \langle B \in F \rangle \text{ obtain } U \text{ where } U \subseteq T \subseteq U \text{ and } U \subseteq B
\]
\[
\text{by auto}
\]
\[
\text{with } I \langle B \subseteq C \rangle \text{ have } \exists U \subseteq T. (D \subseteq U \land U \subseteq C)
\]
\[
\text{by blast}
\]
\[
\text{with } I \langle C \in \text{Pow}(X) \rangle \text{ have } C \in F \text{ by simp}
\]
\[
\text{thus thesis by blast}
\]
qed
ultimately show $F \{\text{is a filter on} \} X$
unfolding IsFilter_def by simp
qed

If $N$ is a (set) neighborhood of $A$ in $T$, then exist an open set $U$ such that $N$ contains $U$ which contains $A$. This is similar to the Metamath’s theorem with the same name, except that here we do not need assume that $T$ is a topology (which is a bit worrying).

**Lemma nei2**: assumes $N \in (\{\text{set neighborhood system of} \} T)(A)$ shows $\exists U \subseteq T. (A \subseteq U \land U \subseteq N)$

proof -
\[
\text{from asms have } A \in \text{Pow}(\bigcup T) \text{ using nei_fun arg_in_domain}
\]
\[
\text{by blast}
\]
\[
\text{with asms show thesis}
\]
unfolding SetNeighSystem_def using ZF_fun_from_tot_val1
by simp
qed

An open set $U$ covering a set $A$ is a set neighborhood of $A$.

**Lemma open_superset_nei**: assumes $V \in T. A \subseteq V$
shows $V \in (\{\text{set neighborhood system of} \} T)(A)$

proof -
\[
\text{from asms have}
\]
\[
(\{\text{set neighborhood system of} \} T)(A) = \{N \in \text{Pow}(\bigcup T). \exists U \subseteq T. (A \subseteq U \land U \subseteq N)\}
\]
\[
\text{using nei_val by blast}
\]
\[
\text{with asms show thesis by auto}
\]
qed

An open set is a set neighborhood of itself.

**Corollary open_is_nei**: assumes $V \in T$
shows $V \in (\{\text{set neighborhood system of} \} T)(V)$

using asms open_superset_nei by simp

An open neighborhood of $x$ is a set neighborhood of $\{x\}$.

**Corollary open_nei_singl**: assumes $V \in T. x \in V$
shows $V \in (\{\text{set neighborhood system of} \} T)\{x\}$

using asms open_superset_nei by simp
The Cartesian product of two neighborhoods is a neighborhood in the product topology. Similar to the Metamath's theorem with the same name.

**lemma neitx:**

assumes T {is a topology} S {is a topology} and
A ∈ (set neighborhood system of} T(C) and
B ∈ (set neighborhood system of} S(D)
shows A×B ∈ (set neighborhood system of} (T×S)(C×D)

**proof** -

have A×B ⊆ ∪(T×S)

**proof** -

from assms(3,4) have A×B ⊆ (∪T)×(∪S) using nei_val_subset(2) by blast

with assms(1,2) show thesis using Top_1_4_T1 by simp

**qed**

let \( F = (\{\text{set neighborhood system of}\} (T×S))(C×D) \) \{ assume C=0 ∨ D=0

with assms(1,2) show \( A×B ⊆ ∪(T×S) \) have \( A×B ∈ F \)

using Top_1_4_T1(1) nei_empty by auto

**moreover**

\{ assume C≠0 D≠0

from assms(3) obtain \( U_A \) where

\( U_A ∈ T \subseteq U_A \subseteq A \) using nei2 by blast

from assms(4) obtain \( U_B \) where

\( U_B ∈ S \subseteq U_B \subseteq B \) using nei2 by blast

from assms(1,2) \( \langle U_A ∈ T \rangle \langle U_B ∈ S \rangle \langle C ⊆ U_A \rangle \langle D ⊆ U_B \rangle \)

have \( U_A×U_B ∈ T×S \) and \( C×D ⊆ U_A×U_B \)

using prod_open_open_prod by auto

then have \( U_A×U_B ∈ F \) using open_superset_nei by simp

from \( \langle U_A ∈ A \rangle \langle U_B ∈ B \rangle \) have \( U_A×U_B ⊆ A×B \) by auto

have \( F \{\text{is a filter on}\} (∪(T×S)) \)

**proof** -

from assms(1,2) have \( (T×S) \{\text{is a topology}\} \)

using Top_1_4_T1(1) by simp

moreover have \( C×D ⊆ ∪(T×S) \)

**proof** -

from assms(3,4) have \( C×D ⊆ (∪T)×(∪S) \)

using nei_val_subset(1) by blast

with assms(1,2) show thesis using Top_1_4_T1(3) by simp

**qed**

moreover from \( \langle C≠0 \rangle \langle D≠0 \rangle \) have \( C×D ≠ 0 \) by simp

ultimately show \( F \{\text{is a filter on}\} (∪(T×S)) \)

using nei_filter by simp

**qed**

with \( \langle U_A×U_B ∈ F \rangle \langle A×B ⊆ ∪(T×S) \rangle \langle U_A×U_B ⊆ A×B \rangle \)

have \( A×B ∈ F \) using is_filter_def_split(5) by simp

} ultimately show thesis by auto

918
Any neighborhood of an element of the closure of a subset intersects the subset. This is practically the same as `neigh_inter_nempty`, just formulated in terms of set neighborhoods of singletons. Compare with Metamath’s theorem with the same name.

**Lemma neindisj:** assumes \( T \) \{is a topology\} \( A \subseteq \bigcup T \ x \in \text{Closure}(A,T) \) and \( N \in \{\text{set neighborhood system of} \ T\}(x) \)

shows \( N \cap A \neq 0 \)

**Proof** -

let \( X = \bigcup T \)

from \( \text{assms(1)} \) have \( \text{cntx: topology0}(T) \) unfolding `topology0_def` by simp

with \( \text{assms(2,3)} \) have \( x \in X \) using `topology0.Top_3_L11(1)` by blast

with \( \text{assms} \) show thesis using `neigh_from_nei` `neigh_inter_nempty` by simp

qed

73 Topology - examples

theory Topology_ZF_examples imports Topology_ZF Cardinal_ZF

begin

This theory deals with some concrete examples of topologies.

73.1 CoCardinal Topology

In this section we define and prove the basic properties of the co-cardinal topology on a set \( X \).

The collection of subsets of a set whose complement is strictly bounded by a cardinal is a topology given some assumptions on the cardinal.

**Definition**

\[
\text{CoCardinal}(X,T) \equiv \{F \in \text{Pow}(X). \ X - F \prec T\} \cup \{0\}
\]

For any set and any infinite cardinal we prove that \( \text{CoCardinal}(X,\mathbb{Q}) \) forms a topology. The proof is done with an infinite cardinal, but it is obvious that the set \( \mathbb{Q} \) can be any set equipollent with an infinite cardinal. It is a topology also if the set where the topology is defined is too small or the cardinal too large; in this case, as it is later proved the topology is a discrete topology. And the last case corresponds with \( \mathbb{Q}=1 \) which translates in the indiscrete topology.
lemma CoCar_is_topology:
  assumes InfCard (Q)
  shows CoCardinal(X,Q) {is a topology}
proof -
  let T = CoCardinal(X,Q)
  
  fix M
  assume A:M∈Pow(T)
  hence M⊆T by auto
  then have M⊆Pow(X) using CoCardinal_def by auto
  moreover
  { assume B:M=0
    then have ∪M∈T using CoCardinal_def by auto
  }
  moreover
  { assume B:M={0}
    then have ∪M∈T using CoCardinal_def by auto
  }
  moreover
  { assume B:M ≠ 0 M≠{0}
    from B obtain T where C:T∈M and T≠0 by auto
    with A have D:X-T ≺ (Q) using CoCardinal_def by auto
    from C have X-∪M⊆X-T by blast
    with D have X-∪M≺ (Q) using subset_imp_lepoll lesspoll_trans1
    by blast
  }
  ultimately have ∪M∈T using CoCardinal_def by auto
  
  moreover
  { fix U and V
    assume U∈T and V∈T
    then have A:U=0 ∨ (U∈Pow(X) ∧ X-U≺ (Q)) and
    B:V=0 ∨ (V∈Pow(X) ∧ X-V≺ (Q)) using CoCardinal_def by auto
    hence D:U∈Pow(X)V∈Pow(X) by auto
    have C:X-(U ∩ V)=X-(U∪V) by fast
    with A B C have U∩V=0 ∨ (U∩V∈Pow(X) ∧ X-(U ∩ V)≺ (Q)) using less_less_imp_un_less
    by auto
    then have U∩V∈T using CoCardinal_def by auto
  }
  ultimately show thesis using IsATopology_def by auto
qed

We can use theorems proven in topology0 context for the co-cardinal topol-
ogy.

**Theorem** topology0_CoCardinal:
assumes InfCard(T)
shows topology0(CoCardinal(X,T))
using topology0_def CoCar_is_topology assms by auto

It can also be proven that if CoCardinal(X,T) is a topology, X≠0, Card(T) and T≠0; then T is an infinite cardinal, X≺T or T=1. It follows from the fact that the union of two closed sets is closed. Choosing the appropriate cardinals, the cofinite and the cocountable topologies are obtained.

The cofinite topology is a very special topology because it is closely related to the separation axiom \( T_1 \). It also appears naturally in algebraic geometry.

**Definition**
Cofinite (CoFinite _ 90) where
CoFinite X ≡ CoCardinal(X,nat)

Cocountable topology in fact consists of the empty set and all cocountable subsets of \( X \).

**Definition**
Cocountable (CoCountable _ 90) where
CoCountable X ≡ CoCardinal(X,csucc(nat))

### 73.2 Total set, Closed sets, Interior, Closure and Boundary

There are several assertions that can be done to the CoCardinal(X,T) topology. In each case, we will not assume sufficient conditions for CoCardinal(X,T) to be a topology, but they will be enough to do the calculations in every possible case.

The topology is defined in the set \( X \).

**Lemma** union_cocardinal:
assumes T≠0
shows \( \bigcup \text{CoCardinal}(X,T) = X \)
proof-
  have X:X-X=0 by auto
  have 0 ≲ 0 by auto
  with assms have 0≺11 ≲T using not_0_is_lepoll_1 lepoll_imp_lesspoll_succ
  by auto
  then have 0≺T using lesspoll_trans2 by auto
  with X have (X-X)≺T by auto
  then have X∈CoCardinal(X,T) using CoCardinal_def by auto
  hence X∈∪ CoCardinal(X,T) by blast
  then show ∪ CoCardinal(X,T)=X using CoCardinal_def by auto
qed

The closed sets are the small subsets of \( X \) and \( X \) itself.

**Lemma** closed_sets_cocardinal:
assumes \( T \neq 0 \)
shows \( D \) \{is closed in\} CoCardinal\((X,T)\) \iff (\( D \in \text{Pow}(X) \land D \prec T \)) \lor D = X
proof-
{ 
  assume A:D \subseteq X \land D \in \text{CoCardinal}\((X,T)\) \land D \neq X
  from A(1,3) have X-(X-D) = D \land D \neq 0 by auto
  with A(2) have D \prec T using CoCardinal_def by simp
}
with assms have D \{is closed in\} CoCardinal\((X,T)\) \implies (\( D \in \text{Pow}(X) \land D \prec T \)) \lor D = X using IsClosed_def
moreover
{ 
  assume A:D \prec TD \subseteq X
  from A(2) have X-(X-D) = D by blast
  with A(1) have X-(X-D) \prec T by auto
  then have X-D \in CoCardinal\((X,T)\) using CoCardinal_def by auto
}
with assms have (\( D \in \text{Pow}(X) \land D \prec T \)) \implies D \{is closed in\} CoCardinal\((X,T)\) using union_cocardinal
IsClosed_def by auto
moreover
have X-X = 0 by auto
then have X-X \in CoCardinal\((X,T)\) using CoCardinal_def by auto
with assms have X \{is closed in\} CoCardinal\((X,T)\) using union_cocardinal
IsClosed_def by auto
ultimately show thesis by auto
qed

The interior of a set is itself if it is open or 0 if it isn’t open.

lemma interior_set_cocardinal:
assumes noC: \( T \neq 0 \) and A\( \subseteq X \)
shows \( \text{Interior}(A,\text{CoCardinal}(X,T)) = (\text{if } ((X-A) \prec T) \text{ then } A \text{ else } 0) \)
proof-
from assms(2) have dif_dif:X-(X-A) = A by blast
{ 
  assume \( (X-A) \prec T \)
  then have \( (X-A) \in \text{Pow}(X) \land (X-A) \prec T \) by auto
  with noC have \( (X-A) \) \{is closed in\} CoCardinal\((X,T)\) using closed_sets_cocardinal
  by auto
  with noC have \( X-(X-A) \in \text{CoCardinal}(X,T) \) using IsClosed_def union_cocardinal
  by auto
  with dif_dif have A \in CoCardinal\((X,T)\) by auto
  hence \( A \subseteq \{U \subseteq \text{CoCardinal}(X,T) . U \subseteq A\} \) by auto
  hence a1: A \subseteq \bigcup \{U \subseteq \text{CoCardinal}(X,T) . U \subseteq A\} by auto
  have a2: \( \bigcup \{U \subseteq \text{CoCardinal}(X,T) . U \subseteq A\} \subseteq A \) by blast
  from a1 a2 have \( \text{Interior}(A,\text{CoCardinal}(X,T)) = A \) using Interior_def
  by auto
}
moreover
\{ 
  assume as:-((X-A) \prec T) 
  
  fix U 
  assume U \subseteq A 
  hence X-A \subseteq X-U by blast 
  then have Q:X-A \subseteq X-U using subset_imp_lepoll by auto 
  
  assume X-U \prec T 
  with Q have X-A \prec T using lesspoll_trans1 by auto 
  with as have False by auto 
  
  hence -((X-U) \prec T) by auto 
  then have U \notin CoCardinal(X,T) \setminus U=0 using CoCardinal_def by auto 
  
  hence \{U \in CoCardinal(X,T). U \subseteq A\} \subseteq \{0\} by blast 
  then have Interior(A,CoCardinal(X,T))=0 using Interior_def by auto 
  
  ultimately show thesis by auto 
\}

qed

X is a closed set that contains A. This lemma is necessary because we cannot use the lemmas proven in the topology0 context since \(T \neq \emptyset\) is too weak for \(CoCardinal(X,T)\) to be a topology.

\begin{enumerate}
\item \textbf{lemma} \texttt{X\_closedcov\_cocardinal:}
\item \texttt{assumes} \texttt{T\neq0 A \subseteq X}
\item \texttt{shows} \texttt{X \in ClosedCovers(A,CoCardinal(X,T)) using ClosedCovers_def}
\item \texttt{using} \texttt{union_cocardinal closed_sets_cocardinal assms} by \texttt{auto}
\end{enumerate}

The closure of a set is itself if it is closed or \(X\) if it isn’t closed.

\begin{enumerate}
\item \textbf{lemma} \texttt{closure\_set\_cocardinal:}
\item \texttt{assumes} \texttt{T\neq0 A \subseteq X}
\item \texttt{shows} \texttt{Closure(A,CoCardinal(X,T))=(if (A \prec T) then A else X)}
\item \texttt{proof-}
\item \texttt{assume A \prec T}
\item \texttt{with assms have A \{is closed in\} CoCardinal(X,T) using closed_sets_cocardinal}
\item \texttt{by auto}
\item \texttt{with assms(2) have A \in \{D \in Pow(X). D \{is closed in\} CoCardinal(X,T) \land A \subseteq D\}} by \texttt{auto}
\item \texttt{with assms(1) have S:A \in ClosedCovers(A,CoCardinal(X,T)) using ClosedCovers_def}
\item \texttt{using union_cocardinal by auto}
\item \texttt{hence 11:\bigcap ClosedCovers(A,CoCardinal(X,T)) \subseteq A} by \texttt{blast}
\item \texttt{from S have 12:A \subseteq \bigcap ClosedCovers(A,CoCardinal(X,T))}
\item \texttt{unfolding ClosedCovers_def by auto}
\item \texttt{from 11 12 have Closure(A,CoCardinal(X,T))=A using Closure_def}
\item \texttt{by auto}
\end{enumerate}

moreover

923
\{  
    assume as: ¬ A \prec T  
    \{  
      \text{fix } U  
      assume A \subseteq U  
      then have Q:A \subseteq U \text{ using } \text{subset_imp_lepoll} \text{ by } \text{auto}  
      \{  
        assume U \prec T  
        with Q have A \prec T \text{ using } \text{lesspoll_trans1} \text{ by } \text{auto}  
        with as have False \text{ by } \text{auto}  
      \}  
      hence ¬ U \prec T \text{ by } \text{auto}  
      with assms(1) have ¬(U \{\text{is closed in}\text{ CoCardinal}(X,T)) \lor U=X \text{ using } \text{closed_sets_cocardinal} \text{ by } \text{auto}  
    \}  
  \}  
  \text{ultimately show thesis } \text{ by } \text{auto}  
\}  
\text{qed}  

The boundary of a set is empty if \( A \) and \( X - A \) are closed, \( X \) if not \( A \) neither \( X - A \) are closed and; if only one is closed, then the closed one is its boundary.

\text{lemma boundary_cocardinal:}  
\text{assumes } T \neq 0  
\text{shows } \text{Boundary}(A,\text{CoCardinal}(X,T)) = (\text{if } A \prec T \text{ then (if } (X-A) \prec T \text{ then 0 else } A) \text{ else (if } (X-A) \prec T \text{ then } X-A \text{ else } X))  
\text{proof}  
\text{from assms(2) have } X-A \subseteq X \text{ by } \text{auto}  
\{  
  \text{assume AS: } A \prec T \text{ X-A } \prec T  
  \text{with assms } <X-A \subseteq X> \text{ have}  
  \text{Closure}(X-A,\text{CoCardinal}(X,T)) = X-A \text{ and } \text{Closure}(A,\text{CoCardinal}(X,T)) = A  
  \text{using } \text{closure_set_cocardinal} \text{ by } \text{auto}  
  \text{with assms(1) have } \text{Boundary}(A,\text{CoCardinal}(X,T)) = 0  
  \text{using } \text{Boundary_def} \text{ union_cocardinal} \text{ by } \text{auto}  
\}  
\text{moreover}  
\{  
  \text{assume AS: } ¬(A \prec T) \text{ X-A } \prec T  
  \text{with assms } <X-A \subseteq X> \text{ have}  
\}  

924
Closure(X-A,CoCardinal(X,T)) = X-A and Closure(A,CoCardinal(X,T)) = X

using closure_set_cocardinal by auto

with assm(1) have Boundary(A,CoCardinal(X,T)) = X-A using Boundary_def

union_cocardinal by auto

} moreover

{ 
  assume AS: ~ (A≺T) ~ (X-A≺T)
  with assm X≺X have
  Closure(X-A,CoCardinal(X,T)) = X and Closure(A,CoCardinal(X,T)) = X
  using closure_set_cocardinal by auto
  with assm(1) have Boundary(A,CoCardinal(X,T)) = X using Boundary_def
  union_cocardinal by auto

} moreover

{ 
  assume AS: A≺T ~ (X-A≺T)
  with assms X≺X have
  Closure(X-A,CoCardinal(X,T)) = X and Closure(A,CoCardinal(X,T)) = A
  using closure_set_cocardinal by auto
  with assm have Boundary(A,CoCardinal(X,T)) = A using Boundary_def
  union_cocardinal by auto

} ultimately show thesis by auto

qed

If the set is too small or the cardinal too large, then the topology is just the discrete topology.

lemma discrete_cocardinal:
  assumes X≺T
  shows CoCardinal(X,T) = Pow(X)

proof

{ 
  fix U
  assume U∈CoCardinal(X,T)
  then have U ∈ Pow(X) using CoCardinal_def by auto

} then show CoCardinal(X,T) ⊆ Pow(X) by auto

{ 
  fix U
  assume A: U ∈ Pow(X)
  then have X-U ⊆ X by auto
  then have X-U≺X using subset_imp_lepoll by auto
  then have X-U≺T using lesspoll_trans1 assm by auto
  with A have U∈CoCardinal(X,T) using CoCardinal_def

925
by auto

} then show Pow(X) ⊆ CoCardinal(X,T) by auto
qed

If the cardinal is taken as T=1 then the topology is indiscrete.

lemma indiscrete_cocardinal:
  shows CoCardinal(X,1) = {0,X}
proof
{
  fix Q
  assume Q ∈ CoCardinal(X,1)
  then have Q ∈ Pow(X) and Q=0 ∨ X-Q≺1 using CoCardinal_def by auto
  then have Q ∈ Pow(X) and Q=0 ∨ X-Q=0 using lesspoll_succ_iff lepoll_0_iff
by auto
  then have Q=0 ∨ Q=X by blast
}
then show CoCardinal(X,1) ⊆ {0, X} by auto
have 0 ∈ CoCardinal(X,1) using CoCardinal_def by auto
moreover have 0≺1 and X-X=0 using lesspoll_succ_iff by auto
then have x ∈ CoCardinal(X,1) using CoCardinal_def by auto
ultimately show {0, X} ⊆ CoCardinal(X,1) by auto
qed

The topological subspaces of the CoCardinal(X,T) topology are also CoCardinal topologies.

lemma subspace_cocardinal:
  shows CoCardinal(X,T) {restricted to} Y = CoCardinal(Y∩X,T)
proof
{
  fix M
  assume M ∈ (CoCardinal(X,T) {restricted to} Y)
  then obtain A where A1: A ∈ CoCardinal(X,T) M=Y ∩ A using RestrictedTo_def
by auto
  then have M ∈ Pow(X ∩ Y) using CoCardinal_def by auto
moreover have A1 have (Y ∩ X)-M = (Y ∩ X)-A using CoCardinal_def by auto
with ⊥(Y ∩ X)-M = (Y ∩ X)-A> have (Y ∩ X)-M ⊆ X-A by auto
then have (Y ∩ X)-M ⊆ X-A using subset_imp_lepoll by auto
with A1 have (Y ∩ X)-M ≺ T ∨ M=0 using lesspoll_trans1 CoCardinal_def
by auto
  ultimately have M ∈ CoCardinal(Y∩X, T) using CoCardinal_def
by auto
}
then show CoCardinal(X,T) {restricted to} Y ⊆ CoCardinal(Y∩X,T) by auto
{
  fix M

926
let A = M \cup (X-Y)
assume A: M \in \text{CoCardinal}(Y \cap X, T)
{
assume M=0
hence M=0 \cap Y by auto
then have M \in \text{CoCardinal}(X, T) \{\text{restricted to} Y\} using \text{RestrictedTo_def}
CoCardinal_def by auto
}
moreover
{
assume AS: M \neq 0
from A AS have A1: (M \in \text{Pow}(Y \cap X) \land (Y \cap X)-M \prec T) using \text{CoCardinal_def}
by auto
hence A \in \text{Pow}(X) by blast
moreover
have X-A=(Y \cap X)-M by blast
with A1 have X-A \prec T by auto
ultimately have A \in \text{CoCardinal}(X, T) using \text{CoCardinal_def} by auto
then have AT: Y \cap A \in \text{CoCardinal}(X, T) \{\text{restricted to} Y\} using \text{RestrictedTo_def}
by auto
have Y \cap A=Y \cap M by blast
also from A1 have \ldots=M by auto
finally have Y \cap A=M by simp
with AT have M \in \text{CoCardinal}(X, T) \{\text{restricted to} Y\} by auto
ultimately have M \in \text{CoCardinal}(X, T) \{\text{restricted to} Y\} by auto
}
then show CoCardinal(Y \cap X, T) \subseteq \text{CoCardinal}(X, T) \{\text{restricted to} Y\} by auto
qed

73.3 Excluded Set Topology

In this section, we consider all the subsets of a set which have empty intersection with a fixed set.

The excluded set topology consists of subsets of \( X \) that are disjoint with a fixed set \( U \).

definition ExcludedSet(X,U) \equiv \{F \in \text{Pow}(X). U \cap F=0\} \cup \{X\}

For any set; we prove that ExcludedSet(X,Q) forms a topology.

theorem excludedset_is_topology:
shows ExcludedSet(X,Q) \{is a topology\}
proof-
{
fix M
assume M \in \text{Pow}(\text{ExcludedSet}(X,Q))

927
then have \( A : M \subseteq \{ F \in \text{Pow}(X). Q \cap F = 0 \} \cup \{ X \} \) using ExcludedSet_def by auto

hence \( \bigcup M \in \text{Pow}(X) \) by auto

moreover

\{ 
  have \( B : Q \cap M = \bigcup \{ Q \cap T. T \in M \} \) by auto
  \{
    assume \( X \notin M \)
    with \( A \) have \( M \subseteq \{ F \in \text{Pow}(X). Q \cap F = 0 \} \) by auto
    with \( B \) have \( Q \cap \bigcup M = 0 \) by auto
  \}
  moreover
  \{ 
    assume \( X \in M \)
    with \( A \) have \( \bigcup M = X \) by auto
  \}
  ultimately have \( Q \cap \bigcup M = 0 \lor \bigcup M = X \) by auto
\}

ultimately have \( \bigcup M \in \text{ExcludedSet}(X,Q) \) using ExcludedSet_def by auto

moreover

\{ 
  fix \( U, V \)
  assume \( U \in \text{ExcludedSet}(X,Q) \) \( V \in \text{ExcludedSet}(X,Q) \)
  then have \( U \in \text{Pow}(X) \) \( V \in \text{Pow}(X) \) \( U = X \lor U \cap Q = 0 \lor V = X \lor V \cap Q = 0 \) using ExcludedSet_def by auto
  hence \( U \in \text{Pow}(X) \) \( V \in \text{Pow}(X) \) \( U \cap V = X \lor Q \cap (U \cap V) = 0 \) by auto
  then have \( (U \cap V) \in \text{ExcludedSet}(X,Q) \) using ExcludedSet_def by auto
\}

ultimately show thesis using IsATopology_def by auto

qed

We can use topology0 when discussing excluded set topology.

\textbf{theorem} topology0_excludedset:

\textbf{shows} topology0(\text{ExcludedSet}(X,T))

\textbf{using} topology0_def excludedset_is_topology by auto

Choosing a singleton set, it is considered a point in excluded topology.

\textbf{definition}

\( \text{ExcludedPoint}(X,p) \equiv \text{ExcludedSet}(X,\{p\}) \)

\textbf{73.4 Total set, closed sets, interior, closure and boundary}

Here we discuss what are closed sets, interior, closure and boundary in excluded set topology.

The topology is defined in the set \( X \)

\textbf{lemma} union_excludedset:
shows $\bigcup \text{ExcludedSet}(X, T) = X$

proof-
  have $X \in \text{ExcludedSet}(X, T)$ using ExcludedSet_def by auto
  then show thesis using ExcludedSet_def by auto
qed

The closed sets are those which contain the set $(X \cap T)$ and 0.

**lemma closed_sets_excludedset:**
  shows $D \in \text{closed in} \text{ExcludedSet}(X, T) \iff (D \in \text{Pow}(X) \land (X \cap T) \subseteq D) \lor D = 0$

proof-

\[
\begin{align*}
\{ & \text{fix } x \\
& \text{assume } A: D \subseteq X \land X - D \in \text{ExcludedSet}(X, T) \land D \neq 0 \land x \in T \land x \in X \\
& \text{from } A(1) \text{ have } B: X - (X - D) = D \text{ by auto} \\
& \text{from } A(2) \text{ have } T \cap (X - D) = 0 \lor X - D = X \text{ using ExcludedSet_def by auto} \\
& \text{hence } T \cap (X - D) = 0 \lor X - (X - D) = X - X \text{ by auto} \\
& \text{with } B \text{ have } T \cap (X - D) = 0 \lor D = X - X \text{ by auto} \\
& \text{hence } T \cap (X - D) = 0 \lor D = 0 \text{ by auto} \\
& \text{with } A(3) \text{ have } T \cap (X - D) = 0 \text{ by auto} \\
& \text{with } A(4) \text{ have } x \not\in X - D \text{ by auto} \\
& \text{with } A(5) \text{ have } x \in D \text{ by auto} \\
\}
\]
moreover

\[
\begin{align*}
& \text{assume } A: X \cap T \subseteq D \subseteq X \\
& \text{from } A(1) \text{ have } X - D \subseteq X - (X \cap T) \text{ by auto} \\
& \text{also have } ... = X - T \text{ by auto} \\
& \text{finally have } T \cap (X - D) = 0 \text{ by auto} \\
& \text{moreover} \\
& \text{have } X - D \in \text{Pow}(X) \text{ by auto} \\
& \text{ultimately have } X - D \in \text{ExcludedSet}(X, T) \text{ using ExcludedSet_def by auto} \\
\}
\]
ultimately show thesis using IsClosed_def union_excludedset ExcludedSet_def by auto

qed

The interior of a set is itself if it is $X$ or the difference with the set $T$

**lemma interior_set_excludedset:**
  assumes $A \subseteq X$
  shows $\text{Interior}(A, \text{ExcludedSet}(X, T)) = (\text{if } A = X \text{ then } X \text{ else } A - T)$

proof-

\[
\begin{align*}
& \text{assume } A: A \neq X \\
& \text{from asms have } A - T \in \text{ExcludedSet}(X, T) \text{ using ExcludedSet_def by auto} \\
& \text{then have } A - T \subseteq \text{Interior}(A, \text{ExcludedSet}(X, T)) \text{ using Interior_def by auto} \\
& \text{moreover}
\]

929
\[
\begin{align*}
\{ & \text{fix } U \\
& \text{assume } U \in \text{ExcludedSet}(X, T) \subseteq A \\
& \text{then have } T \cap U = 0 \lor U = X \cup A \text{ using ExcludedSet_def by auto} \\
& \text{with } A \text{ assms have } T \cap U = 0U \subseteq A \text{ by auto} \\
& \text{then have } U = \emptyset \subseteq A - T \text{ by auto} \\
\} & \text{then have Interior}(A, \text{ExcludedSet}(X, T)) \subseteq A - T \text{ using Interior_def by auto} \\
& \text{ultimately have Interior}(A, \text{ExcludedSet}(X, T)) = A - T \text{ by auto} \\
\}
\end{align*}
\]

moreover
\[
\begin{align*}
& \text{have } X \in \text{ExcludedSet}(X, T) \text{ using ExcludedSet_def} \\
& \text{union_excludedset by auto} \\
& \text{then have Interior}(X, \text{ExcludedSet}(X, T)) = X \text{ using topology0.Top_2_L3} \\
& \text{topology0_excludedset by auto} \\
& \text{ultimately show thesis by auto} \\
\end{align*}
\]

qed

The closure of a set is itself if it is 0 or the union with T.

**lemma closure_set_excludedset:**

assumes \( A \subseteq X \)

shows \( \text{Closure}(A, \text{ExcludedSet}(X, T)) = (\text{if } A = 0 \text{ then } 0 \text{ else } A \cup (X \cap T)) \)

proof-

have \( 0 \in \text{ClosedCovers}(0, \text{ExcludedSet}(X, T)) \) using ClosedCovers_def

\text{closed_sets_excludedset by auto} \\
then have \( \text{Closure}(0, \text{ExcludedSet}(X, T)) \subseteq 0 \) using Closure_def by auto \\
hence \( \text{Closure}(0, \text{ExcludedSet}(X, T)) = 0 \) by blast

moreover
\[
\begin{align*}
& \text{assume } A: A \neq 0 \\
& \text{with assms have } (A \cup (X \cap T)) \{ \text{is closed in} \} \text{ExcludedSet}(X, T) \text{ using closed_sets_excludedset} \\
& \text{by blast} \\
& \text{then have } (A \cup (X \cap T)) \in \{ D \in \text{Pow}(X). D \{ \text{is closed in} \} \text{ExcludedSet}(X, T) \\
& \wedge A \subseteq D \} \\
& \text{using assms by auto} \\
& \text{then have } (A \cup (X \cap T)) \in \text{ClosedCovers}(A, \text{ExcludedSet}(X, T)) \text{ unfolding ClosedCovers_def} \\
& \text{using union_excludedset by auto} \\
& \text{then have } \bigcap \text{ClosedCovers}(A, \text{ExcludedSet}(X, T)) \subseteq (A \cup (X \cap T)) \text{ by blast} \\
& \}
\end{align*}
\]

\[
\begin{align*}
& \text{fix } U \\
& \text{assume } U \in \text{ClosedCovers}(A, \text{ExcludedSet}(X, T)) \\
& \text{then have } U \{ \text{is closed in} \} \text{ExcludedSet}(X, T) \text{ and } A \subseteq U \text{ using ClosedCovers_def} \\
& \text{union_excludedset by auto} \\
& \text{then have } U = 0 \lor (X \cap T) \subseteq U \text{ and } A \subseteq U \text{ using closed_sets_excludedset}
\end{align*}
\]

930
by auto
with \( A \) have \((X \cap T) \subseteq U \) by auto
hence \((X \cap T) \cup A \subseteq U \) by auto
\}
with assms have \((A \cup (X \cap T)) \subseteq (\bigcap \text{ClosedCovers}(A, \text{ExcludedSet}(X, T)))\)
using topology0.Top_3_L3 topology0_excludedset union_excludedset
by auto
with 11 have \( \bigcap \text{ClosedCovers}(A, \text{ExcludedSet}(X, T)) = (A \cup (X \cap T)) \) by auto
then have \( \text{Closure}(A, \text{ExcludedSet}(X, T)) = (A \cup (X \cap T)) \) using Closure_def
by auto
\}
ultimately show thesis by auto
qed

The boundary of a set is 0 if \( A \) is \( X \) or 0, and \( X \cap T \) in other case.

lemma boundary_excludedset:
\[\text{assumes } A \subseteq X \]
\[\text{shows } \text{Boundary}(A, \text{ExcludedSet}(X, T)) = (\text{if } A = 0 \lor A = X \text{ then } 0 \text{ else } X \cap T)\]
proof-
\{ 
  have \( \text{Closure}(0, \text{ExcludedSet}(X, T)) = 0 \)
  \( \bigcap \text{Closure}(X - 0, \text{ExcludedSet}(X, T)) = X \)
  using closure_set_excludedset by auto
  then have \( \text{Boundary}(0, \text{ExcludedSet}(X, T)) = 0 \)
  using Boundary_def using union_excludedset assms by auto
\}
moreover
\{ 
  have \( X - X = 0 \) by blast
  then have \( \text{Closure}(X, \text{ExcludedSet}(X, T)) = X \) and \( \text{Closure}(X - X, \text{ExcludedSet}(X, T)) = 0 \)
  using closure_set_excludedset by auto
  then have \( \text{Boundary}(X, \text{ExcludedSet}(X, T)) = 0 \) unfolding Boundary_def using
  union_excludedset by auto
\}
moreover
\{ 
  assume \( A \neq 0 \) and \( A \neq X \)
  then have \( X - A \neq 0 \) using assms by auto
  with assms \( A \neq 0 \) \( \subseteq X \) have \( \text{Closure}(A, \text{ExcludedSet}(X, T)) = A \cup (X \cap T) \)
  using closure_set_excludedset by simp
  moreover
  from \( \subseteq X \) have \( X - A \subseteq X \) by blast
  with \( A \neq 0 \) have \( \text{Closure}(X - A, \text{ExcludedSet}(X, T)) = (X - A) \cup (X \cap T) \)
  using closure_set_excludedset by simp
  ultimately have \( \text{Boundary}(A, \text{ExcludedSet}(X, T)) = X \cap T \)
using Boundary_def union_excludedset by auto
}
ultimately show thesis by auto
qed

73.5 Special cases and subspaces

This section provides some miscellaneous facts about excluded set topologies.

The excluded set topology is equal in the sets \( T \) and \( X \cap T \).

**Lemma** smaller_excludedset:

- shows \( \text{ExcludedSet}(X, T) = \text{ExcludedSet}(X, (X \cap T)) \)

**Proof**

- show \( \text{ExcludedSet}(X, T) \subseteq \text{ExcludedSet}(X, X \cap T) \) and \( \text{ExcludedSet}(X, X \cap T) \subseteq \text{ExcludedSet}(X, T) \)
- unfolding ExcludedSet_def by auto

qed

If the set which is excluded is disjoint with \( X \), then the topology is discrete.

**Lemma** empty_excludedset:

- assumes \( T \cap X = 0 \)
- shows \( \text{ExcludedSet}(X, T) = \text{Pow}(X) \)

**Proof**

- from assms show \( \text{ExcludedSet}(X, T) \subseteq \text{Pow}(X) \) using smaller_excludedset
- unfolding ExcludedSet_def by auto
- from assms show \( \text{Pow}(X) \subseteq \text{ExcludedSet}(X, T) \) unfolding ExcludedSet_def by blast

qed

The topological subspaces of the \( \text{ExcludedSet} \ X \ T \) topology are also ExcludedSet topologies.

**Lemma** subspace_excludedset:

- shows \( \text{ExcludedSet}(X, T) \ {\text{restricted to}} \ Y = \text{ExcludedSet}(Y \cap X, T) \)

**Proof**

- fix \( M \)
- assume \( M \in (\text{ExcludedSet}(X, T) \ {\text{restricted to}} \ Y) \)
- then obtain \( A \) where \( A1:A: \text{ExcludedSet}(X, T) M=Y \cap A \) unfolding RestrictedTo_def by auto
- then have \( M \in \text{Pow}(X \cap Y) \) unfolding ExcludedSet_def by auto
- moreover
- from \( A1 \) have \( T \cap M = 0 \lor M = Y \cap X \) unfolding ExcludedSet_def by blast
- ultimately have \( M \in \text{ExcludedSet}(Y \cap X, T) \) unfolding ExcludedSet_def by auto

- then show \( \text{ExcludedSet}(X, T) \ {\text{restricted to}} \ Y \subseteq \text{ExcludedSet}(Y \cap X, T) \) by auto

932
\begin{verbatim}
{ fix M
  let A = M \cup ((X\cap Y-T)-Y)
  assume A:M \in ExcludedSet(Y\cap X,T)
  { assume M = Y \cap X
    then have M \in ExcludedSet(X,T) \{restricted to\} Y unfolding RestrictedTo_def
       ExcludedSet_def by auto
  }
  moreover
  { assume AS:M \neq Y \cap X
    from A AS have A1:(M\inPow(Y \cap X) \land T\cap M=0) unfolding ExcludedSet_def
       by auto
    then have A\inPow(X) by blast
    moreover
    have T\cap A=T\cap M by blast
    with A1 have T\cap A=0 by auto
    ultimately have A \inExcludedSet(X,T) unfolding ExcludedSet_def by auto
    then have AT:Y \cap A \inExcludedSet(X,T) \{restricted to\} Y unfolding RestrictedTo_def
       by auto
    have Y \cap A=Y \cap M by blast
    also have \ldots=\ldots using A1 by auto
    finally have Y\cap A = M by simp
    with AT have M \inExcludedSet(X,T) \{restricted to\} Y by auto
  }
  ultimately have M \inExcludedSet(X,T) \{restricted to\} Y by auto
  then show ExcludedSet(Y \cap X,T) \subseteq ExcludedSet(X,T) \{restricted to\} Y by auto
  qed

73.6 Included Set Topology

In this section we consider the subsets of a set which contain a fixed set.
The family defined in this section and the one in the previous section are
dual; meaning that the closed set of one are the open sets of the other.

We define the included set topology as the collection of supersets of some
fixed subset of the space X.

definition
IncludedSet(X,U) \equiv \{F\inPow(X) \land U \subseteq F\} \cup \{0\}

In the next theorem we prove that IncludedSet X Q forms a topology.

theorem includedset_is_topology:
  shows IncludedSet(X,Q) \{is a topology\}
\end{verbatim}
proof-
{  
  fix M  
  assume M ∈ Pow(IncludedSet(X,Q))  
  then have A:M⊆(F∈Pow(X). Q ⊆ F)∪{0} using IncludedSet_def by auto  
  then have ∪M∈Pow(X) by auto  
  moreover  
  have Q ⊆ ∪M∪ M=0 using A by blast  
  ultimately have ∪M∈IncludedSet(X,Q) using IncludedSet_def by auto  
}
moreover
{  
  fix U V  
  assume U∈IncludedSet(X,Q) V∈IncludedSet(X,Q)  
  then have U∈Pow(X)V∈Pow(X)U=0∨ Q⊆UV=0∨ Q⊆V using IncludedSet_def by auto  
  then have (U ∩ V)∈IncludedSet(X,Q) using IncludedSet_def by auto  
}
ultimately show thesis using IsATopology_def by auto
qed

We can reference the theorems proven in the topology0 context when discussing the included set topology.

theorem topology0Includedset:
  shows topology0(IncludedSet(X,T))
  using topology0_def includedset_is_topology by auto

Choosing a singleton set, it is considered a point excluded topology. In the following lemmas and theorems, when necessary it will be considered that T≠0 and T⊆X. These cases will appear in the special cases section.

definition
  IncludedPoint (IncludedPoint _ _ 90) where
  IncludedPoint X p ≡ IncludedSet(X,{p})

73.7 Basic topological notions in included set topology

This section discusses total set, closed sets, interior, closure and boundary for included set topology.

The topology is defined in the set X.

lemma union_includedset:
  assumes T⊆X  
  shows ∪IncludedSet(X,T) = X

proof-
  from assms have X ∈ IncludedSet(X,T) using IncludedSet_def by auto  
  then show ∪IncludedSet(X,T) = X using IncludedSet_def by auto
qed
The closed sets are those which are disjoint with $T$ and $X$.

**lemma closed_sets_includedset:**

assumes $T \subseteq X$

shows $D \{\text{is closed in} \} \text{IncludedSet}(X,T) \iff (D \in \text{Pow}(X) \land (D \cap T) = 0) \lor D = X$

**proof**-

- have $X - X = 0$ by blast
- then have $X - X \in \text{IncludedSet}(X,T)$ using IncludedSet_def by auto
- moreover
  - assume $A: D \subseteq X - D \in \text{IncludedSet}(X,T)$ $D \neq X$
    - from $A(2)$ have $T \subseteq (X - D) \lor X - D = 0$ using IncludedSet_def by auto
    - with $A(1)$ have $T \subseteq (X - D) \lor D = X$ by blast
    - with $A(3)$ have $T \subseteq (X - D)$ by auto
      - hence $D \cap T = 0$ by blast
  - moreover
    - assume $A: D \cap T = 0$ $D \subseteq X$
      - from $A(1)$ assms have $T \subseteq (X - D)$ by blast
      - then have $X - D \in \text{IncludedSet}(X,T)$ using IncludedSet_def by auto
  - ultimately show thesis using IsClosed_def union_includedset assms by auto

**qed**

The interior of a set is itself if it is open or the empty set if it isn’t.

**lemma interior_set_includedset:**

assumes $A \subseteq X$

shows $\text{Interior}(A, \text{IncludedSet}(X,T)) = (\text{if } T \subseteq A \text{ then } A \text{ else } 0)$

**proof**-

- fix $x$
  - assume $A: \text{Interior}(A, \text{IncludedSet}(X,T)) \neq 0$ $x \in T$
    - have $\text{Interior}(A, \text{IncludedSet}(X,T)) \in \text{IncludedSet}(X,T)$ using topology0.Top_2_L2 topology0_includedset by auto
    - with $A(1)$ have $T \subseteq \text{Interior}(A, \text{IncludedSet}(X,T))$ using IncludedSet_def by auto
    - with $A(2)$ have $x \in \text{Interior}(A, \text{IncludedSet}(X,T))$ by auto
      - then have $x \in A$ using topology0.Top_2_L1 topology0_includedset by auto
  - moreover
    - assume $T \subseteq A$
      - with assms have $A \in \text{IncludedSet}(X,T)$ using IncludedSet_def by auto
      - then have $\text{Interior}(A, \text{IncludedSet}(X,T)) = A$ using topology0.Top_2_L3 topology0_includedset by auto
  - ultimately show thesis by auto

**qed**

935
The closure of a set is itself if it is closed or the whole space if it is not.

**lemma** closure_set_includedset:  
assumes $A \subseteq X$  
$T \subseteq X$  
shows $\text{Closure}(A, \text{IncludedSet}(X,T)) = (\text{if } T \cap A = 0 \text{ then } A \text{ else } X)$

**proof** -  
{  
  assume $A \subseteq X$  
  then $A$ (is closed in) $\text{IncludedSet}(X,T)$ using closed_sets_includedset  
  with assms(1) have $\text{Closure}(A, \text{IncludedSet}(X,T)) = A$ using topology0.Top_3_L8  
  topology0_includedset union_includedset assms(2) by auto
}

moreover {  
  assume $A \subseteq X$  
  have $X \subseteq \bigcap \text{ClosedCovers}(A, \text{IncludedSet}(X,T))$ using ClosedCovers_def  
  closed_sets_includedset union_includedset assms by auto
}

moreover {  
  fix $U$  
  assume $U \subseteq \bigcap \text{ClosedCovers}(A, \text{IncludedSet}(X,T))$  
  then have $U$ (is closed in) $\text{IncludedSet}(X,T) \subseteq U$ using ClosedCovers_def  
  by auto
}

ultimately have $\bigcap \text{ClosedCovers}(A, \text{IncludedSet}(X,T))$  
ultimately have $\bigcap \bigcup \text{ClosedCovers}(A, \text{IncludedSet}(X,T)) = X$ by auto

ultimately have $\text{Closure}(A, \text{IncludedSet}(X,T)) = X$  
using Closure_def by auto

ultimately show thesis by auto

qed

The boundary of a set is $X - A$ if $A$ contains $T$ completely, is $A$ if $X - A$ contains $T$ completely and $X$ if $T$ is divided between the two sets. The case where $T = 0$ is considered as a special case.

**lemma** boundary_includedset:  
assumes $A \subseteq X$  
$T \subseteq X$  
$T \neq 0$  
shows $\text{Boundary}(A, \text{IncludedSet}(X,T)) = (\text{if } T \subseteq A \text{ then } X - A \text{ else } (\text{if } T \cap A = 0 \text{ then } A \text{ else } X))$

**proof** -  
{  
  from $A \subseteq X$ have $X - A \subseteq X$ by auto
}
assume $T \subseteq A$
with assms(2,3) have $T \cap A \neq 0$ and $T \cap (X-A) = 0$ by auto
with assms(1,2) $<X-A \subseteq X>$ have
  \[
  \text{Closure}(A, \text{IncludedSet}(X,T)) = X \quad \text{and} \quad \text{Closure}(X-A, \text{IncludedSet}(X,T)) = (X-A)
  \]
  using closure_set_includedset by auto
with assms(2) have $\text{Boundary}(A, \text{IncludedSet}(X,T)) = X-A$
  using Boundary_def union_includedset by auto
}
moreover
{
  assume $\neg (T \subseteq A)$ and $T \cap A = 0$
  with assms(2) have $T \cap (X-A) \neq 0$ by auto
  with assms(1,2) $<T \cap A = 0 > <X-A \subseteq X>$ have
    \[
    \text{Closure}(A, \text{IncludedSet}(X,T)) = A \quad \text{and} \quad \text{Closure}(X-A, \text{IncludedSet}(X,T)) = X
    \]
    using closure_set_includedset by auto
  with assms(1,2) have $\text{Boundary}(A, \text{IncludedSet}(X,T)) = A$
    using Boundary_def union_includedset by auto
}
moreover
{
  assume $\neg (T \subseteq A)$ and $T \cap A \neq 0$
  with assms(1,2) have $T \cap (X-A) \neq 0$ by auto
  with assms(1,2) $<T \cap A \neq 0 > <X-A \subseteq X>$ have
    \[
    \text{Closure}(A, \text{IncludedSet}(X,T)) = X \quad \text{and} \quad \text{Closure}(X-A, \text{IncludedSet}(X,T)) = X
    \]
    using closure_set_includedset by auto
  with assms(2) have $\text{Boundary}(A, \text{IncludedSet}(X,T)) = X$
    using Boundary_def union_includedset by auto
}
ultimately show thesis by auto
qed

73.8 Special cases and subspaces

In this section we discuss some corner cases when some parameters in our definitions are empty and provide some facts about subspaces in included set topologies.

The topology is discrete if $T = 0$

**Lemma smaller_includedset:**
shows $\text{IncludedSet}(X,0) = \text{Pow}(X)$

**Proof**
show $\text{IncludedSet}(X,0) \subseteq \text{Pow}(X)$ and $\text{Pow}(X) \subseteq \text{IncludedSet}(X,0)$
unfolding $\text{IncludedSet_def}$ by auto

qed
If the set which is included is not a subset of \( X \), then the topology is trivial.

**Lemma empty_includedset:**

assumes \(- (T \subseteq X)\)

shows \( \text{IncludedSet}(X, T) \subseteq \{0\} \)

proof

from assms show \( \text{IncludedSet}(X, T) \subseteq \{0\} \) and \( \{0\} \subseteq \text{IncludedSet}(X, T) \)

unfolding IncludedSet_def by auto

qed

The topological subspaces of the \( \text{IncludedSet}(X, T) \) topology are also \( \text{IncludedSet} \) topologies. The trivial case does not fit the idea in the demonstration because if \( Y \subseteq X \) then \( \text{IncludedSet}(Y \cap X, Y \cap T) \) is never trivial. There is no need for a separate proof because the only subspace of the trivial topology is itself.

**Lemma subspace_includedset:**

assumes \( T \subseteq X \)

shows \( \text{IncludedSet}(X, T) \) \( \{\text{restricted to}\} \ Y = \text{IncludedSet}(Y \cap X, Y \cap T) \)

proof

fix \( M \)

assume \( M \in (\text{IncludedSet}(X, T) \) \( \{\text{restricted to}\} \ Y) \)

then obtain \( A \) where \( A: \text{IncludedSet}(X, T) M = Y \cap A \)

unfolding RestrictedTo_def by auto

then have \( M \in \mathcal{P}(Y \cap X) \)

unfolding IncludedSet_def by auto

moreover

from \( A \) have \( Y \cap T \subseteq M \lor M = 0 \)

unfolding IncludedSet_def by blast

ultimately have \( M \in \text{IncludedSet}(Y \cap X, Y \cap T) \)

unfolding IncludedSet_def by auto

then show \( \text{IncludedSet}(X, T) \) \( \{\text{restricted to}\} \ Y \subseteq \text{IncludedSet}(Y \cap X, Y \cap T) \)

by auto

fix \( M \)

let \( A = M \cup T \)

assume \( A: M \in \text{IncludedSet}(Y \cap X, Y \cap T) \)

{ assume \( M=0 \)

then have \( M \in \text{IncludedSet}(X, T) \) \( \{\text{restricted to}\} \ Y \)

unfolding RestrictedTo_def

IncludedSet_def by auto

}

moreover

{ assume \( AS: M \neq 0 \)

from \( A \) \( AS \) have \( A: M \in \mathcal{P}(Y \cap X) \land Y \cap T \subseteq M \)

unfolding IncludedSet_def by auto

then have \( A \in \mathcal{P}(X) \)

using assms by blast

938
moreover
have $T \subseteq A$ by blast
ultimately have $A \in \text{IncludedSet}(X,T)$ unfolding IncludedSet_def by auto
then have $AT : Y \cap A \in \text{IncludedSet}(X,T)$ {restricted to} $Y$
unfolding RestrictedTo_def by auto
from $A1$ have $Y \cap A = Y \cap M$ by blast
also from $A1$ have $\ldots = M$ by auto
finally have $Y \cap A = M$ by simp
with $AT$ have $M \in \text{IncludedSet}(X,T)$ {restricted to} $Y$
by auto
}
ultimately have $M \in \text{IncludedSet}(X,T)$ {restricted to} $Y$ by auto
}
thus $\text{IncludedSet}(Y \cap X, Y \cap T) \subseteq \text{IncludedSet}(X,T)$ {restricted to} $Y$ by auto
qed
end

74 More examples in topology

theory Topology_ZF_examples_1
imports Topology_ZF_1 Order_ZF
begin

In this theory file we reformulate the concepts related to a topology in relation with a base of the topology and we give examples of topologies defined by bases or subbases.

74.1 New ideas using a base for a topology

74.2 The topology of a base

Given a family of subsets satisfying the base condition, it is possible to construct a topology where that family is a base of. Even more, it is the only topology with such characteristics.

definition
TopologyWithBase (TopologyBase _ 50) where
U {satisfies the base condition} $\longrightarrow$ TopologyBase U $\equiv$ THE T. U {is a base for} T

If a collection $U$ of sets satisfies the base condition then the topology constructed from it is indeed a topology and $U$ is a base for this topology.

theorem Base_topology_is_a_topology:
assumes U {satisfies the base condition}
shows (TopologyBase U) {is a topology} and U {is a base for} (TopologyBase U)
proof
  from assms obtain T where U {is a base for} T using
    Top_1_2_T1(2) by blast
  then have ∃!T. U {is a base for} T using same_base_same_top ex1I[where
    P= λT. U {is a base for} T] by blast
  with assms show U {is a base for} T using
    same_base_same_top ex1I[where
      P= λT. U {is a base for} T] by blast
  with assms show (TopologyBase U) {is a topology} using
    Top_1_2_T1(1) IsAbaseFor_def by auto
  qed

A base doesn’t need the empty set.

lemma base_no_0:
  shows B{is a base for}T ↔ (B-{0}){is a base for}T
proof
  { fix M
    assume M ∈ {⋃A . A ∈ Pow(B)}
    then obtain U where M=⋃Q∈Pow(B) by auto
    hence M=⋃(Q-{0})Q-{0}∈Pow(B-{0}) by auto
    hence M∈{⋃A . A ∈ Pow(B - {0})} by auto
  } hence {⋃A . A ∈ Pow(B)} ⊆ {⋃A . A ∈ Pow(B - {0})} by blast
  moreover
  { fix M
    assume M∈{⋃A . A ∈ Pow(B - {0})}
    then obtain U where M=⋃Q∈Pow(B - {0}) by auto
    hence M=⋃(Q)Q∈Pow(B) by auto
    hence M∈{⋃A . A ∈ Pow(B)} by auto
  } hence {⋃A . A ∈ Pow(B - {0})} ⊆ {⋃A . A ∈ Pow(B)}
    by auto
    ultimately have {⋃A . A ∈ Pow(B - {0})} = {⋃A . A ∈ Pow(B)} by auto
    then show B{is a base for}T ↔ (B-{0}){is a base for}T using IsAbaseFor def
    by auto
  qed

The interior of a set is the union of all the sets of the base which are fully
contained by it.

lemma interior_set_base_topology:
  assumes U {is a base for} T T{is a topology}
  shows Interior(A,T) = ⋃{T∈U. T⊆A}
proof
  have {T∈U. T⊆A}⊆U by auto

940
with assms(1) have \( \bigcup \{ T \in U. T \subseteq A \} \subseteq T \)
  using IsAbaseFor_def by auto
moreover have \( \bigcup \{ T \in U. T \subseteq A \} \subseteq A \)
  by blast
ultimately show \( \bigcup \{ T \in U. T \subseteq A \} \subseteq \text{Interior}(A,T) \)
  using assms(2) topology0.Top_2_L5 topology0_def by auto

\[
\{ \text{fix } x \text{ assume } x \in \text{Interior}(A,T) \text{ with assms(2,3) have } B: x \in \bigcup Q \text{ using topology0_def topology0.Top_3_L11(1) by blast} \}
\]
with assms have \( V \in U x \in V \subseteq A \) using topology0.Top_2_L2 topology0_def
  by auto
hence \( x \in \bigcup \{ T \in U. T \subseteq A \} \) by auto

thus \( \text{Interior}(A,T) \subseteq \bigcup \{ T \in U. T \subseteq A \} \)
  by auto
qed

In the following, we offer another lemma about the closure of a set given a basis for a topology. This lemma is based on cl_inter_neigh and inter_neigh_cl. It states that it is only necessary to check the sets of the base, not all the open sets.

lemma closure_set_base_topology:
  assumes U {is a base for} Q Q {is a topology} A \( \subseteq \bigcup Q \)
  shows \( \text{Closure}(A,Q) = \{ x \in \bigcup Q. \ \forall T \in U. x \in T \longrightarrow A \cap T \neq 0 \} \)
proof
  \{
  fix x
  assume A: \( x \in \text{Closure}(A,Q) \)
  with assms(2,3) have B: \( x \in \bigcup Q \) using topology0_def topology0.Top_3_L11(1)
    by blast
  moreover
  \{
  fix T
  assume T \in U x \in T
  with assms(1) have \( T \cap Q \in T \) using base_sets_open by auto
  with assms(2,3) A have \( A \cap T \neq 0 \) using topology0_def topology0.cl_inter_neigh
    by auto
  \}
  hence \( \forall T \in U. x \in T \longrightarrow A \cap T \neq 0 \) by auto
  ultimately have \( x \in \{ x \in \bigcup Q. \ \forall T \in U. x \in T \longrightarrow A \cap T \neq 0 \} \) by auto
  \}
thus \( \text{Closure}(A,Q) \subseteq \{ x \in \bigcup Q. \ \forall T \in U. x \in T \longrightarrow A \cap T \neq 0 \} \)
  by auto

  \{
  fix x
  assume AS: \( x \in \bigcup Q. \ \forall T \in U. x \in T \longrightarrow A \cap T \neq 0 \)
  hence \( x \in \bigcup Q \) by blast
  moreover
  \}
{ fix R
  assume R :∈ Q
  with assms(1) obtain W where RR:W⊆U R=∪W using
    IsAbaseFor_def by auto
  { assume x ∈ R
    with RR(2) obtain WW where TT:WW∈WxWW by auto
    { assume R ∩ A = 0
      with RR(2) TT(1) have WW∩A=0 by auto
      with TT(1) RR(1) have W⊆U WW∩A=0 by auto
      with AS have x∈∪Q-WW by auto
      with TT(2) have False by auto
    } hence R∩A≠0 by auto
  }
  hence ∀U∈Q. x∈U → U∩A≠0 by auto
  ultimately have x∈Closure(A,Q) using assms(2,3) topology0_def topology0.inter_neigh_cl
    by auto
  } then show {x ∈ ∪Q . ∀T∈U. x ∈ T −→ A ∩ T ≠ 0} ⊆ Closure(A,Q)
    by auto
qed

The restriction of a base is a base for the restriction.

lemma subspace_base_topology:
  assumes B {is a base for} T
  shows (B {restricted to} Y) {is a base for} (T {restricted to} Y)
proof -
  from assms have (B {restricted to} Y) ⊆ (T {restricted to} Y)
  unfolding IsAbaseFor_def RestrictedTo_def by auto
  moreover have (T {restricted to} Y) = {∪A. A ∈ Pow(B {restricted to} Y))
  proof
    { fix U assume U ∈ (T {restricted to} Y)
      then obtain  with WW∈T and U = W∩Y unfolding RestrictedTo_def
      by blast
      with assms obtain C where C∈Pow(B) and W=∪C unfolding IsAbaseFor_def
      by blast
      let A={V∩Y. V∈C}
      from <C∈Pow(B)> <U = W∩Y> <W=∪C> have
      A ∈ Pow(B {restricted to} Y) and U=(∪A)
      unfolding RestrictedTo_def by auto
      hence U ∈ {∪A. A ∈ Pow(B {restricted to} Y)) by blast
    } thus (T {restricted to} Y) ⊆ {∪A. A ∈ Pow(B {restricted to} Y))
      by auto
  }
{ fix U assume U ∈ {∪ A. A ∈ Pow(B {restricted to} Y)}
then obtain A where A: A ⊆ (B {restricted to} Y) and U = (∪A)
by auto
let A₀ = {C ∈ B. Y ∩ C ∈ A}
from A have A₀ ⊆ B and A = A₀ {restricted to} Y unfolding RestrictedTo_def
by auto
with ⟨U = (∪A)⟩ have A₀ ⊆ B and U = ∪(A₀ {restricted to} Y)
by auto
with assms have U ∈ (T {restricted to} Y) unfolding RestrictedTo_def
IsAbaseFor_def
by auto
} thus {∪ A. A ∈ Pow(B {restricted to} Y)} ⊆ (T {restricted to} Y)
by blast
qed
ultimately show thesis unfolding IsAbaseFor_def by simp
qed

If the base of a topology is contained in the base of another topology, then
the topologies maintain the same relation.

theorem base_subset:
assumes B{is a base for}T2{is a base for}T2B ⊆ B2
shows T ⊆ T2
proof
{ fix x
assume x ∈ T
with assms(1) obtain M where M ⊆ Bx = ∪ M using IsAbaseFor_def by auto
with assms(3) have M ⊆ B²x = ∪ M by auto
with assms(2) show x ∈ T₂ using IsAbaseFor_def by auto
}
qed

74.3 Dual Base for Closed Sets

A dual base for closed sets is the collection of complements of sets of a base
for the topology.

definition
DualBase (DualBase _ _ 80) where
B{is a base for}T \implies DualBase B T \equiv (∪ T - U. U ∈ B) ∪ (∪ T)

lemma closed_inter_dual_base:
assumes D{is closed in}TB{is a base for}T
obtains M where M ⊆ DualBase B T \equiv \bigcap M
proof-
assume K: \bigwedge M. M ⊆ DualBase B T \implies D = \bigcap M \implies thesis
{ assume AS:D \neq \bigcup T

943
from assms(1) have D:D ∈ ℙ(⋃T)⋃T-D ∈ T using IsClosed_def by auto
hence A:⋃(⋃T-D)=D|=T-D∈T by auto
with assms(2) obtain Q where QQ:Q∈ℙ(⋃T-D)=⋃Q using IsAbaseFor_def by auto

{ assume Q=0
  then have ∪Q=0 by auto
  with QQ(2) have ∪T-D=0 by auto
  with D(1) have D=⋃T by auto
  with AS have False by auto
}
hence QNN:Q≠0 by auto
from QQ(2) A(1) have D=⋃T-⋃Q by auto
with QNN have D=⋃T-R. R∈Q by auto
moreover
with assms(2) QQ(1) have {⋃T-R. R∈Q}⊆DualBase B T using DualBase_def by auto
with calculation K have thesis by auto
}
moreover
{ assume AS:D=⋃T
  with assms(2) have {⋃T}⊆DualBase B T using DualBase_def by auto
  moreover
  have ∪T = ∩{⋃T} by auto
  with calculation K AS have thesis by auto
}
ultimately show thesis by auto
qed

We have already seen for a base that whenever there is a union of open sets,
we can consider only basic open sets due to the fact that any open set is a
union of basic open sets. What we should expect now is that when there is
an intersection of closed sets, we can consider only dual basic closed sets.

lemma closure_dual_base:
  assumes U {is a base for} QQ{is a topology}A⊆⋃Q
  shows Closure(A,Q)=⋂{T∈DualBase U Q. A⊆T}
proof
  from assms(1) have T:⋃Q∈DualBase U Q using DualBase_def by auto
  moreover
  { fix T
    assume A:T∈DualBase U Q A⊆T
    with assms(1) obtain R where (T=⋃Q-R R∈U)∨T=⋃Q using DualBase_def by auto
    with A(2) assms(1,2) have (T is closed in)Q A⊆T∈ℙ(⋃Q) using
topology0.Top_3_L1 topology0_def
    topology0.Top_3_L9 base_sets_open by auto
  }

  944
then have $\{T \in \text{DualBase} \cup Q. A \subseteq T\} \subseteq \{T \in \text{Pow}(\bigcup Q). (T \text{ is closed in } Q) \wedge A \subseteq T\}$
   by blast
   with calculation assms(3) have $\bigcap \{T \in \text{Pow}(\bigcup Q). (T \text{ is closed in } Q) \wedge A \subseteq T\} \subseteq \{T \in \text{DualBase} \cup Q. A \subseteq T\}$
   by auto
then show $\text{Closure}(A,Q) \subseteq \{T \in \text{DualBase} \cup Q. A \subseteq T\}$ using $\text{Closure}_\text{def}$ $\text{ClosedCovers}_\text{def}$
   by auto

\begin{itemize}
\item fix $x$
\item assume $A \colon x \in \bigcap \{T \in \text{DualBase} \cup Q. A \subseteq T\}$
\item fix $T$
\item assume $B \colon x \in T \in U$
\item assume $C \colon A \cap T = 0$
\item from $B(2)$ assms(1) have $\bigcup Q - T \in \text{DualBase} \cup Q$ using $\text{DualBase}_\text{def}$
   by auto
\item moreover
\item from $C$ assms(3) have $A \subseteq \bigcup Q - T$ by auto
\item moreover
\item from $B(1)$ have $x \notin \bigcup Q - T$ by auto
\item ultimately have $x \notin \bigcap \{T \in \text{DualBase} \cup Q. A \subseteq T\}$ by auto
\item with $A$ have $\text{False}$ by auto
\item hence $A \cap T \neq 0$ by auto
\item hence $\forall T \in U. x \in T \rightarrow A \cap T \neq 0$ by auto
\item moreover
\item from $T \ A$ assms(3) have $x \in \bigcup Q$ by auto
\item with calculation assms have $x \in \text{Closure}(A,Q)$ using $\text{closure}_\text{set}_\text{base}_\text{topology}$
   by auto
\item thus $\bigcap \{T \in \text{DualBase} \cup Q. A \subseteq T\} \subseteq \text{Closure}(A, Q)$ by auto
\end{itemize}
qed

74.4 Partition topology

In the theory file Partitions_ZF.thy; there is a definition to work with partitions. In this setting is much easier to work with a family of subsets.

**definition**

$\text{IsAPartition} \ _\{\text{is a partition of}\}_\{\_90\}$ where

$(U \ \{\text{is a partition of}\} \ X) \equiv (\bigcup U = X \land (\forall A \in U. \ \forall B \in U. A = B \lor A \cap B = 0) \land 0 \notin U)$

A subcollection of a partition is a partition of its union.

**lemma** subpartition:

assumes $U \ \{\text{is a partition of}\} X V \subseteq U$
shows $V \ \{\text{is a partition of}\} \bigcup V$
using assms unfolding $\text{IsAPartition}_\text{def}$ by auto
A restriction of a partition is a partition. If the empty set appears it has to be removed.

**Lemma restriction_partition:**
assumes $U$ {is a partition of} $X$
shows $((U \{\text{restricted to} \ Y\})-{\emptyset})$ {is a partition of} $(X \cap Y)$
using assms unfolding IsAPartition_def RestrictedTo_def
by fast

Given a partition, the complement of a union of a subfamily is a union of a subfamily.

**Lemma diff_union_is_union_diff:**
assumes $R \subseteq P$ {is a partition of} $X$
shows $X - \bigcup R = \bigcup (P-R)$
proof
{
fix $x$
assume $x \in X - \bigcup R$
hence $P: x \in X \setminus \bigcup R$ by auto
{
fix $T$
assume $T \in R$
with P(2) have $x \notin T$ by auto
}
with P(1) assms(2) obtain $Q$ where $Q \in (P-R)x \in Q$ using IsAPartition_def
by auto
hence $x \in \bigcup (P-R)$ by auto
thus $X - \bigcup R \subseteq \bigcup (P-R)$ by auto
{
fix $x$
assume $x \in \bigcup (P-R)$
then obtain $Q$ where $Q \in \bigcup (P-R)\cap Q$ by auto
hence $C: Q \in P \cap Q \subseteq \bigcup Qx \in Q$ by auto
then have $x \in \bigcup P$ by auto
with assms(2) have $x \in X$ using IsAPartition_def by auto
moreover
{
assume $x \in \bigcup R$
then obtain $t$ where $G: t \in R \setminus x \in t$ by auto
with C(3) assms(1) have $t \cap Q \neq \emptyset$ by auto
with assms(2) C(1,3) have $t = Q$ using IsAPartition_def
by blast
with C(2) G(1) have False by auto
}
} hence $x \notin \bigcup R$ by auto
ultimately have $x \in X - \bigcup R$ by auto
thus $\bigcup (P-R) \subseteq X - \bigcup R$ by auto
qed
74.5 Partition topology is a topology.

A partition satisfies the base condition.

**lemma** partition_base_condition:
assumes P {is a partition of} X
shows P {satisfies the base condition}
**proof**
{
  fix U V
  assume AS:U\in P \land V\in P
  with asms have A:U=V \lor U\cap V=0 using IsAPartition_def by auto
  {
    fix x
    assume ASS:x\in U\cap V
    with A have U=V by auto
    with AS ASS have U\in P \land U\subseteq U \cap V by auto
    hence \exists W\in P. x\in W \land W \subseteq U \cap V by auto
  }
  hence (\forall x \in U \cap V. \exists W\in P. x\in W \land W \subseteq U \cap V) by auto
  then show thesis using SatisfiesBaseCondition_def by auto
  qed
}

Since a partition is a base of a topology, and this topology is uniquely determined; we can built it. In the definition we have to make sure that we have a partition.

**definition**
PartitionTopology (PTopology _ _ 50) where
(U {is a partition of} X) \implies PTopology X U \equiv TopologyBase U

**theorem** Ptopology_is_a_topology:
assumes U {is a partition of} X
shows (PTopology X U) {is a topology} and U {is a base for} (PTopology X U)
using asms Base_topology_is_a_topology partition_base_condition
  PartitionTopology_def by auto

**lemma** topology0_ptopology:
assumes U {is a partition of} X
shows topology0(PTopology X U)
using Ptopology_is_a_topology topology0_def asms by auto

74.6 Total set, Closed sets, Interior, Closure and Boundary

The topology is defined in the set X

**lemma** union_ptopology:
assumes U {is a partition of} X
shows \bigcup (PTopology X U)=X
The closed sets are the open sets.

**Lemma closed_sets_ptopology:**

** Assumes** \( T \) {is a partition of} \( X \)

** Shows** \( D \) {is closed in} \( (PTopology X T) \) \( \iff \) \( D \in (PTopology X T) \)

** Proof**

** From** **Assms**

** Have** \( B: T \) {is a base for} \( (PTopology X T) \) **Using** \( Ptopology_is_a_topology(2) \)

** By** **Auto**

\[
\begin{align*}
&\text{fix } D \\
&\quad \text{assume } D \text{ {is closed in} } (PTopology X T) \\
&\quad \text{with Assms have } A:D \subseteq \text{Pow}(X)X-D \subseteq \text{PTopology X T) using } \text{IsClosed_def union_ptopology by auto} \\
&\quad \text{from A(2) B obtain } R \text{ where } Q: R \subseteq T X-D=\bigcup R \text{ using } \text{Top_1_2_L1[where B=T and U=X-D]} \text{ by auto} \\
&\quad \text{from A(1) have } X-(X-D)=D \text{ by blast} \\
&\quad \text{with Q(2) have } D=X-\bigcup R \text{ by auto} \\
&\quad \text{with Q(1) assms have } D=\bigcup (T-R) \text{ using } \text{diff_union_is_union_diff by auto} \\
&\quad \text{with B show } D \in (PTopology X T) \text{ using } \text{IsAbaseFor_def by auto} \\
&\text{moreover from Q have } D \subseteq \bigcup T \text{ by auto} \\
&\quad \text{with Assms have } D \subseteq X \text{ using } \text{IsAPartition_def by auto} \\
&\quad \text{with Calculation assms show } D \text{ {is closed in} } (PTopology X T) \text{ using } \text{IsClosed_def union_ptopology by auto} \\
&\text{qed}
\end{align*}
\]

There is a formula for the interior given by an intersection of sets of the dual base. Is the intersection of all the closed sets of the dual basis such that they do not complement \( A \) to \( X \). Since the interior of \( X \) must be inside \( X \), we have to enter \( X \) as one of the sets to be intersected.

**Lemma interior_set_ptopology:**

** Assumes** \( U \) {is a partition of} \( X \) \( A \subseteq X \)

** Shows** \( \text{Interior}(A,(PTopology X U))=\bigcap \{ T \in \text{DualBase U (PTopology X U)} . T=X\setminus T \cup A \neq X} \)

948
proof
{
fix x
assume x∈Interior(A,(PTopology X U))
with assms obtain R where A:x∈R∈(PTopology X U)R⊆A
  using topology0.open_open_neigh topology0_ptopology
  topology0.Top_2_L2 topology0.Top_2_L1
by auto
with assms obtain B where B:B⊆UR=⋃B
  using Ptopology_is_a_topology(2)
IsAbaseFor_def
by auto
from A(1,3) assms
have XX:x∈XX∈{T∈DualBase U (PTopology X U). T=X∨T∪A≠X}
  using union_ptopology[of UX] DualBase_def[of U] Ptopology_is_a_topology(2)[of UX]
  by (safe,blast,auto)
moreover
from B(2) A(1) obtain S where C:S⊆UR by auto
{
fix T
assume AS:T∈DualBase U (PTopology X U)T∪A≠X
from AS(1) assms obtain w where (T=X∧w∈U)∨(T=X)
  using DualBase_def union_ptopology Ptopology_is_a_topology(2)
by auto
with assms(2) AS(2) have D:T=X∧w∈U by auto
from D(2) have w⊆U by auto
with assms(1) have w⊆⋃(PTopology X U) using Ptopology_is_a_topology(2)
Top_1_2_L5[of UPTopology X U]
  by auto
with assms(1) have w⊆X using union_ptopology by auto
with D(1) have X-T=w by auto
with D(2) have X-T∈U by auto
{
  assume x∈X-T
  with C B(1) have S∈US∩(X-T)≠0 by auto
  with ⟨X-T∈U⟩ assms(1) have X-T=S using IsAPartition_def by auto
  with ⟨X-T=w⟩ ⟨T=X∧w⟩ have X-S=T by auto
  with AS(2) have X-S∪A≠X by auto
  from A(3) B(2) C(1) have S⊆A by auto
  hence X-A⊆X-S by auto
  with assms(2) have X-S∪A=X by auto
  with ⟨X-S∪A≠X⟩ have False by auto
}
then have x∈T using XX by auto
}
ultimately have x∈⋃{T∈DualBase U (PTopology X U). T=X∨T∪A≠X}
  by auto
}
thus Interior(A,(PTopology X U))⊆⋃{T∈DualBase U (PTopology X U). T=X∨T∪A≠X}
  by auto
{
fix x

949
assume \( p: x \in \bigcap \{ T \in \text{DualBase \ U (PTopology X U)} \mid T \neq X \cup \{ A \} \} \)

hence noE: \( \bigcap \{ T \in \text{DualBase \ U (PTopology X U)} \mid T \neq X \cup \{ A \} \} \neq 0 \) by auto

{ fix \( T \)
  assume \( T \in \text{DualBase \ U (PTopology X U)} \)
  with \( \text{assms(1)} \) obtain \( w \) where \( T = X \cup (w \in \text{PTopology X U}) \land T = X - w \) using \( \text{DualBase_def} \)
  with \( \text{assms(1)} \) have \( T = X \cup (w \in \text{PTopology X U}) \land T = X - w \) using \( \text{base_sets_open} \)
  with \( \text{assms(1)} \) have \( T \) is closed in \( \text{PTopology X U} \) using \( \text{topology0.Top_3_L1}[\text{where } T = \text{PTopology X U}] \)
  topology0_ptopology topology0.Top_3_L9[\text{where } T = \text{PTopology X U}]
  by auto

moreover from \( \text{assms(1)} \) \( p \) have \( X \in \bigcap \{ T \in \text{DualBase \ U (PTopology X U)} \mid T \neq X \cup \{ A \} \} \) and

\( X: x \in X \) using \( \text{PTopology_is_a_topology(2)} \)
  \( \text{DualBase_def union_ptopology by auto} \)
  with calculation \( \text{assms(1)} \) have \( (\bigcap \{ T \in \text{DualBase \ U (PTopology X U)} \mid T \neq X \cup \{ A \} \} \} \) \( \text{is closed in} \) \( \text{PTopology X U} \)
  using \( \text{topology0.Top_3_L4}[\text{where } K = \{ T \in \text{DualBase \ U (PTopology X U)} \mid T \neq X \cup \{ A \} \} \] topology0_ptopology[\text{where } U = U \text{ and } X = X]
  by auto
  with \( \text{assms(1)} \) have \( ab: (\bigcap \{ T \in \text{DualBase \ U (PTopology X U)} \mid T \neq X \cup \{ A \} \} \in (\text{PTopology X U}) \)
  using \( \text{closed_sets_ptopology by auto} \)
  with \( \text{assms(1)} \) obtain \( B \) where \( B \in \text{Pow(U)} \) \( (\bigcap \{ T \in \text{DualBase \ U (PTopology X U)} \mid T \neq X \cup \{ A \} \} ) = \bigcup B \)
  using \( \text{PTopology_is_a_topology(2)} \) \( \text{IsAbaseFor_def by auto} \)
  with \( p \) obtain \( R \) where \( x \in R \in R \subseteq (\bigcap \{ T \in \text{DualBase \ U (PTopology X U)} \mid T \neq X \cup \{ A \} \} ) \)
  by auto
  with \( \text{assms(1)} \) have \( R: x \in R \in \text{R} \subseteq (\bigcap \{ T \in \text{DualBase \ U (PTopology X U)} \mid T \neq X \cup \{ A \} \} ) \) \( \text{R} \subseteq \text{DualBase \ U (PTopology X U)} \)
  using \( \text{base_sets_open} \) \( \text{PTopology_is_a_topology(2)} \) \( \text{DualBase_def union_ptopology by (safe, blast, simp, blast)} \)

} assume \( (X - R) \cup A \neq X \)

with \( R(4) \) have \( X - R \in \{ T \in \text{DualBase \ U (PTopology X U)} \mid T \neq X \cup \{ A \} \} \) by auto

hence \( \bigcap \{ T \in \text{DualBase \ U (PTopology X U)} \mid T \neq X \cup \{ A \} \} \subseteq X - R \) by auto

with \( R(3) \) have \( R \subseteq X - R \) using \( \text{subset_trans [where } A = R \text{ and } C = X - R \] by auto

hence \( R = 0 \) by blast

with \( R(1) \) have \( \text{False by auto} \)

hence \( I: (X - R) \cup A = X \) by auto

{ 950 \)
fix y
assume ASR: y ∈ R
with R(2) have y ∈ \bigcup (PTopology X U) by auto
with assms(1) have XX: y ∈ X using union_ptopology by auto
with I have y ∈ (X - R) ∪ A by auto
with XX have y ∉ R \forall y ∈ A by auto
with ASR have y ∈ A by auto
}
hence R ⊆ A by auto
with R(1,2) have \( \exists R ∈ \text{PTopology X U}) . (x ∈ R \cap A) \) by auto
with assms(1) have \( x \in \text{Interior}(A,(PTopology X U)) \) using topology0.Top_2_L6
topology0_ptopology by auto
}
thus \( \bigcap \{ T ∈ \text{DualBase U PTopology X U} : T = X \lor T \cup A \neq X \} \subseteq \text{Interior}(A,\text{PTopology X U}) \)
by auto
qed

The closure of a set is the union of all the sets of the partition which intersect with A.

lemma closure_set_ptopology:
assumes U {is a partition of} X \( A \subseteq X \)
shows \( \text{Closure}(A,\text{PTopology X U})=\bigcup \{ T \in U : T \cap A \neq 0 \} \)
proof
{
fix x
assume A: x ∈ \text{Closure}(A,\text{PTopology X U})
with assms have x ∈ \bigcup (PTopology X U) using topology0.Top_3_L11[where T=PTopology X U
and A=A] topology0_ptopology union_ptopology by auto
with assms(1) have x ∈ \bigcup U using Top_1_2_L5[where B=U and T=PTopology X U] Ptopology_is_a_topology(2) by auto
then obtain W where B:x ∈ W \subseteq U by auto
with A have x ∈ \text{Closure}(A,\text{PTopology X U}) \cap W by auto
moreover from assms B(2) have W ∈ (PTopology X U)A \subseteq X using base_sets_open Ptopology_is_a_topology by (safe,blast)
with calculation assms(1) have A ∩ W \neq 0 using topology0_ptopology[where U=U and X=X]
topology0.cl_inter_neigh union_ptopology by auto
with B have x ∈ \bigcup \{ T \in U : T \cap A \neq 0 \} by blast
}
thus \( \text{Closure}(A,\text{PTopology X U}) \subseteq \bigcup \{ T \in U : T \cap A \neq 0 \} \) by auto
{
fix x
assume x ∈ \bigcup \{ T \in U : T \cap A \neq 0 \}
then obtain T where A:x ∈ TT \subseteq UT \cap A \neq 0 by auto
from assms have A \subseteq \bigcup (PTopology X U) using union_ptopology by auto
moreover

951
from A(1,2) assms(1) have x⊂∪(PTopology X U) using Top_1_2_L5[where B=U and T=PTopology X U]
  Ptopology_is_a_topology(2) by auto
moreover
{ 
  fix Q 
  assume B:Q⊂(PTopology X U)x⊂Q 
  with assms(1) obtain H where C:Q=∪MM⊂U using 
    Ptopology_is_a_topology(2) 
    IsAbaseFor_def by auto 
  from B(2) C(1) obtain R where D:R⊂MX⊂R by auto 
  with C(2) A(1,2) have R∪T≠O⊂UT⊂U by auto 
  with assms(1) have R=T using IsAPartition_def by auto 
  with C(1) D(1) have R=T using IsAPartition_def by auto 
  with A(3) have Q∩A≠0 by auto 
}
then have ∀Q⊂(PTopology X U). x⊂Q → Q∩A≠0 by auto 
  with calculation assms(1) have x⊂Closure(A,(PTopology X U)) using 
    topology0.inter_neigh_cl 
    topology0_ptopology by auto 
}
then show ∪{T ∈ U . T ∩ A ≠ 0} ⊆ Closure(A, PTopology X U) by auto 
qed

The boundary of a set is given by the union of the sets of the partition which
have non empty intersection with the set but that are not fully contained in
it. Another equivalent statement would be: the union of the sets of the par-
tition which have non empty intersection with the set and its complement.

lemma boundary_set_ptopology: 
  assumes U {is a partition of} XA⊂X 
  shows Boundary(A,(PTopology X U))=∪{T⊂U . T∩A≠0 ∧ ~(T⊂A)} 
proof-
  from assms have Closure(A,(PTopology X U))=∪{T⊂U . T∩A≠0} using 
    closure_set_ptopology by auto 
  moreover 
  from assms(1) have Interior(A,(PTopology X U))=∪{T⊂U . T ⊆ A} using 
    interior_set_base_topology Ptopology_is_a_topology[where U=U and 
    X=X] by auto 
  with calculation assms have A:Boundary(A,(PTopology X U))=∪{T⊂U . T∩A≠0} - ∪{T⊂U . T ⊆ A} 
    using topology0.Top_3_L12 topology0_ptopology union_ptopology 
    by auto 
  from assms(1) have {(T⊂U . T∩A≠0)} {is a partition of} U{(T⊂U . T∩A≠0)} 
    using subpartition by blast 
  moreover 
  { 

952
fix \( T \)
assume \( T \in U \subseteq A \)
with assms(1) have \( T \cap A \neq T \neq 0 \) using IsAPartition_def by auto
with \( \langle T \in U \rangle \) have \( T \neq 0 \) using \( T \cap A \neq 0 \) by auto
\}
then have \( \{ T \in U . \ T \subseteq A \} \subseteq \{ T \in U . \ T \cap A \neq 0 \} \) by auto
ultimately have \( \bigcup \{ T \in U . \ T \cap A \neq 0 \} - \bigcup \{ T \in U . \ T \subseteq A \} = \{(T \in U . \ T \cap A \neq 0) - (T \in U . \ T \subseteq A)\} \) by blast
with calculation A show thesis by auto
qed

74.7 Special cases and subspaces

The discrete and the indiscrete topologies appear as special cases of this partition topologies.

lemma discrete_partition:
shows \( \{\{x\}.x \in X\} \) {is a partition of} \( X \)
using IsAPartition_def by auto

lemma indiscrete_partition:
assumes \( X \neq 0 \)
shows \( \{X\} \) {is a partition of} \( X \)
using assms IsAPartition_def by auto

theorem discrete_ptopology:
shows \( \{\{x\}.x \in X\}\subseteq \{\{x\}.x \in X\} = \text{Pow}(X) \)
proof
\{
fix \( t \)
assume \( t \in \{\{x\}.x \in X\} \)
hence \( t \subseteq \bigcup \{\{x\}.x \in X\} \) by auto
then have \( t \in \text{Pow}(X) \) using union_ptopology
discrete_partition by auto
\}
thus \( \{\{x\}.x \in X\}\subseteq \text{Pow}(X) \) by auto
\{
fix \( t \)
assume \( A.t \in \text{Pow}(X) \)
have \( \bigcup \{\{x\}.x \in t\} = t \) by auto
moreover
from \( A \) have \( \{\{x\}.x \in t\} \in \text{Pow}(\{\{x\}.x \in X\}) \) by auto
hence \( \bigcup \{\{x\}.x \in t\} \in \bigcup A . A \in \text{Pow}(\{\{x\}.x \in X\}) \) by auto
ultimately
have \( t \in \{\{x\}.x \in X\} \) using Ptopology_is_a_topology(2)
discrete_partition IsAbaseFor_def by auto
\}
thus \( \text{Pow}(X) \subseteq \{\{x\}.x \in X\} \) by auto

953
theorem indiscrete_ptopology:
  assumes \( X \neq 0 \)
  shows \((\text{PTopology } X \{X\}) = \{0,X\}\)
proof
{ 
  fix \( T \)
  assume \( T \in (\text{PTopology } X \{X\}) \)
  with assms obtain \( M \) where \( M \subseteq \{X\} \cup M = T \) using Ptopology_is_a_topology(2)
    indiscrete_partition IsAbaseFor_def by auto
  then have \( T = 0 \lor T = X \) by auto
}
then show \((\text{PTopology } X \{X\}) \subseteq \{0,X\}\) by auto
from assms have \( 0 \in (\text{PTopology } X \{X\}) \) using Ptopology_is_a_topology(1)
empty_open
  indiscrete_partition by auto
moreover
from assms have \( \bigcup (\text{PTopology } X \{X\}) \in (\text{PTopology } X \{X\}) \) using union_open
Ptopology_is_a_topology(1)
  indiscrete_partition by auto
with assms have \( X \in (\text{PTopology } X \{X\}) \) using union_ptopology indiscrete_partition
  by auto
ultimately show \( \{0,X\} \subseteq (\text{PTopology } X \{X\}) \) by auto
qed

The topological subspaces of the \((\text{PTopology } X U)\) are partition topologies.

lemma subspace_ptopology:
  assumes \( U \{\text{is a partition of}\} X \)
  shows \((\text{PTopology } X U) \{\text{restricted to}\} Y = (\text{PTopology } (X \cap Y)) ((U \{\text{restricted to}\} Y)-\{0\}))\)
proof
  from assms have \( U \{\text{is a base for}\} (\text{PTopology } X U) \) using Ptopology_is_a_topology(2)
    by auto
  then have \((U\{\text{restricted to}\} Y) \{\text{is a base for}\} (\text{PTopology } X U) \{\text{restricted to}\} Y \)
    using subspace_base_topology by auto
  then have \((U\{\text{restricted to}\} Y)-\{0\} \{\text{is a base for}\} (\text{PTopology } X U) \{\text{restricted to}\} Y \)
    using base_no_0
    by auto
  moreover
from assms have \((U\{\text{restricted to}\} Y)-\{0\} \{\text{is a partition of}\} (X \cap Y)\)
    using restriction_partition by auto
  then have \((U\{\text{restricted to}\} Y)-\{0\} \{\text{is a base for}\} (\text{PTopology } (X \cap Y)) ((U \{\text{restricted to}\} Y)-\{0\}))\)
    using Ptopology_is_a_topology(2) by auto
ultimately show thesis using same_base_same_top by auto
qed
74.8 Order topologies

74.9 Order topology is a topology

Given a totally ordered set, several topologies can be defined using the order relation. First we define an open interval, notice that the set defined as Interval is a closed interval; and open rays.

**definition**

`IntervalX` where

```
IntervalX(X,r,b,c) ≡ (Interval(r,b,c) ∩ X) - {b,c}
```

**definition**

`LeftRayX` where

```
LeftRayX(X,r,b) ≡ {c ∈ X. ⟨c,b⟩ ∈ r} - {b}
```

**definition**

`RightRayX` where

```
RightRayX(X,r,b) ≡ {c ∈ X. ⟨b,c⟩ ∈ r} - {b}
```

Intersections of intervals and rays.

**lemma** `inter_two_intervals`:

**assumes** `bu ∈ X` `bv ∈ X` `cu ∈ X` `cv ∈ X`

**shows** `IntervalX(X,r,bu,cu) ∩ IntervalX(X,r,bv,cv) = IntervalX(X,r,GreaterOf(r,bu,bv),SmallerOf(r,cu,cv))`

**proof**

1. **fix x**

   **assume ASS:** `x ∈ IntervalX(X,r,bu,cu) ∩ IntervalX(X,r,bv,cv)`

   **then have x ∈ IntervalX(X,r,bu,cu) ∩ IntervalX(X,r,bv,cv) by auto**

2. **then have BB:** `x ∈ X` `x ∈ IntervalX(X,r,bu,cu) ∩ IntervalX(X,r,bv,cv)`

   **using IntervalX_def assms by auto**

3. **moreover**

   **have x ∈ X by auto**

4. **moreover**

   **have x ∈ GreaterOf(r,bu,bv) ∩ SmallerOf(r,cu,cv)**

   **proof**

   1. **show x ∈ GreaterOf(r,bu,bv) using GreaterOf_def BB(6,3) by (cases ⟨bu,bv⟩ ∈ r, simp, simp)**

   2. **show x ∈ SmallerOf(r,cu,cv) using SmallerOf_def BB(7,4) by (cases ⟨cu,cv⟩ ∈ r, simp, simp)**

5. **qed**

6. **moreover**

   **have ⟨bu,x⟩ ∈ r ∩ ⟨x,cv⟩ ∈ r using BB(2,5), Order_ZF_2_L1A by auto**

7. **then have x ∈ Interval(r,GreaterOf(r,bu,bv),SmallerOf(r,cu,cv)) using Order_ZF_2_L1 by auto**

8. **ultimately**

   **have x ∈ IntervalX(X,r,GreaterOf(r,bu,bv),SmallerOf(r,cu,cv)) using**
IntervalX_def T by auto

} then show IntervalX(X, r, bu, cu) ∩ IntervalX(X, r, GreaterOf(r, bu, bv), SmallerOf(r, cu, cv)) ⊆ IntervalX(X, r, GreaterOf(r, bu, bv), SmallerOf(r, cu, cv)) by auto

{ fix x
  assume x ∈ IntervalX(X, r, GreaterOf(r, bu, bv), SmallerOf(r, cu, cv))
  then have x ∈ X by auto
  moreover from BB(2) have CC: ⟨GreaterOf(r, bu, bv), x⟩ ∈ r ∧ ⟨x, SmallerOf(r, cu, cv)⟩ ∈ r using Order_ZF_2_L1A by auto
  
  \{ assume AS: ⟨bu, bv⟩ ∈ r
    then have GreaterOf(r, bu, bv) = bv using GreaterOf_def by auto
    then have ⟨bv, x⟩ ∈ r using CC(1) by auto
    with AS have ⟨bu, x⟩ ∈ r ⟨bv, x⟩ ∈ r using assms IsLinOrder_def trans_def by (safe, blast)
  \}

  moreover
  \{ assume AS: ⟨bu, bv⟩ /∈ r
    then have GreaterOf(r, bu, bv) = bu using GreaterOf_def by auto
    then have ⟨bu, x⟩ ∈ r using CC(1) by auto
    from AS have ⟨bv, bu⟩ ∈ r using assms IsLinOrder_def IsTotal_def
    assms by auto
    with ⟨⟨bu, x⟩ ∈ r⟩ have ⟨bu, x⟩ ∈ r ⟨bv, x⟩ ∈ r using assms IsLinOrder_def trans_def by (safe, blast)
  \}

  ultimately have R: ⟨bu, x⟩ ∈ r ⟨bv, x⟩ ∈ r by auto
  moreover
  \{ assume AS: x = bu
    then have ⟨bv, bu⟩ ∈ r using R(2) by auto
    then have GreaterOf(r, bu, bv) = bu using GreaterOf_def assms IsLinOrder_def antisym_def by auto
    then have False using AS BB(3) by auto
  \}
  moreover
  \{ assume AS: x = bv
    then have ⟨bu, bv⟩ ∈ r using R(1) by auto
    then have GreaterOf(r, bu, bv) = bv using GreaterOf_def by auto
    then have False using AS BB(3) by auto
  \}

  ultimately have ⟨bu, x⟩ ∈ r ⟨bv, x⟩ ∈ r x ≠ bu ≠ bv by auto
moreover 
{
\{ 
assume AS:\langle cu,cv \rangle \in r 
then have SmallerOf (r, cu, cv) = cu using SmallerOf_def by auto 
then have \langle x, cu \rangle \in r using CC(2) by auto 
with AS have \langle x, cu \rangle \in r \langle x, cv \rangle \in r using assms IsLinOrder_def trans_def 
by (safe , blast) 
\}
moreover 
{ 
assume AS:\langle cu,cv \rangle \notin r 
then have SmallerOf (r, cu, cv) = cv using SmallerOf_def by auto 
then have \langle x, cv \rangle \in r using CC(2) by auto 
from AS have \langle cv, cu \rangle \in r using assms IsLinOrder_def IsTotal_def 
by auto 
with \langle x, cv \rangle \in r \langle x, cu \rangle \in r using assms IsLinOrder_def trans_def by (safe , blast) 
\}
ultimately have R: \langle x, cv \rangle \in r \langle x, cu \rangle \in r by auto 
moreover 
{ 
assume AS:x=cv 
then have \langle cv, cu \rangle \in r using R(2) by auto 
then have SmallerOf (r, cu, cv) = cv using SmallerOf_def assms IsLinOrder_def antisym_def by auto 
then have False using AS BB(4) by auto 
\}
moreover 
{ 
assume AS:x=cu 
then have \langle cu, cv \rangle \in r using R(1) by auto 
then have SmallerOf (r, cu, cv) = cu using SmallerOf_def by auto 
then have False using AS BB(4) by auto 
\}
ultimately have \langle x, cu \rangle \in r \langle x, cv \rangle \in r \neq cu \neq cv by auto 
\}
ultimately 
have x\in IntervalX(X, r, bu, cu) \ x\in IntervalX(X, r, bv, cv) using Order_ZF_2_L1 
IntervalX_def 
assms by auto 
then have x\in IntervalX(X, r, bu, cu) \ Intersection intervalX(X, r, bv, cv) by auto 
\}
then show IntervalX (X, r, GreaterOf (r, bu, bv), SmallerOf (r, cu, cv)) \subseteq IntervalX(X, r, bu, cu) \ Intersection intervalX(X, r, bv, cv) 
by auto 
qed
lemma inter_ray_interval:
  assumes bv∈Xbu∈Xcv∈XIsLinOrder(X,r)
  shows RightRayX(X,r,bu)∩IntervalX(X,r,bv,cv)=IntervalX(X,r,GreaterOf(r,bu,bv),cv)
proof
  { fix x
    assume x∈RightRayX(X,r,bu)∩IntervalX(X,r,bv,cv)
    then have x∈RightRayX(X,r,bu) by auto
    then have BB:x∈X x̸=bu x̸=bv x̸=cv ⟨bu,x⟩∈rx∈Interval(r,bv,cv) using RightRayX_def IntervalX_def
      by auto
    then have ⟨bv,x⟩∈r ⟨x,cv⟩∈r using Order_ZF_2_L1A by auto
    with ⟨x,cv⟩∈r have x∈Interval(r,GreaterOf(r,bu,bv),cv) using BB(1-4) IntervalX_def GreaterOf_def
      by (simp)
  } then show RightRayX(X,r,bu)∩IntervalX(X,r,bv,cv)⊆IntervalX(X,r,GreaterOf(r,bu,bv),cv)
    by auto
  { fix x
    assume x∈IntervalX(X,r,GreaterOf(r,bu,bv),cv)
    then have x∈Interval(r,GreaterOf(r,bu,bv),cv) using BB(1-4) IntervalX_def GreaterOf_def
      by (simp)
    moreover
    { assume AS:⟨bu,bv⟩∈r
      then have GreaterOf(r,bu,bv)=bv using GreaterOf_def by auto
      then have ⟨bv,x⟩∈r using R(1) by auto
        with AS have ⟨bu,x⟩∈r ⟨bv,x⟩∈r using assms unfolding IsLinOrder_def trans_def by (safe,blast)
    } moreover
    { assume AS:⟨bu,bv⟩∉r
      then have GreaterOf(r,bu,bv)=bu using GreaterOf_def by auto
      then have ⟨bu,x⟩∈r using R(1) by auto
        from AS have ⟨bv,bu⟩∈r using assms unfolding IsLinOrder_def IsTotal_def using assms by auto
        with ⟨⟨bu,x⟩∈r have ⟨⟨bv,bu⟩∈r ⟨bv,x⟩∈r using assms unfolding IsLinOrder_def trans_def by (safe,blast)
  }

958
ultimately have $T: (bu,x) \in r$ $(bv,x) \in r$ by auto
moreover

{ 
  assume AS: $x = bu$
  then have $(bv, bu) \in r$ using $T(2)$ by auto
  then have $\text{GreaterOf}(r, bu, bv) = bu$ unfolding $\text{GreaterOf}_\text{def}$ using assms unfolding $\text{IsLinOrder}_\text{def}$
  antisym_def by auto
  with $\langle x \neq \text{GreaterOf}(r, bu, bv) \rangle$ have False using AS by auto
}
moreover

{ 
  assume AS: $x = bv$
  then have $(bu, bv) \in r$ using $T(1)$ by auto
  then have $\text{GreaterOf}(r, bu, bv) = bv$ unfolding $\text{GreaterOf}_\text{def}$ by auto
  with $\langle x \neq \text{GreaterOf}(r, bu, bv) \rangle$ have False using AS by auto
}
ultimately have $\langle bu, x \rangle \in r$ $\langle bv, x \rangle \in r$ $x \neq bu$ $x \neq bv$ by auto

with calculation $\langle x \in X \rangle$ have $x \in \text{RightRay}_X(X, r, bu)$ $x \in \text{Interval}_X(X, r, bv, cv)$ unfolding $\text{RightRay}_X_\text{def}$ $\text{Interval}_X_\text{def}$ by auto
from $\langle \langle x, cu \rangle \in r, \langle x, cv \rangle \in r \rangle$ $\langle bv, x \rangle \in r$ $\langle x, cv \rangle \in r, \langle bx \neq cv \rangle$ have False by auto
ultimately have $\langle bu, x \rangle \in r$ $\langle bv, x \rangle \in r$ $x \neq bu$ $x \neq bv$ by auto

with calculation $\langle x \in X \rangle$ have $x \in \text{RightRay}_X(X, r, bu)$ $x \in \text{Interval}_X(X, r, bv, cv)$ unfolding $\text{RightRay}_X_\text{def}$ $\text{Interval}_X_\text{def}$ by auto
then have $x \in \text{RightRay}_X(X, r, bu)$ $\cap$ $\text{Interval}_X(X, r, bv, cv)$ by auto
then show $\text{Interval}_X(X, r, \text{GreaterOf}(r, bu, bv), cv) \subseteq \text{RightRay}_X(X, r, bu)$ $\cap$ $\text{Interval}_X(X, r, bv, cv)$ by auto
qed

lemma inter_lray_interval: 
  assumes $bv \in X$ $cu \in X$ $cv \in X$ $\text{IsLinOrder}(X, r)$
  shows $\text{LeftRay}_X(X, r, cu) \cap \text{Interval}_X(X, r, bv, cv) = \text{Interval}_X(X, r, bv, \text{SmallerOf}(r, cu, cv))$
proof 
  { 
    fix $x$ assume $x \in \text{LeftRay}_X(X, r, cu)$ $\cap$ $\text{Interval}_X(X, r, bv, cv)$
    then have $B: x \neq cu \in X$ $\langle x, cu \rangle \in r$ $\langle x, cv \rangle \in r$ $\langle bx \neq cv \rangle$ unfolding $\text{LeftRay}_X_\text{def}$ $\text{Interval}_X_\text{def}$ $\text{Interval}_\text{def}$ by auto
    from $\langle (x, cu) \in r, \langle x, cv \rangle \in r \rangle$ have $C: (x, \text{SmallerOf}(r, cu, cv)) \in r$ using $\text{SmallerOf}_\text{def}$ by (cases $\langle cu, cv \rangle \in r, \text{simp}^*$)
    from $B(7,1)$ have $x \neq \text{SmallerOf}(r, cu, cv)$ using $\text{SmallerOf}_\text{def}$ by (cases $\langle cu, cv \rangle \in r, \text{simp}^*$)
    then have $x \in \text{Interval}_X(X, r, bv, \text{SmallerOf}(r, cu, cv))$ using $B$ $C$ $\text{Interval}_X_\text{def}$ $\text{Order}_ZF_2_L1$ by auto
    then show $\text{LeftRay}_X(X, r, cu) \cap \text{Interval}_X(X, r, bv, cv) \subseteq \text{Interval}_X(X, r, bv, \text{SmallerOf}(r, cu, cv))$ by auto
  } 

fix x assume x∈IntervalX(X,r,bv,SmallerOf(r,cu,cv))
then have R:x∈X⟨bv,x⟩∈r⟨x,SmallerOf(r,cu,cv)⟩∈rx̸=bv̸=SmallerOf(r,cu,cv)
using IntervalX_def Interval_def
by auto
then have ⟨bv,x⟩∈rx̸=bv by auto
moreover
{}
assume AS:⟨cu,cv⟩∈r
then have SmallerOf(r,cu,cv)=cu using SmallerOf_def by auto
then have ⟨x,cu⟩∈r using R(3) by auto
with AS have ⟨x,cu⟩∈r ⟨x,cv⟩∈r using assms unfolding IsLinOrder_def
trans_def by (safe, blast)
}
moreover
{}
assume AS:⟨cu,cv⟩∉r
then have SmallerOf(r,cu,cv)=cv using SmallerOf_def by auto
then have ⟨x,cv⟩∈r using R(3) by auto
from AS have ⟨cv,cu⟩∈r using assms IsLinOrder_def IsTotal_def
assms by auto
with ⟨⟨x,cv⟩∈r⟩ have ⟨x,cv⟩∈r ⟨x,cu⟩∈r using assms IsLinOrder_def
trans_def by (safe, blast)
}
ultimately have T:⟨x,cv⟩∈r ⟨x,cu⟩∈r by auto
moreover
{}
assume AS:x=cu
then have ⟨cu,cv⟩∈r using T(1) by auto
then have SmallerOf(r,cu,cv)=cu using SmallerOf_def assms IsLinOrder_def
antisym_def by auto
with ⟨x̸=SmallerOf(r,cu,cv)⟩ have False using AS by auto
}
moreover
{}
assume AS:x=cv
then have ⟨cv,cu⟩∈r using T(2) by auto
then have SmallerOf(r,cu,cv)=cv using SmallerOf_def assms IsLinOrder_def
antisym_def by auto
with ⟨x̸=SmallerOf(r,cu,cv)⟩ have False using AS by auto
}
ultimately have ⟨⟨x,cu⟩∈r⟩ ⟨x,cv⟩∈r̸=cu̸=cv by auto
}
with calculation ⟨x∈X⟩ have x∈LeftRayX(X,r,cu) x∈IntervalX(X, r, bv, cv) using LeftRayX_def IntervalX_def Interval_def
by auto
then have x∈LeftRayX(X, r, cu) ∩ IntervalX(X, r, bv, cv) by auto
}
then show IntervalX(X, r, bv, SmallerOf(r, cu, cv)) ⊆ LeftRayX(X, r,
Lemma `inter_lray_rray`:

- Assumes: $bu \in X$, $cv \in X$, $\text{IsLinOrder}(X, r)$
- Shows: $\text{LeftRay}_X(X, r, bu) \cap \text{RightRay}_X(X, r, cv) = \text{Interval}_X(X, r, cv, bu)$

Proof:

1. Fix $x$ assume $x \in \text{LeftRay}_X(X, r, bu) \cap \text{LeftRay}_X(X, r, cv)$
   - Then have $B: x \in X$, $\langle x, bu \rangle \in r$ $\langle x, cv \rangle \notin r$ using `LeftRay_def` by auto
   - Then have $C: x \in \text{SmallerOf}(r, bu, cv)$ using `SmallerOf_def` by (cases $\langle bu, cv \rangle \in r$, auto)

2. From $B$ have $D: x \notin \text{SmallerOf}(r, bu, cv)$ using `SmallerOf_def` by (cases $\langle bu, cv \rangle \in r$, auto)

3. From $B$, $C$, $D$ have $x \in \text{LeftRay}_X(X, r, \text{SmallerOf}(r, bu, cv))$ using `LeftRay_def` by auto
   - Then show $\text{LeftRay}_X(X, r, bu) \cap \text{LeftRay}_X(X, r, cv) \subseteq \text{LeftRay}_X(X, r, \text{SmallerOf}(r, bu, cv))$ by auto

4. Fix $x$ assume $x \in \text{LeftRay}_X(X, r, \text{SmallerOf}(r, bu, cv))$
   - Then have $R: x \in X$, $\langle x, \text{SmallerOf}(r, bu, cv) \rangle \in r$ $\langle x, cv \rangle \notin r$ using `SmallerOf_def` by auto
   - Assume $AS: \langle bu, cv \rangle \in r$
     - Then have $\text{SmallerOf}(r, bu, cv) = bu$ using `SmallerOf_def` by auto
     - Then have $\langle x, bu \rangle \in r$ using `R(2)` by auto
     - With $AS$ have $\langle x, bu \rangle \in r$ $\langle x, cv \rangle \notin r$ using `IsLinOrder_def` `trans_def` by (safe, blast)
   - Moreover
     1. Assume $AS: \langle bu, cv \rangle \notin r$
        - Then have $\text{SmallerOf}(r, bu, cv) = cv$ using `SmallerOf_def` by auto
        - Then have $\langle x, cv \rangle \in r$ using `R(2)` by auto
        - From $AS$ have $\langle cv, bu \rangle \in r$ using `IsLinOrder_def` `IsTotal_def`
        - With $\langle x, cv \rangle \in r$ have $\langle x, bu \rangle \in r$ using `IsLinOrder_def` `trans_def` by (safe, blast)
      - Ultimately have $T: \langle x, cv \rangle \in r$ $\langle x, bu \rangle \in r$ by auto
moreover
{ 
  assume AS: x = bu
  then have (bu, cv) \in r using T(1) by auto
  then have SmallerOf(r, bu, cv) = bu using SmallerOf_def assms IsLinOrder_def
  antisym_def by auto
  with \langle x \neq \text{SmallerOf}(r, bu, cv) \rangle have False using AS by auto
}
moreover
{ 
  assume AS: x = cv
  then have (cv, bu) \in r using T(2) by auto
  then have SmallerOf(r, bu, cv) = cv using SmallerOf_def assms IsLinOrder_def
  antisym_def by auto
  with \langle x \neq \text{SmallerOf}(r, bu, cv) \rangle have False using AS by auto
}
ultimately have \langle x, bu \rangle \in r \langle x, cv \rangle \in r x \neq bu \neq cv by auto
with \langle x \in X \rangle have x \in \text{LeftRayX}(X, r, bu) \cap \text{LeftRayX}(X, r, cv) using LeftRayX_def by auto
then show \text{LeftRayX}(X, r, \text{SmallerOf}(r, bu, cv)) \subseteq \text{LeftRayX}(X, r, bu) \cap \text{LeftRayX}(X, r, cv) by auto
qed

lemma \text{inter_rray_rray}:
assumes bu \in X \cap \text{cv} \in X \cap \text{IsLinOrder}(X, r)
shows \text{RightRayX}(X, r, bu) \cap \text{RightRayX}(X, r, cv) = \text{RightRayX}(X, r, \text{GreaterOf}(r, bu, cv))
proof
{ 
  fix x
  assume x \in \text{RightRayX}(X, r, bu) \cap \text{RightRayX}(X, r, cv)
  then have B: \langle x \in X, (bu, x) \in r, (cv, x) \in r, x \neq bu \neq cv \rangle using RightRayX_def by auto
  then have C: \langle x \in \text{GreaterOf}(r, bu, cv), (x, x) \in r \rangle using GreaterOf_def by (cases (bu, cv) \in r, auto)
  from B have D: \langle x \neq \text{GreaterOf}(r, bu, cv) \rangle using GreaterOf_def by (cases (bu, cv) \in r, auto)
  from B C D have x \in \text{RightRayX}(X, r, \text{GreaterOf}(r, bu, cv)) using RightRayX_def by auto
}
then show \text{RightRayX}(X, r, bu) \cap \text{RightRayX}(X, r, cv) \subseteq \text{RightRayX}(X, r, \text{GreaterOf}(r, bu, cv)) by auto
{ 
  fix x
  assume x \in \text{RightRayX}(X, r, \text{GreaterOf}(r, bu, cv))
  then have R: \langle x \in X, (GreaterOf(r, bu, cv), x) \in r, x \neq \text{GreaterOf}(r, bu, cv) \rangle using RightRayX_def by auto

962
assume AS:⟨bu,cv⟩∈r
then have \( \text{GreaterOf}(r, bu, cv) = cv \) using GreaterOf_def by auto
with AS have ⟨bu,x⟩∈r ⟨cv,x⟩∈r using assms IsLinOrder_def trans_def by(auto)
then have ⟨cv,x⟩∈r using R(2) by auto

moreover
{ assume AS:⟨bu,cv⟩∉r
then have \( \text{GreaterOf}(r, bu, cv) = bu \) using GreaterOf_def by auto
then have ⟨bu,x⟩∈r using R(2) by auto
from AS have ⟨cv,bu⟩∈r using assms IsLinOrder_def IsTotal_def
assms by auto
with ⟨⟨bu,x⟩∈r⟩ have ⟨cv,x⟩∈r ⟨bu,x⟩∈r using assms IsLinOrder_def trans_def by(safe, blast)
}
ultimately have T:⟨cv,x⟩∈r ⟨bu,x⟩∈r by auto
moreover
{ assume AS:x=bu
then have ⟨cv,bu⟩∈r using T(1) by auto
then have \( \text{GreaterOf}(r, bu, cv) = bu \) using GreaterOf_def assms IsLinOrder_def
antisym_def by auto
with ⟨x≠GreaterOf(r,bu,cv)⟩ have False using AS by auto
}
moreover
{ assume AS:x=cv
then have ⟨bu,cv⟩∈r using T(2) by auto
then have \( \text{GreaterOf}(r, bu, cv) = cv \) using GreaterOf_def assms IsLinOrder_def
antisym_def by auto
with ⟨x≠GreaterOf(r,bu,cv)⟩ have False using AS by auto
}
ultimately have ⟨bu,x⟩∈r ⟨cv,x⟩∉r xy≠bux≠cv by auto

with ⟨x∈X⟩ have x∈RightRayX(X,r, bu) ∩ RightRayX(X,r, cv) using RightRayX_def by auto
then show RightRayX(X,r, GreaterOf(r, bu, cv)) ⊆ RightRayX(X,r, bu) ∩ RightRayX(X,r, cv) by auto
qed

The open intervals and rays satisfy the base condition.

lemma intervals_rays_base_condition:
assumes IsLinOrder(X,r)
shows \( \{\text{IntervalX}(X,r, b,c). \ ⟨ b,c⟩∈X×X\}∪\{\text{LeftRayX}(X,r,b). b∈X\}∪\{\text{RightRayX}(X,r,b). b∈X\} \) {satisfies the base condition}
proof-
let I=\{\text{IntervalX}(X,r,b,c). \ ⟨ b,c⟩∈X×X\}

963
let R={RightRayX(X,r,b). b∈X}
let L={LeftRayX(X,r,b). b∈X}
let B={IntervalX(X,r,b,c). ⟨b,c⟩∈X×X}∪{LeftRayX(X,r,b). b∈X}∪{RightRayX(X,r,b). b∈X}

fix U V
assume A:U∈BV∈B
then have dU:U∈I∨U∈L∨U∈R by auto
{assume S:V∈I

assume U∈I
with S obtain bu cu bv cv where A:U=IntervalX(X,r,bu,cu)V=IntervalX(X,r,bv,cv)bu∈X
by auto
then have SmallerOf(r,cu,cv)∈XGreaterOf(r,bu,bv)∈X by (cases ⟨cu,cv⟩∈r,simp add:SmallerOf_def A,simp add:GreaterOf_def A)
with A have U∩V∈B using inter_two_intervals assms by auto
}
moreover
{assume U∈L
with S obtain bu bv cv where A:U=LeftRayX(X, r,bu)V=IntervalX(X,r,bv,cv)bu∈Xbv∈Xcv
by auto
then have SmallerOf(r,bu,cv)∈X using SmallerOf_def by (cases ⟨bu,cv⟩∈r,auto)
with A have U∩V∈B using inter_lray_interval assms by auto
}
moreover
{assume U∈R
with S obtain cu bv cv where A:U=RightRayX(X,r,cu)V=IntervalX(X,r,bv,cv)cu∈Xbv∈Xcv
by auto
then have GreaterOf(r,cu,bv)∈X using GreaterOf_def by (cases ⟨cu,bv⟩∈r,auto)
with A have U∩V∈B using inter_rray_interval assms by auto
}
ultimately have U∩V∈B using dU by auto
}
moreover
{assume S:V∈L

assume U∈I
with S obtain bu bv cv where A:U=LeftRayX(X, r,bu)V=IntervalX(X,r,bv,cv)bu∈Xbv∈Xcv
by auto
then have SmallerOf(r,bu,cv)∈X using SmallerOf_def by (cases ⟨bu,cv⟩∈r, auto)
have $U \cap V = V \cap U$ by auto
with $A \langle \text{SmallerOf}(r, bu, cv) \in X \rangle$ have $U \cap V \in B$ using $\text{inter_lray_interval}$ assms by auto

moreover
{
  assume $U \in \mathbb{R}$
  with $S$ obtain $bu$ $cv$ where $A: V = \text{LeftRay}(X, r, bu) U = \text{RightRay}(X, r, cv)$ $bu \in X$ $cv \in X$
  by auto
  have $U \cap V = V \cap U$ by auto
  with $A$ have $U \cap V \in B$ using $\text{inter_lray} \_	ext{ray}$ assms by auto
}

moreover
{
  assume $U \in \mathbb{L}$
  with $S$ obtain $bu$ $bv$ where $A: U = \text{LeftRay}(X, r, bu) V = \text{LeftRay}(X, r, bv)$ $bu \in X$ $bv \in X$
  by auto
  then have $\text{SmallerOf}(r, bu, bv) \in X$ using $\text{SmallerOf}_\text{def}$ by (cases $(bu, bv) \in r$, auto)
  with $A$ have $U \cap V \in B$ using $\text{inter_lray_lray}$ assms by auto
}

ultimately have $U \cap V \in B$ using $dU$ by auto

moreover
{
  assume $S: V \in \mathbb{R}$
  
  assume $U \in \mathbb{I}$
  with $S$ obtain $cu$ $bv$ $cv$ where $A: V = \text{RightRay}(X, r, cu) U = \text{Interval}(X, r, bv, cv)$ $cu \in X$ $bv \in X$ $cv \in X$
  by auto
  then have $\text{GreaterOf}(r, cu, bv) \in X$ using $\text{GreaterOf}_\text{def}$ by (cases $(cu, bv) \in r$, auto)
  have $U \cap V = V \cap U$ by auto
  with $A \langle \text{GreaterOf}(r, cu, bv) \in X \rangle$ have $U \cap V \in B$ using $\text{inter_rray}_\text{interval}$ assms by auto
}

moreover
{
  assume $U \in \mathbb{L}$
  with $S$ obtain $bu$ $cv$ where $A: U = \text{LeftRay}(X, r, bu) V = \text{RightRay}(X, r, cv)$ $bu \in X$ $cv \in X$
  by auto
  then have $U \cap V \in B$ using $\text{inter_lray} \_	ext{ray}$ assms by auto
}

moreover
{
  assume $U \in \mathbb{R}$
  with $S$ obtain $cu$ $cv$ where $A: U = \text{RightRay}(X, r, cu) V = \text{RightRay}(X, r, cv)$ $cu \in X$ $cv \in X$
  by auto
  then have $\text{GreaterOf}(r, cu, cv) \in X$ using $\text{GreaterOf}_\text{def}$ by (cases
\( (cu,cv) \in r, \text{auto} \)

with \( A \) have \( U \cap V \in B \) using \text{inter\_ray\_ray assms} by auto

\[
\text{ultimately have } U \cap V \in B \text{ using } dU \text{ by } \text{auto}
\]

ultimately have \( S : U \cap V \in B \) using \( dV \) by auto

\[
\text{fix } x \\
\text{assume } x \in U \cap V \\
\text{then have } x \in U \cap V \land U \cap V \subseteq U \cap V \text{ by auto} \\
\text{then have } \exists W \in (B) \land x \in W \land W \subseteq U \cap V \text{ using } S \text{ by blast} \\
\text{then have } \exists W \in (B). x \in W \land W \subseteq U \cap V \text{ by blast}
\]

hence \( (\forall x \in U \cap V. \exists W \in (B). x \in W \land W \subseteq U \cap V) \) by auto

then show \text{thesis using } \text{SatisfiesBaseCondition\_def} \text{ by auto}

qed

Since the intervals and rays form a base of a topology, and this topology is uniquely determined; we can built it. In the definition we have to make sure that we have a totally ordered set.

\text{definition}

OrderTopology (OrdTopology \_ \_ 50) where

\( \text{IsLinOrder}(X,r) \implies \text{OrdTopology } X \ r \equiv \text{TopologyBase} \{ \text{Interval}(X,r,b,c). \langle b,c \rangle \in X \times X \} \cup \{ \text{LeftRay}(X,r,b). b \in X \} \cup \{ \text{RightRay}(X,r,b). b \in X \} \)

\text{theorem Ordtopology\_is\_a\_topology:}

\( \text{assumes } \text{IsLinOrder}(X,r) \)

\( \text{shows } (\text{OrdTopology } X \ r) \{ \text{is a topology} \} \text{ and } \{ \text{Interval}(X,r,b,c). \langle b,c \rangle \in X \times X \} \cup \{ \text{LeftRay}(X,r,b). b \in X \} \cup \{ \text{RightRay}(X,r,b). b \in X \} \{ \text{is a base for} \} (\text{OrdTopology } X \ r) \)

\( \text{using assms } \text{Base\_topology\_is\_a\_topology} \text{ intervals\_rays\_base\_condition} \)

OrdTopology\_def by auto

\text{lemma topology0\_ordtopology:}

\( \text{assumes } \text{IsLinOrder}(X,r) \)

\( \text{shows } \text{topology0}(\text{OrdTopology } X \ r) \)

\( \text{using } \text{Ordtopology\_is\_a\_topology} \text{ topology0\_def assms by auto} \)

\text{7.4.10 Total set}

The topology is defined in the set \( X \), when \( X \) has more than one point

\text{lemma union\_ordtopology:}

\( \text{assumes } \text{IsLinOrder}(X,r) \exists x \ y. x \neq y \land x \in X \land y \in X \)

\( \text{shows } \bigcup (\text{OrdTopology } X \ r) = X \)

\text{proof}

\( \text{let } B = \{ \text{Interval}(X,r,b,c). \langle b,c \rangle \in X \times X \} \cup \{ \text{LeftRay}(X,r,b). b \in X \} \cup \{ \text{RightRay}(X,r,b). b \in X \} \)

\text{966}
have base: B \{ is a base for\} (OrdTopology X r) using Ordtopology_is_a_topology(2) by auto from assms(2) obtain x y where T: x \neq y \land x \in X \land y \in X by auto then have B: x \in \text{LeftRayX}(X, r, x) \lor x \in \text{RightRayX}(X, r, y) using \text{LeftRayX_def}, \text{RightRayX_def} assms(1) \text{IsLinOrder_def}, \text{IsTotal_def} by auto then have \text{x} \in \bigcup B using T by auto then have \text{x} \in \bigcup (OrdTopology X r) using Top_1_2_L5 base by auto \{ fix z assume z \in X \{ assume x = z then have z \in \bigcup (OrdTopology X r) using x by auto \} moreover \{ assume x \neq z \quad with \quad z T have z \in \text{LeftRayX}(X, r, x) \lor z \in \text{RightRayX}(X, r, y) x \in X using \text{LeftRayX_def}, \text{RightRayX_def} assms(1) \text{IsLinOrder_def}, \text{IsTotal_def} by auto then have z \in \bigcup B by auto then have z \in \bigcup (OrdTopology X r) using Top_1_2_L5 base by auto \} ultimately have z \in \bigcup (OrdTopology X r) by auto \} then show X \subseteq \bigcup (OrdTopology X r) by auto have \bigcup B \subseteq X using \text{IntervalX_def}, \text{LeftRayX_def}, \text{RightRayX_def} by auto then show \bigcup (OrdTopology X r) \subseteq X using Top_1_2_L5 base by auto qed

The interior, closure and boundary can be calculated using the formulas proved in the section that deals with the base.

The subspace of an order topology doesn’t have to be an order topology.

### 74.11 Right order and Left order topologies.

Notice that the left and right rays are closed under intersection, hence they form a base of a topology. They are called right order topology and left order topology respectively.

If the order in \( X \) has a minimal or a maximal element, is necessary to consider \( X \) as an element of the base or that limit point wouldn’t be in any basic open set.
74.11.1 Right and Left Order topologies are topologies

lemma leftrays_base_condition:
assumes IsLinOrder(X,r)
shows \{LeftRayX(X,r,b). b \in X\} \cup \{X\} \{satisfies the base condition\}
proof-
{
  fix U V
  assume U \in \{LeftRayX(X,r,b). b \in X\} \cup \{X\} V \in \{LeftRayX(X,r,b). b \in X\} \cup \{X\}
  then obtain b c where A:\((b \in X \land U = LeftRayX(X,r,b)) \lor U = X(c \in X \land V = LeftRayX(X,r,c)) \lor V = X\) unfolding LeftRayX_def by auto
  then have \((U \cap V = LeftRayX(X,r,SmallerOf(r,b,c)) \lor U = X(c \in X \land V = LeftRayX(X,r,c)) \lor V = X)\)
using inter_lray_lray assms by auto
moreover
have b \in X \land c \in X \rightarrow SmallerOf(r,b,c) \in X unfolding SmallerOf_def by (cases (b,c) \in r,auto)
ultimately have U \cap V \in \{LeftRayX(X,r,b). b \in X\} \cup \{X\} by auto
hence \forall x \in U \cap V. \exists W \in \{LeftRayX(X,r,b). b \in X\} \cup \{X\}. x \in W \land W \subseteq U \cap V by blast
}
moreover
then show thesis using SatisfiesBaseCondition_def by auto
qed

lemma rightrays_base_condition:
assumes IsLinOrder(X,r)
shows \{RightRayX(X,r,b). b \in X\} \cup \{X\} \{satisfies the base condition\}
proof-
{
  fix U V
  assume U \in \{RightRayX(X,r,b). b \in X\} \cup \{X\} V \in \{RightRayX(X,r,b). b \in X\} \cup \{X\}
  then obtain b c where A:\((b \in X \land U = RightRayX(X,r,b)) \lor U = X(c \in X \land V = RightRayX(X,r,c)) \lor V = X\) unfolding RightRayX_def by auto
  then have \((U \cap V = RightRayX(X,r,GreaterOf(r,b,c)) \lor U = X(c \in X \land V = RightRayX(X,r,c)) \lor V = X)\)
using inter_rray_rray assms by auto
moreover
have b \in X \land c \in X \rightarrow GreaterOf(r,b,c) \in X using GreaterOf_def by (cases (b,c) \in r,auto)
ultimately have U \cap V \in \{RightRayX(X,r,b). b \in X\} \cup \{X\} by auto
hence \forall x \in U \cap V. \exists W \in \{RightRayX(X,r,b). b \in X\} \cup \{X\}. x \in W \land W \subseteq U \cap V by blast
}
then show thesis using SatisfiesBaseCondition_def by auto
qed

definition LeftOrderTopology (LOrdTopology _ _) 50 where
  IsLinOrder(X,r) \implies LOrdTopology X r \equiv TopologyBase \{LeftRayX(X,r,b). b \in X\} \cup \{X\}
RightOrderTopology (ROrdTopology _ _ 50) where
IsLinOrder(X,r) \implies ROrdTopology X r \equiv TopologyBase \{RightRayX(X,r,b). b \in X\} \cup \{X\}

theorem LOrdtopology_ROrdtopology_are_topologies:
  assumes IsLinOrder(X,r)
  shows \(\text{LOrdTopology X r is a topology}\) and \(\{LeftRayX(X,r,b). b \in X\} \cup \{X\}\) is a base for \(\text{LOrdTopology X r}\)
  and \(\text{ROrdTopology X r is a topology}\) and \(\{RightRayX(X,r,b). b \in X\} \cup \{X\}\) is a base for \(\text{ROrdTopology X r}\)
  using Base_topology_is_a_topology leftrays_base_condition assms rightrays_base_condition
  LeftOrderTopology_def RightOrderTopology_def by auto

lemma topology0_lordtopology_rordtopology:
  assumes IsLinOrder(X,r)
  shows topology0(LOrdTopology X r) and topology0(ROrdTopology X r)
  using LOrdtopology_ROrdtopology_are_topologies topology0_def assms
  by auto

74.11.2 Total set

The topology is defined on the set \(X\)

lemma union_lordtopology_rordtopology:
  assumes IsLinOrder(X,r)
  shows \(\bigcup \text{LOrdTopology X r}=X\) and \(\bigcup \text{ROrdTopology X r}=X\)
  using Top_1_2_L5[OF LOrdtopology_ROrdtopology_are_topologies(2)[OF assms]]
  Top_1_2_L5[OF LOrdtopology_ROrdtopology_are_topologies(4)[OF assms]]
  unfolding LeftRayX_def RightRayX_def by auto

74.12 Union of Topologies

The union of two topologies is not a topology. A way to overcome this fact is to define the following topology:

definition joinT (joinT _ 90) where
(\forall T \in M. T \text{ is a topology} \land (\forall Q \in M. \bigcup Q = \bigcup T)) \implies (joinT M \equiv \text{THE} T. (\bigcup M) \text{ is a subbase for} T)

First let’s proof that given a family of sets, then it is a subbase for a topology.

The first result states that from any family of sets we get a base using finite intersections of them. The second one states that any family of sets is a subbase of some topology.

theorem subset_as_subbase:
  shows \(\cap A. A \in \text{FinPow}(B)\) \text{satisfies the base condition}
  proof
  { fix U V

969
assume $A, U \in \{ \bigcap A. A \in \text{FinPow}(B) \} \land V \in \{ \bigcap A. A \in \text{FinPow}(B) \}$
then obtain $M, R$ where $\text{MR} : \text{Finite}(M) \land \text{Finite}(R) \land M \subseteq B \land R \subseteq B$
$U = \bigcap M \land V = \bigcap R$
using $\text{FinPow_def}$ by auto

\[
\begin{align*}
\text{fix } x \\
\text{assume } \text{AS} : x \in U \land V \\
\text{then have } N : M \neq 0 \land R \neq 0 \text{ using } \text{MR}(5,6) \text{ by auto} \\
\text{have finite}(M \cup R) \text{ using } \text{MR}(1,2) \text{ by auto} \\
\text{moreover} \\
\text{from } N \text{ have } \bigcap (M \cup R) \subseteq M \land \bigcap (M \cup R) \subseteq R \text{ by auto} \\
\text{then have } \bigcap (M \cup R) \subseteq U \land V \text{ using } \text{MR}(5,6) \text{ by auto} \\
\text{moreover} \\
\text{fix } S \\
\text{assume } S \in M \cup R \\
\text{then have } x \in S \text{ using } \text{AS} \text{ MR}(5,6) \text{ by auto} \\
\text{then have } x \in \bigcap (M \cup R) \text{ using } N \text{ by auto} \\
\text{ultimately have } \exists W \in \{ \bigcap A. A \in \text{FinPow}(B) \}. x \in W \land W \subseteq U \land V \text{ by blast} \\
\text{then have } (\forall x \in U \land V. \exists W \in \{ \bigcap A. A \in \text{FinPow}(B) \}. x \in W \land W \subseteq U \land V) \text{ by auto} \\
\text{then have } \forall U, V. ((U \in \{ \bigcap A. A \in \text{FinPow}(B) \} \land V \in \{ \bigcap A. A \in \text{FinPow}(B) \}) \\
\rightarrow (\forall x \in U \land V. \exists W \in \{ \bigcap A. A \in \text{FinPow}(B) \}. x \in W \land W \subseteq U \land V)) \text{ by auto} \\
\text{then show } \{ \bigcap A. A \in \text{FinPow}(B) \} \{ \text{satisfies the base condition} \} \\
\text{using } \text{SatisfiesBaseCondition_def} \text{ by auto} \\
\text{qed} \\
\end{align*}
\]

theorem $\text{Top_subbase}$:
assumes $T = \{ \bigcup A. A \in \text{Pow}(\{ \bigcap A. A \in \text{FinPow}(B) \}) \}$
shows $T \{ \text{is a topology} \}$ and $B \{ \text{is a subbase for} \} T$
proof-
\[
\begin{align*}
\text{fix } S \\
\text{assume } S \in B \\
\text{then have } \{ S \} \in \text{FinPow}(B) \land \{ S \} = S \text{ using } \text{FinPow_def} \text{ by auto} \\
\text{then have } \{ S \} \in \text{Pow}(\{ \bigcap A. A \in \text{FinPow}(B) \}) \text{ by } \text{blast} \\
\text{then have } \bigcup \{ S \} \in \{ \bigcup A. A \in \text{FinPow}(B) \} \text{ by blast} \\
\text{then have } S \in \{ \bigcup A. A \in \text{FinPow}(B) \} \text{ by auto} \\
\text{then have } S \in T \text{ using } \text{assms} \text{ by auto} \\
\end{align*}
\]

970
then have $B \subseteq T$ by auto
moreover
have $\bigcap \{ A. A \in \text{FinPow}(B)\}$ {satisfies the base condition} using subset_as_subbase by auto
then have $T$ {is a topology} and $\{ \bigcap A. A \in \text{FinPow}(B)\}$ {is a base for} $T$
  using Top_1_2_T1 assms by auto
ultimately show $T$ {is a topology} and $B$ {is a subbase for} $T$
  using IsAsubBaseFor_def by auto
qed

A subbase defines a unique topology.

**Theorem same_subbase_same_top:**

assumes $B$ {is a subbase for} $T$ and $B$ {is a subbase for} $S$
shows $T = S$
  using IsAsubBaseFor_def assms same_base_same_top by auto

end

### 75 Properties in Topology

**Theory Topology_ZF_properties imports Topology_ZF_examples Topology_ZF_examples_1**

begin

This theory deals with topological properties which make use of cardinals.

#### 75.1 Properties of compactness

It is already defined what is a compact topological space, but the is a generalization which may be useful sometimes.

**Definition IsCompactOfCard (\_{is compact of cardinal}\_ {in}_ 90):**

where $K$ {is compact of cardinal} $Q$ {in} $T$ $\equiv$ $(\text{Card}(Q) \land K \subseteq \bigcup T \land (\forall M \in \text{Pow}(T). K \subseteq M \longrightarrow (\exists N \in \text{Pow}(M). K \subseteq N \land N \neq Q)))$

The usual compact property is the one defined over the cardinal of the natural numbers.

**Lemma Compact_is_card_nat:**

shows $K$ {is compact in} $T$ $\iff$ $(K$ {is compact of cardinal} $\cap$ {in} $T)$

proof

{ assume $K$ {is compact in} $T$
  then have sub:$K \subseteq \bigcup T$ and reg:$(\forall M \in \text{Pow}(T). K \subseteq M \longrightarrow (\exists N \in \text{FinPow}(M). K \subseteq N))$
    using IsCompact_def by auto
}

971
{ 
  fix $M$
  assume $M \in \text{Pow}(T) \subseteq \bigcup M$
  with reg obtain $N$ where $N \in \text{FinPow}(M) \subseteq N$ by blast
  then have $\text{Finite}(N)$ using $\text{FinPow}\_\text{def}$ by auto
  then obtain $n$ where $A:n \in \text{nat} \Rightarrow n$ using $\text{Finite}\_\text{def}$ by auto
  from $A(1)$ have $n \prec \text{nat}$ using $\text{n}\_\text{lesspoll}\_\text{nat}$ by auto
  with $A(2)$ have $N \subseteq \text{nat}$ using $\text{lesspoll}\_\text{def}$ $\text{eq}\_\text{lepoll}\_\text{trans}$ by auto
  moreover
  { assume $N \approx \text{nat}$
    then have $\text{nat} \approx N$ using $\text{eqpoll}\_\text{sym}$ by auto
    with $A(2)$ have $n \approx \text{nat}$ using $\text{eqpoll}\_\text{trans}$ by auto
    moreover
    { assume $N \approx \text{nat}$ then have $\text{nat} \approx N$ using $\text{eqpoll}\_\text{sym}$ by auto
      with $A(2)$ have $n \approx \text{nat}$ using $\text{eqpoll}\_\text{sym}$ by auto
      then have $n \approx \text{nat}$ using $\text{eqpoll}\_\text{sym}$ by auto
      then have $\text{False}$ using $\text{lesspoll}\_\text{def}$ by auto
    }
    then have $\neg (N \approx \text{nat})$ by auto
  with calculation $\langle K \subseteq \bigcup N \rangle \forall N \in \text{FinPow}(M)$ have $N \prec \text{nat} \subseteq \bigcup N N \in \text{Pow}(M)$ using $\text{lesspoll}\_\text{def}$ $\text{FinPow}\_\text{def}$ by auto
  hence $\exists N \in \text{Pow}(M). K \subseteq \bigcup N \wedge N \prec \text{nat}$ by auto
  with sub show $K\{\text{is compact of cardinal}\} \text{nat}\{\text{in}\}T$ using $\text{IsCompactOfCard}\_\text{def}$ $\text{Card}\_\text{nat}$ by auto
  }

  { assume $(K\{\text{is compact of cardinal}\} \text{nat}\{\text{in}\}T)$
    then have $\forall M \in \text{Pow}(T). K \subseteq \bigcup M \rightarrow (\exists N \in \text{Pow}(M).$ $K \subseteq \bigcup N \wedge N \prec \text{nat})$ using $\text{IsCompactOfCard}\_\text{def}$ by auto
    { fix $M$
      assume $M \in \text{Pow}(T) \subseteq \bigcup M$
      with reg have $\exists N \in \text{Pow}(M). K \subseteq \bigcup N \wedge N \prec \text{nat}$ by auto
      then obtain $N$ where $N \in \text{FinPow}(M) \subseteq N$ by blast
      then have $N \subseteq \text{nat}$ using $\text{lesspoll}\_\text{nat}\_\text{is}\_\text{Finite}$ $\text{FinPow}\_\text{def}$ by auto
      hence $\exists N \in \text{FinPow}(M). K \subseteq \bigcup N$ by auto
    }
    with sub show $K\{\text{is compact in}\}T$ using $\text{IsCompact}\_\text{def}$ by auto
  }
qed

Another property of this kind widely used is the Lindelöf property; it is
the one on the successor of the natural numbers.

definition
  $\text{IsLindelof} \ _\{\text{is lindelof in}\}_ {\_90}$ where
  $K \{\text{is lindelof in}\} T \equiv K\{\text{is compact of cardinal}\} csucc(\text{nat})\{\text{in}\}T$

972
It would be natural to think that every countable set with any topology is Lindeloef; but this statement is not provable in ZF. The reason is that to build a subcover, most of the time we need to choose sets from an infinite collection which cannot be done in ZF. Additional axioms are needed, but strictly weaker than the axiom of choice.

However, if the topology has not many open sets, then the topological space is indeed compact.

**Theorem card_top_comp:**

**Assumptions:**

- \(\text{Card}(Q) \subseteq T\)
- \(K \subseteq T\)

**Shows:**

- \((K)\) is compact of cardinal \((Q)\) in \(T\)

**Proof:**

- Fix \(M\) assume \(M : M \subseteq T \subseteq \bigcup M\)
- From \((1)\) \(\text{assms}(2)\) have \(M < Q\) using \(\text{subset_imp_lepoll lesspoll_trans1}\) by blast
- With \((2)\) have \(\exists N : \text{Pow}(M). K \subseteq N \land N < Q\) by auto

**With assms(1,3) show thesis unfolding IsCompactOfCard_def by auto**

**Qed**

The union of two compact sets is compact; of any cardinality.

**Theorem union_compact:**

**Assumptions:**

- \(K\) is compact of cardinal \((Q)\) in \(T\)
- \(K1\) is compact of cardinal \((Q)\) in \(T\)

**Shows:**

- \((K \cup K1)\) is compact of cardinal \((Q)\) unfolding IsCompactOfCard_def

**Proof:**

- Fix \(x\) assume \(x \in K\) then show \(x \in \bigcup T\) using \(\text{assms}(1)\) unfolding IsCompactOfCard_def by blast
- Next
- Fix \(x\) assume \(x \in K1\) then show \(x \in \bigcup T\) using \(\text{assms}(2)\) unfolding IsCompactOfCard_def by blast
- Next
- Fix \(M\) assume \(M \subseteq T \subseteq \bigcup M\)
- Then have \(K \subseteq \bigcup NK1 \subseteq \bigcup M\) by auto
- With \((M \subseteq T)\) have \(\exists N : \text{Pow}(M). K \subseteq N \land N < Q\) \(\exists N : \text{Pow}(M). K1 \subseteq N \land N < Q\) by auto
- Then obtain \(NK, NK1\) where \(NK \subseteq \text{Pow}(M)NK1 \subseteq \text{Pow}(M)K \subseteq \bigcup NK1 \subseteq \bigcup NK1 NK \subseteq QNK1 < Q\) by auto
- Then have \(NK \cup NK1 \subseteq \bigcup (NK \cup NK1)NK \cup NK1 \subseteq \text{Pow}(M)\) using \(\text{assms}(3)\) less_less_imp_un_less by auto
- Then show \(\exists N : \text{Pow}(M). K \cup K1 \subseteq N \land N < Q\) by auto

**Qed**

If a set is compact of cardinality \(Q\) for some topology, it is compact of cardinality \(Q\) for every coarser topology.

973
theorem compact_coarser:
  assumes \( T \subseteq T \) and \( \bigcup T = \bigcup T \) and \( (K) \{is compact of cardinal\}Q\{in\}T \)
  shows \( (K) \{is compact of cardinal\}Q\{in\}T \)
proof-
  { 
    fix \( M \)
    assume \( \text{AS:} \; M \in \text{Pow}(T)K \subseteq \bigcup M \)
    then have \( \exists N \in \text{Pow}(M).K \subseteq N \land N \not\subseteq Q \) using assms(1) unfolding IsCompactOfCard_def
    by auto
    then show \( (K) \{is compact of cardinal\}Q\{in\}T \) using assms(3) unfolding IsCompactOfCard_def by auto
  }
qed

If some set is compact for some cardinal, it is compact for any greater cardinal.

theorem compact_greater_card:
  assumes \( Q \preq Q1 \) and \( (K) \{is compact of cardinal\}Q\{in\}T \) and \( \text{Card}(Q1) \)
  shows \( (K) \{is compact of cardinal\}Q1\{in\}T \)
proof-
  { 
    fix \( M \)
    assume \( \text{AS:} \; M \in \text{Pow}(T)K \subseteq \bigcup M \)
    have \( \bigcup T - R \in T \) using assms(2) IsClosed_def by auto
    have \( K - R \subseteq (\bigcup T - R) \) using assms(1) IsCompactOfCard_def by auto
    with \( \langle \bigcup T - R \rangle \) have \( K \subseteq \bigcup \langle M \cup (\bigcup T - R) \rangle \) and \( M \cup (\bigcup T - R) \in \text{Pow}(T) \)
    proof (safe)
      { 
        fix \( x \)
      }
  }
qed

A closed subspace of a compact space of any cardinality, is also compact of the same cardinality.

theorem compact_closed:
  assumes \( K \{is compact of cardinal\} \; Q \{in\} T \)
  and \( R \{is closed in\} T \)
  shows \( (K \cap R) \{is compact of cardinal\} \; Q \{in\} T \)
proof-
  { 
    fix \( M \)
    assume \( \text{AS:} \; M \in \text{Pow}(T)K \cap R \subseteq \bigcup M \)
    have \( \bigcup T - R \in T \) using assms(2) IsClosed_def by auto
    have \( K - R \subseteq (\bigcup T - R) \) using assms(1) IsCompactOfCard_def by auto
    with \( \langle \bigcup T - R \rangle \) have \( K \subseteq \bigcup \langle M \cup (\bigcup T - R) \rangle \) and \( M \cup (\bigcup T - R) \in \text{Pow}(T) \)
    proof (safe)
      { 
        fix \( x \)
      }
  }
qed

974
assume $x \in M$
with AS(1) show $x \in T$ by auto 
}
{
  fix $x$
  assume $x \in K$
  have $x \in R \lor x \notin R$ by auto
  with AS(2) $\langle K-R \subseteq (\bigcup T-R) \rangle$ have $x \in \bigcup M \lor x \notin (\bigcup T-R)$ by auto
  then show $x \in \bigcup (M \cup \{\bigcup T-R\})$ by auto
}
qed
with assms(1)
have $\exists N \in \mathit{Pow}(M \cup \{\bigcup T-R\}). K \subseteq \bigcup N \land N \prec Q$ unfolding IsCompactOfCard_def by auto
then obtain $N$ where cub: $N \in \mathit{Pow}(M \cup \{\bigcup T-R\})$ $K \subseteq \bigcup N$ $N \prec Q$ by auto
have $N-\{\bigcup T-R\} \in \mathit{Pow}(M)$ $K \cap R \subseteq \bigcup (N-\{\bigcup T-R\})$ $N-\{\bigcup T-R\} \prec Q$
proof (safe)
{
  fix $x$
  assume $x \notin N \land x \notin M$
  then show $x = \bigcup T-R$ using cub(1) by auto
}
{
  fix $x$
  assume $x \in K \land x \in R$
  then have $x \notin \bigcup T-R \land x \in K$ by auto
  then show $x \in \bigcup (N-\{\bigcup T-R\})$ using cub(2) by blast
}
have $N-\{\bigcup T-R\} \subseteq N$ by auto
with cub(3) show $N-\{\bigcup T-R\} \prec Q$ using subset_imp_lepoll lesspoll_trans1 by blast
qed
then have $\exists N \in \mathit{Pow}(M). K \cap R \subseteq \bigcup N \land N \prec Q$ by auto
}\then have $\forall M \in \mathit{Pow}(T). (K \cap R \subseteq \bigcup M \rightarrow (\exists N \in \mathit{Pow}(M). K \cap R \subseteq \bigcup N \land N \prec Q))$ by auto
then show thesis using IsCompactOfCard_def assms(1) by auto
qed

75.2 Properties of numerability

The properties of numerability deal with cardinals of some sets built from the topology. The properties which are normally used are the ones related to the cardinal of the natural numbers or its successor.

definition
IsFirstOfCard (_ {is of first type of cardinal} 90) where
  $(T \{is of first type of cardinal\} Q) \equiv \forall x \in \bigcup T. (\exists B. (B \{is a base for\} T) \land (\{b \in B. x \in b\} \prec Q))$
definition
IsSecondOfCard (_ {is of second type of cardinal} _ 90) where
(T {is of second type of cardinal} Q) \equiv (\exists B. (B {is a base for} T) \land (B \prec Q))

definition
IsSeparableOfCard (_ {is separable of cardinal} _ 90) where
T {is separable of cardinal} Q \equiv \exists U \in \text{Pow}(\bigcup T). \text{Closure}(U, T) = \bigcup T \land U \prec Q

definition
IsFirstCountable (_ {is first countable} 90) where
(T {is first countable}) \equiv T {is of first type of cardinal} csucc(nat)

definition
IsSecondCountable (_ {is second countable} 90) where
(T {is second countable}) \equiv (T {is of second type of cardinal} csucc(nat))

definition
IsSeparable (_ {is separable} 90) where
T {is separable} \equiv T {is separable of cardinal} csucc(nat)

If a set is of second type of cardinal Q, then it is of first type of that same cardinal.

theorem second_imp_first:
assumes T {is of second type of cardinal} Q
shows T {is of first type of cardinal} Q
proof
from assms have \exists B. (B {is a base for} T) \land (B \prec Q) using IsSecondOfCard_def
by auto
then obtain B where base: (B {is a base for} T) \land (B \prec Q) by auto
{ fix x
  assume x \in \bigcup T
  have {b \in B. x \in b} \subseteq B by auto
  then have {b \in B. x \in b} \prec B using subset_imp_lepoll by auto
  with base have \{b \in B. x \in b\} \prec Q using lesspoll_trans1 by auto
  with base have (B {is a base for} T) \land \{b \in B. x \in b\} \prec Q by auto
}
then have \forall x \in \bigcup T. \exists B. (B {is a base for} T) \land \{b \in B. x \in b\} \prec Q by auto
then show thesis using IsFirstOfCard_def by auto
qed

A set is dense iff it intersects all non-empty, open sets of the topology.

lemma dense_int_open:
assumes T {is a topology} and A \subseteq \bigcup T
shows \text{Closure}(A, T) = \bigcup T \iff (\forall U \in T. U \neq 0 \rightarrow A \cap U \neq 0)
proof
assume AS: \text{Closure}(A, T) = \bigcup T
{
75.3 Relations between numerability properties and choice principles

It is known that some statements in topology aren't just derived from choice axioms, but also equivalent to them. Here is an example

The following are equivalent:

- Every topological space of second cardinality \( \text{csucc}(Q) \) is separable of cardinality \( \text{csucc}(Q) \).

- The axiom of \( Q \) choice.

In the article [4] there is a proof of this statement for \( Q = \mathbb{N} \), with more equivalences.

If a topology is of second type of cardinal \( \text{csucc}(Q) \), then it is separable of the same cardinal. This result makes use of the axiom of choice for the cardinal \( Q \) on subsets of \( \bigcup T \).

**theorem** \( Q\_\text{choice} \_\text{imp} \_\text{second} \_\text{imp} \_\text{separable} \):
- assumes \( T \{ \text{is of second type of cardinal} \} \text{csucc}(Q) \)
- and \{the axiom of\} \( Q \ \{ \text{choice holds for subsets} \} \bigcup T \)
- and \( T \{ \text{is a topology} \} \)
- shows \( T \{ \text{is separable of cardinal} \} \text{csucc}(Q) \)

**proof**
- from \( \text{assms}(1) \) have \( \exists B \ (B \ \{ \text{is a base for} \} T \) \land (B \ \prec \text{csucc}(Q)) \) using \( \text{IsSecondOfCard} \_\text{def} \) by auto
then obtain $B$ where base:$(B \text{ is a base for } T) \land (B < \text{csucc}(Q))$ by auto

let $N=\lambda b \in B. b$

let $B=B-\{0\}$

have $B-\{0\} \subseteq B$ by auto

with base have prec:$B-\{0\} < \text{csucc}(Q)$ using subset_imp_lepoll lesspoll_trans1 by blast

from base have baseOpen:$\forall b \in B. b \in T$ using base_sets_open by auto

from assms(2) have car:Card(Q) and reg:$\forall M N. (M \lesscsucc Q \land (\forall t \in M. Nt \neq 0 \land \forall t \in M. ft \in Nt))$ using AxiomCardinalChoice_def by auto

then have $(B \lesscsucc Q \land (\forall t \in B. Nt \neq 0 \land \forall t \in B. ft \in Nt))$ by blast

with prec have $(\forall t \in B. Nt \subseteq T) \longrightarrow (\exists f. f:Pi(B,\lambda t. Nt))$ using Card_less_csucc_eq_le car by blast

then have $(\forall t \in B. Nt \subseteq T) \longrightarrow (\exists f. f:Pi(B,\lambda t. Nt))$ by blast

with baseOpen have $\exists f. f:Pi(B,\lambda t. Nt) \land (\forall t \in B. ft \in Nt)$ by auto

{ fix $U$
  assume $U \in T$ and $U \neq 0$
  then obtain $b$ where $A1:b \in B-\{0\}$ and $b \subseteq U$ using Top_1_2_L1 base by blast
    with $f2$ have $fb \in U$ by auto
    with $A1$ have $\{fb. b \in B\} \cap U \neq 0$ by auto
  }

then have $r: \forall U \in T. U \neq 0 \longrightarrow \{fb. b \in B\} \cap U \neq 0$ by auto

have $\{fb. b \in B\} \subseteq T$ using $f2$ baseOpen by auto

moreover

with $r$ have Closure($\{fb. b \in B\}, T) = \bigcup T$ using dense_int_open assms(3) by auto

moreover

have $ffun:f:B \rightarrow \text{range}(f)$ using $f$ range_of_fun by auto

then have $f: \text{surj}(B, \text{range}(f))$ using $f2$ baseOpen by auto

moreover

have $\exists f. f:Pi(B,\lambda t. Nt) \land (\forall t \in B. ft \in Nt)$ by auto

with $A1$ have $\{fb. b \in B\} \subseteq \bigcup T$ using $f2$ baseOpen by auto

ultimately show thesis using IsSeparableOfCard_def by auto

qed

The next theorem resolves that the axiom of Q choice for subsets of $\bigcup T$
is necessary for second type spaces to be separable of the same cardinal $\text{csucc}(Q)$.

theorem second_imp_separable_imp_Q_choice:

assumes $\forall T. (T \text{ is a topology}) \land (T \text{ is of second type of cardinal} \text{csucc}(Q))) \longrightarrow (T \text{ is separable of cardinal} \text{csucc}(Q))$


978
and Card(Q)
shows {the axiom of} Q {choice holds}
proof-
{
  fix N M
  assume AS:M \subseteq Q \land (\forall t \in M. Nt \neq 0)

  then obtain h where inj:h \inj(M,Q) using lepoll_def by auto
  then have bij:converse(h):bij(range(h),M) using inj_bij_range bij_converse_bij
  by auto
  let T={(N(converse(h)i)) \times \{i\}. i \in range(h)}
  {
    fix j
    assume AS2:j \in range(h)
    from bij have converse(h):range(h) \to M using bij_def inj_def by auto
    with AS2 have converse(h)j \in M by simp
    with AS have N(converse(h)j) \neq 0 by auto
    then have (N(converse(h)j)) \times \{j\} \neq 0 by auto
  }
  then have noEmpty:0 \notin T by auto
  moreover
  {
    fix A B
    assume AS2:A \in TB \cap T \neq 0
    then obtain j t where A_def:A=N(converse(h)j) \times \{j\} and B_def:B=N(converse(h)t) \times \{t\}
      and Range:j \in range(h) t \in range(h) by auto
    from AS2(3) obtain x where x \in A \cap B by auto
    with A_def B_def have j=t by auto
    with A_def B_def have A=B by auto
  }
  then have (\forall A \in T. \forall B \in T. A=B \lor A \cap B=0) by auto
  ultimately
  have Part:T {is a partition of} \bigcup T unfolding IsAPartition_def by auto
  let \tau=PTopology \bigcup T T
  from Part have top:\tau {is a topology} and base:T {is a base for} \tau
    using Ptopology_is_a_topology by auto
  let f={((i,(N(converse(h)i))) \times \{i\}. i \in range(h))
  have f:range(h) \to T using functionI[of f] Pi_def by auto
  then have f \in surj(range(h),T) unfolding surj_def using apply_equality
  by auto
  moreover
  have range(h) \subseteq Q using inj unfolding inj_def range_def domain_def
  Pi_def by auto
  ultimately have T \subseteq Q using surj_fun_inv[of range(h)TQ] assms(2)
  Card_is_Ord lepoll_trans
  subset_imp_lepoll by auto
  then have T \prec csucc(Q) using Card_less_csucc_eq_le assms(2) by auto

  979
with base have \((\tau\{\text{is of second type of cardinal}\})\text{csucc}(Q)\) using \text{IsSecondOfCard_def} by auto
with top have \((\tau\{\text{is separable of cardinal}\})\text{csucc}(Q)\) using \text{assms}(1) by auto

then obtain \(D\) where \(\text{sub}:D\in\text{Pow}(\bigcup\tau)\) and \(\text{clos}:\text{Closure}(D,\tau)=\bigcup\tau\) and \(\text{cardd}:D<-\text{csucc}(Q)\) using \text{IsSeperableOfCard_def} by auto

then have \(D\subsetneq Q\) using \text{Card_less_csucc_eq_le} \text{assms}(2) by auto
then obtain \(r\) where \(r:r\in\text{inj}(D,Q)\) using \text{lepoll_def} by auto
then have \(D\triangleq Q\) using \text{Card_less_csucc_eq_le} \text{assms}(2) by auto
then obtain \(r\) where \(r:r\in\text{inj}(D,Q)\) using \text{lepoll_def} by auto
then have \(D\triangleq Q\) using \text{Card_less_csucc_eq_le} \text{assms}(2) by auto

then have \(D\triangleq Q\) using \text{Card_less_csucc_eq_le} \text{assms}(2) by auto
then have \(r\) where \(r:r\in\text{inj}(D,Q)\) using \text{lepoll_def} by auto
then have \(D\triangleq Q\) using \text{Card_less_csucc_eq_le} \text{assms}(2) by auto

then have \(D\triangleq Q\) using \text{Card_less_csucc_eq_le} \text{assms}(2) by auto
then have \(r\) where \(r:r\in\text{inj}(D,Q)\) using \text{lepoll_def} by auto
then have \(D\triangleq Q\) using \text{Card_less_csucc_eq_le} \text{assms}(2) by auto

let \(R=\lambda i\in\text{range}(h).\{j\in\text{range}(r).\text{converse}(r)j\in((\text{N}(\text{converse}(h)i))\times\{i\})\}

{ fix \(i\) assume \(\text{AS}:i\in\text{range}(h)\)
then have \(T:((\text{N}(\text{converse}(h)i))\times\{i\})\in T\) by auto
then have \(P:((\text{N}(\text{converse}(h)i))\times\{i\})\in\tau\) using aux unfolding \text{IsAbaseFor_def} by blast
with \(P\) have \((\text{N}(\text{converse}(h)i))\times\{i\}\neq 0\) by auto
with \(T\) noEmpty have \(D\cap(\text{N}(\text{converse}(h)i))\times\{i\}\neq 0\) by auto
then obtain \(x\) where \(x\in D\) and \(px:x\in((\text{N}(\text{converse}(h)i))\times\{i\})\) by auto
with \(surj2\) obtain \(j\) where \(j\in\text{range}(r)\) and \(\text{converse}(r)j=x\) unfolding \text{surj_def} by blast
with \(px\) have \(j\in\{j\in\text{range}(r).\text{converse}(r)j\in((\text{N}(\text{converse}(h)i))\times\{i\})\}\)
by auto
then have \(R\neq 0\) using \text{beta_if[of range(h) _ i]} \text{AS} by auto
}
then have \(\forall i\in\text{range}(h).\ R\neq 0\) by auto
{ fix \(i\) \(j\)
assume \(i:i\in\text{range}(h)\) and \(j:j\in R\)
from \(j\) \(i\) have \(\text{converse}(r)j\in((\text{N}(\text{converse}(h)i))\times\{i\})\) using \text{beta_if}
by auto
}
then have \(\forall i\in\text{range}(h).\ \forall j\in R.\ \text{converse}(r)j\in((\text{N}(\text{converse}(h)i))\times\{i\})\)
by auto
let \(E=\{\text{(m,fst(\text{converse}(r)(\mu j.\ j\in R(hm))))}.\ m\in M\}\)
have \(\text{ff:function}(E)\) unfolding \text{function_def} by auto
moreover
{ fix \(m\)
assume \(M:m\in M\)
with \(\text{inj}\) have \(\text{hm}:hm\in\text{range}(h)\) using \text{apply_rangeI} \text{inj_def} by auto
}

980
{
  fix j
  assume j ∈ R(hm) with hm have j ∈ range(r) using beta_if by auto
  from r have r: surj(D, range(r)) using fun_is_surj inj_def by auto
  with ⟨j ∈ range(r)⟩ obtain d where d ∈ D and rd = j using surj_def
  by auto
  then have j ∈ Q using r inj_def by auto
}
then have subcar: R(hm) ⊆ Q by blast
from nonE hm obtain ee where P: ee ∈ R(hm) by blast
with subcar have ee ∈ Q by auto
then have Ord(ee) using assms(2) Card_is_Ord Ord_in_Ord by auto
with P have (λ j. j ∈ R(hm)) ∈ R(hm) using LeastI[where i=ee and P=λ j.
  j ∈ R(hm)] by auto
with pp hm have converse(r)(µ j. j ∈ R(hm)) ∈ ((N(converse(h)(hm))) × {hm}) by auto
  then have converse(r)(µ j. j ∈ R(hm)) ∈ ((N(m)) × {hm}) using left_inverse[OF
  inj M] by simp
  then have fst(converse(r)(µ j. j ∈ R(hm))) ∈ (N(m)) by auto
} ultimately have thesis1: ∀ m ∈ M. Em ∈ (N(m)) using function_apply_equality
by auto
  { fix e
    assume e ∈ E
    then obtain m where m ∈ M and e = ⟨m, Em⟩ using function_apply_equality
    ff by auto
    with thesis1 have e ∈ Sigma(M, λ t. Nt) by auto
  } then have E ∈ Pow(Sigma(M, λ t. Nt)) by auto
  with ff have E ∈ Pi(M, λ m. Nm) using Pi_iff by auto
  then have (∃ f. f: Pi(M, λ t. Nt) ∧ (∀ t ∈ M. ft ∈ Nt)) using thesis1 by auto
  { then show thesis using AxiomCardinalChoiceGen_def assms(2) by auto
qed

Here is the equivalence from the two previous results.

theorem Q_choice_eq_secon_imp_sepa: assumes Card(Q)
  shows (∀ T. (T is a topology) ∧ (T is of second type of cardinal) → (T is separable of cardinal))
  using Q_choice_imp_second_imp_separable choice_subset_imp_choice
  using second_imp_separable_imp_Q_choice assms by auto

Given a base injective with a set, then we can find a base whose elements
are indexed by that set.

**Lemma** base_to_indexed_base:
- Assumes \( \text{B} \subseteq \text{Q} \) \( \text{B} \) \{is a base for\} \( \text{T} \)
- Shows \( \exists \text{N.} \{\text{Ni.} \text{i} \in \text{Q}\} \{\text{is a base for}\} \( \text{T} \)

**Proof**
- From assms obtain \( f \) where \( f \text{def}\) : \( \text{ff} \in \text{inj(} \text{B,} \text{Q} \) \) unfolding lepoll_def by auto
  
  **let** \( \text{ff} = \{\langle \text{b,fb} \rangle . \text{b} \in \text{B}\} \)
  
  **have** \( \text{domain(ff)=B} \) by auto
  
  moreover
  
  have **relation(ff) unfolding relation_def by auto**
  moreover
  
  have **function(ff) unfolding function_def by auto**
  ultimately
  
  have **fun:ff:B \rightarrow range(ff) using function_imp_Pi[of ff] by auto**
  then have \( \text{injj:ff \in inj(B,range(ff)) unfolding inj_def} \)
  proof
  
  \[
  \{ \text{fix w x} \\
  \text{assume AS:w\in Bx\in B}{\langle b,f b \rangle . b \in B} . w = {\langle b,f b \rangle . b \in B} . x \\
  \text{then have fw=fx using apply_equality[OF _ fun] by auto} \\
  \text{then have w=x using f_def inj_def AS(1,2) by auto} \\
  \}
  \]
  
  then show \( \forall \text{w\in B.} \forall \text{x\in B.} \{\langle b,f b \rangle . b \in B}\} . \text{w} = \{\langle b,f b \rangle . b \in B}\} \rightarrow \text{w} = \text{x} \) by auto
  
  qed
  
  then have **bij:ff\in bij(B,range(ff)) using inj_bij_range by auto**
  from fun have **range(ff)={fb. b \in B} by auto**
  with \( f \text{def} \) have **ran:range(ff)\subseteq Q using inj_def by auto**
  let \( \text{N} = \{\langle \text{i,(if i \in range(ff) then converse(ff)i else 0)} . \text{i} \in \text{Q}\} \)
  have **FN:function(N) unfolding function_def by auto**
  have **B \subseteq \{\text{Ni.} \text{i} \in \text{Q}\}**
  proof
  
  \[
  \{ \text{fix t} \\
  \text{assume a:t\in B} \\
  \text{from bij have rr:ff:B \rightarrow range(ff) unfolding bij_def inj_def by auto} \\
  \text{have ig:fft=ft using a apply_equality[OF _ rr] by auto} \\
  \text{have r:fft\in range(fft) using apply_type[OF rr a].} \\
  \text{from ig have t:fft\in Q using apply_type[OF _ a] f_def unfolding inj_def} \\
  \text{by auto} \\
  \}
  \]
  
  with \( r \) have **N(fft)=converse(fft)(fft) using function_apply_equality[OF _ FN] by auto**
  then have \( N(fft)=t \) using left_inverse[OF injj a] by auto
  then have \( t=N(fft) \) by auto
  then have \( \exists i\in \text{Q.} t=Ni \) using t(1) by auto
  then show \( t\in\{\text{Ni.} \text{i} \in \text{Q}\} \) by simp
  qed
  moreover
  
  have **\forall r\in\{\text{Ni.} \text{i} \in \text{Q}\}-B. r=0**

982
proof
  fix r
  assume r∈{Ni. i∈Q}←B
  then obtain j where R:j∈Qr=Njr∈B by auto
  { assume AS:j∈range(ff)
    with R(1) have Nj=converse(ff)j using function_apply_equality[OF _ FN] by auto
    then have Nj∈B using apply_funtype[OF inj_is_fun[OF bij_is_inj[OF bij_converse_bij[OF bij]]] AS]
      by auto
    then have False using R(3,2) by auto
  } then have j∉range(ff) by auto
  then show r=0 using function_apply_equality[OF _ FN] R(1,2) by auto
qed
ultimately have {Ni. i∈Q}=B ∨ {Ni. i∈Q}=B ∪ {0} by blast
moreover
have (B ∪ {0})-{0}=B-{0} by blast
then have (B ∪ {0})-{0} {is a base for}T using base_no_0[of BT assms(2)] by auto
then have B∪{0} {is a base for}T using assms(2) by auto
ultimately have {Ni. i∈Q} {is a base for}T by auto
then show thesis by auto
qed

75.4 Relation between numerability and compactness

If the axiom of Q choice holds, then any topology of second type of cardinal
csucc(Q) is compact of cardinal csucc(Q)

theorem compact_of_cardinal_Q:
  assumes {the axiom of} Q {choice holds for subsets} (Pow(Q))
T{is of second type of cardinal}csucc(Q)
T{is a topology}
shows ((∪T){is compact of cardinal}csucc(Q){in}T)
proof-
  from assms(1) have CC:Card(Q) and reg:∀ M N. (M Subseteq Q ∧ (∀ t∈M. Nt∉0 ∧ Nt∈Pow(Q)))
  → (∃ f:Pi(M,λ t. Nt) ∧ (∀ t∈M. ft∈Nt)) using
    AxiomCardinalChoice_def by auto
  from assms(2) obtain R where R⊆QR{is a base for}T unfolding IsSecondOfCard_def
    using Card_less_csucc_eq_le CC by auto
  with base_to_indexed_base obtain N where base:{Ni. i∈Q} {is a base for}T
  by blast
  { fix M
    assume A:∪T⊆∪(M∈Pow(T)
    let α=λU∈M. {i∈Q. N(i)⊆U}
    have inj:α∈inj(M,Pow(Q)) unfolding inj_def

983
proof
{
  show \((\lambda U \in M. \{ i \in Q . \ N(i) \subseteq U \}) \in M \rightarrow \text{Pow}(Q)\) using lam_type[of \(M\)] \%t. \(\text{Pow}(Q)\) by auto
  {  
    fix \(w \ x\)
    assume \(AS: w \in M \times M \{ i \in Q . \ N(i) \subseteq w \} = \{ i \in Q . \ N(i) \subseteq x \}\)
    from \(AS(1,2)\) \(A(2)\) have \(w \in x\) by auto
    then have \(w = \text{Interior}(w,T) = \text{Interior}(x,T)\) using assms(3) topology0.Top_2_L3[of \(T\)]
      topology0_def[of \(T\)] by auto
    then have \(\bigcup \{ \{ B \in \{ N(i). i \in Q \}. B \subseteq w \} \} = \bigcup \{ \{ B \in \{ N(i). i \in Q \}. B \subseteq x \} \}\)
      using interior_set_base_topology assms(3) base by auto
    {  
      fix \(b\)
      assume \(b \in w\)
      then have \(b \in \bigcup \{ \{ B \in \{ N(i). i \in Q \}. B \subseteq w \} \}\) using assms(1) by auto
      then obtain \(S\) where \(S: S \in \{ N(i). i \in Q \}\) \(b \in S\) \(S \subseteq w\) by blast
      then obtain \(j\) where \(j: j \in QS = N(j)\) by auto
      then have \(j \in \{ i \in Q . \ N(i) \subseteq w \}\) using assms(3) by auto
      then have \(N(j) \subseteq \bigcup \{ \{ B \in \{ N(i). i \in Q \}. B \subseteq x \} \}\) by auto
      then have \(b \in \bigcup \{ \{ B \in \{ N(i). i \in Q \}. B \subseteq w \} \}\) by auto
      then have \(b \in x\) using assms(2) by auto
    }
    moreover
    {  
      fix \(b\)
      assume \(b \in x\)
      then have \(b \in \bigcup \{ \{ B \in \{ N(i). i \in Q \}. B \subseteq x \} \}\) using assms(2) by auto
      then obtain \(S\) where \(S: S \in \{ N(i). i \in Q \}\) \(b \in S\) \(S \subseteq x\) by blast
      then obtain \(j\) where \(j: j \in QS = N(j)\) by auto
      then have \(j \in \{ i \in Q . \ N(i) \subseteq x \}\) using assms(3) by auto
      then have \(j \in \{ i \in Q . \ N(i) \subseteq w \}\) using assms(3) by auto
      then have \(N(j) \subseteq \bigcup \{ \{ B \in \{ N(i). i \in Q \}. B \subseteq w \} \}\) by auto
      then have \(b \in \bigcup \{ \{ B \in \{ N(i). i \in Q \}. B \subseteq x \} \}\) by auto
      then have \(b \in x\) using assms(2) by auto
    }
    ultimately have \(w = x\) by auto
  }
  then show \(\forall w \in M. \forall x \in M. (\lambda U \in M. \{ i \in Q . \ N(i) \subseteq U \}) w = (\lambda U \in M. \{ i \in Q . \ N(i) \subseteq U \}) x \rightarrow w = x\) by auto
}  
qed

let \(X = \lambda i \in Q. \{ u \in V \in M. \ N(i) \subseteq V \}\)
let \(M = \{ i \in Q. \ X_i \neq 0 \}\)
have \(\text{subM}: M \subseteq Q\) by auto
then have \(\text{ddd}: \mathcal{M} \subseteq Q\) using subset_imp_lepoll by auto
then have \(M \subseteq Q \forall i \in M. \ X_i \neq 0 \forall i \in M. \ X_i \subseteq \text{Pow}(Q)\) by auto

984
then have $M \subseteq \forall i \in M. \overline{X_i} \subseteq \text{Pow}(Q)$ using \text{subset_imp_lepoll}

by auto

then have $(\exists f. f: \Pi(M, \lambda t. X_t) \land (\forall t \in M. f t \in X_t))$ using \text{reg[of MX]}

by auto

then obtain $f$ where $f: f: \Pi(M, \lambda t. X_t)(!! t \in M. f t \in X_t)$ using \text{reg[of MX]}

by auto

{ fix $m$
  assume $S: m \in M$
  from $f(2) S$
  obtain $YY$ where $YY : (YY \in M \land f m = \alpha YY)$ by auto
  then have $Y : (YY \in M \land f m = \alpha YY)$ by auto
  moreover
  { fix $U$
    assume $U \in M \land (f m = \alpha U)$
    then have $U = YY$ using \text{inj inj_def YY} by auto
  }
  then have $r : \forall x. x \in M \land (f m = \alpha x) \Rightarrow x = YY$ by blast
  have $\exists ! YY. YY \in M \land f m = \alpha YY$ using \text{ex1I[of $Y. Y \in M \land f m = \alpha Y$,OF Y r]} by auto
  qed
}

then have $\exists ! YY. YY \in M \land f m = \alpha YY$ using \text{ex1I[of $Y. Y \in M \land f m = \alpha Y$,OF Y r]} by auto

let $YY_m = \{ \langle m, (\text{THE } YY. YY \in M \land f m = \alpha YY) \rangle. m \in M \}$

have aux: $\forall m \in M. (YY_m) \in M \land f m = \alpha (YY_m)$

proof
  fix $m$
  assume $C: m \in M$
  then have $\exists ! YY. YY \in M \land f m = \alpha YY$ using \text{ex1YY} by auto
  then have $(\text{THE } YY. YY \in M \land f m = \alpha YY) \in M \land f m = \alpha (\text{THE } YY. YY \in M \land f m = \alpha YY)$
    using \text{theI[of $Y. Y \in M \land f m = \alpha YY$]} by blast
  then show $(YY_m) \in M \land f m = \alpha (YY_m)$ apply (simp only: aux[OF C]) done
qed

have $tt : \forall m \in M. (YY_m) \in M \land f m = \alpha (YY_m)$

proof-
  fix $m$
  assume $C: m \in M$
  then have $QQ : m \in Q$ by auto
  from $D$ have $t : (YY_m) \in M \land f m = \alpha (YY_m)$ using $ree$ by blast
  then have $f m = \alpha (YY_m)$ by blast
  then have $(\alpha (YY_m)) \in (\lambda i \in Q. \{ \alpha U. U \in \{V \in M. N(i) \subseteq V\}\}) m$ using $f(2)$ [OF D]
  by auto
  then have $(\alpha (YY_m)) \in \{ \alpha U. U \in \{V \in M. N(m) \subseteq V\}\}$ using $QQ$ by auto
  then obtain $U$ where $U \in \{V \in M. N(m) \subseteq V\}$ and $\alpha (YY_m) = \alpha U$ by auto
  then have $r : U \cap N(m) \subseteq U \land (YY_m) = \alpha U (YY_m) \in M$ using $t$ by auto
  then have $YY_m = U$ using \text{inj_apply_equality[OF inj]} by blast
  then show $N(m) \subseteq YY_m$ using $r$ by auto
qed

985
then have \((\bigcup_{m \in M} N(m)) \subseteq (\bigcup_{m \in M} YYmm)\)
proof- { 
  fix \(s\)
  assume \(s \in (\bigcup_{m \in M} N(m))\)
  then obtain \(t\) where \(r \colon t \in M\) \(\subseteq N(t)\) by auto
  then have \(s \in YYmt\) using \(tt[\text{OF } r(1)]\) by blast
  then have \(s \in (\bigcup_{m \in M} YYmm)\) using \(r(1)\) by blast
}
then show thesis by blast qed
moreover
{
  fix \(x\)
  assume \(AT \colon x \in \bigcup T\)
  with \(A\) obtain \(U\) where \(BB \colon U \in M\) \(\subseteq T x \in U\) by auto
  then obtain \(j\) where \(BC \colon j \in Q N(j) \subseteq U x \in N(j)\) using \(\text{point_open_base_neigh[OF } base, of \ U x]\) by auto
  then obtain \(j\) where \(BC \colon j \in Q N(j) \subseteq U x \in N(j)\) using \(r(1)\) by auto
}
then have \(\bigcup T \subseteq (\bigcup_{m \in M} N(m))\) by blast
ultimately have covers: \(\bigcup T \subseteq (\bigcup_{m \in M} YYmm)\) using \(\text{subset_trans[of } \bigcup T (\bigcup_{m \in M} N(m)) (\bigcup_{m \in M} YYmm)\]
by auto
have relation(YYm) unfolding relation_def by auto
moreover
have \(f \colon \text{function}(YYm)\) unfolding function_def by auto
moreover
have \(d \colon \text{domain}(YYm) = M\) by auto
moreover
have \(r \colon \text{range}(YYm) = YYm M\) by auto
ultimately
have fun: \(YYm \colon M \rightarrow YYm M\) using \(\text{function_imp_Pi[of } YYm\) by auto
have \(YYm \subseteq \text{surj}(M, YYm M)\) using \(\text{fun_is_surj[OF } \text{fun}]\) \(r\) by auto
with \(\text{surj}_\text{fun_inv[OF this subMQ Card_is_Ord[OF CC]]}\)
have \(YYm M \subseteq M\) by auto
with \(\text{ddd}\) have \(\text{Rw:YYm M} \subseteq Q\) using \(\text{lepoll_trans}\) by blast
{
  fix \(m\) assume \(m \in M\)
  then have \(m, YYmm) \in YYm\) using \(\text{function_apply_Pair[OF } f] d\) by blast
  then have \(YYmm \subseteq YYm M\) by auto
}
then have \(l1 : \{YYmm, m \in M\} \subseteq YYm M\) by blast
{
  fix \(t\) assume \(t \in YYm M\)
  then have \(\exists x \in M. (x, t) \in YYm\) unfolding \(\text{image_def}\) by auto
  then obtain \(r\) where \(S \colon r \in M(r, t) \in YYm\) by auto
  have \(YYm r = t\) using \(\text{apply_equality[OF } S(2) \text{ fun}]\) by auto
}
986
with \( S(1) \) have \( t \in \{ \text{YYmm}. \ m \in \text{M} \} \) by auto
}

with \( \text{l}1 \) have \( \{ \text{YYmm}. \ m \in \text{M} \} = \text{YYmM} \) by blast

with covers have \( \{ \text{YYmm}. \ m \in \text{M} \} \subseteq \text{Pow(M)} \wedge \bigcup T \subseteq \bigcup \{ \text{YYmm}. \ m \in \text{M} \} \wedge \{ \text{YYmm}. \ m \in \text{M} \} \prec \text{csucc(Q)} \) using \text{ree}

Card_less_csucc_eq_le[OF CC] by blast

then have \( \exists \text{N} \in \text{Pow(M)}. \bigcup T \subseteq \bigcup \text{N} \wedge \text{N} \prec \text{csucc(Q)} \) by auto

then have \( \forall \text{M} \in \text{Pow(T)}. \bigcup \text{T} \subseteq \bigcup \text{M} \rightarrow (\exists \text{N} \in \text{Pow(M)}. \bigcup \text{T} \subseteq \bigcup \text{N} \wedge \text{N} \prec \text{csucc(Q)}) \) by auto

then show thesis using IsCompactOfCard_def Card_csucc CC Card_is_Ord by auto
qed

In the following proof, we have chosen an infinite cardinal to be able to apply the equation \( Q \times Q \approx Q \). For finite cardinals; both, the assumption and the axiom of choice, are always true.

\textbf{theorem} second_imp_compact_imp_Q_choice_PowQ:
assumes \( \forall \text{T}. (\text{T is a topology} \wedge (\text{T is of second type of cardinal}) \text{csucc(Q)}) \)
\( \rightarrow (\bigcup \text{T}) \text{is compact of cardinal} \text{csucc(Q}) \{\text{in} \text{T}\}
\) and \( \text{InfCard(Q)} \)
shows \{the axiom of} \ Q {choice holds for subsets} \ (\text{Pow(Q)})

proof-
{
fix \text{N} \text{ M}
assume AS: \text{M} \preceq Q \wedge (\forall \text{t} \in \text{M}. \text{Nt} \neq 0 \wedge \text{Nt} \subseteq \text{Pow(Q)})
then obtain \text{h} where \text{h} \in \text{inj(M,Q)} using lepoll_def by auto

have discTop:\text{Pow(Q} \times \text{M}) \ {is a topology} using Pow_is_top by auto
{
fix \text{A}
assume AS: \text{A} \in \text{Pow(Q} \times \text{M})
\text{have A} = \bigcup \{ \text{i}. \text{i} \in \text{A} \} \) by auto
with AS have \( \exists \text{T} \in \text{Pow(}\{\text{i}\}. \text{i} \in \text{Q} \times \text{M})\}. \text{A} = \bigcup \text{T} \) by auto
then have \text{A} \in \{ \bigcup \text{U}. \text{U} \in \text{Pow(}\{\text{i}\}. \text{i} \in \text{Q} \times \text{M})\} \) by auto
}
moreover
{
fix \text{A}
assume AS: \text{A} \in \{ \bigcup \text{U}. \text{U} \in \text{Pow(}\{\text{i}\}. \text{i} \in \text{Q} \times \text{M})\}
then have \text{A} \in \text{Pow(Q} \times \text{M}) \) by auto
}
ultimately
have base:\{\text{x}. \text{x} \in \text{Q} \times \text{M} \} \ {is a base for} \ \text{Pow(Q} \times \text{M}) \ unfolding \text{IsAbaseFor_def by blast}
let \text{f} = \{\text{i}. \text{i} \in \text{Q} \times \text{M}\}

have \text{fff} : \text{f} \in \text{Pow(}\{\text{i}\}. \text{i} \in \text{Q} \times \text{M}) \) using Pi_def function_def by auto
then have \text{f} \in \text{inj(}\text{Q} \times \text{M},\{\text{i}\}. \text{i} \in \text{Q} \times \text{M})\} unfolding inj_def using apply_equality

987
then have $f \in \text{bij}(Q \times M, \{\{i\}. i \in Q \times M\})$ unfolding bij_def surj_def using fff

apply_equality fff by auto
then have $Q \times M \approx \{\{i\}. i \in Q \times M\}$ using eqpoll_def by auto
then have $\{\{i\}. i \in Q \times M\} \approx Q \times M$ using eqpoll_sym by auto
then have $\{\{i\}. i \in Q \times M\} \leq Q \times Q$ using AS prod_lepoll_mono[of QQMQ] lepoll_refl[of Q]

lepoll_trans by blast
then have $\{\{i\}. i \in Q \times M\} \leq Q \times M$ using eqpoll_def by auto
then have $\{\{i\}. i \in Q \times M\} \approx Q \times M$ using eqpoll_sym by auto
then have $\{\{i\}. i \in Q \times M\} \leq Q \times M$ using AS prod_lepoll_mono[of QQMQ] lepoll_refl[of Q]

then have $\{\{i\}. i \in Q \times M\} \leq Q \times Q$ using lepoll_trans by blast
then have $\{\{i\}. i \in Q \times M\} \leq Q \times Q$ using InfCard_square_eqpoll assms(2) lepoll_eq_trans by auto
then have $\{\{i\}. i \in Q \times M\} \leq Q \times Q$ using lepoll_trans by blast
then have $\{\{i\}. i \in Q \times M\} \leq Q \times M$ using lepoll_refl[of Q]

then have $\{\{i\}. i \in Q \times M\} \leq Q \times Q$ using InfCard_square_eqpoll assms(2) lepoll_eq_trans by auto
then have $\{\{i\}. i \in Q \times M\} \leq Q \times Q$ using lepoll_trans by blast
then have $\{\{i\}. i \in Q \times M\} \leq Q \times M$ using lepoll_refl[of Q]

then have $\{\{i\}. i \in Q \times M\} \leq Q \times Q$ using InfCard_square_eqpoll assms(2) lepoll_eq_trans by auto
then have $\{\{i\}. i \in Q \times M\} \leq Q \times Q$ using lepoll_trans by blast
then have $\{\{i\}. i \in Q \times M\} \leq Q \times M$ using lepoll_refl[of Q]

then have $\{\{i\}. i \in Q \times M\} \leq Q \times Q$ using InfCard_square_eqpoll assms(2) lepoll_eq_trans by auto
then have $\{\{i\}. i \in Q \times M\} \leq Q \times Q$ using lepoll_trans by blast
then have $\{\{i\}. i \in Q \times M\} \leq Q \times Q$ using lepoll_refl[of Q]

then have $\{\{i\}. i \in Q \times M\} \leq Q \times Q$ using InfCard_square_eqpoll assms(2) lepoll_eq_trans by auto
then have $\{\{i\}. i \in Q \times M\} \leq Q \times Q$ using lepoll_trans by blast
then have $\{\{i\}. i \in Q \times M\} \leq Q \times Q$ using lepoll_refl[of Q]

then have $\{\{i\}. i \in Q \times M\} \leq Q \times Q$ using InfCard_square_eqpoll assms(2) lepoll_eq_trans by auto
then have $\{\{i\}. i \in Q \times M\} \leq Q \times Q$ using lepoll_trans by blast
then have $\{\{i\}. i \in Q \times M\} \leq Q \times Q$ using lepoll_refl[of Q]

then have $\{\{i\}. i \in Q \times M\} \leq Q \times Q$ using InfCard_square_eqpoll assms(2) lepoll_eq_trans by auto
then have $\{\{i\}. i \in Q \times M\} \leq Q \times Q$ using lepoll_trans by blast
then have $\{\{i\}. i \in Q \times M\} \leq Q \times Q$ using lepoll_refl[of Q]
fix t
assume AA: t ∈ Mₙ ≠ {0}
from AA(1) AS have Nₜ ≠ 0 by auto
with AA(2) obtain U where G: U ∈ Nₜ and notEm: U ≠ 0 by blast
then have U × {t} ⊆ cub using AA by auto
then have U × {t} ⊆ U by auto
with G notEm AA have ∃ s. (s, t) ∈ U by auto

then have ∀ t ∈ M. (Nₜ ≠ {0}) → (∃ s. (s, t) ∈ U) using S_def(2) by blast
from S_def(1) have B: ∀ f ∈ S. ∃ t ∈ M. ∃ U ∈ Nₜ. f = U × {t} by auto
from A B have ∀ t ∈ M. (Nₜ ≠ {0}) → (∃ s. (s, t) ∈ U) using S_def(2) by blast
then have bij2: converse(r): bij(range(r), S) by auto
let R = λ t ∈ M. { j ∈ range(r). converse(r) j ∈ (U × {t}. U ∈ Nₜ)}

then have ∀ t ∈ M. (Nₜ ≠ {0}) → Rt ≠ 0 by auto
then have nonE: ∀ t ∈ M. Nₜ ≠ {0} → Rt ≠ 0 by auto

then have ∀ t U V. U × {t} = V × {t} using beta_if AA by auto
then have pp: ∀ t ∈ M. ∀ j ∈ Rt. converse(r) j ∈ (U × {t}. U ∈ Nₜ) using beta_if by auto
have reg: ∀ t U V. U × {t} = V × {t} → U = V
proof-

fix t U V
assume AA: U × {t} = V × {t}

fix v
assume v ∈ V
then have {v, t} ∈ V × {t} by auto
then have {v, t} ∈ U × {t} using AA by auto
then have v ∈ U by auto

989
then have $V \subseteq U$ by auto
moreover
{  
  fix $u$
  assume $u \in U$
  then have $\langle u, t \rangle \in U \times \{t\}$ by auto
  then have $u \in V$ by auto
}
then have $U \subseteq V$ by auto
ultimately have $U = V$ by auto
}
then show thesis by auto
qed

let $E = \{(t, if N_t = \{0\} then 0 else (THE U. converse(r)(\mu j. j \in R_t) = U \times \{t\})\}$.

have $ff : \text{function}(E)$ unfolding $\text{function_def}$ by auto
moreover
{  
  fix $t$
  assume $pm : t \in M$
  {  
    assume $\text{nonEE} : N_t \neq \{0\}$
    {  
      fix $j$
      assume $j \in R_t$
      with $pm(1)$ have $j \in \text{range}(r)$ using $\text{beta_if}$ by auto
      from $r$ have $r : \text{surj}(S, \text{range}(r))$ using $\text{fun_is_surj inj_def}$ by auto
      with $\langle j \in \text{range}(r) \rangle$ obtain $d$ where $d \in S$ and $rd = j$ using $\text{surj_def}$
      by auto
      then have $j \in Q$ using $r \text{ inj_def}$ by auto
    }
    then have $\text{sub} : R_t \subseteq Q$ by blast
    from $\text{nonE pm nonEE}$ obtain $ee$ where $P : ee \in R_t$ by blast
    with $\text{sub}$ have $ee \in Q$ by auto
    then have $\text{Ord}(ee)$ using $\text{assms}(2)$ $\text{Card_is_Ord Ord_in_Ord InfCard_is_Card}$
    by blast
    with $P$ have $(\mu j. j \in R_t) \in R_t$ using $\text{LeastI[where i=ee and P=\lambda j. j\inRt]}$
    by auto
    with $pp pm$ have $\text{converse}(r)(\mu j. j \in R_t) \in \{U \times \{t\}. U \in N_t\}$ by auto
    then obtain $W$ where $\text{converse}(r)(\mu j. j \in R_t) = W \times \{t\}$ and $s : W \in N_t$ by auto
    then have $\text{(THE U. converse(r)(\mu j. j \in R_t) = U \times \{t\}) = W}$ using $\text{reg}$ by auto
    with $s$ have $\text{(THE U. converse(r)(\mu j. j \in R_t) = U \times \{t\})} \in N_t$ by auto
  }
  then have $\text{(if N_t = \{0\} then 0 else (THE U. converse(r)(\mu j. j \in R_t) = U \times \{t\}))} \in N_t$
  by auto
}
ultimately have thesis1: ∀t∈M. Et∈Nt using function_apply_equality by auto

{  
  fix e 
  assume e∈E 
  then obtain m where m∈M and e=(m,Em) using function_apply_equality 
  ff by auto 
  with thesis1 have e∈Sigma(M,λt. Nt) by auto 
}
then have E∈Pow(Sigma(M,λt. Nt)) by auto 
with ff have E∈Pi(M,λm. Nm) using Pi_iff by auto 
then have (∃f. f:Pi(M,λt. Nt) ∧ (∀t∈M. ft∈Nt)) using thesis1 by auto 
then show thesis using AxiomCardinalChoice_def assms(2) InfCard_is_Card by auto qed

The two previous results, state the following equivalence:

theorem Q_choice_Pow_eq_secon_imp_comp: 
  assumes InfCard(Q) 
  shows (∀T. (T is a topology) ∧ (T is of second type of cardinal)csucc(Q))) 
         −→ ((∪T)(is compact of cardinal)csucc(Q){in}T)
         ←→ (the axiom of Q {choice holds for subsets} (Pow(Q)))
         using second_imp_compact_imp_Q_choice_PowQ compact_of_cardinal_Q assms by auto 

In the next result we will prove that if the space (κ,Pow(κ)), for κ an infinite cardinal, is compact of its successor cardinal; then all topological spaces which are of second type of the successor cardinal of κ are also compact of that cardinal.

theorem Q_csuccQ_comp_eq_Q_choice_Pow: 
  assumes InfCard(Q) (Q){is compact of cardinal}csucc(Q){in}Pow(Q) 
  shows ∀T. (T is a topology) ∧ (T is of second type of cardinal)csucc(Q)) 
         −→ ((∪T)(is compact of cardinal)csucc(Q){in}T)
proof 
  fix T 
  {  
    assume top:T {is a topology} and sec:T{is of second type of cardinal}csucc(Q) 
    from assms have Card(csucc(Q)) Card(Q) using InfCard_is_Card Card_is_Ord 
    Card_csucc by auto 
    moreover 
    have ∪T⊆∪T by auto 
    moreover 
    {  
      fix M 
      assume MT:M∈Pow(T) and cover:∪T⊆∪M 
      from sec obtain B where B {is a base for} T B<csucc(Q) using IsSecondOfCard_def by auto 
  } 
} 

991
with \( \langle \text{Card}(Q) \rangle \) obtain \( N \) where base: \( \{N_i. \ i \in Q\} \) is a base for \( T \) using \text{Card_less_csucc_eq_le}

\[
\text{base_to_indexed_base \ by \ blast}
\]

let \( S=\langle u, \{i \in Q. N_i \subseteq u\} \rangle. \ u \in M \)

have function(S) unfolding function_def by auto

then have \( S:M \to \text{Pow}(Q) \) using \text{Pi_iff}

by auto

then have \( S \in \text{inj}(M, \text{Pow}(Q)) \) unfolding inj_def

proof

\[
\begin{aligned}
\text{fix } w \ x \\
\text{assume AS: } w \in M x \in M \{\langle u, \{i \in Q. N_i \subseteq u\}\rangle. \ u \in M\} \ w = \{\langle u, \{i \in Q. N_i \subseteq u\}\rangle. \ u \in M\} \ x
\end{aligned}
\]

with \( \langle S:M \to \text{Pow}(Q) \rangle \) have \( \text{ASS: } i \in Q. N_i \subseteq w = \{i \in Q. N_i \subseteq x\} \)

using \text{apply_equality} by auto

from \( \text{AS}(1,2) \) \( T \) have \( w \in \text{Interior}(w, T) x \in \text{Interior}(x, T) \) using \text{topology0.Top_2_L3[of T]}

\[
\begin{aligned}
\text{topology0_def[of } T \text{] by auto}
\end{aligned}
\]

then have \( \text{UN: } w= (\bigcup \{B \in \{N(i). i \in Q\}. B \subseteq w\}) x = (\bigcup \{B \in \{N(i). i \in Q\}. B \subseteq x\}) \)

using \text{interior_set_base_topology} \( \text{top base by auto} \)

\[
\begin{aligned}
\text{fix } b \\
\text{assume } b \in w
\end{aligned}
\]

then have \( b \in (\bigcup \{B \in \{N(i). i \in Q\}. B \subseteq w\}) \) using \( \text{UN}(1) \) by auto

then obtain \( S \) where \( S: S \in \{N(i). i \in Q\} \) \( b \in S \subseteq w \) by blast

then obtain \( j \) where \( j:j \in QS=N(j) \) by auto

then have \( j \in \{i \in Q. N(i) \subseteq w\} \) using \( \text{S}(3) \) by auto

then have \( N(j) \subseteq \{x \in N(j) j \in Q\} \) using \( \text{S}(2) \) \( \text{ASS j by auto} \)

then have \( b \in (\bigcup \{B \in \{N(i). i \in Q\}. B \subseteq x\}) \) by auto

then have \( b \in x \) using \( \text{UN}(2) \) by auto

moreover

\[
\begin{aligned}
\text{fix } b \\
\text{assume } b \in x
\end{aligned}
\]

then have \( b \in (\bigcup \{B \in \{N(i). i \in Q\}. B \subseteq x\}) \) using \( \text{UN}(2) \) by auto

then obtain \( S \) where \( S: S \in \{N(i). i \in Q\} \) \( b \in S \subseteq x \) by blast

then obtain \( j \) where \( j:j \in QS=N(j) \) by auto

then have \( j \in \{i \in Q. N(i) \subseteq x\} \) using \( \text{S}(3) \) by auto

then have \( j \in \{i \in Q. N(i) \subseteq w\} \) using \( \text{ASS} \) by auto

then have \( N(j) \subseteq \{w \in N(j) j \in Q\} \) using \( \text{S}(2) \) \( j(2) \) by auto

then have \( b \in (\bigcup \{B \in \{N(i). i \in Q\}. B \subseteq w\}) \) by auto

then have \( b \in w \) using \( \text{UN}(2) \) by auto

ultimately have \( w=x \) by auto

\[
\begin{aligned}
\text{then show } \forall w \in M. \forall x \in M. \{\langle u, \{i \in Q. N_i \subseteq u\}\rangle. \ u \in M\} \ w = \{\langle u, \{i \in Q. N_i \subseteq u\}\rangle. \ u \in M\} \ x \to w = x \text{ by auto}
\end{aligned}
\]
qed

then have \( S \in \text{bij}(M, \text{range}(S)) \) using \text{fun_is_surj} unfolding \text{bij_def}

\text{inj_def surj_def} by force

have \( \text{range}(S) \subseteq \text{Pow}(Q) \) by auto

then have \( \text{range}(S) \in \text{Pow}(\text{Pow}(Q)) \) by auto

moreover

have \( (\bigcup(\text{range}(S))) \in \text{is_closed_in} \ \text{Pow}(Q) \) using \text{IsClosed_def}

by auto

from this(2) \( \text{compact_closed[OF assms(2) this(1)]} \) have \( (\bigcup(\text{range}(S))) \in \text{Pow}(\text{range}(S)) \) by auto

moreover

have \( \bigcup(\text{range}(S)) \subseteq \bigcup(\text{range}(S)) \) by auto

ultimately have \( \exists S \in \text{Pow}(\text{range}(S)). (\bigcup(\text{range}(S))) \subseteq \bigcup S \)

using \text{IsCompactOfCard_def} by auto

then obtain \( SS \) where \( SS \subseteq \text{range}(S) \)

ultimately have \( \bigcup(\text{range}(S)) = \bigcup SS \) by auto

moreover

have \( \text{converse(\text{restrict(\text{converse}(S),SS)}} \) by auto

ultimately obtain \( RR \) where \( \text{converse(\text{restrict(\text{converse}(S),SS)}} = \text{converse(\text{\bigcup(\text{range}(S))})} \)

moreover

fix \( x \)

assume \( x \in \bigcup T \)

with \( \text{cover} \) have \( x \in \bigcup M \) by auto

then obtain \( R \) where \( R \in M \ x \in R \) by auto

with \( \text{MT} \) have \( R \in T \ x \in R \) by auto

then have \( \exists V \in \{ \text{Ni. i} \in Q \}. V \subseteq R \)

using \text{point_open_base_neigh} by force

then obtain \( j \) where \( j \in Q \ Nj \subseteq R \) and \( x_p : x \in Nj \) by auto

with \( \langle R \in M \rangle \)

have \( \text{SR} \in \text{range}(S) \)

using \text{apply_equality} by auto

from \text{exI}[where \( P = \lambda t. t \in \text{range}(S) \) and \( j \in t \), OF this] have \( \exists A \in \text{range}(S) \).

\( j \in A \) unfolding \text{Bex_def}

by auto

then have \( j \in (\bigcup(\text{range}(S))) \) by auto

then have \( j \in SS \) using \text{SS_def}(2) by blast

then obtain \( SR \) where \( SR \in SS \ j \in SR \) by auto

moreover

have \( \text{converse(\text{restrict(\text{converse}(S),SS)}} \in \text{surj(\text{\bigcup(\text{range}(S))})} \)

using \text{rr bij_def} by auto

ultimately obtain \( RR \) where \( \text{converse(\text{restrict(\text{converse}(S),SS)}}) = \text{SR} \)

and \( p : RR \in \text{surj(\text{\bigcup(\text{range}(S))})} \)

ultimately obtain \( RR \) using \text{surj_def} by blast

then have \( \text{converse(\text{\bigcup(\text{range}(S))})} \in \text{surj(\text{\bigcup(\text{range}(S))})} \)

by auto

moreover

993
have converse(restrict(converse(S),SS))∈inj(converse(S)SS,SS)
using rr unfolding bij_def by auto
moreover
ultimately have RR=converse(restrict(converse(S),SS))SR
using left_inverse[OF _ p] by force
moreover
with r1 have restrict(converse(S),SS)∈SS→converse(S)SS unfolding bij_def inj_def by auto
then have relation(restrict(converse(S),SS)) using Pi_def relation_def by auto
ultimately have RR=restrict(converse(S),SS)SR by auto
moreover
with SR∈range(S) using SS_def(1) by auto
from con left_inverse[OF _ this] have converse(converse(S))(converse(S)SR)=SR unfolding bij_def
by auto
ultimately have converse(converse(S))RR=SR by auto
moreover
with SR∈range(S) using SS_def(1) by auto
have converse(S):range(S)→M using bij_def inj_def by auto
with eq:RR=converse(S)SR unfolding restrict by auto
ultimately have SR=range(S)
moreover
have SR∈M using apply_funtype by auto
ultimately have SR={i∈Q. Ni⊆RR}
then have unlock_le by auto
moreover
have converse(S):range(S)→M using con bij_def inj_def by auto
with eq:RR∈M using apply_funtype
ultimately have SR={i∈Q. Ni⊆RR} using <S:M→Pow(Q)> apply_equality
by auto
then have Nj⊆RR using ⊆ by auto
with x_p have x∈RR by auto
with p have x∈⋃(converse(S)SS) by auto
}
then have ⋃T⊆⋃(converse(S)SS) by blast
moreover
{
from con have converse(S)SS={converse(S)R. R∈SS} using image_function[of converse(S) SS]
SS_def(1) unfolding range_def bij_def inj_def Pi_def by auto
have {converse(S)R. R∈SS}⊆{converse(S)R. R∈range(S)} using SS_def(1)
by auto
moreover
have converse(S):range(S)→M using con unfolding bij_def inj_def
by auto
then have {converse(S)R. R∈range(S)}⊆M using apply_funtype by force

994
ultimately
  have $(\text{converse}(S)SS) \subseteq M$ by auto
}
then have $\text{converse}(S)SS \in \text{Pow}(M)$ by auto
moreover
with rr have $\text{converse}(S)SS \approx SS$ using eqpoll_def by auto
then have $\text{converse}(S)SS \prec \text{csucc}(Q)$ using SS_def(3) eq_lesspoll_trans
by auto
  ultimately
  have $\exists N \in \text{Pow}(M). \bigcup N \subseteq \bigcup N \land N \prec \text{csucc}(Q)$ by auto
}
then have $\forall M \in \text{Pow}(T). \bigcup M \subseteq \bigcup M \rightarrow (\exists N \in \text{Pow}(M). \bigcup N \subseteq \bigcup N \land N \prec \text{csucc}(Q))$
by auto
ultimately have $(\bigcup T)\{\text{is compact of cardinal}\} \text{csucc}(Q)\{\text{in}\} T$ unfolding IsCompactOfCard_def
by auto
}
then show $(T\{\text{is a topology}\}) \land (T\{\text{is of second type of cardinal}\} \text{csucc}(Q))$
  $\rightarrow ((\bigcup T)\{\text{is compact of cardinal}\} \text{csucc}(Q)\{\text{in}\} T)$
by auto
qed

theorem Q_disc_is_second_card_csuccQ:
  assumes InfCard(Q)
  shows $\text{Pow}(Q)\{\text{is of second type of cardinal}\} \text{csucc}(Q)$
proof-
  
  fix A
  assume AS:A \in \text{Pow}(Q)
  have $A=\bigcup \{\{i\}. i \in A\}$ by auto
  with AS have $\exists T \in \text{Pow}(\{\{i\}. i \in Q\}). A=\bigcup T$ by auto
  then have $A \in \{\bigcup U. U \in \text{Pow}(\{\{i\}. i \in Q\})\}$ by auto
}
moreover
  
  fix A
  assume AS:A \in \{\bigcup U. U \in \text{Pow}(\{\{i\}. i \in Q\})\}
  then have $A \in \text{Pow}(Q)$ by auto
}
ultimately
have base:{\{x\}. x \in Q} \{is a base for\} \text{Pow}(Q) unfolding IsAbaseFor_def
by blast
let f=\{(i,\{i\}). i \in Q\}
have f\in Q\rightarrow{\{x\}. x \in Q} unfolding Pi_def function_def by auto
then have f\in inj(Q,\{\{x\}. x \in Q\}) unfolding inj_def using apply_equality
by auto
moreover
from $f\in Q\rightarrow{\{x\}. x \in Q}$ have $f\in \text{surj}(Q,\{\{x\}. x \in Q\})$ unfolding surj_def
using apply_equality
by auto
ultimately have \( f \in \text{bij}(Q,\{\{x\}. x \in Q\}) \) unfolding bij_def by auto
then have \( Q \approx \{\{x\}. x \in Q\} \) using eqpoll_def by auto
then have \( \{\{x\}. x \in Q\} \approx Q \) using eqpoll_sym by auto
then have \( \{\{x\}. x \in Q\} \leq Q \) using eqpoll_imp_lepoll by auto
then have \( \{\{x\}. x \in Q\} \prec \text{csucc}(Q) \) using Card_less_csucc_eq_le assms InfCard_is_Card by auto
with base show thesis using IsSecondOfCard_def by auto
qed

This previous results give us another equivalence of the axiom of \( Q \) choice that is apparently weaker (easier to check) to the previous one.

\textbf{theorem} \( Q \text{ disc comp csuccQ eq Q choice csuccQ} \):

assumes \( \text{InfCard}(Q) \)
shows \( (Q\{\text{is compact of cardinal}\}\text{csucc}(Q)\{\text{in}\}(\text{Pow}(Q))) \longleftrightarrow (\{\text{the axiom of} Q\{\text{choice holds for subsets}\}(\text{Pow}(Q))) \)
proof
assume \( Q\{\text{is compact of cardinal}\}\text{csucc}(Q) \{\text{in}\}\text{Pow}(Q) \)
with assms show \( (\{\text{the axiom of} Q\{\text{choice holds for subsets}\}(\text{Pow}(Q))) \) using \( Q\text{ choice Pow_eq secon_imp comp Q csuccQ comp eq Q choice Pow} \)
by auto
next
assume \( (\{\text{the axiom of} Q\{\text{choice holds for subsets}\}(\text{Pow}(Q))) \)
with assms show \( Q\{\text{is compact of cardinal}\}\text{csucc}(Q)\{\text{in}\}(\text{Pow}(Q)) \) using \( Q\text{ disc is second card csuccQ Q choice Pow_eq secon_imp comp Pow is top[of Q]} \)
by force
qed

end

76 Topology 5

\textbf{theory} Topology_ZF_5 \textbf{imports} Topology_ZF_properties Topology_ZF_examples_1 Topology_ZF_4
begin

76.1 Some results for separation axioms

First we will give a global characterization of \( T_1 \)-spaces; which is interesting because it involves the cardinal \( N \).

\textbf{lemma} (in topology0) \( T_1\_cocardinal\_coarser: \)
shows \( (T \{\text{is } T_1\}) \longleftrightarrow (\text{CoFinite } (\bigcup T) \subseteq T \)
proof
{ assume \( AS: T \{\text{is } T_1\} \)


996
fix x assume p:x∈∪T
{
  fix y assume y∈(∪T)-(x)
  with AS p obtain U where U∈T y∈U x∉U using isT1_def by blast
  then have U∈T y∈∪(∪T)-(x) by auto
  then have ∃U∈T. y∈U ∧ U⊂(∪T)-(x) by auto
}
then have ∀y∈(∪T)-(x). ∃U∈T. y∈U ∧ U⊂(∪T)-(x) by auto
then have ∪T-{x}∈T using open_neigh_open by auto
with p have {x} {is closed in}T using IsClosed_def by auto
then have pointCl:∀x∈∪T. {x} {is closed in}T by auto
{
  fix A
  assume AS2:A∈FinPow(∪T)
  let p={(x, {x}). x∈A}
  have p∈A→{{x}. x∈A} using Pi_def unfolding function_def by auto
  then have p:bij(A,{{x}. x∈A}) unfolding bij_def inj_def surj_def
  using apply_equality
  by auto
  then have A≈{{x}. x∈A} unfolding eqpoll_def by auto
  with AS2 have Finite({{x}. x∈A}) unfolding FinPow_def using eqpoll_imp_Finite_iff
  by auto
  then have {{x}. x∈A}∈FinPow({D ∈ Pow(∪T). D {is closed in} T})
  using AS2 pointCl unfolding FinPow_def
  by (safe, blast+)
  then have (∪{{x}. x∈A}) {is closed in} T using fin_union_cl_is_cl
  by auto
  moreover
  have ∪{{x}. x∈A}=A by auto
  ultimately have A {is closed in} T by simp
}
then have reg:∀A∈FinPow(∪T). A {is closed in} T by auto
{
  fix U
  assume AS2:U ∈ CoCardinal(∪T,nat)
  then have U∈Pow(∪T) U=0 ∨ (∪T)-U≈nat using CoCardinal_def by auto
  then have U∈Pow(∪T) U=0 ∨ Finite(∪T-U) using lesspoll_nat_is_Finite
  by auto
  then have U∈Pow(∪T) U∈TV(∪T-U) {is closed in} T using empty_open
  topSpaceAssum
    reg unfolding FinPow_def by auto
    then have U∈Pow(∪T) U∈TV(∪T-{(∪T-U)})∈T using IsClosed_def by auto
  moreover
    have (∪T-{(∪T-U)})=U by blast
    ultimately have U∈T by auto
}

997
then show \((\text{CoFinite} \ (\bigcup T)) \subseteq T\) using \text{Cofinite_def} by auto 

assume \((\text{CoFinite} \ (\bigcup T)) \subseteq T\) then have \(\text{AS:CoCardinal}(\bigcup T, \text{nat}) \subseteq T\) using \text{Cofinite_def} by auto 

fix \(x \ y\) assume \(\text{AS2:} x \in \bigcup T \ y \in \bigcup T x \neq y\) have \(\text{Finite}\{y\}\) by auto 
then obtain \(n\) where \(\{y\} \approx n\) \(n \in \text{nat}\) using \text{Finite_def} by auto 
then have \(\{y\} \prec \text{nat}\) using \(\text{n_lesspoll_nat eq_lesspoll_trans}\) by auto 
then have \(\{y\} \ {\text{is closed in}} \ \text{CoCardinal}(\bigcup T, \text{nat})\) using \text{closed_sets_cocardinal} 
\(\text{AS2}(2)\) by auto 
then have \((\bigcup T) - \{y\} \in \text{CoCardinal}(\bigcup T, \text{nat})\) using \text{union_cocardinal} \text{IsClosed_def} by auto 
with \(\text{AS}\) have \((\bigcup T) - \{y\} \in T\) by auto 
moreover with \(\text{AS2}(1,3)\) have \(\forall x \ (x \in (\bigcup T) - \{y\}) \land y \notin (\bigcup T) - \{y\}\) by auto 
ultimately have \(\exists V \in T. \ x \in V \land y \notin V\) by (safe, auto) 
then show \(T \ {\text{is T}_1}\) using \text{isT1_def} by auto 

qed 

In the previous proof, it is obvious that we don’t need to check if ever cofinite set is open. It is enough to check if every singleton is closed.

corollary (in \text{topology0}) \text{T1_iff_singleton_closed:}\nshows \((T \ {\text{is T}_1}) \iff (\forall x \in \bigcup T. \ x\ {\text{is closed in}} T)\) 
proof
assume \(T \ {\text{is T}_1}\) 

fix \(x \ p: x \in \bigcup T\) 

fix \(y\) assume \(y \in (\bigcup T) - \{x\}\) 
with \(\text{AS}\) obtain \(U\) where \(U \in T \ y \in U \ x \notin U\) using \text{isT1_def} by blast 
then have \(U \in T \ y \in U \ U \subseteq (\bigcup T) - \{x\}\) by auto 
then have \(\exists U \in T. \ y \in U \land U \subseteq (\bigcup T) - \{x\}\) by auto 
then have \(\forall y \in (\bigcup T) - \{x\}. \ \exists U \in T. \ y \in U \land U \subseteq (\bigcup T) - \{x\}\) by auto 
then have \(\bigcup T - \{x\} \in T\) using \text{open_neigh_open} by auto 
with \(p\) have \(\{x\} \ {\text{is closed in}} T\) using \text{IsClosed_def} by auto 
then show \(\text{pointCl:} \forall x \in \bigcup T. \ x\ {\text{is closed in}} T\) by auto
next 
assume \(\text{pointCl:} \forall x \in \bigcup T. \ x\ {\text{is closed in}} T\) 

fix \(A\) 
assume \(\text{AS2:} A \in \text{FinPow}(\bigcup T)\) 
let \(p=\{(x,\{x\}). \ x \in A\} \)
have \( p \in A \rightarrow \{ x \}. x \in A \) using \( \text{Pi_def} \) unfolding \( \text{function_def} \) by \( \text{auto} \)
then have \( p : \text{bij}(A, \{ x \}. x \in A) \) unfolding \( \text{bij_def} \) inj_def surj_def
using apply_equality
by \( \text{auto} \)
then have \( A \approx \{ x \}. x \in A \) unfolding \( \text{bij_def} \) inj_def surj_def
by \( \text{auto} \)
then have \( A \approx \{ x \}. x \in A \) unfolding \( \text{eqpoll_def} \) by \( \text{auto} \)
with \( \text{AS2} \) have \( \text{Finite}(\{ x \}. x \in A) \) unfolding \( \text{FinPow_def} \) using \( \text{eqpoll_imp_Finite_iff} \) by \( \text{auto} \)
then have \( \{ \bigcup \{ x \}. x \in A \} \in \text{FinPow}(\bigcup T) \) using \( \text{AS2} \) pointCl unfolding \( \text{FinPow_def} \)
by \( \text{(safe, blast+)} \)
then have \( \bigcup \{ x \}. x \in A \in T \) using \( \text{fin_union_cl_is_cl} \) by \( \text{auto} \)
moreover
then have \( \bigcup \{ x \}. x \in A = A \) by \( \text{auto} \)
ultimately have \( A ~ T \) by simp
\}
then have \( \text{reg:} \forall A \in \text{FinPow}(\bigcup T). A \in T \) by \( \text{auto} \)
\{
fix \( U \)
assume \( \text{AS2:} U \in \text{CoCardinal}(\bigcup T, \text{nat}) \)
then have \( U \in \text{Pow}(\bigcup T) \) U=0 \( \lor \) ((\bigcup T)-U)\textless nat using \( \text{CoCardinal_def} \) by \( \text{auto} \)
then have \( U \in \text{Pow}(\bigcup T) \) U=0 \( \lor \) Finite((\bigcup T)-U) using \( \text{lesspoll_nat_is_Finite} \) by \( \text{auto} \)
then have \( U \in \text{Pow}(\bigcup T) \) U=0 \( \lor \) Finite((\bigcup T)-U) T {is closed in} T using \( \text{empty_open} \) topSpaceAssum
reg unfolding \( \text{FinPow_def} \) by \( \text{auto} \)
then have \( U \in \text{Pow}(\bigcup T) \) U\in T\-(\bigcup T-U) \( \in \) T using \( \text{IsClosed_def} \) by \( \text{auto} \)
moreover
then have \( \bigcup T-(\bigcup T-U) = U \) by \( \text{blast} \)
ultimately have \( U \in T \) by \( \text{auto} \)
\}
then have \( \{ \text{CoFinite} (\bigcup T) \} \subseteq T \) using \( \text{CoFinite_def} \) by \( \text{auto} \)
then show \( T \) {is \( T_1 \)} using \( \text{Ti_cocardinal_coarser} \) by \( \text{auto} \)
qed

Secondly, let’s show that the CoCardinal \( X Q \) topologies for different sets \( Q \) are all ordered as the partial order of sets. (The order is linear when considering only cardinals)

lemma order_cocardinal_top:
fixes \( X \)
assumes \( Q_1 \ll Q_2 \)
shows \( \text{CoCardinal}(X,Q_1) \subseteq \text{CoCardinal}(X,Q_2) \)
proof
fix \( x \)
assume \( x \in \text{CoCardinal}(X,Q_1) \)
then have \( x \in \text{Pow}(X) \) x=0\( \lor \) (X-x)\textless Q_1 using \( \text{CoCardinal_def} \) by \( \text{auto} \)
with \( \text{assms} \) have \( x \in \text{Pow}(X) \) x=0\( \lor \) (X-x)\textless Q_2 using \( \text{lesspoll_trans2} \) by \( \text{auto} \)
then show \( x \in \text{CoCardinal}(X,Q_2) \) using \( \text{CoCardinal_def} \) by \( \text{auto} \)
corollary \texttt{cocardinal\_is\_T1}:
\begin{itemize}
\item \texttt{fixes} \(X \ K\)
\item \texttt{assumes} \(\text{InfCard}(K)\)
\item \texttt{shows} \(\text{CoCardinal}(X,K) \text{\{is\ T}_1\})
\end{itemize}
\texttt{proof-}
\begin{itemize}
\item have \(\text{nat} \leq K\) using \text{InfCard}\_def \texttt{assms\ auto}
\item then have \(\text{nat} \subseteq K\) using \text{le\_imp\_subset\ by\ \texttt{auto}}
\item then have \(\text{nat} \subseteq K\text{\neq\0}\) using \text{subset\_imp\_lepoll\ by\ \texttt{auto}}
\item then have \(\text{CoCardinal}(X,\text{nat}) \subseteq \text{CoCardinal}(X,K) \cup \text{CoCardinal}(X,K)=X\) using \text{order\_cocardinal\_top}
\item union\_cocardinal\_by\ \texttt{auto}
\item then show thesis using \text{topology0}\_T1\_cocardinal\_coarser \text{topology0}\_\text{CoCardinal}
\item \texttt{assms\ Cofinite\_def}
\item \texttt{by\ \texttt{auto}}
\end{itemize}
\texttt{qed}

In \(T_2\)-spaces, filters and nets have at most one limit point.

\begin{itemize}
\item \texttt{lemma (in topology0) T2\_imp\_unique\_limit\_filter:}
\item \texttt{assumes} \(T \\{\text{is T}_2\}\) \(\mathcal{F} \\{\text{is a filter on}\} \cup T\ \mathcal{F} \rightarrow_{F} x \ \mathcal{F} \rightarrow_{F} y\)
\item \texttt{shows} \(x=y\)
\item \texttt{proof-}
\item \{ \begin{itemize}
\item assume \(x \neq y\)
\item from \texttt{assms(3,4)} have \(x \in \cup T\) \(y \in \cup T\) using \text{FilterConverges}\_def \texttt{assms(2)}
\item by \texttt{auto}
\item with \(\langle x \neq y \rangle\) have \(\exists U \in T. \exists V \in T. x \in \text{Interior}(A,T) \land y \in \text{Interior}(A,T)\)
\item using \text{Top}_2\_\text{L3}\ by \texttt{auto}
\item then have \(U \in \mathcal{F}\) \(V \in \mathcal{F}\) using \text{FilterConverges}\_def \texttt{assms(2)} \texttt{assms(3,4)}
\item by \texttt{auto}
\item then have \(U \cap V \in \mathcal{F}\) using \text{IsFilter}\_def \texttt{assms(2)} by \texttt{auto}
\item with \(\langle U \cap V = 0 \rangle\) have \(0 \in \mathcal{F}\) by \texttt{auto}
\item then have \texttt{False\ using\ IsFilter\_def\ assms(2)\ by\ \texttt{auto}}
\end{itemize}
\item \texttt{then show thesis by \texttt{auto}}
\end{itemize}
\texttt{qed}

\begin{itemize}
\item \texttt{lemma (in topology0) T2\_imp\_unique\_limit\_net:}
\item \texttt{assumes} \(T \\{\text{is T}_2\}\) \(N \\{\text{is a net on}\} \cup T\ N \rightarrow_{N} x \ N \rightarrow_{N} y\)
\item \texttt{shows} \(x=y\)
\item \texttt{proof-}
\item \begin{itemize}
\item have \((\text{Filter} \ N.\cdot(\cup T)) \{\text{is a filter on}\} (\cup T) (\text{Filter} \ N.\cdot(\cup T)) \rightarrow_{F} x\ (\text{Filter} \ N.\cdot(\cup T)) \rightarrow_{F} y\)
\item using \text{filter\_of\_net\_is\_filter(1)} \text{net\_conver\_filter\_of\_net\_conver} \texttt{assms(2)} \texttt{assms(3,4) by \texttt{auto}}
\end{itemize}
\end{itemize}

with assms(1) show thesis using T2_imp_unique_limit_filter by auto

qed

In fact, $T_2$-spaces are characterized by this property. For this proof we build a filter containing the union of two filters.

lemma (in topology0) unique_limit_filter_imp_T2: 
  assumes $\forall x \in \bigcup T. \forall y \in \bigcup T. (\langle \mathcal{F} \rangle \text{ is a filter on} \bigcup T) \land (\mathcal{F} \rightarrow F \ x) \land (\mathcal{F} \rightarrow F \ y) \rightarrow x=y$
  shows $T \{\text{is } T_2\}$
proof-
  { 
    fix $x \ y$
    assume $x \in \bigcup T \ y \in \bigcup T \ x \neq y$
    { 
      assume $\forall U \in T. \forall V \in T. (x \in U \land y \in V) \rightarrow U \cap V \neq 0$
      let $U_x = \{ A \in \text{Pow}\{\bigcup T\}. x \in \text{int}(A)\}$
      let $U_y = \{ A \in \text{Pow}\{\bigcup T\}. y \in \text{int}(A)\}$
      let $FF = U_x \cup U_y \cup \{ A \cap B. \langle A, B \rangle \in U_x \times U_y\}$
      have sat: $FF \{\text{satisfies the filter base condition}\}$
      proof
        { 
          fix $A \ B$
          assume $A \in FF \ B \in FF$
          { 
            assume $A \in U_x$
            { 
              assume $B \in U_x$
              with $\langle x \in \bigcup T \rangle \langle A \in U_x \rangle$ have $A \cap B \in U_x$ using neigh_filter(1)
              IsFilter_def by auto
              then have $A \cap B \in FF$ by auto
            }
            moreover
            { 
              assume $B \in U_y$
              with $\langle A \in U_x \rangle$ have $A \cap B \in FF$ by auto
            }
            moreover
            { 
              assume $B \in \{ A \cap B. (A, B) \in U_x \times U_y\}$
              then obtain $AA \ BB$ where $B=AA \cap BB$ $AA \in U_x \ BB \in U_y$ by auto
              with $\langle x \in \bigcup T \rangle \langle A \in U_x \rangle$ have $A \cap B = (A \cap AA) \cap BB$ $A \cap AA \in U_x$ using neigh_filter(1)
              IsFilter_def by auto
              with $\langle BB \in U_y \rangle$ have $A \cap B \in \{ A \cap B. (A, B) \in U_x \times U_y\}$ by auto
              then have $A \cap B \in FF$ by auto
            } 
          } 
          ultimately have $A \cap B \in FF \ using \ B \in FF \ by \ auto$
        }
        moreover
        { 
        }


1001
assume \( A \in U_y \)

\{ 
    assume \( B \in U_y \)
    with \( \forall y \in T. \forall A \in U_y \) have \( A \cap B \in U_y \) using neigh_filter(1)

IsFilter_def by auto
then have \( A \cap B \in FF \) by auto
\}

moreover
\{ 
    assume \( B \in U_x \)
    with \( A \in U_y \) have \( B \cap A \in FF \) by auto
    moreover have \( A \cap B = B \cap A \) by auto
    ultimately have \( A \cap B \in FF \) by auto
\}

moreover
\{ 
    assume \( B \in \{ A \cap B. (A,B) \in U_x \times U_y \} \)
    then obtain \( AA \ BB \) where \( B = AA \cap BB \) \( AA \in U_x \) \( BB \in U_y \) by auto
    with \( \forall y \in T. \forall A \in U_y \) have \( A \cap B = AA \cap (A \cap BB) \) \( A \cap BB \in U_y \) using neigh_filter(1)

IsFilter_def by auto
with \( AA \in U_x \) have \( A \cap B \in \{ \forall y \in T. \forall A \in U_y \} \) by auto
then have \( A \cap B \in FF \) by auto
\}

ultimately have \( A \cap B \in FF \) using \( B \in FF \) by auto
\}

moreover
\{ 
    assume \( A \in \{ A \cap \forall y \in U_y \} \)
    then obtain \( AA \ BB \) where \( A = AA \cap BB \) \( AA \in U_x \) \( BB \in U_y \) by auto
    \{ 
        assume \( B \in U_y \)
        with \( \forall y \in T. \forall A \in U_y \) have \( B \cap BB \in U_y \) using neigh_filter(1)
        morefrom \( A = AA \cap BB \) have \( A \cap B = AA \cap (B \cap BB) \) by auto
        ultimately have \( A \cap B \in FF \) using \( AA \in U_x \) \( B \in BB \in U_y \) by auto
    \}
    moreover
    \{ 
        assume \( B \in U_x \)
        with \( \forall y \in T. \forall A \in U_y \) have \( B \cap AA \in U_x \) using neigh_filter(1)
        morefrom \( A = AA \cap BB \) have \( A \cap B = (B \cap AA) \cap BB \) by auto
        ultimately have \( A \cap B \in FF \) using \( B \cap AA \in U_x \) \( BB \in U_y \) by auto
    \}
    moreover
    \{ 
        assume \( B \in \{ A \cap \forall y \in U_y \} \)
        then obtain \( AA \ BB \) where \( B = AA \cap BB \) \( AA \in U_x \) \( BB \in U_y \) by auto
        from \( B = AA \cap BB \) \( A = AA \cap BB \) have \( A \cap B = (AA \cap AA) \cap (BB \cap BB) \)
\\}
by auto
  moreover
  from \(<A \in Ux> <A \in Ux> x \in \bigcup T>\) have \(A \cap A \in Ux\) using neigh_filter(1)
IsFilter_def by auto
moreover
from \(<B \in Uy> <B \in Uy> y \in \bigcup T>\) have \(B \cap B \in Uy\) using neigh_filter(1)
IsFilter_def by auto
ultimately have \(A \cap B \in FF\) by auto
  }
ultimately have \(A \cap B \in FF\) using \(<B \in FF>\) by auto
  }
ultimately have \(A \cap B \in FF\) using \(<A \in FF>\) by auto
then have \(\exists D \in FF. D \subseteq A \cap B\) unfolding Bex_def by auto
  }
then have \(\forall A \in FF. \forall B \in FF. \exists D \in FF. D \subseteq A \cap B\) by force
moreover
have \(\bigcup T \cup Ux\) using \(<x \in \bigcup T>\) neigh_filter(1) IsFilter_def by auto
then have \(FF \neq \emptyset\) by auto
moreover
  
  assume \(0 \in FF\)
moreover
have \(0 \notin Ux\) using \(<x \in \bigcup T>\) neigh_filter(1) IsFilter_def by auto
moreover
have \(0 \notin Uy\) using \(<y \in \bigcup T>\) neigh_filter(1) IsFilter_def by auto
ultimately have \(0 \in \{A \cap B. \langle A, B\rangle \in Ux \times Uy\}\) by auto
then obtain \(A\ B\) where \(0 = A \cap B\ A \in Ux B \in Uy\) by auto
then have \(x \in \text{int}(A) \text{ y} \in \text{int}(B)\) by auto
moreover
with \(<0 = A \cap B>\) have \(\text{int}(A) \cap \text{int}(B) = \emptyset\) using Top_2_L1 by auto
moreover
have \(\text{int}(A) \subseteq \text{int}(B) \in T\) using Top_2_L2 by auto
ultimately have \(\text{False}\) using \(<\forall U \in T. \forall V \in T. x \in U \land y \in V \longrightarrow U \cap V \neq \emptyset>\)
by auto
  }
then have \(0 \notin FF\) by auto
ultimately show thesis using SatisfiesFilterBase_def by auto
qed
moreover
have \(FF \subseteq \text{Pow}(\bigcup T)\) by auto
ultimately have \(\text{bas:FF}\ \{\text{is a base filter}\} \{A \in \text{Pow}(\bigcup T). \exists D \in FF. D \subseteq A\}
\bigcup \{A \in \text{Pow}(\bigcup T). \exists D \in FF. D \subseteq A\} = \bigcup T\)
using base_unique_filter_set2[of FF] by auto
then have \(\text{fil:}\{A \in \text{Pow}(\bigcup T). \exists D \in FF. D \subseteq A\} \ \{\text{is a filter on}\ \bigcup T\}
using basic_filter sat by auto
have \(\forall U \in \text{Pow}(\bigcup T). x \in \text{int}(U) \longrightarrow (\exists D \in FF. D \subseteq U)\) by auto
then have \(\{A \in \text{Pow}(\bigcup T). \exists D \in FF. D \subseteq A\} \longrightarrow x \text{ using convergence_filter_base2[OF fil bas(1)] < x < \bigcup T>}\) by auto
moreover
then have $\forall U \in \text{Pow}(\bigcup T). y \in \text{int}(U) \rightarrow (\exists D \in \text{FF} \cdot D \subseteq A)$ by auto

ultimately have $x=y$ using assms fil \(\langle x \in \bigcup T \rangle \langle y \in \bigcup T \rangle\) by blast with \(\langle x \neq y \rangle\) have False by auto

then have $\exists U \in T. \exists V \in T. x \in U \land y \in V \land U \cap V = \emptyset$ by blast

then show thesis using isT2_def by auto

qed

lemma (in topology0) unique_limit_net_imp_T2:
assumes $\forall x \in \bigcup T. \forall y \in \bigcup T. \forall N. ((N \text{ is a net on } \bigcup T) \land (N \rightarrow_N x) \land (N \rightarrow_N y)) \rightarrow x=y$
shows $T \{\text{is } T_2\}$
proof-
\{
  fix x y \(\triangle\)
  assume $x \in \bigcup T \land y \in \bigcup T \land \{\text{is a filter on } \bigcup T\} \land (\text{Net}(\triangle) \rightarrow_F x) \land (\text{Net}(\triangle) \rightarrow_F y)$
  then have $\{\text{Net}(\triangle)\} \{\text{is a net on } \bigcup T\} \land (\text{Net}(\triangle) \rightarrow_N x) \land (\text{Net}(\triangle) \rightarrow_N y)$
    using filter_conver_net_of_filter_conver net_of_filter_is_net by auto
    with \(\langle x \in \bigcup T \rangle \langle y \in \bigcup T \rangle\) have $x=y$ using assms by blast
  \}
then have $\forall x \in \bigcup T. \forall y \in \bigcup T. \forall \triangle. ((\triangle \{\text{is a filter on } \bigcup T\} \land (\triangle \rightarrow_F x) \land (\triangle \rightarrow_F y)) \rightarrow x=y$ by auto
then show thesis using unique_limit_filter_imp_T2 by auto
qed

This results make easy to check if a space is $T_2$.

The topology which comes from a filter as in \(\triangle \{\text{is a filter on } \bigcup T\} \cup \triangle \rightarrow (\triangle \cup \text{cons}(\emptyset, \emptyset))\) is a topology} is not $T_2$ generally. We will see in this file later on, that the exceptions are a consequence of the spectrum.

corollary filter_T2_imp_cardi:
assumes $(\triangle \cup \text{cons}(\emptyset, \emptyset)) \{\text{is } T_2\} \\triangle \{\text{is a filter on } \bigcup T\} \land (\triangle \rightarrow_F x) \land (\triangle \rightarrow_F y)$
shows $\bigcup \triangle = \{x\}$
proof-
\{
  fix y assume $y \in \bigcup \triangle$
  then have $\triangle \rightarrow_F y \langle \text{in} \rangle (\text{\triangle} \cup \text{cons}(\emptyset, \emptyset))$ using lim_filter_top_of_filter assms(2) by auto
  moreover
  have $\triangle \rightarrow_F x \langle \text{in} \rangle (\text{\triangle} \cup \text{cons}(\emptyset, \emptyset))$ using lim_filter_top_of_filter assms(2,3) by auto
  moreover
  have $\bigcup \triangle = \bigcup (\text{\triangle} \cup \text{cons}(\emptyset, \emptyset))$ by auto
  ultimately
  have $y=x$ using topology0.T2_imp_unique_limit_filter[of topology0_filter[of...
There are more separation axioms that just $T_0$, $T_1$ or $T_2$

definition
\isT3 (_{is T_3} 90)
where T{is T_3} \equiv (T{is T_1}) \land (T{is regular})

definition
\IsNormal (_{is normal} 90)
where T{is normal} \equiv \forall A. A{is closed in}T \rightarrow (\forall B. B{is closed in}T \\
\land A\cap B=0 \rightarrow \\
(\exists U\in T. \exists V\in T. A\subseteq U\land B\subseteq V\land U\cap V=0))

definition
\isT4 (_{is T_4} 90)
where T{is T_4} \equiv (T{is T_1}) \land (T{is normal})

lemma (in topology0) T4_is_T3:
assumes T{is T_4}
shows T{is T_3}
proof-
from assms have nor: T{is normal} using \isT4_def by auto
from assms have T{is T_1} using \isT4_def by auto
then have Cofinite (\bigcup T) \subseteq T using T1_cocardinal_coarser by auto
{
  fix A
  assume AS: A{is closed in}T
  
  fix x
  assume x elem T-A
  have Finite({x}) by auto
  then obtain n where \{x\} \approx n n\in\mathbb{N} unfolding Finite_def by auto
  then have \{x\} \subseteq n n\in\mathbb{N} using eqpoll_imp_lepoll by auto
  then have \{x\} \approx n n\in\mathbb{N} unfolding n_lesspoll_nat lesspoll_trans1 by auto
  with \langle x elem T-A \rangle have \{x\} {is closed in} (Cofinite (\bigcup T)) using Cofinite_def
  closed_sets_cocardinal by auto
  then have \bigcup T-{x} elem Cofinite(\bigcup T) unfolding IsClosed_def using union_cocardinal
  Cofinite_def
  by auto
  with \langle Cofinite (\bigcup T) \subseteq T \rangle have \bigcup T-{x} elem T by auto
  with \langle x elem T-A \rangle have \{x\} {is closed in}T \cap \{x\} = 0 unfolding IsClosed_def
  by auto
  with nor AS have \\exists U\in T. \exists V\in T. A\subseteq U\cap \{x\} \subseteq V\land \{x\} = 0 unfolding IsNormal_def
  by blast
}
then have $\exists U \in T. \exists V \in T. A \subseteq U \land x \in V \land V \cap x = 0$ by auto
}
then have $\forall x \in \bigcup T - A. \exists U \in T. \exists V \in T. A \subseteq U \land x \in V \land U \cap V = 0$ by auto
}
then have $T$ is regular) using IsRegular_def by blast
with $\langle T \{T_1\} \rangle$ show thesis using isT3_def by auto
qed

lemma (in topology0) T3_is_T2:
  assumes $T$ is $T_3$ shows $T$ is $T_2$
proof-
  from assms have $T$ is regular) using isT3_def by auto
  from assms have $T$ is $T_1$ using isT3_def by auto
  then have Cofinite $(\bigcup T) \subseteq T$ using T1_cocardinal_coarser by auto
  {  
    fix x y  
    assume $x \in \bigcup T \land y \in \bigcup T \land x \neq y$  
    have Finite($\{x\}$) by auto
    then obtain n where $\{x\} \approx n \land n \in \text{nat}$ unfolding Finite_def by auto
    then have $\{x\} \leq n \land n \in \text{nat}$ using eqpoll_imp_lepoll by auto
    then have $\langle x \in \bigcup T \rangle$ have $\{x\}$ is closed in $(\text{Cofinite} (\bigcup T))$ using Cofinite_def closed_sets_cocardinal by auto
    then have $\bigcup T - \{x\} \in \text{Cofinite}(\bigcup T)$ unfolding IsClosed_def using union_cocardinal Cofinite_def by auto
    with $\langle \text{Cofinite} (\bigcup T) \subseteq T \rangle$ have $\bigcup T - \{x\} \in T$ by auto
    with $\langle \{x\} \subseteq U \land y \in V \land V \cap x = 0 \rangle$ unfolding IsRegular_def by force
    then have $\exists U \in T. \exists V \in T. x \in U \land y \in V \land V \cap x = 0$ by auto
  }  
  then show thesis using isT2_def by auto
qed

Regularity can be rewritten in terms of existence of certain neighborhoods.

lemma (in topology0) regular_imp_exist_clos_neig:
  assumes $T$ is regular) and $U \in T$ and $x \in U$
  shows $\exists V \in T. x \in V \land \text{cl}(V) \subseteq U$
proof-
  from assms(2) have $(\bigcup T - U) \{\text{is closed in} \} T$ using Top_3_L9 by auto moreover
  from assms(2,3) have $x \in \bigcup T$ by auto moreover
  note assms(1,3) ultimately obtain $A \land B$ where $A \in T$ and $B \in T$ and $A \cap B = 0$
  and $(\bigcup T - U) \subseteq A$ and $x \in B$
  unfolding IsRegular_def by blast
  from $\langle A \cap B = 0 \rangle$ $\langle B \in T \rangle$ have $B \subseteq \bigcup T - A$ by auto
with \( A \in T \) have \( \text{cl}(B) \subseteq \bigcup T - A \) using Top_3.L9 Top_3.L13 by auto

moreover from \( \bigcup (T - U) \subseteq A \) assms(3) have \( U - A \subseteq U \) by auto

moreover note \( x \in B \) \( \langle B \in T \rangle \)

ultimately have \( B \in T \) \( \wedge x \in B \wedge \text{cl}(B) \subseteq U \) by auto

then show thesis by auto

qed

lemma (in topology0) exist_clos_neig_imp_regular:

assumes \( \forall x \in \bigcup T. \forall U \in T. x \in U \rightarrow (\exists V \in T. x \in V \wedge \text{cl}(V) \subseteq U) \)

shows \( T \text{ is regular} \)

proof-

\{
fix \( F \)
assume \( F \text{ is closed in } T \)
\{
fix \( x \)
assume \( x \in \bigcup T - F \)
with \( \langle F \text{ is closed in } T \rangle \) have \( x \in \bigcup T \bigcup T - F \subseteq T \) unfolding IsClosed_def by auto

with assms \( x \in \bigcup T - F \) have \( \exists V \in T. x \in V \wedge \text{cl}(V) \subseteq \bigcup T - F \) by auto

then obtain \( V \) where \( V \in T x \in V \land \text{cl}(V) \subseteq \bigcup T - F \) by auto

ultimately have \( F \subseteq \text{int}(\bigcup T - V) \) using Top_3.L11(2)[of \( \bigcup T - V \)] by auto

moreover have \( F \subseteq \text{int}(\bigcup T - V) \) by auto

ultimately have \( \text{int}(\bigcup T - V) \subseteq \text{int}(\bigcup T - V) \) by auto

then have \( U \in T \) \( \exists V \in T. A \subseteq U \wedge x \in V \) \( \wedge U \cap V = 0 \) by auto

ultimately have \( V \in T \) \( \exists V \in T. F \subseteq \text{int}(\bigcup T - V) \land x \in V \land (\text{int}(\bigcup T - V)) \cap V = 0 \) unfolding Top_2.L2 by auto

then have \( \forall x \in \bigcup T - F. \exists U \in T. F \subseteq x \in U \land \cap V = 0 \) by auto

\}
\}

then show thesis using IsRegular_def by blast

qed

lemma (in topology0) regular_eq:

shows \( T \text{ is regular} \iff (\forall x \in \bigcup T. \forall U \in T. x \in U \rightarrow (\exists V \in T. x \in V \wedge \text{cl}(V) \subseteq U)) \)

using regular_imp_exist_clos_neig exist_clos_neig_imp_regular by force

A Hausdorff space separates compact spaces from points.

theorem (in topology0) T2_compact_point:

assumes \( T \text{ is T}_2 \) \( A \text{ is compact in } T \) \( x \in \bigcup T x \notin A \)

shows \( \exists U \in T. \exists V \in T. A \subseteq U \wedge x \in V \wedge U \cap V = 0 \)

proof-

\{
assume \( A = 0 \)
then have $A \subseteq 0 \land x \in \bigcup T \cap (0 \cap (\bigcup T) = 0)$ using assms(3) by auto
then have thesis using empty_open topSpaceAssum unfolding IsATopology_def by auto

moreover
{
assume noEmpty: $A \neq 0$
let $U = \{(U, V) \in T \times T. \ x \in U \cup U \cap V = 0\}$
{
fix $y$ assume $y \in A$
with $\langle x \notin A \rangle$ assms(4) have $x \neq y$ by auto
moreover from $\langle y \in A \rangle$ have $x \in \bigcup T \in T$ using assms(2,3) unfolding IsCompact_def by auto
ultimately obtain $U$ $V$ where $U \in T$ $V \in T$ $U \cap V = 0$
then have $\exists \langle U, V \rangle \in U. y \in V$ by auto
then have $\forall y \in A. \exists \langle U, V \rangle \in U. y \in V$ by auto
then have $\forall y \in A. \exists (U, V) \in U. y \in V$ by auto
then have $A \subseteq \bigcup \{\text{snd}(B). B \in U\}$ by auto
moreover have $\{\text{snd}(B). B \in U\} \in \text{Pow}(T)$ by auto
ultimately have $\exists N \in \text{FinPow}(\{\text{snd}(B). B \in U\}). A \subseteq \bigcup N$ using assms(2) unfolding IsCompact_def by auto
then obtain $N$ where $ss: N \in \text{FinPow}(\{\text{snd}(B). B \in U\})$ $A \subseteq \bigcup N$ by auto
then have $N \subseteq \{\text{snd}(B). B \in U\}$ unfolding FinPow_def by auto
from $ss$ have $\text{Finite}(N)N \subseteq \{\text{snd}(B). B \in U\}$ unfolding FinPow_def by auto
then have $N \subseteq n$ using eqpoll_imp_lepoll by auto
from noEmpty $\langle A \cup N \rangle$ have $\text{NoEmpty}: N \neq 0$ by auto
let $QQ = \{\langle n, \{\text{fst}(B). B \in \{A \cup U. \text{snd}(A) = n\}\} \rangle. n \in N\}$
have $QQ_1: QQ \ni \{\text{fst}(B). B \in \{A \cup U. \text{snd}(A) = n\}\}. n \in N$ unfolding Pi_def function_def domain_def by auto
{
fix $n$ assume $n \in N$
with $\langle n \subseteq \{\text{snd}(B). B \in U\} \rangle$ obtain $B$ where $n = \text{snd}(B)$. $B \in U$ by auto
then have $\text{fst}(B) \subseteq \{\text{fst}(B). B \in \{A \cup U. \text{snd}(A) = n\}\}$ by auto
then have $\{\text{fst}(B). B \in \{A \cup U. \text{snd}(A) = n\}\} \neq 0$ by auto
moreover
from $\langle n \subseteq N \rangle$ have $\langle \text{fst}(B). B \in \{A \cup U. \text{snd}(A) = n\}\rangle \subseteq QQ$ by auto
with $QQ_1$ have $QQ_n = \{\text{fst}(B). B \in \{A \cup U. \text{snd}(A) = n\}\}$ by auto
do auto
ultimately have $QQ_n \neq 0$ by auto
}
then have $\forall n \in N. QQ_n \neq 0$ by auto
with $\langle n \in \text{nat}, \langle n \subseteq n \rangle \rangle$ have $\exists f. f : \Pi(N, \lambda t. QQ_t) \land (\forall t \in N. ft \in QQ_t)$ using finite_choice unfolding AxiomCardinalChoiceGen_def by auto
then obtain $f$ where $fPI: f : \Pi(N, \lambda t. QQ_t) \land (\forall t \in N. ft \in QQ_t)$ by auto

1008
from fPI(1) NnoEmpty have range(f)\neq 0 unfolding Pi_def range_def domain_def converse_def by (safe,blast)
{
  fix t assume t\in\text{N}
  then have ft\in\text{QQt} using fPI(2) by auto
  with \langle t\in\text{N} \rangle have ft\in\bigcup (\text{QQN}) \text{ QQt}\subseteq \bigcup (\text{QQN}) using func_imagedef QQPi by auto
}
then have reg: \forall t\in\text{N}. ft\in\bigcup (\text{QQN}) \text{ \forall } \text{t}\in\text{N}. \text{ QQt}\subseteq \bigcup (\text{QQN}) by auto
{
  fix tt assume tt\in f
  with fPI(1) have tt\in\text{Sigma(\text{N}, ()(\text{QQ}))} unfolding Pi_def by auto
  then have tt\in (\bigcup xa\in\text{N}. \bigcup y\in\text{QQxa}. \{\langle xa,y \rangle \}) unfolding Sigma_def by auto
  then obtain xa y where xa\in\text{N} y\in\text{QQxa} tt=\langle xa,y \rangle by auto
  with reg(2) have y\in\bigcup (\text{QQN}) by blast
  with \langle \text{tt}=\langle xa,y \rangle \rangle \langle \text{xa}\in\text{N} \rangle have \text{tt}\in (\bigcup xa\in\text{N}. \bigcup y\in\text{QQN}. \{\langle xa,y \rangle \}) by auto
  then have \text{tt}\in\text{N}\times (\bigcup (\text{QQN})) unfolding Sigma_def by auto
}
then have ffun: f: \text{N}\to \bigcup (\text{QQN}) using fPI(1) unfolding Pi_def by auto
then have f\in\text{surj(\text{N},range(f))} using fun_is_surj by auto
with \langle \text{N}\subseteq \text{n} \rangle \langle \text{n}\in\text{nat} \rangle have range(f)\subseteq \text{N} using surj_fun_inv_2 nat_into_Ord by auto
with \langle \text{N}\subseteq \text{n} \rangle have range(f)\subseteq \text{N} using lepoll_trans by blast
with \langle \text{n}\in\text{nat} \rangle have Finite(range(f)) using n_lesspoll_nat lesspoll_nat_is_Finite lesspoll_trans1 by auto
moreover from ffun have rr: range(f)\subseteq \bigcup (\text{QQN}) unfolding Pi_def by auto
then have range(f)\subseteq T by auto
ultimately have range(f)\in\text{FinPow(T)} unfolding FinPow_def by auto
then have \bigcap range(f)\in T using fin_inter_open_open \langle range(f)\neq 0 \rangle by auto
moreover
{
  fix S assume S\in range(f)
  with rr have S\in\bigcup (\text{QQN}) by blast
  then have \exists B\in(\text{QQN}). S \in B using Union_iff by auto
  then obtain B where B\in(\text{QQN}) S\subseteq B by auto
  then have \exists rr\in\text{N}. (rr,B)\in\text{QQ} unfolding image_def by auto
  then have \exists rr\in\text{N}. B=\{fst(B), B\in\{A\in\text{U}. \text{snd(A)}=rr\} \} by auto
  with \langle S\in B \rangle obtain rr where \langle S,rr \rangle\in\text{U} by auto
  then have x\in S by auto
}
then have x\in\bigcap range(f) using \langle range(f)\neq 0 \rangle by auto
moreover
{
  fix y assume y\in(\bigcup \text{N})\cap (\bigcap range(f))
  then have reg:\forall S\in range(f). y\in S \land (\exists t\in\text{N}. y=t) by auto
  then obtain t where t\in\text{N} y\in t by auto
  then have \langle t, \{fst(B), B\in\{A\in\text{U}. \text{snd(A)}=t\} \} \rangle\in\text{QQ} by auto
}
then have \( ft \in \text{range}(f) \) using apply_rangeI ffun by auto
with \( \langle t \in \mathbb{N} \rangle \) fPI(2) have \( ft \in \mathbb{Q} \) by auto
with \( \langle t \in \mathbb{N} \rangle \) have \( ft \in \{\text{fst}(B). \ B \in \{A \in U. \ \text{snd}(A)=t\}\} \) using apply_equality

\[ \mathbb{Q} \cap \mathbb{P} = 0 \]

ultimately have thesis by auto
qed

A Hausdorff space separates compact spaces from other compact spaces.

\[ \text{thm (in topology0) \ T2\_compact\_compact:} \]
assumes \( T \{\text{is T}\_2\} A \{\text{is compact in}\} T \ B \{\text{is compact in}\} T \ A \cap B = 0 \)
shows \( \exists U \in T. \ \exists V \in T. \ A \subseteq U \wedge B \subseteq V \wedge U \cap V = 0 \)

proof-
{ assume \( B = 0 \)
then have \( A \subseteq \cup T \wedge B \subseteq \emptyset \wedge (\cup T) \cap B = 0 \) using assms(2) unfolding IsCompact_def
by auto moreover
have \( 0 \in T \) using empty_open topSpaceAssum by auto moreover
have \( \cup T \in T \) using topSpaceAssum unfolding IsATopology_def by auto
ultimately have thesis by auto
}
moreover
{ assume noEmpty: \( B \neq 0 \)
let \( U = \{\langle U, V \rangle \in T \times T. \ A \subseteq U \wedge U \cap V = 0\} \)
{ fix \( y \) assume \( y \in B \)
then have \( y \in \cup T \) using assms(3) unfolding IsCompact_def by auto
with \( \langle y \in B \rangle \) have \( \exists U \in T. \ \exists V \in T. \ A \subseteq U \wedge y \in V \wedge U \cap V = 0 \) using T2\_compact\_point
assms(1,2,4) by auto
then have \( \exists \langle U, V \rangle \in U. \ y \in V \) by auto
}
then have \( \forall y \in B. \ \exists \langle U, V \rangle \in U. \ y \in V \) by auto
then have \( B \subseteq \cup \{\text{snd}(B). \ B \in U\} \) by auto
moreover have \( \{\text{snd}(B). \ B \in U\} \subseteq \text{Pow}(T) \) by auto
ultimately have \( \exists N \in \text{FinPow}(\{\text{snd}(B, B \in U)\}). \ B \subseteq \cup N \) using assms(3) unfolding IsCompact_def by auto
then obtain \( N \) where ss: \( N \in \text{FinPow}(\{\text{snd}(B). B \in U\}) \) \( B \subseteq \cup N \) by auto
with \( \langle \text{snd}(B). B \in U \rangle \in \text{Pow}(T) \) have \( B \subseteq \cup N \) using \( \text{FinPow}\_\text{def} \) by auto

1010
then have \( N : B \subseteq U \cup N \subseteq T \) using topSpaceAssum unfolding IsATopology_def by auto

from ss have Finite(N)\( N \subseteq \{\text{snd}(B). B \subseteq U\} \) unfolding FinPow_def by auto
then obtain \( n \) where \( n \in \text{nat} \approx n \) unfolding Finite_def by auto
then have \( N \subseteq n \) using eqpoll_imp_lepoll by auto
from noEmpty \( \langle B \subseteq U \rangle \) have NnoEmpty: \( N \neq 0 \) by auto
let \( QQ = \{\langle n, \{\text{fst}(B). B \subseteq \{A \in U. \text{snd}(A) = n\}\} \rangle. n \in N\} \) unfolding Pi_def function_def domain_def by auto
unfolding with \( \langle n, \{\text{fst}(B). B \subseteq \{A \in U. \text{snd}(A) = n\}\} \rangle. n \in N \} \) unfolding Pi_def function_def domain_def by auto

\[
\text{ultimately have } QQ \neq 0 \text{ by auto}
\]
then have \( \forall n \in N. QQ \neq 0 \) by auto
with \( \langle n \in \text{nat} \rangle \langle n \leq n \rangle \) have \( \exists f. f : \Pi(N, \lambda t. QQ) \land (\forall t \in N. ft \in QQ) \) unfolding \( \text{finite}_\text{choice} \) unfolding AxiomCardinalChoiceGen_def by auto
then obtain \( f \) where \( fPI : f : \Pi(N, \lambda t. QQ) \land (\forall t \in N. ft \in QQ) \) by auto
from \( fPI(1) \) NnoEmpty have range(f)\( \neq 0 \) unfolding Pi_def range_def domain_def converse_def by (safe, blast)

\[
\text{ultimately have } QQ = \{\text{fst}(B). B \subseteq \{A \in U. \text{snd}(A) = n\}\} \text{ using } \text{apply}_\text{equality}
\]
by auto
ultimately have \( QQ \neq 0 \) by auto
then have \( \forall n \in N. QQ \neq 0 \) by auto
with \( \langle n \in \text{nat} \rangle \langle n \leq n \rangle \) have \( \exists f. f : \Pi(N, \lambda t. QQ) \land (\forall t \in N. ft \in QQ) \) unfolding \( \text{finite}_\text{choice} \) unfolding AxiomCardinalChoiceGen_def by auto
ultimately have \( QQ = \{\text{fst}(B). B \subseteq \{A \in U. \text{snd}(A) = n\}\} \) using \( \text{apply}_\text{equality} \)
by auto
ultimately have \( QQ \neq 0 \) by auto
then have \( \forall n \in N. QQ \neq 0 \) by auto
with \( \langle n \in \text{nat} \rangle \langle n \leq n \rangle \) have \( \exists f. f : \Pi(N, \lambda t. QQ) \land (\forall t \in N. ft \in QQ) \) unfolding \( \text{finite}_\text{choice} \) unfolding AxiomCardinalChoiceGen_def by auto
ultimately have \( QQ \neq 0 \) by auto
then have \( \forall n \in N. QQ \neq 0 \) by auto
with \( \langle n \in \text{nat} \rangle \langle n \leq n \rangle \) have \( \exists f. f : \Pi(N, \lambda t. QQ) \land (\forall t \in N. ft \in QQ) \) unfolding \( \text{finite}_\text{choice} \) unfolding AxiomCardinalChoiceGen_def by auto
ultimately have \( QQ \neq 0 \) by auto
then have \( \forall n \in N. QQ \neq 0 \) by auto
with \( \langle n \in \text{nat} \rangle \langle n \leq n \rangle \) have \( \exists f. f : \Pi(N, \lambda t. QQ) \land (\forall t \in N. ft \in QQ) \) unfolding \( \text{finite}_\text{choice} \) unfolding AxiomCardinalChoiceGen_def by auto
ultimately have \( QQ \neq 0 \) by auto
then have \( \forall n \in N. QQ \neq 0 \) by auto
with \( \langle n \in \text{nat} \rangle \langle n \leq n \rangle \) have range(f)\( \subseteq n \) using \( \text{lepoll}_\text{trans} \) by blast
with \( \langle \in \text{nat} \rangle \) have \( \text{Finite}(\text{range}(f)) \) using \( \text{n_lesspoll_nat} \), \( \text{lesspoll_nat_is_Finite} \)
lesspoll_trans1 by auto
moreover from \( \text{ffun} \) have \( \text{rr}: \text{range}(f) \subseteq \bigcup (\text{QQN}) \) unfolding \( \text{Pi_def} \) by auto
ultimately have \( \text{range}(f) \subseteq \text{T} \) using \( \text{fin_inter_open_open} \) \( \langle \text{range}(f) \neq 0 \rangle \) by auto
moreover
\{
  \text{fix} \ S \ \text{assume} \ S \in \text{range}(f)
  \text{with} \text{rr} \ \text{have} \ S \in \bigcup (\text{QQN}) \text{ by blast}
  \text{then} \text{have} \ \exists B \in (\text{QQN}). \ S \in B \ \text{using} \ \text{Union_iff} \ \text{by auto}
  \text{then} \text{obtain} \ B \ \text{where} \ B \in (\text{QQN}) \ S \in B \ \text{by auto}
  \text{then} \text{have} \ \exists \text{rr} \in \text{N}. \langle \text{rr}, B \rangle \in \text{QQ} \ \text{unfolding} \ \text{image_def} \ \text{by auto}
  \text{then} \text{obtain} \ B \text{ where} \langle \text{rr}, B \rangle \in (\text{QQN}) \ S \in B \ \text{by auto}
  \text{then obtain} \text{rr} \in \text{N}. \ B = \{ \text{A} \in \text{U}. \ \text{snd(A)} = \text{rr} \} \ \text{by auto}
  \text{with} \langle \text{S}, \text{rr} \rangle \in \text{U} \ \text{by auto}
  \text{then have} \ A \subseteq S \ \text{by auto}
}\}
then have \( \bigcap \text{range}(f) \subseteq \bigcup (\text{QQN}) \) using \( \langle \text{range}(f) \neq 0 \rangle \) by auto
moreover
\{
  \text{fix} \ y \ \text{assume} \ y \in (\bigcup \text{N}) \cap (\bigcap \text{range}(f))
  \text{then} \text{have} \ \text{reg}: (\forall S \in \text{range}(f). \ y \in S) \land (\exists t \in \text{N}. \ y \in t) \ \text{by auto}
  \text{then} \text{obtain} \ t \ \text{where} t \in \text{N} \ y \in t \ \text{by auto}
  \text{then have} \langle t, \{ \text{fst}(B). \ B \in \{ \text{A} \in \text{U}. \ \text{snd(A)} = \text{t} \} \rangle \in \text{QQ} \ \text{by auto}
  \text{then have} \langle t \rangle \in \text{range}(f) \ \text{using} \ \text{apply_rangeI} \ \text{ffun} \ \text{by auto}
  \text{with} \ \text{reg} \ \text{have} \ y \in \text{ft} \ \text{by auto}
  \text{with} \langle \text{t} \in \text{N} \rangle \ \text{fPI(2)} \ \text{have} \text{ft} \in \text{QQt} \ \text{by auto}
  \text{with} \langle \text{t} \in \text{N} \rangle \ \text{have} \text{ft} \in \{ \text{A} \in \text{U}. \ \text{snd(A)} = \text{t} \} \ \text{using} \ \text{apply_equality} \ \text{QQPi} \ \text{by auto}
  \text{then have} \langle \text{ft}, t \rangle \in \text{U} \ \text{by auto}
  \text{then have} \text{ft} \cap t = 0 \ \text{by auto}
  \text{with} \langle \text{y} \in t \rangle \ \text{yft} \ \text{have} \ False \ \text{by auto}
}\}
then have \( (\bigcap \text{range}(f)) \cap (\bigcup \text{N}) = 0 \) by blast
note \( \text{NN} \)
ultimately have \( \text{thesis} \) by auto
\}
ultimately show \( \text{thesis} \) by auto
qed

A compact Hausdorff space is normal.

corollary \( \langle \text{in} \ \text{topology0} \rangle \) \( \text{T2_compact_is_normal} \):
\( \text{assumes} T \{ \text{is} \ T_2 \} \ (\bigcup T) \{ \text{is compact in} T \} \)
\( \text{shows} T \{ \text{is normal} \} \) unfolding \( \text{IsNormal_def} \)
proof
  from \( \text{assms(2)} \) have \( \text{car_nat} : (\bigcup T) \{ \text{is compact of cardinal} \} \ \text{nat} \{ \text{in} T \} \)
  using \( \text{Compact_is_card_nat} \) by auto
  \{ 
    \text{fix} A B \ \text{assume} A \{ \text{is closed in} T \} B \{ \text{is closed in} T \} A \cap B = 0
\}
then have com:\((\bigcup T)\cap A)\{is compact of cardinal\}nat\{in\}T \ (\bigcup T)\cap B)\{is compact of cardinal\}nat\{in\}T using compact_closed[OF car_nat]
by auto
from \(A\{is closed in\}T\)<B\{is closed in\}T> have \((\bigcup T)\cap A=A(\bigcup T)\cap B=B
unfolding IsClosed_def by auto
with com have \(A\{is compact in\}T B\{is compact in\}T
by auto
then have \(A\{is compact in\}T B\{is compact in\}T
using Compact_is_card_nat by auto
with \(A \cap B=0\\)
show \(\exists U \in T. \exists V \in T. A \subseteq U \land B \subseteq V \land U \cap V = 0\) using T2_compact_compact assms(1) by auto
}\nthen show \(\forall A. A \{is closed in\} T \longrightarrow (\forall B. B \{is closed in\} T \land A \cap B = 0 \longrightarrow (\exists U \in T. \exists V \in T. A \subseteq U \land B \subseteq V \land U \cap V = 0))
by auto
qed

76.2 Hereditability
A topological property is hereditary if whenever a space has it, every sub-
space also has it.
definition IsHer (_{is hereditary} 90)
where \(P \{is hereditary\} \equiv \forall T. T\{is a topology\} \land P(T) \longrightarrow (\forall A \in \text{Pow}(\bigcup T). P(T\{restricted to\}A))\)

lemma subspace_of_subspace:
assumes \(A \subseteq B \subseteq \bigcup T\)
shows \(T\{restricted to\}A=(T\{restricted to\}B)\{restricted to\}A\)
proof
from assms have \(S:\forall S \in T. A \cap (B\cap S)=A \cap S\) by auto
then show \(T \{restricted to\} A \subseteq T \{restricted to\} B \{restricted to\}\)
by auto
unfolding RestrictedTo_def
from S show \(T \{restricted to\} B \{restricted to\} A \subseteq T \{restricted to\} A\)
by auto
qed

The separation properties \(T_0, T_1, T_2, T_3\) are hereditary.

theorem regular_here:
assumes \(T\{is regular\} A \in \text{Pow}(\bigcup T)\) shows \((T\{restricted to\}A)\{is regular\}
proof-
{ fix \(C\)
assume \(A:C\{is closed in\}(T\{restricted to\}A)\)
{fix \(y\) assume \(y\in \bigcup (T\{restricted to\}A)\) \(y\not\in C\)
with \(A\) have \((\bigcup (T\{restricted to\}A))-C \subseteq (T\{restricted to\}A)\) \(C \subseteq \bigcup (T\{restricted to\}A)\) \(y\in \bigcup (T\{restricted to\}A)\) \(y\not\in C\)
unfolding IsClosed_def

1013
by auto
moreover
with assms(2) have $\bigcup (T\text{restricted to} A) = A$ unfolding RestrictedTo_def
by auto
ultimately have $A-C \subseteq T\text{restricted to} A$ y $\forall y \in C \subseteq \text{Pow}(A)$ by auto
then obtain $S$ where $S \subseteq T\text{restricted to} A - C$ y $\forall y \in C$ unfolding RestrictedTo_def
by auto
then have $y \in A - C \cap S = A - C$ by auto
with $\langle C \subseteq \text{Pow}(A) \rangle$ have $y \in A \cap S = A - S$ by auto
then have $y \in S \subseteq A - S$ by auto
with assms(2) have $y \in S \subseteq U \cup - T - S$ by auto
moreover
from $\langle S \subseteq T \rangle$ have $U - (U \cup - T - S) = S$ by auto
moreover
with $\langle S \subseteq T \rangle$ have $(U \cup - T - S) \{\text{is closed in}\} T$ using IsClosed_def by auto
ultimately have $y \in U \cup - (U \cup - T - S) \{\text{is closed in}\} T$ by auto
with assms(1) have $\forall y \in (U \cup - T) \cap S \subseteq U \cup y \in V \cup U \cup V = 0$
unfolding IsRegular_def by auto
with $\langle y \in U \cup - (U \cup - T - S) \rangle$ have $\exists U \subseteq T \cup S \cup y \subseteq V \cup U \cup V = 0$ by auto
then obtain $U \cup V$ where $U \subseteq V \subseteq T - S \cup y \subseteq U \cup V \subseteq U \cup V = 0$ by auto
then have $A \cap U \subseteq (T\text{restricted to} A) \cap S \subseteq (T\text{restricted to} A) \cap U \subseteq (A \cup U) \cup (A \cap V) = 0$
unfolding RestrictedTo_def using $\langle C \subseteq U \cup - T - S \rangle$ by auto
moreover
with $\langle C \subseteq \text{Pow}(A) \rangle \langle y \in A \rangle$ have $C \subseteq A \cup U \cup y \cup A \cap V$ by auto
ultimately have $\exists U \subseteq (T\text{restricted to} A)$. $\exists V \subseteq (T\text{restricted to} A)$. $C \subseteq U \cup y \subseteq V \cup U \cup V = 0$
by auto
then have $\forall x \in (T\text{restricted to} A) \cup - C$. $\exists U \subseteq (T\text{restricted to} A)$. $\exists V \subseteq (T\text{restricted to} A)$
then have $\forall C. C\{\text{is closed in}\}(T\text{restricted to} A) \rightarrow (\forall x \in (T\text{restricted to} A) \cup - C$. $\exists U \subseteq (T\text{restricted to} A)$. $\exists V \subseteq (T\text{restricted to} A)$. $C \subseteq U \cup x \subseteq V \cup U \cup V = 0$
by blast
then show thesis using IsRegular_def by auto
qed

corollary here_regular:
shows IsRegular \{is hereditary\} using regular_here IsHer_def by auto

theorem T1_here:
assumes $T\{\text{is} T_1\}$ $A \subseteq \text{Pow}(\bigcup T)$ shows $(T\text{restricted to} A)\{\text{is} T_1\}$
proof-
from assms(2) have $\bigcup (T\text{restricted to} A) = A$ unfolding RestrictedTo_def
by auto

{ fix x y
assume $x \in A \cap y \in A \neq y$
with $\langle A \subseteq \text{Pow}(\bigcup T) \rangle$ have $x \subseteq \bigcup T \subseteq \bigcup T - y$ by auto
then have $\exists U \subseteq T$. $x \subseteq U \cap y \subseteq U$ using assms(1) isT1_def by auto
}

1014
then obtain $U$ where $U \in Tx \cap U \notin U$ by auto
with $\langle x \in A \rangle$ have $A \cap U \in (T \{\text{restricted to} \}A)$ $x \in A \cap U$ $y \notin A \cap U$ unfolding $\text{RestrictedTo_def}$
by auto
then have $\exists U \in (T \{\text{restricted to} \}A)$. $x \in U \land y \notin U$ by blast
}
with $\text{un}$ have $\forall x y$. $x \in \bigcup (T \{\text{restricted to} \}A)$ $\land x \neq y$ $\rightarrow$ $(\exists U \in (T \{\text{restricted to} \}A)$. $x \in U \land y \notin U$)
by auto
then show thesis using $\text{isT1_def}$ by auto
qed

**corollary here_T1:**
shows $\text{isT1}$ {is hereditary} using $\text{T1_here IsHer_def}$ by auto

**lemma here_and:**
assumes $P$ {is hereditary} $Q$ {is hereditary}
shows $(\forall T. P(T) \land Q(T))$ {is hereditary} using $\text{assms unfolding IsHer_def}$
by auto

**corollary here_T3:**
shows $\text{isT3}$ {is hereditary} using $\text{here_and[OF here_T1 here_regular]}$ unfolding $\text{IsHer_def isT3_def}$.

**lemma T2_here:**
assumes $T$ {is $T_2$} $A \in \text{Pow}(\bigcup T)$ shows $(T \{\text{restricted to} \}A) \{\text{is $T_2$}\}$
proof-
from $\text{assms}(2)$ have $\text{un:} \bigcup (T \{\text{restricted to} \}A)=A$ unfolding $\text{RestrictedTo_def}$
by auto
{
fix $x y$
assume $x \in A \land y \notin x \neq y$
with $\langle A \in \text{Pow}(\bigcup T) \rangle$ have $x \in \bigcup T \land y \notin \bigcup T \neq y$ by auto
then have $\exists U \in T$. $\exists V \in T$. $x \in U \land y \in V \land U \cap V = 0$ using $\text{assms}(1)$ $\text{isT2_def}$ by auto

then obtain $U V$ where $U \in T \land V \in Tx \subset U \land y \notin U \land U \cap V = 0$ by auto
with $\langle x \in A \rangle$ $\langle y \in A \rangle$ have $A \cap U \in (T \{\text{restricted to} \}A) \land A \cap V \in (T \{\text{restricted to} \}A)$
$x \in A \cap U$ $y \in A \cap V$ $(A \cap U) \cap (A \cap V) = 0$ unfolding $\text{RestrictedTo_def}$ by auto
then have $\exists U \in (T \{\text{restricted to} \}A)$. $\exists V \in (T \{\text{restricted to} \}A)$. $x \in U \land y \in V \land U \cap V = 0$
unfolding $\text{Bex_def}$ by auto
}
with $\text{un}$ have $\forall x y$. $x \in \bigcup (T \{\text{restricted to} \}A)$ $\land y \in \bigcup (T \{\text{restricted to} \}A)$
$\land x \neq y$ $\rightarrow$ $(\exists U \in (T \{\text{restricted to} \}A)$. $\exists V \in (T \{\text{restricted to} \}A)$. $x \in U \land y \in V \land U \cap V = 0$)
by auto
then show thesis using $\text{isT2_def}$ by auto
qed

**corollary here_T2:**
shows $\text{isT2}$ {is hereditary} using $\text{T2_here IsHer_def}$ by auto
lemma T0_here:
  assumes T{is T0} A ∈ Pow(⋃T) shows (T{restricted to}A){is T0}
proof-
  from assms(2) have un:⋃(T{restricted to}A)=A unfolding RestrictedTo_def by auto
  { fix x y
    assume x∈A y∈Ax≠y
    with ⟨A∈Pow(⋃T)⟩ have x∈Ty∈Tx≠y by auto
    then have ∃U∈T. (x∈U∧y∉U)∨(y∈U∧x∉U) unfolding RestrictedTo_def by auto
    then obtain U where U∈T (x∈U∧y∉U)∨(y∈U∧x∉U) unfolding RestrictedTo_def by auto
    then have ∃U∈(T{restricted to}A). (x∈U∧y∉U)∨(y∈U∧x∉U) unfolding RestrictedTo_def by auto
  }
  with un have ∀x y. x∈⋃(T{restricted to}A) ∧ y∈⋃(T{restricted to}A)
  ∧ x≠y → (∃U∈(T{restricted to}A). (x∈U∧y∉U)∨(y∈U∧x∉U)) by auto
  then show thesis using isT0_def by auto
qEd

corollary here_T0:
  shows isT0 {is hereditary} using T0_here IsHer_def by auto

76.3 Spectrum and anti-properties

The spectrum of a topological property is a class of sets such that all topologies defined over that set have that property.

The spectrum of a property gives us the list of sets for which the property doesn’t give any topological information. Being in the spectrum of a topological property is an invariant in the category of sets and function; meaning that equipollent sets are in the same spectra.

definition Spec (_ {is in the spectrum of} _) ≡ ∀T. ((T{is a topology} ∧ ⋃T≈K) → P(T))

lemma equipollent_spect:
  assumes A≈B B {is in the spectrum of} P
  shows A {is in the spectrum of} P
proof-
  from assms(2) have ∀T. ((T{is a topology} ∧ ⋃T≈B) → P(T)) using Spec_def by auto
  then have ∀T. ((T{is a topology} ∧ ⋃T≈A) → P(T)) using eqpoll_trans[OF _ assms(1)] by auto
  then show thesis using Spec_def by auto
qEd
theorem eqpoll_iff_spec:  
  assumes A≈B 
  shows (B {is in the spectrum of} P) ⟷ (A {is in the spectrum of} P) 
proof 
  assume B {is in the spectrum of} P 
  with assms equipollent_spect show A {is in the spectrum of} P by auto 
next 
  assume A {is in the spectrum of} P 
  moreover 
  from assms have B≈A using eqpoll_sym by auto 
  ultimately show B {is in the spectrum of} P using equipollent_spect by auto 
qed 

From the previous statement, we see that the spectrum could be formed only by representative of classes of sets. If AC holds, this means that the spectrum can be taken as a set or class of cardinal numbers.

Here is an example of the spectrum. The proof lies in the indiscrète filter \( \{A\} \) that can be build for any set. In this proof, we see that without choice, there is no way to define the spectrum of a property with cardinals because if a set is not comparable with any ordinal, its cardinal is defined as 0 without the set being empty.

theorem T4_spectrum:  
  shows (A {is in the spectrum of} isT4) ⟷ A ≲ 1 
proof 
  assume A {is in the spectrum of} isT4 
  then have reg: \( \forall T. ((T \text{ is a topology} \land \bigcup T \approx A) \rightarrow (T \text{ is T}\_2)) \) using Spec_def by auto 
  { 
    assume A≠0 
    then obtain x where x∈A by auto 
    then have x∈\( \bigcup \{A\} \) by auto 
    moreover 
    then have \( \{A\} \) {is a filter on}\( \bigcup \{A\} \) using IsFilter_def by auto 
    moreover 
    then have \( \{\{A\}\cup\{0\}\} \) {is a topology} \land \( \bigcup \{\{A\}\cup\{0\}\}=A \) using top_of_filter by auto 
    then have top: \( \{\{A\}\cup\{0\}\} \) {is a topology} \land \( \bigcup \{\{A\}\cup\{0\}\}=A \) using eqpoll_refl by auto 
  } 
  moreover 
  then have \( \{\{A\}\cup\{0\}\} \) {is T}\_2 using reg by auto 
  then have \( \{\{A\}\cup\{0\}\} \) {is T}\_3 using topology0.T3_is_T2 topology0.T4_is_T3 topology0_def top by auto 
  ultimately have \( \bigcup \{A\}={x} \) using filter_T2_imp_card1[of \{A\}x] by auto 
  then have \( A={x} \) by auto 
  then have A≈1 using singleton_eqpoll_1 by auto 
} 

moreover
have $A=0 \rightarrow A \approx 0$ by auto
ultimately have $A \approx 1 \lor A \approx 0$ by blast
then show $A \leq 1$ using empty_lepollI eqpoll_imp_lepoll eq_lepoll_trans
by auto
next
assume $A \leq 1$
have $A=0 \lor A \neq 0$ by auto
then obtain $E$ where $A=0 \lor E \in A$ by auto
with $\langle A \leq 1 \rangle$ have $A \approx \emptyset \lor (E \in A)$ by auto
then have $A \approx \emptyset \lor A=1$ using singleton_eqpoll_1 by auto

fix $T$
assume AS: $T$ is a topology $\bigcup T = A$

assume $A=0$
with AS have $T$ is a topology and empty: $\bigcup T = 0$ using eqpoll_trans

have $T \{T \in T \} \{T \in T \}$ using isT2_def by auto
then have $T \{T \in T \}$ using T2_is_T1 by auto
moreover
from empty have $T \subseteq \emptyset$ by auto
with AS(1) have $T = \emptyset$ using empty_open by auto
from empty have $\forall A. A (is closed in) T \rightarrow A = 0$ using IsClosed_def

have $\exists U \in T. \exists V \in T. 0 \subseteq U \land 0 \subseteq V \land U \cap V = 0$

with $\forall A. A (is closed in) T \rightarrow (\forall B. B (is closed in) T \land A \cap B = 0 \rightarrow \langle 0 \subseteq U \in T. \exists V \in T. A (is closed in) T \land A \cap B = 0 \rangle)$
by blast
then have $T \{T \in T \}$ using IsNormal_def by auto
with $\langle T \{T \in T \} \rangle$ have $T \{T \in T \}$ using isT4_def by auto

moreover
assume $A = 1$
with AS have $T$ is a topology and NONempty: $\bigcup T = 1$ using eqpoll_trans[of $\bigcup TA1$] by auto
then have $\bigcup T \leq 1$ using eqpoll_imp_lepoll by auto
moreover
assume $\bigcup T = 0$
then have $0 \approx \bigcup T$ by auto
with NONempty have $0 = 1$ using eqpoll_trans by blast
then have $0 = 1$ using eqpoll_0_is_0 eqpoll_sym by auto
then have False by auto

then have $\bigcup T \neq 0$ by auto
then obtain $R$ where $R \subseteq \bigcup T$ by blast
ultimately have $\bigcup T = \{R\}$ using lepoll_1_is_sing by auto

1018
\{ 
  \text{fix } x, y \\
  \text{assume } x \text{ is closed in } T, Ty \text{ is closed in } T, x \cap y = 0 \\
  \text{then have } x \subseteq \bigcup Ty \subseteq T \text{ using IsClosed_def by auto} \\
  \text{then have } x = 0 \vee y = 0 \text{ using } \langle x \cap y = 0 \rangle \langle \bigcup T = \{R\} \rangle \text{ by force} \\
  \text{assume } x = 0 \\
  \text{then have } x \subseteq 0 \subseteq \bigcup Ty \subseteq T \text{ using } \langle y \subseteq T \rangle \text{ by auto} \\
  \text{moreover} \\
  \text{have } 0 \in T \bigcup T \in T \text{ using } \text{AS(1)} \text{ IsATopology_def empty_open by auto} \\
  \text{ultimately have } \exists U \in T. \exists V \in T. x \subseteq U \land y \subseteq V \land U \cap V = 0 \text{ by auto} \\
  \} \\
  \text{moreover} \\
  \text{assume } x \neq 0 \\
  \text{with } \langle x = 0 \vee y = 0 \rangle \text{ have } y = 0 \text{ by auto} \\
  \text{then have } x \subseteq \bigcup Ty \subseteq 0 \text{ using } \langle x \subseteq T \rangle \text{ by auto} \\
  \text{moreover} \\
  \text{have } 0 \in T \bigcup T \in T \text{ using } \text{AS(1)} \text{ IsATopology_def empty_open by auto} \\
  \text{ultimately have } \exists U \in T. \exists V \in T. x \subseteq U \land y \subseteq V \land U \cap V = 0 \text{ by auto} \\
  \} \\
  \text{ultimately} \\
  \text{have } (\exists U \in T. \exists V \in T. x \subseteq U \land y \subseteq V \land U \cap V = 0) \text{ by blast} \\
  \} \\
  \text{then have } T \{ \text{is normal} \} \text{ using IsNormal_def by auto} \\
  \text{moreover} \\
  \text{fix } x, y \\
  \text{assume } x \in \bigcup Ty \in \bigcup Tx \neq y \\
  \text{with } \langle \bigcup T = \{R\} \rangle \text{ have False by auto} \\
  \text{then have } \exists U \in T. x \in U \land y \not\in U \text{ by auto} \\
  \} \\
  \text{then have } T \{ \text{is } T_1 \} \text{ using isT1_def by auto} \\
  \text{ultimately have } T \{ \text{is } T_4 \} \text{ using isT4_def by auto} \\
  \} \\
  \text{ultimately have } T \{ \text{is } T_4 \} \text{ using } \langle A \approx 0 \vee A \approx 1 \rangle \text{ by auto} \\
  \} \\
  \text{then have } \forall T. (T \{ \text{is a topology} \} \land \bigcup T \approx A) \rightarrow (T \{ \text{is } T_4 \}) \text{ by auto} \\
  \text{then show } A \{ \text{is in the spectrum of} \} \text{ isT4 using Spec_def by auto} \\
\text{qed} \\
\}

If the topological properties are related, then so are the spectra.

lemma P_imp_Q_spec_inv: 
  \text{assumes } \forall T. (T \{ \text{is a topology} \} \land \bigcup T \approx A) \rightarrow (T \{ \text{is } T_4 \}) \text{ by auto} \\
  \text{then show } A \{ \text{is in the spectrum of} \} \text{ isT4 using Spec_def by auto} \\
\text{qed} 

with assms(1) have \( \forall T. T \text{(is a topology)} \land \bigcup T \approx A \longrightarrow P(T) \) by auto
then show thesis using Spec_def by auto
qed

Since we already know the spectrum of \( T_4 \); if we now the spectrum of \( T_0 \), it should be easier to compute the spectrum of \( T_1, T_2 \) and \( T_3 \).

theorem \( T_0 \)_spectrum:
shows \( (A \ {\text{is in the spectrum of}} \ isT_0) \iff A \preceq 1 \)
proof
assume \( A \ {\text{is in the spectrum of}} \ isT_0 \)
then have reg:\( \forall T. ((T \text{(is a topology)} \land \bigcup T \approx A) \longrightarrow (T \text{(is } T_0\text{)}) \) using Spec_def by auto
\{ assume \( A \neq 0 \)
then obtain \( x \) where \( x \in A \) by auto
then have \( x \in \bigcup \{A\} \) by auto
moreover then have \( \{A\} \ {\text{is a filter on}} \bigcup \{A\} \) using IsFilter_def by auto
moreover then have \( (\{A\} \cup \{0\}) \ {\text{is a topology}} \land \bigcup (\{A\} \cup \{0\}) \approx A \) using top_of_filter by auto
then have \( (\{A\} \cup \{0\}) \ {\text{is a topology}} \land \bigcup (\{A\} \cup \{0\}) \approx A \) using eqpoll_refl by auto
then have \( (\{A\} \cup \{0\}) \ {\text{is } T_0 \) using reg by auto
\{ fix \( y \)
assume \( y \in A \neq y \)
with \( \langle (\{A\} \cup \{0\}), T_0 \rangle \) obtain \( U \) where \( U \in (\{A\} \cup \{0\}) \) and \( \text{dis} : (x \in U \land y \notin U) \lor (y \in U \land x \notin U) \) using isT0_def by auto
then have \( U = A \) by auto
with \( \text{dis} : (y \in A) \land (x \in \bigcup \{A\}) \) have False by auto
\}
then have \( \forall y \in A, y = x \) by auto
with \( \langle x \in \bigcup \{A\}, A \approx x \rangle \) have \( A \approx 1 \) by blast
then have \( A \approx 1 \) using singleton_eqpoll_1 by auto
\} moreover
have \( A = 0 \longrightarrow A \approx 0 \) by auto
ultimately have \( A \approx 1 \lor A \approx 0 \) by blast
then show \( A \preceq 1 \) using empty_lepollI eqpoll_imp_lepoll eq_lepoll_trans by auto
next
assume \( A \preceq 1 \)
\{ fix \( T \)
assume \( T \text{(is a topology)} \)
then have \( (T \text{(is } T_4\text{)}) \longrightarrow (T \text{(is } T_0\text{)}) \) using topology0.T4_is_T3 topology0.T3_is_T2 T2_is_T1 T1_is_T0 topology0_def by auto
qed
then have \( \forall T. \{ \text{is a topology} \} \rightarrow ((T \{ \text{is } T_4 \}) \rightarrow (T \{ \text{is } T_0 \})) \) by auto
then have \( (A \{ \text{is in the spectrum of} \} \{ \text{is } T_4 \}) \rightarrow (A \{ \text{is in the spectrum of} \} \{ \text{is } T_0 \}) \) by auto

\[ \text{by auto, qed} \]

**Theorem T1_spectrum:**

shows \( (A \{ \text{is in the spectrum of} \} \{ \text{is } T_1 \}) \leftrightarrow A \leq 1 \)
proof-

- note \( T_2 \_\text{is } T_1 \) \( \text{topology0} \). \( T_3 \_\text{is } T_2 \) \( \text{topology0} \). \( T_4 \_\text{is } T_3 \)
then have \( (A \{ \text{is in the spectrum of} \} \{ \text{is } T_4 \}) \rightarrow (A \{ \text{is in the spectrum of} \} \{ \text{is } T_1 \}) \)
  using \( \text{P_imp_Q_spec_inv[of } isT4isT1\text{]} \) \( \text{topology0_def by auto} \)
moreover

- note \( T_1 \_\text{is } T_0 \)
then have \( (A \{ \text{is in the spectrum of} \} \{ \text{is } T_1 \}) \rightarrow (A \{ \text{is in the spectrum of} \} \{ \text{is } T_0 \}) \)
  using \( \text{P_imp_Q_spec_inv[of } isT1isT0\text{]} \) \( \text{by auto} \)
moreover

- note \( T_0 \_\text{spectrum} \) \( T_4 \_\text{spectrum} \)
ultimately show thesis by blast
qed

**Theorem T2_spectrum:**

shows \( (A \{ \text{is in the spectrum of} \} \{ \text{is } T_2 \}) \leftrightarrow A \leq 1 \)
proof-

- note \( \text{topology0} \). \( T_3 \_\text{is } T_2 \) \( \text{topology0} \). \( T_4 \_\text{is } T_3 \)
then have \( (A \{ \text{is in the spectrum of} \} \{ \text{is } T_4 \}) \rightarrow (A \{ \text{is in the spectrum of} \} \{ \text{is } T_2 \}) \)
  using \( \text{P_imp_Q_spec_inv[of } isT4isT2\text{]} \) \( \text{topology0_def by auto} \)
moreover

- note \( T_2 \_\text{is } T_1 \)
then have \( (A \{ \text{is in the spectrum of} \} \{ \text{is } T_2 \}) \rightarrow (A \{ \text{is in the spectrum of} \} \{ \text{is } T_1 \}) \)
  using \( \text{P_imp_Q_spec_inv[of } isT2isT1\text{]} \) \( \text{by auto} \)
moreover

- note \( T_1 \_\text{spectrum} \) \( T_4 \_\text{spectrum} \)
ultimately show thesis by blast
qed

**Theorem T3_spectrum:**

shows \( (A \{ \text{is in the spectrum of} \} \{ \text{is } T_3 \}) \leftrightarrow A \leq 1 \)
proof-

- note \( \text{topology0} \). \( T_4 \_\text{is } T_3 \)
then have \( (A \{ \text{is in the spectrum of} \} \{ \text{is } T_4 \}) \rightarrow (A \{ \text{is in the spectrum of} \} \{ \text{is } T_3 \}) \)

1021
using P_imp_Q_spec_inv[of isT4isT3] topology0_def by auto
moreover
note topology0.T3_is_T2
then have \((A \{\text{is in the spectrum of} \} \text{isT3}) \longrightarrow (A \{\text{is in the spectrum of} \} \text{isT2})\)
using P_imp_Q_spec_inv[of isT3isT2] topology0_def by auto
moreover
note T2_spectrum T4_spectrum
ultimately show thesis by blast
qed

theorem compact_spectrum:
shows \((A \{\text{is in the spectrum of} \} (\lambda T. (\bigcup T) \{\text{is compact in} \} T)) \longleftrightarrow \text{Finite}(A)\)
proof
assume \((A \{\text{is in the spectrum of} \} (\lambda T. (\bigcup T) \{\text{is compact in} \} T))\)
then have reg:\(\forall T. T\{\text{is a topology}\} \land \bigcup T \approx A \longrightarrow ((\bigcup T) \{\text{is compact in} \} T)\) using Spec_def by auto
have \(\text{Pow}(A)\{\text{is a topology}\} \land \bigcup \text{Pow}(A) = A\) using eqpoll_top by auto
then have \(\text{Pow}(A)\{\text{is a topology}\} \land \bigcup \text{Pow}(A) \approx A\) using eqpoll_refl by auto
with reg have \(A\{\text{compact in} \} \text{Pow}(A)\) by auto
moreover
have \(\bigcup \{x\}. x \in A = A\) by auto
ultimately have \(\exists N \in \text{FinPow}(\{x\}. x \in A)\). \(A \subseteq \bigcup N\) using IsCompact_def by auto
then obtain \(N\) where \(N \in \text{FinPow}(\{x\}. x \in A)\) \(A \subseteq \bigcup N\) using FinPow_def by auto
\{fix \(t\)
assume \(t \in \{x\}. x \in A\)
then obtain \(x\) where \(x \in \text{At} = \{x\}\) by auto
with \(A \subseteq \bigcup N\) have \(x \in \bigcup N\) by auto
then obtain \(B\) where \(B \subseteq \text{N} \times B\) by auto
with \(\bigcup \{x\}. x \in A\) have \(B = \{x\}\) by auto
with \(t = \{x\} < B \in N\) have \(t \in N\) by auto\}
with \(\bigcup \\{x\}. x \in A\) have \(N = \{x\}. x \in A\) by auto
with \(\text{Finite}(N)\) have \(\text{Finite}(\{x\}. x \in A)\) by auto
let \(B = \langle x, \{x\} \rangle. x \in A\)
have \(B: A \rightarrow \{x\}. x \in A\) unfolding Pi_def function_def by auto
then have \(B: \text{bij}(A, \{x\}. x \in A)\) unfolding bij_def inj_def surj_def using apply_equality by auto
then have \(A = \{x\}. x \in A\) using eqpoll_def by auto
with \(\text{Finite}(\{x\}. x \in A)\) show \(\text{Finite}(A)\) using eqpoll_imp_Finite_iff by auto
next

assume Finite(A)
{
  fix T assume T{is a topology} \(\bigcup T \approx A\)
  with <Finite(A)> have Finite(\(\bigcup T\)) using eqpoll_imp_Finite_iff by auto
  then have Finite(Pow(\(\bigcup T\))) using Finite_Pow by auto
  moreover have T \(\subseteq\) Pow(\(\bigcup T\)) by auto
  ultimately have Finite(T) using subset_Finite by auto
  
  { fix M
    assume M \(\in\) Pow(T) \(\bigcup T \subseteq \bigcup M\) with <Finite(T)> have Finite(M) using subset_Finite by auto
    with \(<\bigcup T \subseteq \bigcup M\>) have \(\exists N \, \text{FinPow}(M). \bigcup T \subseteq \bigcup N\) using FinPow_def by auto
  }
  then have \((\bigcup T)\{\text{is compact in}\} T\) unfolding IsCompact_def by auto
}
then show A \{\text{is in the spectrum of}\} (\(\lambda T. (\bigcup T)\{\text{is compact in}\} T\)) using Spec_def by auto
qed

It is, at least for some people, surprising that the spectrum of some properties cannot be completely determined in \(ZF\).

theorem compactK_spectrum:
  assumes \{the axiom of\} \(K\{\text{choice holds for subsets}\}(\text{Pow}(K))\) \(\text{Card}(K)\)
  shows \((A \{\text{is in the spectrum of}\} (\lambda T. ((\bigcup T)\{\text{is compact of cardinal}\} csucc(K)\{\text{in}\} T))) \iff (A \preccurlyeq K)\)
proof
  assume A \{\text{is in the spectrum of}\} (\(\lambda T. ((\bigcup T)\{\text{is compact of cardinal}\} csucc(K)\{\text{in}\} T))\)
  then have reg:\(\forall T. T\{\text{is a topology}\} \land \bigcup T \approx A \rightarrow ((\bigcup T)\{\text{is compact of cardinal}\} csucc(K)\{\text{in}\} T)\) using Spec_def by auto
  then have A\{\text{is compact of cardinal}\} csucc(K) \{\text{in}\} Pow(A) using Pow_is_top[of A] by auto
  then have \(\forall M \in\text{Pow(Pow}(A))\). A \(\subseteq\) M \(\rightarrow\) (\(\exists N \in\text{Pow}(M). A \subseteq N \land N \prec c\text{succ}(K)\)) unfolding IsCompactOfCard_def by auto
  moreover have \({\{x\}. x \in A}\) \(\in\) Pow(Pow(A)) by auto
  moreover have A=\(\bigcup\{\{x\}. x \in A\}\) by auto
  ultimately have \(\exists N \in\text{Pow}\{\{x\}. x \in A\}\). A \(\subseteq\) N \(\land\) N \(\prec\) c\text{succ}(K) by auto
  then obtain N where N \(\in\) Pow(\{\{x\}. x \in A\}) A \(\subseteq\) N \(\prec\) c\text{succ}(K) by auto
  then have N\{\{x\}. x \in A\} \(\prec\) c\text{succ}(K) A \(\subseteq\) N using FinPow_def by auto
  
  { fix t
    assume t\{\{x\}. x \in A\}
    then obtain x where x \(\in\) At=\{x\} by auto
    with \(<\subseteq\bigcup N\>) have x \(\in\) N by auto
    
    
    1023
then obtain $B$ where $B \in \mathbb{N} \times B$ by auto
with \langle N \subseteq \{x \}. x \in A \rangle \) have $B = \{x \}$ by auto
with \langle t=x \rangle \). $B \in \mathbb{N}$ by auto
\}
with \langle N \subseteq \{x \}. x \in A \rangle \) have $N = \{x \}. x \in A \}$ by auto
let $B = \{x, \{x \} \}. x \in A \}$
from \langle N = \{x \}. x \in A \rangle \) have $B : A \to N$ unfolding Pi_def function_def by auto
with \langle N = \{x \}. x \in A \rangle \) have $B : \text{inj}(A, N)$ unfolding inj_def using apply_equality
by auto
then have $A \subseteq N$ using lepoll_def by auto
with \langle N < \text{csucc}(K) \rangle \) have $A < \text{csucc}(K)$ using lesspoll_trans1 by auto
then show $A \subseteq K$ using Card_less_csucc_eq_le assms(2) by auto
next
assume $A \subseteq K$
\}
fix $T$
assume $T$ \{is a topology\}$\bigcup T \approx A$
have $\text{Pow}(\bigcup T)$ \{is a topology\} using Pow_is_top by auto
\}
fix $B$
assume $A \subseteq B \in \text{Pow}(\bigcup T)$
then have $\langle \{i \}. i \in B \rangle \subseteq \langle \{i \} . i \in \bigcup T \rangle$ by auto
moreover
have $B = \bigcup \{\{i \}. i \in B \}$ by auto
ultimately have $\exists S \in \text{Pow}(\{\{i \}. i \in \bigcup T \}). B = \bigcup S$ by auto
then have $B \in (\bigcup U. U \in \text{Pow}(\{\{i \}. i \in \bigcup T \}))$ by auto
\}
moreover
\}
fix $B$
assume $A \subseteq B \in (\bigcup U. U \in \text{Pow}(\{\{i \}. i \in \bigcup T \}))$
then have $B \in \text{Pow}(\bigcup T)$ by auto
\}
ultimately
have base: $\{\{x \}. x \in \bigcup T \} \ \{is a base for\} \text{Pow}(\bigcup T)$ unfolding IsAbaseFor_def
by auto
let $f = \langle \{i, \{i \} \}. i \in \bigcup T \rangle$
have $f : f : \bigcup T \to \{\{x \}. x \in \bigcup T \}$ using Pi_def function_def by auto
moreover
\}
fix $w x$
assume $a : w \in \bigcup T x \in \bigcup T w \approx fx$
with $f$ have $f w = \{w \} f x \approx f x$ using apply_equality by auto
with $a(3)$ have $w = x$ by auto
\}
with $f$ have $f : \text{inj}(\bigcup T, \{\{x \}. x \in \bigcup T \})$ unfolding inj_def by auto
moreover
\}
fix $xa$

1024
assume $x \in \{x\}$. $x \in \bigcup T$
then obtain $x$ where $x \in \bigcup T$ $x = x$ by auto
with $f$ have $f x = x a$ using apply_equality by auto
with $\langle x \in \bigcup T \rangle$ have $\exists x \in \bigcup T$. $f x = x a$ by auto
} then have $\forall x a \in \{x\}. x \in \bigcup T$. $f x = x a$ by blast
ultimately have $f : \text{bij}(\bigcup T, \{x\}. x \in \bigcup T)$ unfolding bij_def surj_def
by auto
with base have $\text{Pow}(\bigcup T)$ {is of second type of cardinal}$\text{csucc}(K)$ unfolding IsSecondOfCard_def by auto
moreover have $\bigcup \text{Pow}(\bigcup T) = \bigcup T$ by auto
with calculation assms(1) $\langle \text{Pow}(\bigcup T)$ {is a topology}$\rangle$ have $\langle \text{Pow}(\bigcup T)$ {is compact of cardinal}$\rangle$ $\text{csucc}(K)\{\in\rangle$ $\text{Pow}(\bigcup T)$
using $\text{compact_of_cardinal}_Q[\text{of } \text{K} \text{Pow}(\bigcup T)]$ by auto
moreover have $T \subseteq \text{Pow}(\bigcup T)$ by auto
ultimately have $\bigcup \text{Pow}(\bigcup T)$ {is compact of cardinal}$\text{csucc}(K)\{\in\rangle T$ using
$\text{compact_coarser}$ by auto
} then show $A$ {is in the spectrum of}$\{\lambda T. (\{\bigcup T\} \{is compact of cardinal\}$ $\text{csucc}(K)\{\in\rangle T)$} using $\text{Spec_def}$ by auto
qed

theorem compactK_spectrum_reverse:
assumes $\forall A. (A$ {is in the spectrum of}$\{\lambda T. (\{\bigcup T\} \{is compact of cardinal\}$
$\text{csucc}(K)\{\in\rangle T)$}) $\longleftrightarrow (A \leq K) \text{InfCard}(K)$
shows $\{\text{the axiom of } K \{\text{choice holds for subsets}\} \langle \text{Pow}(K)\}$
proof
have $K \leq K$ using lepoll_refl by auto
then have $K$ {is in the spectrum of}$\{\lambda T. (\{\bigcup T\} \{is compact of cardinal\}$
$\text{csucc}(K)\{\in\rangle T)$} using assms(1) by auto
moreover have $\text{Pow}(K)$ {is a topology} using $\text{Pow_is_top}$ by auto
moreover have $\bigcup \text{Pow}(K) = K$ by auto
then have $\bigcup \text{Pow}(K) = K$ using eqpoll_refl by auto
ultimately have $K$ {is compact of cardinal}$\text{csucc}(K)\{\in\rangle \text{Pow}(K)$ using $\text{Spec_def}$ by auto
then show thesis using $\text{Q_disc_comp_csuccQ_eq_Q_choice_csuccQ}$ assms(2)
by auto

1025
This last theorem states that if one of the forms of the axiom of choice related to this compactness property fails, then the spectrum will be different. Notice that even for Lindelöf spaces that will happen.

The spectrum gives us the possibility to define what an anti-property means. A space is anti-\( P \) if the only subspaces which have the property are the ones in the spectrum of \( P \). This concept tries to put together spaces that are completely opposite to spaces where \( P(T) \).

**Definition**

\[
\text{antiProperty (\{-is anti-\} 50)}
\]

where \( T\text{\{-is anti-\}P} \equiv \forall \mathcal{A} \in \text{Pow}(\bigcup T). \ P(T\text{\{restricted to\}}A) \rightarrow (\mathcal{A} \text{\{is in the spectrum of\} P})
\]

**Abbreviation**

\[
\text{ANTI}(P) \equiv \forall T. \ (T\text{\{-is anti-\}P})
\]

A first, very simple, but very useful result is the following: when the properties are related and the spectra are equal, then the anti-properties are related in the opposite direction.

**Theorem (in topology0)**

\[
\text{eq_spect_rev_imp_anti:}
\]

**Proof**

\[
\begin{align*}
\text{fix } A \\
\text{assume } A \in \text{Pow}(\bigcup T) & P(T\text{\{restricted to\}}A) \\
\text{with } \text{assms(1)} & Q(T\text{\{restricted to\}}A) \text{ using Top_1_L4 by auto} \\
\text{with } \text{assms(3)} & \forall \mathcal{A} \in \text{Pow}(\bigcup T). \ A \text{\{is in the spectrum of\} Q} \text{ using antiProperty_def by auto} \\
\text{by auto} \\
\text{with } \text{assms(2)} & A \text{\{is in the spectrum of\} P} \text{ by auto} \\
\text{then show thesis using antiProperty_def by auto}
\end{align*}
\]

**Qed**

If a space can be \( P(T) \land Q(T) \) only in case the underlying set is in the spectrum of \( P \); then \( Q(T) \rightarrow \text{ANTI}(P,T) \) when \( Q \) is hereditary.

**Theorem Q_P_imp_Spec:**

**Proof**

\[
\begin{align*}
\text{fix } T \\
\end{align*}
\]

1026
assume $T$ is a topology
{
assume $Q(T)$
{
assume $\neg (T$ is anti-$P)$
then obtain $A$ where $A \in \text{Pow}(\bigcup T)$ $P(T\text{restricted to} A) \neg (A$ is in the spectrum of $P)$
unfolding antiProperty_def by auto
from $\langle Q(T)\rangle$ $\langle T$ is a topology$\rangle$ $\langle A \in \text{Pow}(\bigcup T)$ assms(2) have $Q(T\text{restricted to} A)$
unfolding IsHer_def by auto
moreover
note $\langle P(T\text{restricted to} A)\rangle$ assms(1)
moreover
from $\langle T$ is a topology$\rangle$ have $(T\text{restricted to} A)$ is a topology
using topology0.Top_1_L4
 topology0_def by auto
moreover
from $\langle A \in \text{Pow}(\bigcup T)\rangle$ have $\bigcup (T\text{restricted to} A)=$ $A$ unfolding RestrictedTo_def
by auto
ultimately have $A$ is in the spectrum of $P$ by auto
with $\neg (A$ is in the spectrum of $P)$ have False by auto
}
then have $T$ is anti-$P$ by auto
}
then have $Q(T) \rightarrow (T$ is anti-$P)$ by auto
} then show $(T$ is a topology$) \rightarrow (Q(T) \rightarrow (T$ is anti-$P))$ by auto
qed

If a topological space has an hereditary property, then it has its double-anti property.

**Theorem (in topology0)** her_P_imp_anti2P:
assumes $P$ is hereditary $P(T)$
shows $T$ is anti-$\text{ANTI}(P)$
proof-
{
assume $\neg (T$ is anti-$\text{ANTI}(P))$
then have $\exists A \in \text{Pow}(\bigcup T)$. $(T\text{restricted to} A)$ is anti-$P)$ $\neg (A$ is in the spectrum of $\text{ANTI}(P)$)
unfolding antiProperty_def[of _ $\text{ANTI}(P)$] by auto
then obtain $A$ where $A$ def: $A \in \text{Pow}(\bigcup T)$. $(A$ is in the spectrum of $\text{ANTI}(P))$ $(T\text{restricted to} A)$ is anti-$P$
by auto
from $\langle A \in \text{Pow}(\bigcup T)\rangle$ have tot:$\cup (T\text{restricted to} A)=$ $A$ unfolding RestrictedTo_def
by auto
from $A$ def have $\forall B \in \text{Pow}(\bigcup (T\text{restricted to} A))$. $P((T\text{restricted to} A)\text{restricted to} B) \rightarrow (B$ is in the spectrum of $P)$
unfolding antiProperty_def by auto

1027
have \( \forall B \in \text{Pow}(A). (T\{\text{restricted to} A\}\{\text{restricted to} B\} = T\{\text{restricted to} B\} \) using \text{subspace_of_subspace} \( \langle A \in \text{Pow}(\bigcup T) \rangle \) by auto

then have \( \forall B \in \text{Pow}(A). P(T\{\text{restricted to} B\}) \rightarrow (B\{\text{is in the spectrum of} P\}) \) using \text{reg tot}

by force

moreover

have \( \forall B \in \text{Pow}(A). P(T\{\text{restricted to} B\}) \) using \text{subms} \( \langle A \in \text{Pow}(\bigcup T) \rangle \) unfolding \text{IsHer_def} using \text{topSpaceAssum} by blast

ultimately have \( \text{reg2} : \forall B \in \text{Pow}(A). (B\{\text{is in the spectrum of} P\}) \) have \( \exists T. T\{\text{is a topology}\} \)

\( \wedge \bigcup T = A \wedge \neg (T\{\text{is anti-} P\}) \)

unfolding \text{Spec_def} by auto

then obtain \( S \) where \( S\{\text{is a topology}\} \bigcup S = A \neg (S\{\text{is anti-} P\}) \) by auto

from \( \neg (S\{\text{is anti-} P\}) \) have \( \exists B \in \text{Pow}(S)\). \( P(S\{\text{restricted to} B\}) \wedge \neg (B\{\text{is in the spectrum of} P\}) \)

unfolding \text{antiProperty_def} by auto

then obtain \( B \) where \( B\{\text{is in the spectrum of} P\} \) \( B \in \text{Pow}(\bigcup S) \)

by auto

then have \( B \subseteq \bigcup S \) using \text{subset_imp_lepoll} by auto

with \( \bigcup S = A \) have \( B \subseteq A \) using \text{lepoll_eq_trans} by auto

then obtain \( f \) where \( f \in \text{inj}(B, A) \) unfolding \text{lepoll_def} by auto

then have \( f \in \text{bij}(B, \text{range}(f)) \) using \text{inj_bij_range} by auto

then have \( B = \text{range}(f) \) unfolding \text{eqpoll_def} by auto

with \( B\{\text{def} : \neg (B\{\text{is in the spectrum of} P\}) \} \) using \text{eqpoll_iff_spec} by auto

moreover

with \( \langle f \in \text{inj}(B, A) \rangle \) have \( \text{range}(f) \subseteq A \) unfolding \text{inj_def Pi_def} by auto

with \( \text{reg2} \) have \( \text{range}(f)\{\text{is in the spectrum of} P\} \) by auto

ultimately have \( \text{False} \) by auto

} then show thesis by auto

qed

The anti-properties are always hereditary

\text{theorem anti_here:}

shows \( \text{ANTI}(P)\{\text{is hereditary}\} \)

\text{proof-}

\{\n  \text{fix } T  \\
  \text{assumption } T\{\text{is a topology}\}\text{ANTI}(P, T)  \\
  \{  \\
    \text{fix } A  \\
    \text{assume } A \in \text{Pow}(\bigcup T)  \\
    \text{then have } \bigcup (T\{\text{restricted to} A\}) = A \text{ unfolding } \text{RestrictedTo_def} by auto  \\
  \}  \\
  \text{moreover}  \\
  \{  \\
    \text{fix } B  \\
    \text{assume } B \in \text{Pow}(A)\{\langle T\{\text{restricted to} A\}\rangle\{\text{restricted to} B\} \) with \( \langle A \in \text{Pow}(\bigcup T) \rangle \) have \( B \in \text{Pow}(\bigcup T)\{T\{\text{restricted to} B\}) \) using \text{subspace_of_subspace}
\}
by auto
  with \langle \text{ANTI}(P,T) \rangle \text{ have } B \{\text{is in the spectrum of} \} P \text{ unfolding antiProperty_def }

by auto
} 
ultimately have \forall B \in \text{Pow}(\bigcup (T \{\text{restricted to} \} A)). (P((T \{\text{restricted to} \} A) \{\text{restricted to} \} B)) \rightarrow (B \{\text{is in the spectrum of} \} P) 
by auto
then have \text{ANTI}(P,(T \{\text{restricted to} \} A)) \text{ unfolding antiProperty_def }
by auto
} 
then have \forall A \in \text{Pow} (\bigcup T). \text{ANTI}(P,(T \{\text{restricted to} \} A)) \text{ by auto }
}
then show thesis using IsHer_def by auto
qed

corollary (in topology0) anti_imp_anti3:
 assumes T\{\text{is anti-} \} P
 shows T\{\text{is anti-} \} \text{ANTI}(\text{ANTI}(P))
 using anti_here her_P_imp_anti2P assms by auto

In the article [5], we can find some results on anti-properties.

theorem (in topology0) anti_T0:
 shows (T\{\text{is anti-} \} \text{isT0} ) \iff T = \{0, \bigcup T\}
proof
 assume T = \{0, \bigcup T\}
 { 
  fix A
  assume A \in \text{Pow}(\bigcup T)(T \{\text{restricted to} \} A) \{\text{is T}_0\}
  
  fix B
  assume B \in T \{\text{restricted to} \} A
  then obtain S where S \in T \text{ and } B = A \cap S \text{ unfolding RestrictedTo_def by auto }
  with \langle T = \{0, \bigcup T\} \rangle \text{ have } S \subseteq \{0, \bigcup T\} \text{ by auto }
  then have S \cap S = \bigcup T \text{ by auto }
  with \langle B = A \cap S \rangle \langle A \in \text{Pow}(\bigcup T) \rangle \text{ have } B = 0 \cup B = A \text{ by auto }
 } 
moreover
 { 
  have 0 \in \{0, \bigcup T\} \cup T \subseteq \{0, \bigcup T\} \text{ by auto }
  with \langle T = \{0, \bigcup T\} \rangle \text{ have } 0 \in T(\bigcup T) \subseteq T \text{ by auto }
  then have A \cap 0 \in (T \{\text{restricted to} \} A) \ A \cap (\bigcup T) \in (T \{\text{restricted to} \} A) 
  using RestrictedTo_def by auto
  moreover
  from \langle A \in \text{Pow}(\bigcup T) \rangle \text{ have } A \cap (\bigcup T) = A \text{ by auto }
  ultimately have \emptyset \in (T \{\text{restricted to} \} A) \ A \in (T \{\text{restricted to} \} A) \text{ by auto }
  } 
ultimately have (T \{\text{restricted to} \} A) = \{0, A\} \text{ by auto }

1029
with $\langle T\{\text{restricted to}\}A \rangle \{is \ T_0\}> \{0,A\} \{is \ T_0\}$ by auto

assume $A\neq 0$
then obtain $x$ where $x\in A$ by blast

fix $y$
assume $y\in A \neq y$
with $\langle 0,A \rangle \{is \ T_0\}>$ obtain $U$ where $U\in \{0,A\}$ and $\text{dis}:(x \in U \land y \notin U) \lor (y \in U \land x \notin U)$ using $\text{isT0}_{\text{def}}$ by auto
then have $U=\emptyset$ by auto
with $\text{dis} \langle y\in A \rangle \langle x\in A \rangle$ have False by auto

then have $\forall y\in A. \ y=x$ by auto
with $\langle x\in A \rangle$ have $A=\{x\}$ by blast
then have $A\approx 1$ using $\text{singleton_eqpoll}_{\text{I}}$ by auto
then have $A\{\text{is in the spectrum of}\} \text{isT0}$ using $T_0\_\text{specrum}$ by auto

moreover

assume $A=\emptyset$
then have $A\approx 0$ by auto
then have $A\leq 1$ using $\text{empty_lepoll}_{\text{I}}$ by auto
then have $A\{\text{is in the spectrum of}\} \text{isT0}$ using $T_0\_\text{specrum}$ by auto

ultimately have $A\{\text{is in the spectrum of}\} \text{isT0}$ by auto

then show $T\{\text{is anti-}\} \text{isT0}$ using $\text{antiProperty}_{\text{def}}$ by auto

next
assume $T\{\text{is anti-}\} \text{isT0}$
then have $\forall A\in \text{pow}(\bigcup T). \ (T\{\text{restricted to}\}A \{is \ T_0\}) \rightarrow (A\{\text{is in the spectrum of}\} \text{isT0})$ using $\text{antiProperty}_{\text{def}}$ by auto
then have $\text{reg}: \forall A\in \text{pow}(\bigcup T). \ (T\{\text{restricted to}\}A \{is \ T_0\}) \rightarrow (A\leq 1)$ using $T_0\_\text{specrum}$ by auto

assume $\exists A\in T. \ A\neq 0 \land A\neq \bigcup T$
then obtain $A$ where $A\in T\neq 0A\neq \bigcup T$ by auto
then obtain $x \ y$ where $x\in A \ y\in \bigcup T-A$ by blast
with $\langle A\in T \rangle$ have $s:\{x,y\}\in \text{pow}(\bigcup T) \ x\neq y$ by auto
note $s$
moreover

fix $b_1 \ b_2$
assume $b_1\in \bigcup (T\{\text{restricted to}\}\{x,y\}) \ b_2\in \bigcup (T\{\text{restricted to}\}\{x,y\}) b_1 \neq b_2$
moreover
from $s$ have $\bigcup (T\{\text{restricted to}\}\{x,y\}) = \{x,y\}$ unfolding $\text{RestrictedTo}_{\text{def}}$ by auto
ultimately have $(b_1=x \land b_2=y) \lor (b_1=y \land b_2=x)$ by auto
with \(<x \neq y> have (b1 \in \{x\}\wedge b2 \notin \{x\}) \lor (b2 \in \{x\}\wedge b1 \notin \{x\}) by auto
moreover
from \(<y \in \bigcup T-A> have \{x\} = \{x, y\}\cap A by auto
with \(<A \in T> have \{x\} \in (T\{\text{restricted to}\} \{x, y\}) unfolding \text{RestrictedTo_def by auto}
ultimately have \exists U \in (T\{\text{restricted to}\} \{x, y\}). (b1 \in U \wedge b2 \notin U) \lor (b2 \in U \wedge b1 \notin U) by auto
ultimately have \exists U \in T\{\text{restricted to}\} \{x, y\}. (b1 \in U \wedge b2 \notin U) \lor (b2 \in U \wedge b1 \notin U)
moreover
from topSpaceAssum have 0 \in T \bigcup T \in T using \text{IsATopology_def empty_open by auto}
ultimately show T = \{0, \bigcup T\} by auto
qed

lemma indiscrete_spectrum:
shows (A \{\text{is in the spectrum of}\}(\lambda T. T = \{0, \bigcup T\})) \iff A \approx 1
proof
assume (A \{\text{is in the spectrum of}\}(\lambda T. T = \{0, \bigcup T\}))
then have reg:\(\forall T. ((T\{\text{is a topology}\} \wedge \bigcup T \approx A) \longrightarrow T = \{0, \bigcup T\}) using Spec_def by auto
moreover
have \bigcup \text{Pow}(A) = A by auto
then have \bigcup \text{Pow}(A) = A by auto
moreover
have \text{Pow}(A) \{\text{is a topology}\} using \text{Pow_is_top by auto
ultimately have P:\text{Pow}(A) = \{0, A\} by auto
\{ assume A \neq 0
then obtain x where x \in A by blast
then have \{x\} \in \text{Pow}(A) by auto
with P have \{x\} = A by auto
then have A \approx 1 using singleton_eqpoll_1 by auto
then have A \approx 1 using eqpoll_imp_lepoll by auto
\}
moreover
\{ assume A = 0

then have $A \approx 0$ by auto
then have $A \ll 1$ using empty_lepollI eq_lepoll_trans by auto
}
ultimately show $A \ll 1$ by auto
next
assume $A \ll 1$
{
fix $T$
assume $T\{\text{is a topology}\} \cup T \approx A$
{
assume $A = 0$
with $\langle \cup T \approx A \rangle$ have $\cup T \approx 0$ by auto
then have $\cup T = 0$ using eqpoll_0_is_0 by auto
then have $T \subseteq \{0\}$ by auto
with $\langle T\{\text{is a topology}\} \rangle$ have $T = \{0\}$ using empty_open by auto
then have $T = \{0, \cup T\}$ by auto
}
moreover
{
assume $A \not= 0$
then obtain $E$ where $E \in A$ by blast
with $\langle A \ll 1 \rangle$ have $A = \{E\}$ using lepoll_1_is_sing by auto
then have $A \approx 1$ using singleton_eqpoll_1 by auto
with $\langle \cup T \approx A \rangle$ have NONempty:$\cup T \approx 1$ using eqpoll_trans by blast
then have $\cup T \ll 1$ using eqpoll_imp_lepoll by auto
moreover
{
assume $\cup T = 0$
then have $0 \approx \cup T$ by auto
with NONempty have $0 \ll 1$ using eqpoll_trans by blast
then have $0 = 1$ using eqpoll_0_is_0 eqpoll_sym by auto
then have False by auto
}
then have $\cup T \not= 0$ by auto
then obtain $R$ where $R \in \cup T$ by blast
ultimately have $\cup T = \{R\}$ using lepoll_1_is_sing by auto
moreover
have $T \subseteq \text{Pow}(\cup T)$ by auto
ultimately have $T \subseteq \text{Pow}(\{R\})$ by auto
then have $T \subseteq \{0, \{R\}\}$ by blast
moreover
with $\langle T\{\text{is a topology}\} \rangle$ have $0 \in \cup T \in T$ using IsATopology_def by auto
moreover
note $\langle \cup T = \{R\} \rangle$
ultimately have $T = \{0, \cup T\}$ by auto
}
ultimately have $T = \{0, \cup T\}$ by auto
}
then show \( A \) \{is in the spectrum of\}(\( \lambda T. T = \{0, \bigcup T\} \)) using Spec_def by auto
qed

theorem (in topology0) anti_indiscrete:
shows (\( T \{is anti-\}(\( \lambda T. T = \{0, \bigcup T\} \)) \iff T \{is T_0\})
proof
assume \( T \{is T_0\} \)
{
  fix \( A \)
  assume \( A \in \text{Pow}(\bigcup T) T\{restricted to\}A = \{0, \bigcup (T\{restricted to\}A)\} \)
  then have \( \bigcup (T\{restricted to\}A) = A \{restricted to\}A = \{0, A\} \) using RestrictedTo_def by auto
  from \( T \{is T_0\} \)
  have \( (T\{restricted to\}A) \{is T_0\} \) unfolding isT0_def by auto
  finally have \( A \approx 1 \) using singleton_eqpoll_1 by auto
}
moreover
{
  assume \( A \neq 0 \)
  then obtain \( E \) where \( E \in A \) by blast
  {
    fix \( y \) assume \( y \in A \neq E \)
    with \( E \in A \) have \( y \in \bigcup (T\{restricted to\}A) \) E \in \bigcup (T\{restricted to\}A) \)
    by auto
    with \( \bigcup (T\{restricted to\}A) \{is T_0\} \)
    have \( \exists U \in (T\{restricted to\}A) (E \in U \lor y \notin U) \lor (E \notin U \lor y \in U) \)
    unfolding isT0_def by blast
    then obtain \( U \) where \( U \in (T\{restricted to\}A) \ (E \in U \lor y \in U) \lor (E \notin U \lor y \notin U) \)
    by auto
    with \( \bigcup (T\{restricted to\}A) \{is T_0\} \)
    have \( U = 0 \lor U = A \) by auto
    with \( E \in U \lor y \notin U \lor y \in U \) have False by auto
  }
  then have \( \forall y \in A. y = E \) by auto
  with \( E \in A \) have \( A = \{E\} \) by blast
  then have \( A \approx 1 \) using singleton_eqpoll_1 by auto
  then have \( A \subseteq 1 \) using eqpoll_imp_lepoll by auto
}
ultimately have \( A \subseteq 1 \) by auto
then have \( A \{is in the spectrum of\}(\( \lambda T. T = \{0, \bigcup T\} \)) \) using indiscrete_spectrum
by auto

then show \( T \{is anti-\}(\( \lambda T. T = \{0, \bigcup T\} \)) \) unfolding antiProperty_def by auto
next
assume \(T\{\text{is anti-}\}(\lambda T. T=\{0,1\})\) then have \(\forall A \in \mathbb{P}(\bigcup T). (T\{\text{restricted to}A\}=\{0,1\}(T\{\text{restricted to}A\})\)\) using antiProperty_def
by auto
then have \(\forall A \in \mathbb{P}(\bigcup T). (T\{\text{restricted to}A\}=\{0,1\}(T\{\text{restricted to}A\})\)\)\) using indiscrete_spectrum
by auto
moreover have \(\forall A \in \mathbb{P}(\bigcup T). (T\{\text{restricted to}A\}=\{0,1\}(T\{\text{restricted to}A\})\)\)\) using indiscrete_spectrum
by auto
ultimately have \(\forall A \in \mathbb{P}(\bigcup T). (T\{\text{restricted to}A\}=\{0,1\} A\)\)\) using indiscrete_spectrum
by auto

The conclusion is that being \(T_0\) is just the opposite to being indiscrete.

Next, let’s compute the anti-\(T_i\) for \(i = 1, 2, 3\) or 4. Surprisingly, they are all the same. Meaning, that the total negation of \(T_1\) is enough to negate all of these axioms.
theorem anti_T1:
  shows (T{is anti-}isT1) \iff (IsLinOrder(T,\{(U,V)\in Pow(\bigcup T)\times Pow(\bigcup T). U\subseteq V\}))
proof
  assume T{is anti-}isT1
  let r=\{(U,V)\in Pow(\bigcup T)\times Pow(\bigcup T). U\subseteq V\}
  have antisym(r) unfolding antisym_def by auto
  moreover
  have trans(r) unfolding trans_def by auto
  moreover
  { fix A B
    assume A\in T B\in T
    { assume ¬(A\subseteq B\lor B\subseteq A)
      then have A-B\neq 0B-A\neq 0 by auto
      then obtain x y where x\in A\notin B\notin A x\neq y by blast
      then have (x,y)\subseteq A={x}\subseteq y\in B by auto
      moreover
      from \langle A\in T\rangle \langle B\in T\rangle have {x,y}\subseteq A\in T\{restricted to\} \{x,y\}\subseteq B\in T\{restricted to\} \{x,y\} unfolding RestrictedTo_def by auto
      ultimately have open_set: \{x\}\subseteq T\{restricted to\} \{x,y\} \{y\}\subseteq T \{restricted to\} \{x,y\} by auto
      have x\in \bigcup T\subseteq \bigcup T using \langle A\in T\rangle \langle B\in T\rangle \langle x\in A\rangle \langle y\in B\rangle by auto
      then have sub: \{x,y\}\subseteq Pow(\bigcup T) by auto
      then have tot: \bigcup \{T\{restricted to\} \{x,y\}\}={x,y} unfolding RestrictedTo_def by auto
      { fix s t
        assume s\subseteq \bigcup \{T\{restricted to\} \{x,y\}\} \{x\}\subseteq s \subseteq \bigcup \{T\{restricted to\} \{x,y\}\} \{s\}\neq t
        with tot have s\subseteq \bigcup \{x,y\} \subseteq \{x\} \subseteq s \subseteq \bigcup \{y\} \subseteq \{s\}\neq t by auto
        then have (s=x\cap t=y) \lor (s=y\cap t=x) by auto
        with open_set have \exists U\in \{T\{restricted to\} \{x,y\}\} s\subseteq U \subseteq t\neq U using \langle x\neq y\rangle by auto
      } then have \{T\{restricted to\} \{x,y\}\} \{is T1\} unfolding isT1_def by auto
    } with sub \langle T\{is anti-\}isT1\rangle tot have \{x,y\} \{is in the spectrum of\}isT1 using antiProperty_def by auto
    then have \{x,y\}\subseteq 1 using T1_spectrum by auto
    moreover
    have x\in \{x,y\} by auto
    ultimately have \{x\}=\{x,y\} using lepoll_1_is_sing[of \{x,y\}x] by auto
  } moreover
  have y\in \{x,y\} by auto
  ultimately
  have y\in \{x\} by auto

1035
then have $x=y$ by auto
then have False using $\langle x \in A, y \notin A \rangle$ by auto
\)
then have $A \subseteq B \lor B \subseteq A$ by auto
\)
then have $r$ (is total on) $T$ using IsTotal_def by auto
ultimately
show IsLinOrder($T, r$) using IsLinOrder_def by auto
next
assume IsLinOrder($T, \{U, V\} \in \text{Pow}(\bigcup T) \times \text{Pow}(\bigcup T) \land U \subseteq V$)
then have ordTot: $\forall S \in T, \forall B \in T. S \subseteq B \lor B \subseteq S$ unfolding IsLinOrder_def IsTotal_def by auto
\{ 
  fix $A$
  assume $A \subseteq \text{Pow}(\bigcup T)$ and $T_1: (T\{\text{restricted to} A\})$ (is $T_1$)
  then have tot: $\bigcup (T\{\text{restricted to} A\}) = A$ unfolding RestrictedTo_def by auto
  \{ 
    fix $U$ $V$
    assume $U \in T\{\text{restricted to} A\}$ $V \in T\{\text{restricted to} A\}$
    then obtain $A \cup V$ where $A \cup V = A \cap A \cup V \cap A$ unfolding RestrictedTo_def by auto
    with ordTot have $U \subseteq V \lor V \subseteq U$ by auto
  \}
  then have ordTotSub: $\forall S \in T\{\text{restricted to} A\}. \forall B \in T\{\text{restricted to} A\}. S \subseteq B \lor B \subseteq S$ by auto
  \{ 
    assume $A = 0$
    then have $A = 0$ by auto
    moreover
    have $0 \subseteq 1$ using empty_lepollI by auto
    ultimately have $A \subseteq 1$ using eq_lepoll_trans by auto
    then have $A$ (is in the spectrum of) $\text{is} T_1$ using $T_1$_spectrum by auto
  \}
  moreover
  \{ 
    assume $A \neq 0$
    then obtain $t$ where $t \in A$ by blast
    \{ 
      fix $y$
      assume $y \in A \land t$
      with $<t \in A>$ tot $T_1$ obtain $U$ where $U \in (T\{\text{restricted to} A\}) y \in U \land U$
      unfolding is$T_1$_def by auto
      from $<y \neq t>$ have $t \neq y$ by auto
      with $<y \in A, t \in A>$ tot $T_1$ obtain $V$ where $V \in (T\{\text{restricted to} A\}) t \in V \land V$
      unfolding is$T_1$_def by auto
      with $<y \in U, t \in U>$ have $\neg (U \subseteq V \subseteq U)$ by auto
\}
with ordTotSub \langle U \in (T \{\text{restricted to}\} A) \rangle \langle V \in (T \{\text{restricted to}\} A) \rangle
have False by auto
}
then have \forall y \in A. y=t by auto
with \langle t \in A \rangle have A = \{t\} by blast
then have A \equiv 1 using singleton_eqpoll_1 by auto
then have A \approx 1 using eqpoll_imp_lepoll by auto
then have A \{is in the spectrum of\} isT1 using T1_spectrum by auto
ultimately
have A \{is in the spectrum of\} isT1 by auto
then show T \{is anti-\} isT1 using antiProperty_def by auto
qed

corollary linordtop_here:
shows (\lambda T. IsLinOrder(T,\{(U,V) \in \Pow(\bigcup T) \times \Pow(\bigcup T). U \subseteq V\}) \{\text{hereditary}\}
using anti_T1 anti_here[of isT1] by auto

theorem (in topology0) anti_T4:
shows (T \{is anti-\} isT4) \iff (IsLinOrder(T,\{(U,V) \in \Pow(\bigcup T) \times \Pow(\bigcup T). U \subseteq V\})
proof
assume T \{is anti-\} isT4
let r = \{(U,V) \in \Pow(\bigcup T) \times \Pow(\bigcup T). U \subseteq V\}
have antisym(r) unfolding antisym_def by auto
moreover
have trans(r) unfolding trans_def by auto
moreover
{ fix A B
assume A \in T B \in T
{ assume \neg(\exists B. B \subseteq A) then have A-B \neq 0 \land A \neq 0 by auto
then obtain x y where x \in A \land x \in B \land y \neq A \land y \neq B by blast
then have \{x,y\} \cap A = \{x\} \{x,y\} \cap B = \{y\} by auto
moreover
from \langle A \in T \rangle \langle B \in T \rangle have \{x,y\} \cap A \{\text{restricted to}\} \{x,y\} \cap B \{\text{restricted to}\} \{x,y\} unfolding
RestrictedTo_def by auto
ultimately have open_set: \{x\} \in T \{\text{restricted to}\} \{x,y\} \{y\} \in T \{\text{restricted to}\} \{x,y\} by auto
have x \in \bigcup T \land y \in T using \langle A \in T \rangle \langle B \in T \rangle \langle x \in A \rangle \langle y \in B \rangle by auto
then have sub: \{x,y\} \in \Pow(\bigcup T) by auto
then have tot: \bigcup (T \{\text{restricted to}\} \{x,y\}) = \{x,y\} unfolding RestrictedTo_def by auto
{ fix s t
}
1037
assume $s \in \bigcup (T \{\text{restricted to}\}{x,y}) t \in \bigcup (T \{\text{restricted to}\}{x,y})$ $s \neq t$
with tot have $s \in \{x,y\} t \in \{x,y\} s \neq t$ by auto
then have $(s=x \land t=y) \lor (s=y \land t=x)$ by auto
with open_set have $\exists U \in (T \{\text{restricted to}\}{x,y}) . s \in U \land t \notin U$ using $\langle x \neq y \rangle$ by auto
}
then have $(T \{\text{restricted to}\}{x,y}) \{\text{is T}_1\}$ unfolding isT1_def by auto
moreover

\begin{verbatim}
  \{ fix s
  assume AS:s\{is closed in\}(T\{\text{restricted to}\}{x,y})
  \{ fix t
  assume AS2:t\{is closed in\}(T\{\text{restricted to}\}{x,y}) s \cap t = 0
  have $(T \{\text{restricted to}\}{x,y}) \{\text{is a topology}\}$ using Top_1_L4 by auto
  with tot have $0 \in (T \{\text{restricted to}\}{x,y}) \{x,y\} \in (T \{\text{restricted to}\}{x,y})$
  using empty_open
  union_open[where $A = T \{\text{restricted to}\}{x,y}$] by auto
  moreover
  note open_set
  moreover
  have $T \{\text{restricted to}\}{x,y} \subseteq \text{Pow}(\bigcup (T \{\text{restricted to}\}{x,y}))$ by blast
  with tot have $T \{\text{restricted to}\}{x,y} \subseteq \text{Pow}(\{x,y\})$ by auto
  ultimately have $T \{\text{restricted to}\}{x,y} \subseteq \text{Pow}(\{x\})$ by blast
  moreover have $P: T \{\text{restricted to}\}{x,y} = \text{Pow}(\{x\})$ by simp
  with tot have $\{A \in \text{Pow}(\{x,y\}). A \{\text{is closed in}\}(T \{\text{restricted to}\}{x,y})\} = \{A \in \text{Pow}(\{x\}). A \subseteq \{x\} \land \{x\} - A \in \text{Pow}(\{x\})\}$ using IsClosed_def by simp
  with P have $S: \{A \in \text{Pow}(\{x,y\}). A \{\text{is closed in}\}(T \{\text{restricted to}\}{x,y})\} = T \{\text{restricted to}\}{x,y}$ by auto
  from AS AS2(1) have $s \in \text{Pow}(\{x,y\}) \ t \in \text{Pow}(\{x,y\})$ using IsClosed_def
  tot by auto
  moreover
  note AS2(1) AS
  ultimately have $s \in \text{Pow}(\{x,y\})$ by auto
  moreover have $A \in \text{Pow}(\{x\})$ by auto
  with $S$ AS2(2) have $s \in T \{\text{restricted to}\}{x,y}$ $t \in T \{\text{restricted to}\}{x,y} s \cap t = 0$
  by auto
  then have $\exists U \in T \{\text{restricted to}\}{x,y} . \exists V \in T \{\text{restricted to}\}{x,y} . s \subseteq U \land t \subseteq V \land V \cap W = 0$ by auto
  \}
then have $\forall t. t \{\text{is closed in}\}(T \{\text{restricted to}\}{x,y}) \land s \cap t = 0$ $\rightarrow$ $(\exists U \in T \{\text{restricted to}\}{x,y} . \exists V \in T \{\text{restricted to}\}{x,y} . s \subseteq U \land t \subseteq V \land V \cap W = 0)$ by auto
\end{verbatim}

1038
then have \( \forall s. s \text{ is closed in} (T \text{ restricted to} \{x,y\}) \rightarrow (\forall t. t \text{ is closed in} (T \text{ restricted to} \{x,y\}) \land s \cap t = 0 \rightarrow (\exists U \in (T \text{ restricted to} \{x,y\}). s \subseteq U \land \forall V \subseteq U. V = 0)) \)
by auto
then have \((T \text{ restricted to} \{x,y\}) \text{ is normal}\) using IsNormal_def
by auto
ultimately have \((T \text{ restricted to} \{x,y\}) \text{ is T}_4\) using isT4_def
by auto
with sub \(\langle T \text{ is anti-}\rangle \text{isT4} \) tot have \(\{x,y\} \text{ is in the spectrum of} \)isT4
using antiProperty_def
by auto
then have \((x,y) \leq 1\) using T4_spectrum by auto
moreover have \(x \in \{x,y\}\) by auto
ultimately have \(\{x\} = \{x,y\}\) using lepoll_1_is_sing[of \(\{x,y\}\)\(x\)] by auto
moreover have \(y \in \{x,y\}\) by auto
ultimately have \(y \in \{x\}\) by auto
then have \(x = y\) by auto
then have \(\text{False}\) using \(\langle x \in A \rangle \langle y \notin A \rangle\) by auto
\}
then have \(A \subseteq B \lor B \subseteq A\) by auto
\}
then have \(r \text{ is total on} T\) using IsTotal_def by auto
ultimately show \(\text{IsLinOrder}(T, r)\) using IsLinOrder_def by auto
next
assume \(\text{IsLinOrder}(T, \{\langle U,V \rangle \in \text{Pow}(\bigcup T) \times \text{Pow}(\bigcup T). U \subseteq V\})\)
then have \(T \text{ is anti-}\)isT1 using anti_T1
by auto
moreover have \(\forall T. T \text{ is a topology} \rightarrow (T \text{ is T}_4) \rightarrow (T \text{ is T}_1)\) using topology0.T4_is_T3 topology0.T3_is_T2 T2_is_T1 topology0_def by auto
moreover have \(\forall A. (A \text{ is in the spectrum of} \)isT1) \rightarrow (A \text{ is in the spectrum of} \)isT4) using T4_spectrum
by auto
ultimately show \(T \text{ is anti-}\)isT4 using eq_spect_rev_imp_anti[of \(\text{isT4isT1}\)]
by auto
qed

theorem (in topology0) anti_T3:
shows \(\langle T \text{ is anti-}\rangle\text{isT3} \leftrightarrow (\text{IsLinOrder}(T, \{\langle U,V \rangle \in \text{Pow}(\bigcup T) \times \text{Pow}(\bigcup T). U \subseteq V\})\)
proof
assume \(T \text{ is anti-}\)isT3
moreover
have $\forall T. (\text{is a topology}) \implies (\text{is } T_4) \implies (\text{is } T_3)$ using topology0.T4_is_T3

    topology0_def by auto
moreover have $\forall A. (A \text{ is in the spectrum of } isT3) \implies (A \text{ is in the spectrum of } isT4)$ using T3_spectrum T4_spectrum
    by auto
ultimately have $T(\text{is anti-} isT4) \implies \text{eq_spect_rev_imp_anti[of isT4isT3]}$
    by auto
then show IsLinOrder($T,\{(U,V)\in Pow(\bigcup T)\times Pow(\bigcup T). U\subseteq V\}$) using anti_T4
by auto

next
assume IsLinOrder($T,\{(U,V)\in Pow(\bigcup T)\times Pow(\bigcup T). U\subseteq V\}$)
then have $T(\text{is anti-} isT1) \implies \text{anti_T1}$
by auto
moreover have $\forall T. (\text{is a topology}) \implies (\text{is } T_2) \implies (\text{is } T_1)$ using
    topology0.T3_is_T2 T2_is_T1 topology0_def
by auto
moreover have $\forall A. (A \text{ is in the spectrum of } isT2) \implies (A \text{ is in the spectrum of } isT3)$ using T1_spectrum T3_spectrum
    by auto
ultimately show $T(\text{is anti-} isT3) \implies \text{eq_spect_rev_imp_anti[of isT3isT1]}$
by auto

qed

theorem (in topology0) anti_T2:
    shows $(T(\text{is anti-} isT2) \iff \text{IsLinOrder}(T,\{(U,V)\in Pow(\bigcup T)\times Pow(\bigcup T). U\subseteq V\}))$
proof
assume $T(\text{is anti-} isT2$
moreover have $\forall T. (\text{is a topology}) \implies (\text{is } T_4) \implies (\text{is } T_3)$ using
    topology0.T3_is_T2 topology0_def
moreover have $\forall A. (A \text{ is in the spectrum of } isT2) \implies (A \text{ is in the spectrum of } isT4)$ using T2_spectrum T4_spectrum
    by auto
ultimately have $T(\text{is anti-} isT4) \implies \text{eq_spect_rev_imp_anti[of isT4isT2]}$
by auto
then show IsLinOrder($T,\{(U,V)\in Pow(\bigcup T)\times Pow(\bigcup T). U\subseteq V\}$) using anti_T4
by auto
next
assume IsLinOrder($T,\{(U,V)\in Pow(\bigcup T)\times Pow(\bigcup T). U\subseteq V\}$)
then have $T(\text{is anti-} isT1) \implies \text{anti_T1}$
by auto
moreover have $\forall T. (\text{is a topology}) \implies (\text{is } T_2) \implies (\text{is } T_1)$ using T2_is_T1
by auto
moreover
have \( \forall A. (A \text{ is in the spectrum of } isT1) \longrightarrow (A \text{ is in the spectrum of } isT2) \) using T1_spectrum T2_spectrum by auto

ultimately show \( T \text{ is anti-} isT2 \) using eq_spect_rev_imp_anti[of isT2isT1] by auto

qed

lemma linord_spectrum:
  shows \(((\lambda T. IsLinOrder(T,\{(U,V)\in Pow(\bigcup T)\times Pow(\bigcup T). U\subseteq V\})) \quad \longleftrightarrow \quad A \preceq 1)\)
proof
  assume \( A \text{ is in the spectrum of } (\lambda T. IsLinOrder(T,\{(U,V)\in Pow(\bigcup T)\times Pow(\bigcup T). U\subseteq V\})) \)
  then have \( \forall T. T \text{ is a topology} \quad \bigcup T\approx A \longrightarrow IsLinOrder(T,\{(U,V)\in Pow(\bigcup T)\times Pow(\bigcup T). U\subseteq V\}) \)
    using Spec_def by auto
  
  \{
  assume \( A=0 \)
  moreover
  have \( 0\preceq 1 \) using empty_lepollI by auto
  
  ultimately have \( A\preceq 1 \) using eq_lepoll_trans by auto
  \}

  moreover
  
  \{
  assume \( A\neq 0 \)
  then obtain \( x \) where \( x\in A \) by blast
  moreover
  
  \{
  fix \( y \)
  assume \( y\in A \)
  have \( Pow(A) \text{ is a topology} \) using Pow_is_top by auto
  moreover
  have \( \bigcup Pow(A)=A \) by auto
  
  then have \( \bigcup Pow(A)\approx A \) by auto
  
  note \( \text{reg} \)
  
  ultimately have \( IsLinOrder(Pow(A),\{(U,V)\in Pow(\bigcup Pow(A))\times Pow(\bigcup Pow(A)). U\subseteq V\}) \)
    by auto
  
  then have \( IsLinOrder(Pow(A),\{(U,V)\in Pow(A)\times Pow(A). U\subseteq V\}) \) by auto
  
  with \( \langle x\in A\rangle \langle y\in A\rangle \) have \( \{x\}\subseteq\{y\}\lor\{y\}\subseteq\{x\} \) unfolding IsLinOrder_def
  
  IsTotal_def by auto
  
  then have \( x=y \) by auto
  
  ultimately have \( A=\{x\} \) by blast
  
  then have \( A\approx 1 \) using singleton_eqpoll_1 by auto
  
  then have \( A\preceq 1 \) using eqpoll_imp_lepoll by auto
  \}

ultimately show \( A\preceq 1 \) by auto

next

assume \( A\preceq 1 \)
then have ind:A{is in the spectrum of}(λT. T={0,∪T}) using indiscrete_spectrum by auto 
{
  fix T
  assume AS:T{is a topology} T={0,∪T}
  have trans({⟨U,V⟩∈Pow(∪T)×Pow(∪T). U⊆V}) unfolding trans_def by auto
  moreover
  have antisym({⟨U,V⟩∈Pow(∪T)×Pow(∪T). U⊆V}) unfolding antisym_def by auto
  moreover
  have {⟨U,V⟩∈Pow(∪T)×Pow(∪T). U⊆V}{is total on}T proof-
  \{ fix aa b 
  assume aa∈Tb∈T
  with AS(2) have aa∈{0,∪T}b∈{0,∪T} by auto
  then have aa=0\lor aa=∪T b=0\lor b=∪T by auto
  then have (aa, b) ∈ Collect(Pow(∪T) × Pow(∪T), split(⟨≤⟩)) \lor (b, aa) ∈ Collect(Pow(∪T) × Pow(∪T), split(⟨≤⟩))
  using ⟨aa∈T, b∈T⟩ by auto
  \}
  then show thesis using IsTotal_def by auto
 qed
ultimately have IsLinOrder(T,{⟨U,V⟩∈Pow(∪T)×Pow(∪T). U⊆V}) unfolding IsLinOrder_def by auto 
\}
then have ∀T. T {is a topology} → T = {0,∪T} → IsLinOrder(T, {⟨U,V⟩∈Pow(∪T)×Pow(∪T). U⊆V}) by auto
  then show A{is in the spectrum of}(λT. IsLinOrder(T,{⟨U,V⟩∈Pow(∪T)×Pow(∪T). U⊆V}))
  using P_imp_Q_spec_inv[of λT. IsLinOrder(T,{⟨U,V⟩∈Pow(∪T)×Pow(∪T). U⊆V})]
  by auto
qed

theorem (in topology0) anti_linord:
  shows (T{is anti-}(λT. IsLinOrder(T,{⟨U,V⟩∈Pow(∪T)×Pow(∪T). U⊆V})) ↔ T{is T_1})
proof
  assume AS:T{is anti-}(λT. IsLinOrder(T,{⟨U,V⟩∈Pow(∪T)×Pow(∪T). U⊆V}))
  \{ assume ¬(T{is T_1})
  then obtain x y where x∈∪Ty⊆∪Tx\neq y∀U∈T. x\notin U\lor y\notin U unfolding isT1_def by auto
  \{ assume {x}⊆T{restricted to}{x,y}
  then obtain U where U∈T \{x\}=x,y\cap U unfolding RestrictedTo_def
by auto
moreover
have \( x \in \{x\} \) by auto
ultimately have \( U \subseteq T \) by auto
moreover
\{ 
  assume \( y \in U \)
  then have \( y \in \{x,y\} \cap U \) by auto
  with \( \{x\} = \{x,y\} \cap U \) have \( y \in \{x\} \) by auto
  with \( x \neq y \) have False by auto
\}
then have \( y \notin U \) by auto
moreover
note \( \forall U \in T. \; x \notin U \lor y \in U \)
ultimately have False by auto
\}
then have \( \{x\} \notin T \) by auto
moreover
have \( T \subseteq \{x,y\} \) using \( \{x\} \subseteq \{x,y\} \) unfolding RestrictedTo_def by auto
moreover
have \( \bigcup T \subseteq \{x,y\} \) by auto
ultimately have \( \bigcup T \subseteq \{x\} \) by auto
moreover
have \( \text{IsLinOrder}(\emptyset, \{x,y\}) \) using \( \text{IsTotal} \) on \( \{0, \{x,y\}, \{y\}\} \)
proof-
  have antisym(Collect(Pow({x, y}) \times Pow({x, y}), split((\subseteq)))) using antisym_def by auto
  moreover
  have trans(Collect(Pow({x, y}) \times Pow({x, y}), split((\subseteq)))) using trans_def by auto
  moreover
  have Collect(Pow({x, y}) \times Pow({x, y}), split((\subseteq))) \{ is total on \}{0, \{x, y\}, \{y\}} using IsTotal_def by auto
  ultimately show IsLinOrder(\{0, \{x,y\}, \{y\}\}, \{\emptyset\} \in Pow(\{x,y\}) \times Pow(\{x,y\}), \emptyset \subseteq V) using IsLinOrder_def by auto
qed
ultimately have \( \text{IsLinOrder}(T \subseteq \{x,y\}, \{\emptyset\} \in Pow(\{x,y\}) \times Pow(\{x,y\}, \emptyset \subseteq V) \) using ord_linear_subset by auto
with \( \text{tot} \) have \( \text{IsLinOrder}(T \subseteq \{x,y\}, \{\emptyset\} \in Pow(\bigcup T \subseteq \{x,y\}) \times Pow(\bigcup T \subseteq \{x,y\}), \emptyset \subseteq V) \) by auto
then have \( \text{IsLinOrder}(T \subseteq \{x,y\}, \text{Collect}(Pow(\bigcup T \subseteq \{x,y\})) \times Pow(\bigcup T \subseteq \{x,y\}), \text{split((\subseteq)))} \) by auto
moreover
from \( \{ x \in \bigcup T \} \land \{ y \in \bigcup T \} \) have \( \{ x, y \} \in \text{Pow}(\bigcup T) \) by auto
moreover
note AS
ultimately have \( \{ x, y \} \) is in the spectrum of \( \langle \lambda T. \text{IsLinOrder}(T, \langle U, V \rangle \in \text{Pow}(\bigcup T) \times \text{Pow}(\bigcup T). U \subseteq V \rangle \rangle \) unfolding antiProperty_def
by simp
then have \( \{ x, y \} \preceq 1 \) using linord_spectrum by auto
moreover
have \( x \in \{ x, y \} \) by auto
ultimately have \( \{ x \} = \{ x, y \} \) using lepoll_i_is_sing[of \( \{ x, y \} \)] by auto
moreover
have \( y \in \{ x, y \} \) by auto
ultimately
have \( x = y \) by auto
then have False using \( \langle x \neq y \rangle \) by auto
}
then show \( T \) {is T
next
assume \( T_1 : T \) {is T

{ fix A
assume A_def:A \in \text{Pow}(\bigcup T) \text{IsLinOrder}(\langle T \text{restricted to} A \rangle \langle U, V \rangle \in \text{Pow}(\bigcup (T \text{restricted to} A)) \times \text{Pow}(\bigcup (T \text{restricted to} A)). U \subseteq V \rangle)

{ fix x
assume AS1:x \in A

{ fix y
assume AS:y \in \text{Ax} \neq y
with AS1 have \( \{ x, y \} \in \text{Pow}(\bigcup T) \) using \( \langle A \in \text{Pow}(\bigcup T) \rangle \) by auto
from \( \langle x \in A \rangle \land \langle y \in A \rangle \) have \( \{ x, y \} \in \text{Pow}(A) \) by auto
from \( \langle x, y \rangle \in \text{Pow}(\bigcup T) \) have \( T_{11} : (T \text{restricted to} \{ x, y \}) \{ \text{is T}_1 \} \)
using T1_here T1 by auto
moreover
have tot: \( \bigcup (T \text{restricted to} \{ x, y \}) = \{ x, y \} \) unfolding RestrictedTo_def
using \( \{ x, y \} \in \text{Pow}(\bigcup T) \) by auto
moreover
note AS(2)
ultimately obtain \( U \) where \( x \in U \land y \in U \in (T \text{restricted to} \{ x, y \}) \) unfolding isT1_def by auto
moreover
from AS(2) tot T11 obtain \( V \) where \( y \in V \land x \in V \in (T \text{restricted to} \{ x, y \}) \) unfolding isT1_def by auto
ultimately have \( x \in U \land y \in V \land x \in (T \text{restricted to} \{ x, y \}) \land y \in (T \text{restricted to} \{ x, y \}) \) by auto
then have \( \neg(U \subseteq V \subseteq U \subseteq (T \text{restricted to} \{ x, y \}) \subseteq V \subseteq (T \text{restricted to} \{ x, y \})) \) by auto
then have \( \neg((U, V) \in \text{Pow}(\bigcup (T \text{restricted to} \{ x, y \})) \times \text{Pow}(\bigcup (T \text{restricted to} \{ x, y \}))) \times \text{Pow}(\bigcup (T \text{restricted to} \{ x, y \})) \) by auto

1044
to\{x,y\}). U \subseteq V \{is total on\} (T\{restricted to\}{x,y}))

unfolding IsTotal_def by auto

then have \neg (IsLinOrder((T\{restricted to\}{x,y}),\{(U,V)\in\text{Pow}(\bigcup (T\{restricted to\}{x,y})) \times \text{Pow}(\bigcup (T\{restricted to\}{x,y})). U \subseteq V}))

unfolding IsLinOrder_def by auto

moreover

have \begin{align*}
(T\{restricted to\}A) \{is a topology\} \text{ using Top_1_L4 by auto}
\end{align*}

moreover

note A_def(2) linordtop_here

ultimately have \begin{align*}
\forall B \in \text{Pow}(\bigcup (T\{restricted to\}A)) \{\text{restricted to\}B} \text{ using subspace_of_subspace} \langle A\in\text{Pow}(\bigcup T) \rangle \text{ by auto}
\end{align*}

ultimately have \begin{align*}
\forall B \in \text{Pow}(A) \text{ IsLinOrder}((T\{restricted to\}A)\{restricted to\}B),(U,V)\in\text{Pow}(\bigcup (\{T\{restricted to\}A\}{restricted to\}B)) \times \text{Pow}(\bigcup (\{T\{restricted to\}A\}{restricted to\}B)). U \subseteq V
\end{align*}

moreover

have \begin{align*}
\forall B \in \text{Pow}(A) \text{(T\{restricted to\}A)\{restricted to\}B= T\{restricted to\}B) \text{ using subspace_of_subspace} \langle A\in\text{Pow}(\bigcup T) \rangle \text{ by auto}
\end{align*}

ultimately

have \begin{align*}
\forall B \in \text{Pow}(A) \text{ IsLinOrder}((T\{restricted to\}B),(U,V)\in\text{Pow}(\bigcup (\{T\{restricted to\}A\}{restricted to\}B)) \times \text{Pow}(\bigcup (\{T\{restricted to\}A\}{restricted to\}B)). U \subseteq V
\end{align*}

moreover

have \begin{align*}
\forall B \in \text{Pow}(A) \text{ IsLinOrder}((T\{restricted to\}B),(U,V)\in\text{Pow}(\bigcup (\{x,y\}) \times \text{Pow}(\bigcup (\{x,y\}). U \subseteq V)
\end{align*}

ultimately have \begin{align*}
\text{False}\text{ using tot by auto}
\end{align*}

} 

ultimately have \begin{align*}
A=\{x\} \text{ using AS1 by auto}
\end{align*}

then have \begin{align*}
A\cong 1 \text{ using singleton_eqpoll_1 by auto}
\end{align*}

then have \begin{align*}
A\subseteq 1 \text{ using eqpoll_imp_lepoll by auto}
\end{align*}

then have \begin{align*}
\{\text{is in the spectrum of}\} \langle \lambda T \text{ IsLinOrder}(T,(U,V)\in\text{Pow}(\bigcup T) \times \text{Pow}(\bigcup T). U \subseteq V)\rangle \text{ using linord_spectrum by auto}
\end{align*}

moreover

assume A=0

then have \begin{align*}
A\approx 0 \text{ by auto}
\end{align*}
moreover
  have $0 \leq 1$ using `empty_lepollI` by auto
ultimately have $A \leq 1$ using `eq_lepoll_trans` by auto
then have $A$ is in the spectrum of $(\lambda T. \text{IsLinOrder}(T, \{\langle U, V \rangle \in \text{Pow}(\bigcup T) \times \text{Pow}(\bigcup T). U \subseteq V\}))$ using `linord_spectrum` by auto
ultimately have $A$ is in the spectrum of $(\lambda T. \text{IsLinOrder}(T, \{\langle U, V \rangle \in \text{Pow}(\bigcup T) \times \text{Pow}(\bigcup T). U \subseteq V\}))$ by blast
then show $T$ is anti-$((\lambda T. \text{IsLinOrder}(T, \{\langle U, V \rangle \in \text{Pow}(\bigcup T) \times \text{Pow}(\bigcup T). U \subseteq V\}))$ unfolding `antiProperty_def` by auto
qed

In conclusion, $T_1$ is also an anti-property.

Let’s define some anti-properties that we’ll use in the future.
definition
  `IsAntiComp` ($\langle$ is anti-compact$\rangle$)
where $T$ is anti-compact $\equiv T$ is anti-$((\lambda T. (\bigcup T)$ is compact in$T)$
definition
  `IsAntiLin` ($\langle$ is anti-lindelof$\rangle$)
where $T$ is anti-lindelof $\equiv T$ is anti-$((\bigcup T)$ is lindelof in$T))$

Anti-compact spaces are also called pseudo-finite spaces in literature before the concept of anti-property was defined.
end

77 Topology 6
theory Topology_ZF_6 imports Topology_ZF_4 Topology_ZF_2 Topology_ZF_1
begin
This theory deals with the relations between continuous functions and convergence of filters. At the end of the file there some results about the building of functions in cartesian products.

77.1 Image filter
First of all, we will define the appropriate tools to work with functions and filters together.

We define the image filter as the collections of supersets of of images of sets from a filter.
definition
  ImageFilter (_[_].._ 98)
  where F {is a filter on} X \Rightarrow f:X\rightarrow Y \equiv \{A\in\text{Pow}(Y). \exists D\in\{f(B) . B\in F\}. D\subseteq A\}

Note that in the previous definition, it is necessary to state Y as the final set because f is also a function to every superset of its range. X can be changed by domain(f) without any change in the definition.

lemma base_image_filter:
  assumes F {is a filter on} X f:X \rightarrow Y
  shows \{fB . B \in F\} {is a base filter} (f[\mathcal{F}]..Y) and (f[\mathcal{F}]..Y) {is a filter on} Y
  proof-
    { assume 0 \notin \{fB . B\in F\} by auto
      then obtain B where B\in F and f_B=f_{\mathcal{F}}=0 by auto
      then have B\in \text{Pow}(X) using assms(1) IsFilter_def by auto
      then have f_B=\{fb . b\in B\} by auto
      with f_B have 0=0 by auto
      } then have 0 /\in \{fB . B\in F\} by auto
    moreover
    from assms(1) obtain S where S\subseteq F using IsFilter_def by auto
    then have fS\subseteq \{fB . B\in F\} by auto
    then have nA: \{fB . B\in F\} \neq 0 by auto
    moreover
    { fix A B
      assume A\in \{fB . B\in F\} and B\in \{fB . B\in F\}
      then obtain AB BB where A=fAB B=fBB AB\subseteq F BB\subseteq F by auto
      then have A\cap B=(fAB)\cap (fBB) by auto
      then have I: f(AB\cap BB)\subseteq A\cap B by auto
      moreover
      from assms(1) I \langle AB\cap BB\rangle have AB\cap BB\subseteq F using IsFilter_def by auto
      ultimately have \exists D\in\{fB . B\in F\}. D\subseteq A\cap B by auto
    } then have \forall A\in \{fB . B\in F\}. \forall B\in \{fB . B\in F\}. \exists D\in\{fB . B\in F\}. D\subseteq A\cap B by auto
    ultimately have \text{sbc:}\{fB . B\in F\} \{satisfies the filter base condition\}
    using SatisfiesFilterBase_def by auto
    moreover
    { fix t
      assume t\in \{fB . B\in F\}
      then obtain B where B\in F and \text{im_def:}fB=t by auto
      with assms(1) have B\in \text{Pow}(X) unfolding IsFilter_def by auto
    }
with im_def assms(2) have t=\{fx. x\in B\} using image_fun by auto
with assms(2) \langle B\in\text{Pow}(X)\rangle have t\subseteq Y using apply_funtype by auto
}
then have nB:\{fB . B\in\mathcal{B}\} \subseteq \text{Pow}(Y) by auto
ultimately have \({\{fB . B\in\mathcal{B}\}}\) {is a base filter} \{A \in \text{Pow}(Y) . \exists D\in\{fB . B\in\mathcal{B}\}. D \subseteq A\} by auto
by force
then have \{fB . B\in\mathcal{B}\} \{is a base filter\} \{A \in \text{Pow}(Y) . \exists D\in\{fB . B\in\mathcal{B}\}. D \subseteq A\} by auto
with assms show \{fB . B\in\mathcal{B}\} \{is a base filter\} (f\_{\mathcal{F}}..Y) using ImageFilter_def by auto
moreover
note sbc
moreover
\{ from nA obtain D where I: D\in\{fB . B\in\mathcal{B}\} by blast
moreover from I nB have D\subseteq Y by auto
ultimately have Y\in\{A\in\text{Pow}(Y) . \exists D\in\{fB . B\in\mathcal{B}\}. D\subseteq A\} by auto
\}
then have \bigcup\{A\in\text{Pow}(Y) . \exists D\in\{fB . B\in\mathcal{B}\}. D\subseteq A\}=Y by auto
ultimately show (f\_{\mathcal{F}}..Y) \{is a filter on\} Y using basic_filter ImageFilter_def assms by auto
qed

77.2 Continuous at a point vs. globally continuous

In this section we show that continuity of a function implies local continuity (at a point) and that local continuity at all points implies (global) continuity.

If a function is continuous, then it is continuous at every point.

**lemma cont_global_imp_continuous_x:**
assumes \(x\in\bigcup\tau_1\) IsContinuous(\(\tau_1,\tau_2, f\)) \(f:\bigcup\tau_1\rightarrow\bigcup\tau_2\) \(x\in\bigcup\tau_1\)
shows \(\forall U\in\tau_2. f(x)\in U \rightarrow (\exists V\in\tau_1. x\in V \land f(V)\subseteq U)\)
proof-
\{ fix U
assume AS:U\in\tau_2 f(x)\in U
then have f-(U)\in\tau_1 using assms(2) IsContinuous_def by auto
moreover
from assms(3) have f(f-(U))\subseteq U using function_image_vimage fun_is_fun by auto
moreover
from assms(3) assms(4) AS(2) have x\in f-(U) using func1_1_L15 by auto
ultimately have \(\exists V\in\tau_1. x\in V \land f(V)\subseteq U\) by auto
\}
then show \(\forall U\in\tau_2. f(x)\in U \rightarrow (\exists V\in\tau_1. x\in V \land f(V)\subseteq U)\) by auto

1048
A function that is continuous at every point of its domain is continuous.

**Lemma ccontinuous_all_x_imp_cont_global:**

**Assumes**
\[ \forall x \in \bigcup \tau_1. \ \forall U \in \tau_2. \ f(x) \in U \rightarrow (\exists V \in \tau_1. \ x \in V \land f \subseteq U) \]

**Shows** \[ \text{IsContinuous}(\tau_1, \tau_2, f) \]

**Proof**

1. Fix \( U \)
2. Assume \( U \in \tau_2 \)
3. Fix \( x \)
4. Assume \( A \): \( x \in f - U \)
5. Note \( \langle U \in \tau_2 \rangle \)
6. Moreover, from \( \text{assms}(2) \) have \( f - U \subseteq \bigcup \tau_1 \) using \( \text{func1}_1\_L6A \) by auto
7. With \( \text{assms}(1) \) have \( \forall U \in \tau_2. \ f(x) \in U \rightarrow (\exists V \in \tau_1. \ x \in V \land f \subseteq U) \) by auto
8. Moreover, from \( \text{assms}(2) \) have \( f(x) \in U \) using \( \text{func1}_1\_L15 \) by auto
9. Ultimately, have \( \exists V \in \tau_1. \ x \in V \land f \subseteq U \) by auto
10. Then obtain \( V \) where \( I: \forall V \in \tau_1. \ x \in V \land f \subseteq U \) by auto
11. Moreover, from \( I \) have \( V \subseteq \bigcup \tau_1 \) by auto
12. Moreover, from \( \text{assms}(2) \) have \( V \subseteq f - (fV) \) using \( \text{func1}_1\_L9 \) by auto
13. Ultimately, have \( \forall V \in \tau_1. \ x \in V \land V \subseteq f - (U) \) by auto
14. Hence \( \forall x \in f - (U). \ \forall V \in \tau_1. \ x \in V \land V \subseteq f - (U) \) by auto
15. With \( \text{assms}(3) \) have \( f - (U) \in \tau_1 \) using \( \text{topology0.open_neigh_open topology0_def} \)
16. By auto
17. Hence \( \forall U \in \tau_2. \ f - U \in \tau_1 \) by auto
18. Then show thesis using \( \text{IsContinuous_def} \) by auto

**Qed**

### 77.3 Continuous functions and filters

In this section we consider the relations between filters and continuity.

If the function is continuous then if the filter converges to a point the image filter converges to the image point.

**Lemma (in two_top_spaces0) cont_imp_filter_conver_preserved:**

**Assumes**
\[ \forall \exists \{ \text{is a filter on} \} \ X_1 \ f \{ \text{is continuous} \} \exists \rightarrow F \ x \{ \text{in} \} \tau_1 \]

**Shows** \( (f[\exists]..X_2) \rightarrow_F (f(x)) \{ \text{in} \} \tau_2 \)

**Proof**
from assms(1) assms(3) have \( x \in X_1 \)
  using topology0.FilterConverges_def topol_cntxs_valid(1) X1_def by auto
have topology0(\( \tau_2 \) ) using topol_cntxs_valid(2) by simp
moreover from assms(1) have \( (f[g]..X_2) \) {is a filter on} \( \bigcup \tau_2 \) and
\( \{fB .B\in g\} \) {is a base filter} \( f[g]..X_2 \)
  using base_image_filter fmapAssum X1_def X2_def by auto
moreover have \( \forall U \in \text{Pow}(\bigcup \tau_2) . (fx) \in \text{Interior}(U, \tau_2) \rightarrow (\exists D \in \{fB .B\in g\} . D \subseteq U) \)
proof -
  \{ fix U 
  assume \( U \in \text{Pow}(X_2) . (fx) \in \text{Interior}(U, \tau_2) \)
  with \( x \in X_1 \) have xim: \( x \in f-(\text{Interior}(U, \tau_2)) \) and
  sub: \( f-(\text{Interior}(U, \tau_2)) \in \text{Pow}(X_1) \)
  using func1_1_L6A fmapAssum func1_1_L15 fmapAssum by auto
  note sub
moreover have \( \text{Interior}(U, \tau_2) \in \tau_2 \) using topology0.Top_2_L2 topol_cntxs_valid(2)
by auto
with assms(2) have \( f-(\text{Interior}(U, \tau_2)) \in \tau_1 \) unfolding IsContinuous_def
by auto
with xim have \( x \in \text{Interior}(f-(\text{Interior}(U, \tau_2)), \tau_1) \) using topology0.Top_2_L3 topol_cntxs_valid(1) by auto
moreover from assms(1) assms(3) have \( \{U \in \text{Pow}(X_1) . x \in \text{Interior}(U, \tau_1) \} \subseteq g \)
  using topology0.FilterConverges_def topol_cntxs_valid(1) X1_def by auto
ultimately have \( f-(\text{Interior}(U, \tau_2)) \in \{fB .B\in g\} \) \( D \subseteq U \) by auto
thus thesis by auto
qed
moreover from fmapAssum \( x \in X_1 \) have \( f(x) \in X_2 \)
  by (rule apply_funtype)
hence \( f(x) \in \bigcup \tau_2 \) by simp
ultimately show thesis by (rule topology0.convergence_filter_base2)
qed

Continuity in filter at every point of the domain implies global continuity.

**Lemma** (in two_top_spaces0) filter_conver_preserved_imp_cont:
assumes \( \forall x \in \bigcup \tau_1 . \forall g . ((g \in \text{filter on} X_1) \land (g \rightarrow_{\tau_1} x \in \{\text{in} \tau_1\})) \rightarrow ((f[g]..X_2) \rightarrow_{\tau_2} (fx) \in \{\text{in} \tau_2\}) \)
shows f{is continuous}
proof-

1050
\[
\{ 
\text{fix } x \\
\text{assume as2: } x \in \bigcup \tau_1 \\
\text{with asms have reg:} \\
\forall \tilde{x}. ((\tilde{x} \text{ is a filter on} \ X_1) \land (\tilde{x} \rightarrow_F x \{\text{in}\} \tau_1)) \rightarrow ((f[\tilde{x}]..X_2) \rightarrow_F (fx) \{\text{in}\} \tau_2)
\}
\]

by auto

let Neig = \{U \in \text{Pow}(\bigcup \tau_1) . \ x \in \text{Interior}(U, \tau_1)\}

from as2 have NFil: Neig{is a filter on}X_1 and NCon: Neig \rightarrow_F x \{\text{in}\} \tau_1

using topol_cntxs_valid(1) topology0.neigh_filter by auto

\{
 fix U \\
 assume U\in\tau_2 \ fx\in U \\
 then have U\in\text{Pow}(\bigcup \tau_2) \ fx\in\text{Interior}(U, \tau_2) using topol_cntxs_valid(2)
\}

topology0.Top_2_L3 by auto

moreover

from NCon NFil reg have (f[Neig]..X_2) \rightarrow_F (fx) \{\text{in}\} \tau_2 by auto

moreover have (f[Neig]..X_2) \{is a filter on\} X_2

using base_image_filter(2) NFil fmapAssum by auto

ultimately have \exists(U\in(f[Neig]..X_2))

using topology0.FilterConverges_def topol_cntxs_valid(2) unfolding X1_def X2_def

by auto

moreover

from fmapAssum NFil have \{fB .B\in Neig\} \{is a base filter\} (f[Neig]..X_2)

using base_image_filter(1) X1_def X2_def by auto

ultimately have \exists V\in\{fB .B\in Neig\}. V\subseteq U using basic_element_filter

by blast

then obtain B where B\in Neig fB\subseteq U by auto

moreover

have \text{Interior}(B, \tau_1) \subseteq B using topology0.Top_2_L1 topol_cntxs_valid(1)

by auto

hence f\text{Interior}(B, \tau_1) \subseteq f(B) by auto

moreover have \text{Interior}(B, \tau_1) \subseteq \tau_1

using topology0.Top_2_L2 topol_cntxs_valid(1) by auto

ultimately have \exists V\in\tau_1. x\in V \land fV\subseteq U by force

\}

hence \forall U\in\tau_2. \ fx\in U \rightarrow (\exists V\in\tau_1. x\in V \land fV\subseteq U) by auto

\}

hence \forall x\in\bigcup \tau_1. \forall U\in\tau_2. \ fx\in U \rightarrow (\exists V\in\tau_1. x\in V \land fV\subseteq U) by auto

then show thesis

using ccontinuous_all_x_imp_cont_global fmapAssum X1_def X2_def isContinuous_def tau1_is_top

by auto

qed
78 Topology 7

theory Topology_ZF_7 imports Topology_ZF_5
begin

78.1 Connection Properties

Another type of topological properties are the connection properties. These properties establish if the space is formed of several pieces or just one.

A space is connected iff there is no clopen set other that the empty set and the total set.

definition IsConnected (_{is connected} 70)
  where T {is connected} ≡ ∀ U. (U ∈ T ∧ (U {is closed in} T)) → U=0 ∨ U=∪ T

lemma indiscrete_connected:
  shows {0,X} {is connected}
  unfolding IsConnected_def IsClosed_def by auto

The anti-property of connectedness is called total-disconnectedness.

definition IsTotDis (_{is totally-disconnected} 70)
  where IsTotDis ≡ ANTI(IsConnected)

lemma conn_spectrum:
  shows (A {is in the spectrum of} IsConnected) ↔ A ≼ 1
proof
  assume A {is in the spectrum of} IsConnected
  then have ∀ T. (T {is a topology} ∧ ∪ T=A) → (T {is connected}) using Spec_def by auto
  moreover have Pow(A) {is a topology} using Pow_is_top by auto
  moreover have ∪ (Pow(A))=A by auto
  then have ∪ (Pow(A))≈A by auto
  ultimately have Pow(A) {is connected} by auto
  { assume A≠0
    then obtain E where E∈A by blast
    then have {E}⊂Pow(A) by auto
    moreover have A-{E}⊂Pow(A) by auto
    ultimately have {E}⊂Pow(A) ∧ {E} {is closed in} Pow(A) unfolding IsClosed_def by auto
    with Pow(A) {is connected} have {E}=A unfolding IsConnected_def by auto
    then have A≈1 using singleton_eqpoll_1 by auto

1052
then have 1 ≲ using eqpoll_imp_lepoll by auto
}
moreover
{
  assume A=0
  then have 1 ≲ using empty_lepollI[of 1] by auto
}
ultimately show 1 ≲ by auto
next
assume 1 ≲
{
  fix T
  assume T{is a topology} T= A
  assume T=0 with T{is a topology} have T={0} using empty_open by auto
  then have T{is connected} unfolding IsConnected_def by auto
}
moreover
{
  assume T≠0
  moreover
  from A⊆ T= A have T⊆ using eq_lepoll_trans by auto
  ultimately
  obtain E where T={E} using lepoll_1_is_sing by blast
  moreover
  have T⊆ Pow(T) by auto
  ultimately have T⊆ Pow({E}) by auto
  then have T⊆ {0,{E}} by blast
  with T{is a topology} have {0}⊆ T⊆ {0,{E}} using empty_open by auto
  then have T{is connected} unfolding IsConnected_def by auto
  }
  ultimately have T{is connected} by auto
}
then show A{is in the spectrum of}IsConnected unfolding Spec_def by auto
qed

The discrete space is a first example of totally-disconnected space.

**Lemma discrete_tot_dis:**

shows Pow(X) {is totally-disconnected}

**Proof:**

fix A assume A⊆ Pow(X) and con:(Pow(X){restricted to}A){is connected}
have res:(Pow(X){restricted to}A)=Pow(A) unfolding RestrictedTo_def
using A⊆ Pow(X)
by blast

assume \( A = 0 \) then have \( A \preceq 1 \) using empty_lepoll[of 1] by auto

then have \( A \in \text{the spectrum of} \text{IsConnected} \) using conn_spectrum

by auto

moreover

{ assume \( A \neq 0 \) then obtain \( E \) where \( E \in A \) by blast

then have \( \{E\} \in \text{Pow}(A) \) by auto

moreover

have \( A - \{E\} \in \text{Pow}(A) \) by auto

ultimately have \( \{E\} \in \text{Pow}(A) \) unfolding IsClosed_def

by auto

with con res have \( \{E\} = A \) unfolding IsConnected_def by auto

then have \( A \approx 1 \) using singleton_eqpoll_1 by auto

then have \( A \preceq 1 \) using eqpoll_imp_lepoll by auto

then have \( A \in \text{the spectrum of} \text{IsConnected} \) using conn_spectrum

by auto

ultimately have \( A \in \text{the spectrum of} \text{IsConnected} \) by auto

}

then show thesis unfolding IsTotDis_def antiProperty_def by auto

qed

An space is hyperconnected iff every two non-empty open sets meet.

definition IsHConnected (_{is hyperconnected})

where \( T \in \text{hyperconnected} \equiv \forall U \ V. \ U \in T \land V \in T \land U \cap V = 0 \rightarrow U = 0 \lor V = 0 \)

Every hyperconnected space is connected.

lemma HConn_imp_Conn:

assumes \( T \in \text{hyperconnected} \)

shows \( T \in \text{connected} \)

proof-

{ fix \( U \)

assume \( U \in T \) \( \in \text{closed in} T \)

then have \( U \cup (T - U) \in T \) using IsClosed_def by auto

moreover

have \( (U \cup (T - U)) \cap U = 0 \) by auto

moreover

note assms

ultimately

have \( U = 0 \lor (U \cup (T - U)) = 0 \) using IsHConnected_def by auto

with \( \langle U \in T \rangle \) have \( U = 0 \lor U = T \) by auto

}

then show thesis using IsConnected_def by auto

qed
lemma Indiscrete_HConn:
  shows \{0,X\}\{is hyperconnected\}
  unfolding IsHConnected_def by auto

A first example of an hyperconnected space but not indiscrete, is the cofinite topology on the natural numbers.

lemma Cofinite_nat_HConn:
  assumes \(\neg(X\prec\mathbb{N})\)
  shows (CoFinite X)\{is hyperconnected\}
proof-
{  
fix U V
  assume U\in(CoFinite X)V\in(CoFinite X)U\cap V=0
  then have eq:(X-U)\prec\mathbb{N}V=0(X-V)\prec\mathbb{N}V=0 unfolding Cofinite_def
    CoCardinal_def by auto
  from \(U\cap V=0\) have un:(X-U)\cup(X-V)=X by auto 
  {    
    assume AS:(X-U)\prec\mathbb{N}(X-V)\prec\mathbb{N}
    from un have X\prec\mathbb{N} using less_less_imp_un_less[OF AS InfCard_nat] 
    by auto
    then have False using assms by auto
  }
  with eq(1,2) have U=0\lor V=0 by auto
}
then show (CoFinite X)\{is hyperconnected\} using IsHConnected_def by auto
qed

lemma HConn_spectrum:
  shows \(A\}{is in the spectrum of}IsHConnected\) \iff \(A\approx 1\)
proof
  assume \(A\}{is in the spectrum of}IsHConnected\)
  then have \(\forall T. (T\}{is a topology}\\bigcup T=A) \longrightarrow (T\}{is hyperconnected}\)
  using Spec_def by auto
  moreover
  have \(Pow(A)\}{is a topology}\ using Pow_is_top by auto
  moreover
  have \(\bigcup(Pow(A))=A\) by auto
  then have \(\bigcup(Pow(A))\approx A\) by auto
  ultimately
  have HC.Pow:Pow(A)\{is hyperconnected\} by auto 
  {    
    assume A=0
    then have \(A\approx 1\) using empty_lepollI by auto 
  }
  moreover 
  {    
    assume A\neq 0
    then obtain e where e\in A by blast 
  }
then have \{e\} ∈ Pow(A) by auto
moreover
have A-{e} ∈ Pow(A) by auto
moreover
have \{e\} ∩ (A-{e}) = 0 by auto
moreover
note HC_Pow
ultimately have A-{e} = 0 unfolding IsHConnected_def by blast
with \{e\} ∈ A have A = \{e\} by auto
then have A ≈ 1 using singleton_eqpoll_1 by auto
then have A ≼ 1 using eqpoll_imp_lepoll by auto
}
ultimately show A ≼ 1 by auto
next
assume A ≼ 1
{
  fix T
  assume T\{is a topology\} \cup T \approx A
  {
    assume \cup T = 0
    with \{T\{is a topology\}\} have T = \{0\} using empty_open by auto
    then have T\{is hyperconnected\} unfolding IsHConnected_def by auto
  }
  moreover
  {
    assume \cup T ≠ 0
    moreover
    from \{A ≼ 1\} \cup T \approx A have \cup T \leq 1 using eq_lepoll_trans by auto
    ultimately
    obtain E where \cup T = \{E\} using lepoll_i_is_sing by blast
    moreover
    have T \subseteq Pow(\cup T) by auto
    ultimately have T \subseteq Pow(\{E\}) by auto
    then have T \subseteq \{0,\{E\}\} by blast
    with \{T\{is a topology\}\} have \{0\} \subseteq T \subseteq \{0,\{E\}\} using empty_open by auto
    then have T\{is hyperconnected\} unfolding IsHConnected_def by auto
  }
  ultimately have T\{is hyperconnected\} by auto
}
then show A\{is in the spectrum of\} IsHConnected unfolding Spec_def by auto
qed

In the following results we will show that anti-hyperconnectedness is a separation property between \(T_1\) and \(T_2\). We will show also that both implications are proper.

First, the closure of a point in every topological space is always hyperconnected. This is the reason why every anti-hyperconnected space must be \(T_1\):
every singleton must be closed.

lemma (in topology0) cl_point_imp_HConn:
  assumes x∈∪T
  shows (T{restricted to}Closure({x},T)){is hyperconnected}
proof-
  from assms have sub:Closure({x},T)⊆∪T using Top_3_L11 by auto
  then have tot:∪(T{restricted to}Closure({x},T))=Closure({x},T) unfolding RestrictedTo_def by auto
  { fix A B
    assume AS:A∈(T{restricted to}Closure({x},T))B∈(T{restricted to}Closure({x},T))A∩B=0
    then have B⊆∪((T{restricted to}Closure({x},T)))A⊆∪((T{restricted to}Closure({x},T)))
    unfolding RestrictedTo_def by auto
    with sub have Closure({x},T)-A=Closure({x},T)-(UA) Closure({x},T)-B=Closure({x},T)-(UB)
    unfolding RestrictedTo_def by auto
    with auto have A=0 ∨ B=0 using cl_contains_set assms by blast
  }
  then have thesis unfolding IsHConnected_def by auto
qed

A consequence is that every totally-disconnected space is $T_1$.

lemma (in topology0) tot_dis_imp_T1:

1057
assumes $T$ (is totally-disconnected)
shows $T$ (is $T_1$)
proof
{
  fix $x$ $y$
  assume $y \in \bigcup T x \in \bigcup T y \neq x$
  then have $(T \text{ (restricted to)} \bigcap \{x\}, T)$ (is hyperconnected) using \texttt{cl_point_imp_HConn} by auto
  then have $(T \text{ (restricted to)} \bigcap \{x\}, T)$ (is connected) using \texttt{HConn_imp_Conn} by auto
  moreover from $x \in \bigcup T$
    have $\bigcap \{x\}, T \subseteq \bigcup T$ using \texttt{Top_3_L11(1)} by auto
  moreover
  note \texttt{assms}
  ultimately have $\bigcap \{x\}, T$ (is in the spectrum of) $\text{IsConnected}$ unfolding $\text{IsTotDis_def antiProperty_def}$ by auto
  then have $\bigcap \{x\}, T \triangleleft 1$ using \texttt{conn_spectrum} by auto
  moreover
  from $y \in \bigcup T \land y \neq x$
    have $y \in \bigcup T \setminus \{x\} \land x \not\in \bigcup T \setminus \{x\}$ by auto
  ultimately have $\exists U. y \in U \land x \not\in U$ by force
}
then show \texttt{thesis} unfolding $\text{isT1_def}$ by auto
qed

In the literature, there exists a class of spaces called sober spaces; where the only non-empty closed hyperconnected subspaces are the closures of points and closures of different singletons are different.

definition $\text{IsSober}$ (\{is sober\}90)
  where $T$ (is sober) $\equiv \forall A \in \text{Pow} \{\bigcup T\} - \{0\}. (A$ (is closed in) $T \land ((T \text{ (restricted to)} A)$ (is hyperconnected))) $\Rightarrow (\exists x \in \bigcup T. A = \text{Closure} \{x\}, T) \land (\forall y \in \bigcup T. A = \text{Closure} \{y\}, T) \Rightarrow y = x$)

Being sober is weaker than being anti-hyperconnected.

theorem (in topology0) $\text{anti_HConn_imp_sober}$:
assumes $T$ (is anti-) $\text{IsHConnected}$
shows $T$ (is sober)
proof
{
  fix $A$ assume $A \in \text{Pow} \{\bigcup T\} - \{0\}. A$ (is closed in) $T \land ((T \text{ (restricted to)} A)$ (is hyperconnected))
  with \texttt{assms} have $A$ (is in the spectrum of) $\text{IsHConnected}$ unfolding $\text{antiProperty_def}$ by auto

1058
then have $A \subseteq 1$ using $\text{HConn\_spectrum}$ by auto

moreover with $\langle A \in \text{Pow}(\bigcup T) - \{0\} \rangle$ have $A \neq 0$ by auto

then obtain $x$ where $x \in A$ by auto

ultimately have $A = \{x\}$ using $\text{l epoll\_1\_is\_sing}$ by auto

moreover from $\langle x \in A \rangle \langle A \in \text{Pow}(\bigcup T) - \{0\} \rangle$ have $\{x\} \in \text{Pow}(\bigcup T)$ by auto

ultimately have $\text{Closure}(\{x\}, T) = \{x\}$ unfolding $\text{Closure\_def}$ $\text{ClosedCovers\_def}$ by auto

with $\langle A = \{x\} \rangle$ have $A = \text{Closure}(\{x\}, T)$ by auto

moreover

\[
\begin{align*}
\text{fix } y \text{ assume } y \in \bigcup TA & = \text{Closure}(\{y\}, T) \\
& \text{then have } \{y\} \subseteq \text{Closure}(\{y\}, T) \text{ using } \text{cl\_contains\_set} \text{ by auto} \\
& \text{with } \langle A = \text{Closure}(\{y\}, T) \rangle \text{ have } y \in A \text{ by auto} \\
& \text{with } \langle A = \{x\} \rangle \text{ have } y = x \text{ by auto} \\
\end{align*}
\]

then have $\forall y \in \bigcup T. A = \text{Closure}(\{y\}, T) \rightarrow y = x$ by auto

moreover note $\langle x \in \text{Pow}(\bigcup T) \rangle$

ultimately have $\exists x \in \bigcup T. A = \text{Closure}(\{x\}, T) \wedge (\forall y \in \bigcup T. A = \text{Closure}(\{y\}, T) \rightarrow y = x)$ by auto

\[
\begin{align*}
\text{then show thesis using } \text{IsSober\_def} \text{ by auto} \\
\text{qed}
\end{align*}
\]

Every sober space is $T_0$.

**Lemma (in topology0) sober_imp_T0:**

assumes $T\{\text{is sober}\}$

shows $T\{\text{is }T_0\}$

**proof-**

\[
\begin{align*}
\text{fix } x, y \\
\text{assume } AS: x \in \bigcup T y \in \bigcup T x \neq y \forall U \in T. x \in U \leftrightarrow y \in U \\
& \text{from } \langle x \in \bigcup T \rangle \text{ have } clx: \text{Closure}(\{x\}, T) \{\text{is closed in}T\} \text{ using } \text{cl\_is\_closed} \\
& \text{by auto} \\
& \text{with } \langle x \in \bigcup T \rangle \text{ have } (\bigcup T - \text{Closure}(\{x\}, T)) \in T \text{ using } \text{Top\_3\_L11(1)} \text{ unfolding } \text{IsClosed\_def} \text{ by auto} \\
& \text{moreover} \\
& \text{from } \langle x \in \bigcup T \rangle \text{ have } x \in \text{Closure}(\{x\}, T) \text{ using } \text{cl\_contains\_set} \text{ by auto} \\
& \text{moreover} \\
& \text{note } AS(1, 4) \\
& \text{ultimately have } y \notin (\bigcup T - \text{Closure}(\{x\}, T)) \text{ by auto} \\
& \text{with } AS(2) \text{ have } y \in \text{Closure}(\{x\}, T) \text{ by auto} \\
& \text{with clx have } \text{ineq1:} \text{Closure}(\{y\}, T) \subseteq \text{Closure}(\{x\}, T) \text{ using } \text{Top\_3\_L13} \\
& \text{by auto} \\
& \text{from } \langle y \in \bigcup T \rangle \text{ have } cly: \text{Closure}(\{y\}, T) \{\text{is closed in}T\} \text{ using } \text{cl\_is\_closed} \\
& \text{by auto} \\
& \text{with } \langle y \in \bigcup T \rangle \text{ have } (\bigcup T - \text{Closure}(\{y\}, T)) \in T \text{ using } \text{Top\_3\_L11(1)} \text{ unfolding } \text{IsClosed\_def} \text{ by auto} \\
& \end{align*}
\]

1059
Every $T_2$ space is anti-hyperconnected.

**Theorem (in topology0)**: $T_2$ Imp Anti HConn

**Proof**:

```isar
{ fix TT
assume TT{is a topology} TT{is hyperconnected} TT{is T_2}
{
  assume \( \bigcup TT = 0 \)
  then have \( \bigcup TT \leq 1 \) using empty_lepollI by auto
  then have \( (\bigcup TT) \{\text{is in the spectrum of}\} \text{IsHConnected using HConn_spectrum} \)
  by auto
}
moreover
{
  assume \( \bigcup TT \neq 0 \)
  then obtain \( x \) where \( x \in \bigcup TT \) by blast 
  
  fix \( y \)
  assume \( y \in \bigcup TT x \neq y \)
```
Every anti-hyperconnected space is $T_1$.

**Theorem anti_HConn_imp_T1:**
assumes $T$ is anti-$\text{IsHConnected}$
shows $T$ is $T_1$
proof-
fix $x$ $y$
assume $x \in \bigcup Ty \in \bigcup Tx \neq y$
by auto

with $\langle TT\{\text{is } T_2}\rangle \langle x \in \bigcup TT \rangle$ obtain $U$ $V$ where $U \cap V = 0$
unfolding $\text{is } T_2$ by blast
with $\langle TT\{\text{is hyperconnected}\} \rangle$ have False using $\text{IsHConnected}_\text{def}$

then have $\forall T$. $(T\{\text{is a topology}\} \land (T\{\text{hyperconnected}\}) \land (T\{\text{is } T_2\})) \rightarrow ((\bigcup T)\{\text{is in the spectrum of } \text{IsHConnected}\})$
by auto
moreover note here $T2$
ultimately have $\forall T$. $(T\{\text{is a topology}\} \land (T\{\text{is } T_2\})) \rightarrow (T\{\text{is } \text{IsHConnected}\})$
using $\text{Q-P_imp_Spec}[\text{where } P = \text{IsHConnected} \land Q = \text{is } T2]$ 
by auto
then show thesis using assms topSpaceAssum by auto

qed

with $\langle TT\{\text{is } T_2}\rangle \langle x \in \bigcup TT \rangle$ have $\bigcup TT = \{x\}$ by auto
then have $\bigcup TT \approx 1$ using $\text{singleton_eqpoll}_1$ by auto
then have $(\bigcup TT)\{\text{is in the spectrum of } \text{IsHConnected}\}$ using $\text{HConn_spectrum}$
by auto

ultimately have $\forall T$. $(T\{\text{is a topology}\} \land (T\{\text{is } T_2\})) \rightarrow ((\bigcup T)\{\text{is in the spectrum of } \text{IsHConnected}\})$
by auto

moreover note here $T2$
ultimately have $\forall T$. $T\{\text{is a topology}\} \rightarrow (T\{\text{is } T_2\}) \rightarrow (T\{\text{is } \text{IsHConnected}\})$
using $\text{Q_P_imp_Spec}[\text{where } P = \text{IsHConnected} \land Q = \text{is } T2]$ 
by auto
then show thesis using assms topSpaceAssum by auto

qed
then have \((T\{\text{restricted to}\}{x,y})\{\text{is hyperconnected}\}\) unfolding \text{IsHConnected\_def}\nby auto

with \text{assms}\{x,y\}\in\text{Pow}(\bigcup T) > \text{have} \{x,y\}\{\text{is in the spectrum of}\}\text{IsHConnected}\nunfolding \text{antiProperty\_def}\nby auto

then have \{x,y\}\leq 1 \text{ using } \text{HConn\_spectrum} \text{ by auto}\nmoreover
have \{x,y\} by auto
ultimately have \{x,y\}={x} using \text{lepoll\_1\_is\_sing}[of \{x,y\}] by auto
moreover
have \{y\} by auto
ultimately have \{y\} by auto
then have \{x\} by auto
with \langle x\neq y\rangle \text{ have False by auto}\n}

then have \exists U \in T. x \in U \land y \not\in U \text{ by auto}\n}
then show \text{thesis} using \text{isT1\_def} \text{ by auto}\nqed

There is at least one topological space that is \(T_1\), but not anti-hyperconnected.
This space is the cofinite topology on the natural numbers.

\textbf{lemma} \text{Cofinite\_not\_anti\_HConn}:\n\hspace{1em}\text{shows } \neg((\text{Cofinite }\text{nat})\{\text{is anti-}\}\text{IsHConnected}) \text{ and } (\text{Cofinite }\text{nat})\{\text{is } \text{T}_1\}\n\text{proof-}\n\hspace{1em}\{ \text{assume } (\text{Cofinite }\text{nat})\{\text{is anti-}\}\text{IsHConnected}\nmoreover
have \bigcup (\text{Cofinite }\text{nat})=\text{nat} \text{ unfolding } \text{Cofinite\_def} \text{ using } \text{union\_cocardinal}\nby auto
moreover
have (\text{Cofinite }\text{nat})\{\text{restricted to}\}\text{nat}=(\text{Cofinite }\text{nat}) \text{ using } \text{subspace\_cocardinal}\nunfolding \text{Cofinite\_def}\nby auto
moreover
have \neg(\text{nat}\prec\text{nat}) \text{ by auto}\nthen have (\text{Cofinite }\text{nat})\{\text{is hyperconnected}\} \text{ using } \text{Cofinite\_nat\_HConn}[of \text{nat}] \text{ by auto}\nultimately have \text{nat}\{\text{is in the spectrum of}\}\text{IsHConnected} \text{ unfolding } \text{antiProperty\_def}\nby auto
then have \text{nat}\leq 1 \text{ using } \text{HConn\_spectrum} \text{ by auto}\nmoreover
have 1\in\text{nat} \text{ by auto}\nthen have 1<\text{nat} \text{ using } \text{n\_lesspoll\_nat} \text{ by auto}\nultimately have \text{nat}<\text{nat} \text{ using } \text{lesspoll\_trans1} \text{ by auto}\nthen have \text{False by auto}\n\}\n
1062
then show \( \neg ((\text{CoFinite nat})\{\text{is anti-}\text{IsHConnected}\}) \) by auto
next
show \( (\text{CoFinite nat})\{\text{is T}_1\} \) using \( \text{cocardinal_is_T1 InfCard_nat} \) unfolding \( \text{Cofinite_def} \) by auto
qed

The join-topology build from the cofinite topology on the natural numbers, and the excluded set topology on the natural numbers excluding \{0,1\}; is just the union of both.

**lemma** join_top_cofinite_excluded_set:
shows \( (\text{joinT} \{\text{CoFinite nat},\text{ExcludedSet(nat,\{0,1\})}\})=(\text{CoFinite nat}) \cup \text{ExcludedSet(nat,\{0,1\})} \)

**proof**

have coftop:\( (\text{CoFinite nat})\{\text{is a topology}\} \) unfolding \( \text{Cofinite_def} \) using \( \text{CoCar_is_topology InfCard_nat} \) by auto
moreover
have ExcludedSet(nat,\{0,1\})\{\text{is a topology}\} using \( \text{excludedset_is_topology} \) by auto
moreover
have exuni:\( \bigcup \text{ExcludedSet(nat,\{0,1\})}=\text{nat} \) using \( \text{union_excludedset} \) by auto
moreover
have cofuni:\( \bigcup (\text{CoFinite nat})=\text{nat} \) using \( \text{union_cocardinal} \) unfolding \( \text{Cofinite_def} \) by auto
ultimately have \( (\text{joinT} \{\text{CoFinite nat},\text{ExcludedSet(nat,\{0,1\})}\}) = (\text{THE T. (CoFinite nat) } \cup \text{ExcludedSet(nat,\{0,1\}) \{is a subbase for\} T}) \) using \( \text{joinT_def} \) by auto
moreover
have \( \bigcup (\text{CoFinite nat}) \in \text{CoFinite nat} \) using \( \text{CoCar_is_topology[OF InfCard_nat]} \) unfolding \( \text{Cofinite_def IsATopology_def} \) by auto
with cofuni have \( n:\text{nat}\in\text{CoFinite nat} \) by auto
have Pa:\( (\text{CoFinite nat}) \cup \text{ExcludedSet(nat,\{0,1\})} \{\text{is a subbase for}\}\{\bigcup A. A \in \text{Pow}(\bigcap B. B \in \text{FinPow((CoFinite nat) \cup ExcludedSet(nat,\{0,1\}))})\}\) using \( \text{Top_subbase(2)} \) by auto
have \( \{\bigcup A. A \in \text{Pow}((\bigcap B. B \in \text{FinPow((CoFinite nat) \cup ExcludedSet(nat,\{0,1\}))}))\}=(\text{THE T. (CoFinite nat) } \cup \text{ExcludedSet(nat,\{0,1\})} \{\text{is a subbase for\} T}) \) using \( \text{same_subbase_same_top[where B=(CoFinite nat) \cup ExcludedSet(nat,\{0,1\}), OF _ Pa]\ the_equality[where a=\{\bigcup A. A \in \text{Pow}((\bigcap B. B \in \text{FinPow((CoFinite nat) \cup ExcludedSet(nat,\{0,1\}))}))\} \{\bigcup (CoFinite nat) \cup ExcludedSet(nat,\{0,1\})\} \{\text{is a subbase for\} \text{T}, OF _ Pa}\] by auto
ultimately have equal:\( (\text{joinT} \{\text{CoFinite nat},\text{ExcludedSet(nat,\{0,1\})}\}) = (\bigcup (A. A \in \text{Pow}((\bigcap B. B \in \text{FinPow((CoFinite nat) \cup ExcludedSet(nat,\{0,1\}))}))\) by auto

1063
moreover
have ExcludedSet(nat,{0,1}) ⊆ Pow(⋃ExcludedSet(nat,{0,1}))) by auto
moreover
note cofunl exuni
ultimately have sub:(CoFinite nat)∪ExcludedSet(nat,{0,1}) ⊆ Pow(nat)
by auto
from base have ∀S∈AU. S∈{∩B. B∈FinPow((CoFinite nat)∪ExcludedSet(nat,{0,1})))
by blast
then have ∀S∈AU. 3B∈FinPow((CoFinite nat)∪ExcludedSet(nat,{0,1})).
S=∩B by blast
then have eq:∀S∈AU. 3B∈Pow((CoFinite nat)∪ExcludedSet(nat,{0,1})).
S=∩B unfolding FinPow_def by blast
{ fix S assume S∈AU
  with eq obtain B where B∈Pow((CoFinite nat)∪ExcludedSet(nat,{0,1})))S=∩B
by auto
  with sub have B∈Pow(Pow(nat)) by auto
    { fix x assume x∈∩B
      then have ∀N∈B. x∈N≠0 by auto
        with <B∈Pow(Pow(nat))> have x∈nat by blast
    }
  with <S=∩B> have S∈Pow(nat) by auto
}
then have ∀S∈AU. S∈Pow(nat) by blast
with <U=⋃AU> have U∈Pow(nat) by auto
{ assume 0∈U
  with <U=⋃AU> obtain S where S∈AU0∈S1∈S by auto
  with base obtain BS where S=∩BS and bsbase:BS∈FinPow((CoFinite nat)∪ExcludedSet(nat,{0,1}))) by auto
  with <0∈S1∈S> have ∀M∈BS. 0∈M∈M by auto
  then have ∀M∈BS. M∈ExcludedSet(nat,{0,1})-{nat} unfolding ExcludedPoint_def ExcludedSet_def by auto
moreover
note bsbase n
ultimately have BS∈FinPow(CoFinite nat) unfolding FinPow_def by auto
moreover
from <0∈S1∈S> have S≠0 by auto
with <S=∩BS> have BS≠0 by auto
moreover
note coftop
ultimately have ⋂BS∈CoFinite nat using topology0.fin_inter_open_open[OF topology0_CoCardinal[OF InfCard_nat]]
  unfolding Cofinite_def by auto
with <S=∩BS> have S∈CoFinite nat by auto
with <0∈S1∈S> have nat-S<nat unfolding Cofinite_def CoCardinal_def by auto
moreover
from \( \bigcup_{A \in \mathcal{A}} S \subseteq U \) by auto
then have \( \text{nat-U} \subseteq \text{nat-S} \) by auto
then have \( \text{nat-U} \subseteq \text{nat-S} \) using subset_imp_lepoll by auto
ultimately
have \( \text{nat-U} \) using lesspoll_trans1 by auto
with \( \bigcup_{\mathcal{U} \in \text{Pow}(\text{nat})} \) have \( \mathcal{U} \in \text{CoFinite nat} \) unfolding Cofinite_def CoCardinal_def
by auto
then have \( \text{nat-U} \subseteq \text{nat-S} \) using subset_imp_lepoll by auto
ultimately have \( \text{nat-U} \ll \text{nat-S} \) using lesspoll_trans1 by auto
with \( \bigcup_{\mathcal{U} \in \text{Pow}(\text{nat})} \) have \( \mathcal{U} \in \text{CoFinite nat} \) unfolding Cofinite_def CoCardinal_def
by auto
moreover
\[
\bigcup_{\bigcup_{A \in \mathcal{A}} S \subseteq \bigcup_{B \in \text{FinPow}((\text{CoFinite nat}) \cup \text{ExcludedSet(nat,\{0,1\}))}} \bigcup_{\mathcal{U} \in \text{Pow}(\bigcup_{B \in \text{FinPow}((\text{CoFinite nat}) \cup \text{ExcludedSet(nat,\{0,1\}))}})
\]
\( \subseteq (\text{CoFinite nat}) \cup \text{ExcludedSet(nat,\{0,1\})} \)
by blast
moreover
\{ fix \( \mathcal{U} \in (\text{CoFinite nat}) \cup \text{ExcludedSet(nat,\{0,1\})} \)
assume \( \mathcal{U} \in (\text{CoFinite nat}) \cup \text{ExcludedSet(nat,\{0,1\})} \)
then have \( \mathcal{U} \in \text{FinPow}((\text{CoFinite nat}) \cup \text{ExcludedSet(nat,\{0,1\}))} \)
by unfolding FinPow_def by auto
then have \( \mathcal{U} \in \bigcup_{\bigcup_{B \in \text{FinPow}((\text{CoFinite nat}) \cup \text{ExcludedSet(nat,\{0,1\}))}} \bigcup_{\mathcal{U} \in \text{Pow}(\bigcup_{B \in \text{FinPow}((\text{CoFinite nat}) \cup \text{ExcludedSet(nat,\{0,1\}))}}) \)
by blast
moreover
have \( \mathcal{U} = \bigcup_{\mathcal{U}} \) by auto
ultimately have \( \mathcal{U} \in \bigcup_{\bigcup_{A \in \text{Pow}(\bigcup_{B \in \text{FinPow}((\text{CoFinite nat}) \cup \text{ExcludedSet(nat,\{0,1\}))}}) \}
\bigcup_{\mathcal{U} \in \text{ExcludedSet(nat,\{0,1\})}} \)
by blast
\}
then have \( (\text{CoFinite nat}) \cup \text{ExcludedSet(nat,\{0,1\})} \subseteq (\bigcup_{A \in \text{Pow}(\bigcup_{B \in \text{FinPow}((\text{CoFinite nat}) \cup \text{ExcludedSet(nat,\{0,1\}))}}) \}
\cup (\bigcup_{\mathcal{U} \in \text{ExcludedSet(nat,\{0,1\})}}) \)
by auto
ultimately have \( (\text{CoFinite nat}) \cup \text{ExcludedSet(nat,\{0,1\})} = (\bigcup_{A \in \text{Pow}(\bigcup_{B \in \text{FinPow}((\text{CoFinite nat}) \cup \text{ExcludedSet(nat,\{0,1\}))}}) \}
\cup (\bigcup_{\mathcal{U} \in \text{ExcludedSet(nat,\{0,1\})}}) \)
by auto
with equal show thesis by auto
qed

The previous topology in not \( T_2 \), but is anti-hyperconnected.

**Theorem join_Cofinite_ExclPoint_not_T2:**
shows
\[ \neg ((\text{joinT} \hspace{1em} (\text{CoFinite nat}, \text{ExcludedSet(nat,\{0,1\}))}) \{\text{is } T_2\}) \text{ and } (\text{joinT} \hspace{1em} (\text{CoFinite nat}, \text{ExcludedSet(nat,\{0,1\}))}) \{\text{is anti- } T_2\} \text{ IsHConnected} \]
proof-
have \( (\text{CoFinite nat}) \subseteq (\text{CoFinite nat}) \cup \text{ExcludedSet(nat,\{0,1\})} \) by auto
have \( \bigcup ((\text{CoFinite nat}) \cup \text{ExcludedSet(nat,\{0,1\})}) = (\bigcup (\text{CoFinite nat}) \cup (\bigcup \text{ExcludedSet(nat,\{0,1\})}) \)
by auto
moreover have ...=nat unfolding Cofinite_def using union_cocardinal union_excludedset
by auto
ultimately have tot:\bigcup((\text{CoFinite nat}) \cup \text{ExcludedSet(nat,\{0,1\}))=nat by auto
{
assume (joinT \{\text{CoFinite nat,ExcludedSet(nat,\{0,1\}))\} {is T}_2
then have t2:((\text{CoFinite nat}) \cup \text{ExcludedSet(nat,\{0,1\})){is T}_2 using
join_top_cofinite_excluded_set by auto
with tot have \exists U \in ((\text{CoFinite nat}) \cup \text{ExcludedSet(nat,\{0,1\})). \exists V \in ((\text{CoFinite
nat}) \cup \text{ExcludedSet(nat,\{0,1\})). 0 \in U \land 1 \in V \land U \cap V=0
unfolding ExcludedSet_def by auto
then have U V where U \in (\text{CoFinite nat}) \lor (0 \notin U \land 1 \notin U)V \in (\text{CoFinite
nat}) \lor (0 \notin V \land 1 \notin V)0 \in U \land V \land U \cap V=0
unfoldingRestrictedTo_def by auto
with <0 \in U \land 1 \in V have U \land V \neq 0 using Cofinite_nat_HConn IsHConnected_def
by auto
with \{0,1\} \cap U=0 have False by auto
}\then show \neg((\text{joinT }\{\text{CoFinite nat,ExcludedSet(nat,\{0,1\}))\})\{is T}_2) by auto
{
fix A assume AS:A \in \bigcup((\text{CoFinite nat}) \cup \text{ExcludedSet(nat,\{0,1\}))\{restricted to}A{is hyperconnected}
with tot have A \in \text{Pow(nat)} by auto
then have sub:A \cap \text{nat}=A by auto
have ((\text{CoFinite nat}) \cup \text{ExcludedSet(nat,\{0,1\}))\{restricted to}A=((\text{CoFinite
nat})\{restricted to}A) \cup \text{ExcludedSet(nat,\{0,1\})\{restricted to}A
unfolding RestrictedTo_def by auto
also from sub have ...=(\text{CoFinite A}) \cup \text{ExcludedSet(A,\{0,1\})) using subspace_excludedset[of nat
subspace_cocardinal[of natnatA] unfolding Cofinite_def
by auto
finally have ((\text{CoFinite nat}) \cup \text{ExcludedSet(nat,\{0,1\}))\{restricted to}A=(\text{CoFinite
A}) \cup \text{ExcludedSet(A,\{0,1\})) by auto
with AS(2) have eq:((\text{CoFinite A}) \cup \text{ExcludedSet(A,\{0,1\}))\{is hyperconnected}
by auto
{
assume \{0,1\} \cap A=0
then have (\text{CoFinite A}) \cup \text{ExcludedSet(A,\{0,1\}))=\text{Pow(A)} using empty_excludedset[of
\{0,1\}A] unfolding Cofinite_def CoCardinal_def
by auto
with eq have \text{Pow(A)}\{is hyperconnected\} by auto
then have \text{Pow(A)}\{is connected\} using HConn_imp_Conn by auto
moreover have \text{Pow(A)}\{is anti-}IsConnected using discrete_tot_dis unfolding IsTotDis_def by auto
moreover have \bigcup(\text{Pow(A)}) \subseteq \text{Pow(\bigcup(\text{Pow(A))}} by auto

moreover have $\text{Pow}(A)\{\text{restricted to}\}\bigcup (\text{Pow}(A)) = \text{Pow}(A)$ unfolding $\text{RestrictedTo_def}$ by blast
ultimately have $(\bigcup (\text{Pow}(A)))\{\text{is in the spectrum of}\}\text{IsConnected unfolding } \text{antiProperty_def}$ by auto
then have $A\{\text{is in the spectrum of}\}\text{IsConnected by auto}$
then have $A\leq 1$ using $\text{conn_spectrum}$ by auto
then have $A\{\text{is in the spectrum of}\}\text{IsHConnected using } \text{HConn_spectrum}$ by auto
moreover
\[
\{ \\
\text{assume } AS:\{0,1\}\cap A \neq 0 \\
\text{ assume } A=\{0\}\cup A=\{1\} \\
\text{ then have } A\approx 1 \text{ using } \text{singleton_eqpoll_1 by auto} \\
\text{ then have } A\leq 1 \text{ using } \text{eqpoll_imp_lepoll by auto} \\
\text{ then have } A\{\text{is in the spectrum of}\}\text{IsHConnected using } \text{HConn_spectrum} \\
\}
moreover
\[
\{ \\
\text{assume } AS2:\neg (A=\{0\}\cup A=\{1\}) \\
\text{ assume } AS3:A\subseteq \{0,1\} \\
\text{ with } AS AS2 \text{ have } A_{\text{def}}: A=\{0,1\} \text{ by blast} \\
\text{ then have } \text{ExcludedSet}(A,\{0,1\})=\text{ExcludedSet}(A,A) \text{ by auto} \\
\text{ moreover have } \text{ExcludedSet}(A,A)=\{0,A\} \text{ unfolding } \text{ExcludedSet_def} \\
\}
\text{by blast}
ultimately have $\text{ExcludedSet}(A,\{0,1\})=\{0,A\}$ by auto
moreover have $0\in (\text{CoFinite A})$ using $\text{empty_open[of CoFinite A]}$ 
\text{CoCar_is_topology[of InfCard_nat,of A]} \text{ unfolding } \text{Cofinite_def} 
by auto
moreover have $\bigcup (\text{CoFinite A})=A$ using $\text{union_cocardinal unfolding } \text{Cofinite_def}$ by auto
then have $A\in (\text{CoFinite A})$ using $\text{CoCar_is_topology[of InfCard_nat,of A]}$ unfolding $\text{Cofinite_def}$ 
\text{IsATopology_def by auto} 
ultimately have $(\text{CoFinite A})\cup \text{ExcludedSet}(A,\{0,1\})=(\text{CoFinite A})$ by auto
with $eq\text{have}(\text{CoFinite A})\{\text{is hyperconnected}\}$ by auto
with $A_{\text{def}}\text{ have } \text{hyp}(\text{CoFinite } \{0,1\})\{\text{is hyperconnected}\}$ by auto
have $\{0\}\approx 1\{1\}\approx 1$ using $\text{singleton_eqpoll_1 by auto}$
moreover have $1\prec \text{nat}$ using $\text{n_lesspoll_nat by auto}$
ultimately have \{0\}≺\text{nat}\{1\}≺\text{nat} using eq_lesspoll_trans by auto
moreover
have \{0,1\}≺\{0\}\{0,1\}≺\text{nat} by auto
ultimately have \{1\}∈\text{CoFinite} \{0,1\}\{0\}∈\text{CoFinite} \{0,1\} \{1\}∩\{0\}=0
unfolding Cofinite_def CoCardinal_def
   by auto
with hyp have False unfolding IsHConnected_def by auto
}
then obtain \(t\) where \(t\in A\, t\neq 0\, t\neq 1\) by auto
then have \(\{t\}\in\text{ExcludedSet}(A,\{0,1\})\) unfolding ExcludedSet_def
by auto
moreover
{ have \(\{t\}\approx 1\) using singleton_eqpoll_1 by auto
moreover
have 1≺\text{nat} using n_lesspoll_nat by auto
ultimately have \(\{t\}≺\text{nat}\) using eq_lesspoll_trans by auto
moreover
with \(\langle t\in A \rangle\) have \(A-(A-\{t\})=\{t\}\) by auto
ultimately have \(A-\{t\}∈\text{CoFinite} A\) unfolding Cofinite_def CoCardinal_def
   by auto
}
ultimately have \(\{t\}∈((\text{CoFinite} A)∪\text{ExcludedSet}(A,\{0,1\})) A-\{t\}∈((\text{CoFinite} A)∪\text{ExcludedSet}(A,\{0,1\}))\)
   by auto
with eq have \(A-\{t\}=0\) unfolding IsHConnected_def by auto
with \(\langle t\in A \rangle\) have \(A=\{t\}\) by auto
then have \(A\approx 1\) using singleton_eqpoll_1 by auto
then have \(A\leq 1\) using eqpoll_imp_lepoll by auto
then have \(A\) is in the spectrum of \(\text{IsHConnected}\) using HConn_spectrum
by auto
}
ultimately have \(A\) is in the spectrum of \(\text{IsHConnected}\) by auto
}
ultimately have \(A\) is in the spectrum of \(\text{IsHConnected}\) by auto
}
then have \(((\text{CoFinite} \text{nat})∪\text{ExcludedSet}(\text{nat},\{0,1\}))\) is anti-\(\text{IsHConnected}\)
unfolding antiProperty_def
   by auto
then show \((\text{joinT} \{\text{CoFinite nat}, \text{ExcludedSet}(\text{nat},\{0,1\})\})\) is anti-\(\text{IsHConnected}\)
using join_top_cofinite_excluded_set
by auto
qed

Let’s show that anti-hyperconnected is in fact \(T_1\) and sober. The trick of
the proof lies in the fact that if a subset is hyperconnected, its closure is so
too (the closure of a point is then always hyperconnected because singletons
are in the spectrum); since the closure is closed, we can apply the sober
property on it.
theorem (in topology0) T1_sober_imp_anti_HConn:
  assumes T{is T_1} and T{is sober}
  shows T{is anti-}IsHConnected
proof-
  { fix A assume AS:A ∈ Pow(⋃ T){restricted to}A{is hyperconnected}
    { assume A=0
      then have A ≲ 1 using empty_lepollI by auto
      then have A{is in the spectrum of}IsHConnected using HConn_spectrum
      by auto
    } moreover
    { assume A≠0
      then obtain x where x∈A by blast
      { assume ¬((T{restricted to}Closure(A,T)){is hyperconnected})
        then obtain U V where UV_def:U ∈ (T{restricted to}Closure(A,T))V ∈ (T{restricted to}Closure(A,T))
          unfolding RestrictedTo_def by auto
        from ⟨A∈Pow(⋃ T)⟩ have A⊆Closure(A,T) using cl_contains_set by auto
        then have UCA VCA where UCA∈T VCA∈T U=UCA∩Closure(A,T)V=VCA∩Closure(A,T)
          unfolding RestrictedTo_def by auto
        with ⟨A⊆Closure(A,T)⟩ have UCA∩A⊆UCA∩Closure(A,T) VCA∩A⊆VCA∩Closure(A,T)
          by auto
        moreover
        from ⟨UCA∈T⟩⟨VCA∈T⟩ have UCA∩A∈(T{restricted to}A)VCA∩A∈(T{restricted to}A)
          unfolding RestrictedTo_def by auto
        moreover
        note AS(2)
        ultimately have UCA∩A=0 VCA∩A=0 using IsHConnected_def by auto
        with ⟨A⊆Closure(A,T)⟩ have A⊆Closure(A,T)-UCA VCA⊆Closure(A,T)-VCA
        by auto
        moreover
        { have Closure(A,T)-UCA=Closure(A,T)∩(⋃ T-UCA) VCA=Closure(A,T)∩(⋃ T-VCA)
          using Top_3_L11(1) AS(1) by auto
        }
        moreover
        with ⟨UCA∈T⟩⟨VCA∈T⟩ have (⋃ T-UCA){is closed in}T(⋃ T-VCA){is closed in}TClosure(A,T){is closed in}T
          using Top_3_L9 cl_is_closed AS(1) by auto
        ultimately have (Closure(A,T)-UCA){is closed in}T(Closure(A,T)-VCA){is closed in}T
          using Top_3_L5(1) by auto
      } }
ultimately have Closure(A,T) ⊆ Closure(A,T) - UCA ∪ Closure(A,T) - VCA using Top_3_L13
   by auto
   then have UCA ∩ Closure(A,T) = 0 ∨ VCA ∩ Closure(A,T) = 0 by auto
   with "U = UCA ∩ Closure(A,T)" "V = VCA ∩ Closure(A,T)" have U = 0 ∨ V = 0 by auto
   with "U ≠ 0" "V ≠ 0" have False by auto

then have (T restricted to) Closure(A,T) {is hyperconnected} by auto
moreover have Closure(A,T) {is closed in} T using cl_is_closed AS(1) by auto
moreover from "x ∈ A" have Closure(A,T) ≠ 0 using cl_contains_set AS(1) by auto
moreover from AS(1) have Closure(A,T) ⊆ ∪ T using Top_3_L11(1) by auto
ultimately have Closure(A,T) ∈ Pow(∪ T) - {0} (T restricted to Closure(A,T)) {is hyperconnected}
   by auto
moreover note asms(2)
ultimately have ∃ x ∈ T. (Closure(A,T) = Closure({x}, T) ∧ (∀ y ∈ ∪ T. Closure(A,T) = Closure({y}, T) → y = x)) unfolding IsSober_def
   by auto
   then obtain y where y ∈ ∪ T Closure(A,T) = Closure({y}, T) by auto
   moreover
   { fix z assume z ∈ (∪ T) - {y}
     with asms(1) "y ∈ ∪ T" obtain U where U ∈ T z ∈ U y /∈ U using isT1_def
     by blast
     then have U ∈ T z ∈ U U ⊆ (∪ T) - {y} by auto
     then have ∃ U ∈ T. z ∈ U ∧ U ⊆ (∪ T) - {y} by auto
   } then have ∀ z ∈ (∪ T) - {y}. ∃ U ∈ T. z ∈ U ∧ U ⊆ (∪ T) - {y} by auto
   then have (∪ T) - {y} ∈ T using open_neigh_open by auto
   with "y ∈ ∪ T" have {y} {is closed in} T using IsClosed_def by auto
   with "y ∈ ∪ T" have Closure({y}, T) = {y} using Top_3_L8 by auto
   with "Closure(A,T) = Closure({y}, T)" have Closure(A,T) = {y} by auto
   with AS(1) have A ⊆ {y} using cl_contains_set[of A] by auto
   with "A ≠ 0" have A = {y} by auto
   then have A ≈ 1 using singleton_eqpoll_1 by auto
   then have A ⊆ 1 using eqpoll_imp_lepoll by auto
   then have A {is in the spectrum of} IsHConnected using HConn_spectrum
   by auto
   } ultimately have A {is in the spectrum of} IsHConnected by blast
}
then show thesis using antiProperty_def by auto

1070
A space is ultraconnected iff every two non-empty closed sets meet.

Every ultraconnected space is trivially normal.

Every ultraconnected space is connected.

\begin{proof}
\begin{itemize}
\item A space is ultraconnected iff every two non-empty closed sets meet.
\item Every ultraconnected space is trivially normal.
\item Every ultraconnected space is connected.
\end{itemize}
\end{proof}
lemma UConn_spectrum:
shows (A{is in the spectrum of}IsUConnected) ⟷ A≲1
proof
  assume A_spec:(A{is in the spectrum of}IsUConnected)
  \{ assume A=0
    then have A≲1 using empty_lepollI by auto \}
  moreover
  \{ assume A≠0
    from A_spec have ∀T. (T{is a topology}∧⋃T=A) ⟷ (T{is ultraconnected}) unfolding Spec_def by auto
    moreover
    have Pow(A){is a topology} using Pow_is_top by auto
    moreover
    have ⋃Pow(A)=A by auto
    then have ⋃Pow(A)≈A by auto
    ultimately have ult:Pow(A){is ultraconnected} by auto
    moreover
    from A≠0 obtain b where b∈A by auto
    then have \{b\}{is closed in}Pow(A) unfolding IsClosed_def by auto
      \{ fix c
        assume c∈Ac≠b
        then have \{c\}{is closed in}Pow(A){c}∩\{b}=0 unfolding IsClosed_def by auto
          with ult \{b\}{is closed in}Pow(A) have False using IsUConnected_def by auto
          by auto
        \}
        with b∈A have A={b} by auto
        then have A≈1 using singleton_eqpoll_1 by auto
        then have A≲1 using eqpoll_imp_lepoll by auto
      \}
      ultimately show A≲1 by auto
  next
  assume A≲1
  \{ fix T
    assume T{is a topology}⋃T=A
    \{ assume ∪T=0
      with T{is a topology} have T=∅ using empty_open by auto
      then have T{is ultraconnected} unfolding IsUConnected_def IsClosed_def by auto
    \}
    moreover
    \{
    \}
  \}
assume $\bigcup T \neq 0$

moreover from $A \subseteq 1 \land \bigcup T \approx A$ have $\bigcup T \subseteq 1$ using eq_lepoll_trans by auto

ultimately obtain $E$ where $eq : \bigcup T = \{E\}$ using lepoll_1_is_sing by blast

moreover have $T \subseteq \text{Pow}(\bigcup T)$ by auto

ultimately have $T \subseteq \text{Pow}(\{E\})$ by auto

then have $T \subseteq \{0, \{E\}\}$ using empty_open by auto

then have $T \subseteq \{0, \{E\}\}$ unfolding IsUConnected_def IsClosed_def by (simp only: eq, safe, force)

ultimately have $T \subseteq \{0, \{E\}\}$ by auto

then show $A \subseteq \bigcup T \approx A$ using empty_open by auto

qed

This time, anti-ultraconnected is an old property.

definition anti_UConn :: "T ⇒ bool" where
  anti_UConn_def: "anti_UConn T ≡ T \subseteq \{0, \{E\}\}"

lemma anti_UConnI [intro]: "anti_UConn T" apply (simp only: anti_UConn_def)

proof
  assume $T \subseteq \{0, \{E\}\}$ by auto

then show $T \subseteq \{0, \{E\}\}$ using empty_open by auto

qed

This time, anti-ultraconnected is an old property.

theorem (in topology0) anti_UConn:
  shows $\langle T \text{ is anti-} \rangle \iff T \subseteq \{0, \{E\}\}$

proof
  assume $T \subseteq \{0, \{E\}\}$

  fix $TT$

  assume $TT \subseteq \{0, \{E\}\}$ by auto

  then have $TT \subseteq 1$ using empty_lepollI by auto

  then have $TT \subseteq 1$ using empty_lepollI by auto

  then obtain $t$ where $t \in \bigcup TT$ by blast

  fix $x$

  assume $x \in \bigcup TT$

  then obtain $U$ where $U \in TT \land y \in U \land x \notin U$ using isT1_def

  then have $U \in TT \land y \in U \land x \notin U$ by blast

  then have $\exists U \in TT \cdot y \in U \land x \notin U$ by auto

  then have $\exists U \in TT \cdot y \in U \land x \notin U$ by auto

  qed
then have \( \forall y \in (\bigcup \mathcal{T}) - \{x\}. \exists U \in \mathcal{T}. y \in U \land U \subseteq (\bigcup \mathcal{T}) - \{x\} \) by auto

with \( \langle \mathcal{T} \text{ is a topology} \rangle \) have \( \bigcup \mathcal{T} - \{x\} \in \mathcal{T} \) using topology0.open_neigh_open topology0_def by auto

unfolding with \( p \) have \( \{x\} \) is closed in \( \mathcal{T} \) using IsClosed_def by auto

finally have \( \bigcup \mathcal{T} - \{x\} \in \mathcal{T} \) unfolding topology0_def by auto

with \( t \in \bigcup \mathcal{T} \) have \( t_{\text{cl}} \) is closed in \( \mathcal{T} \) by auto

{ fix \( x \) assume \( x \notin \mathcal{T} \) \}

ultimately have \( \bigcup \mathcal{T} \) is in the spectrum of \( \text{IsUConnected} \) using UConn_spectrum by auto

ultimately have \( \bigcup \mathcal{T} \) is in the spectrum of \( \text{IsUConnected} \) by blast

then have \( (\mathcal{T} \text{ is a topology}) \land (\mathcal{T} \text{ is T}_1) \land (\mathcal{T} \text{ is ultraconnected}) \) \( \implies \) \( ((\bigcup \mathcal{T}) \text{ is in the spectrum of} \text{IsUConnected}) \) by auto

then have \( \forall \mathcal{T}. (\mathcal{T} \text{ is a topology}) \land (\mathcal{T} \text{ is T}_1) \land (\mathcal{T} \text{ is ultraconnected}) \) \( \implies \) \( ((\bigcup \mathcal{T}) \text{ is in the spectrum of} \text{IsUConnected}) \) by auto

moreover note \( \text{here T}_1 \)

ultimately have \( \forall T. T \text{ is a topology} \implies (T \text{ is T}_1) \implies (T \text{ is anti-} \text{IsUConnected}) \) using Q_P_imp_Spec[where \( Q = \text{isT}_1 \) and \( P = \text{IsUConnected} \)] by auto

with \( \langle T \text{ is T}_1 \rangle \) show \( T \text{ is anti-} \text{IsUConnected} \) by auto

next assume \( \text{ASS} : T \text{ is anti-} \text{IsUConnected} \)

{ fix \( x \ y \) assume \( x \in \bigcup \mathcal{T} y \in \bigcup \mathcal{T} \neq y \)

then have \( \text{tot} : \bigcup (T \{\text{restricted to}\} \{x, y\}) = \{x, y\} \) unfolding RestrictedTo_def by auto

{ assume \( \text{AS} : \forall U \in T. x \in U \implies y \in U \)

{ assume \( \{y\} \) is closed in \( (T \{\text{restricted to}\} \{x, y\}) \) moreover from \( \langle x \neq y \rangle \) have \( \{x, y\} - \{y\} = \{x\} \) by auto

1074
ultimately have \( \{x\} \in (T\text{ restricted to}\{x,y\}) \) unfolding IsClosed_def
by (simp only:tot)
then obtain \( U \) where \( U \cap T(x) = \{x,y\} \cap U \) unfolding RestrictedTo_def
by auto
moreover
with \( \langle x \neq y \rangle \) have \( y \notin \{x\} \) \( y \in \{x,y\} \) by (blast+)
with \( \langle \{x\} = (x,y) \cap U \rangle \) have \( y \notin U \) by auto
moreover have \( x \in \{x\} \) by auto
with \( \langle \{x\} = (x,y) \cap U \rangle \) have \( x \in U \) by auto
ultimately have \( x \in \{y\} \notin U \) \( x \in T \) by auto
with \( A \) have False by auto

then have \( y_{no\_cl} : \neg (\{y\} \text{ is closed in} (T\text{ restricted to}\{x,y\})) \) by auto
{
fix \( A \ B \)
assume \( cl : A \) \( (is\_closed\_in) (T\text{ restricted to}\{x,y\})B \) \( (is\_closed\_in) (T\text{ restricted to}\{x,y\})A \)
with tot have \( A \subseteq \{x,y\} \) \( B \subseteq \{x,y\} \) \( A \cap B = 0 \) unfolding IsClosed_def by auto
then have \( x \in A \rightarrow x \notin B \rightarrow y \notin B \subseteq \{x,y\} \) \( B \subseteq \{x,y\} \) by auto
{
assume \( x \in A \)
with \( \langle x \in A \rightarrow x \notin B \rightarrow B \subseteq \{x,y\} \rangle \) have \( B \subseteq \{y\} \) by auto
then have \( B = \emptyset \lor B = \{y\} \) by auto
with \( y_{no\_cl} \) \( cl(2) \) have \( B = \emptyset \) by auto
}
moreover
{
assume \( x \notin A \)
with \( \langle A \subseteq \{x,y\} \rangle \) have \( A \subseteq \{y\} \) by auto
then have \( A = \emptyset \lor A = \{y\} \) by auto
with \( y_{no\_cl} \) \( cl(1) \) have \( A = \emptyset \) by auto
}
ultimately have \( A = \emptyset \lor B = 0 \) by auto
}
thcn have \( (T\text{ restricted to}\{x,y\}) \) \( (is\_ultraconnected) \) unfolding IsUConnected_def
by auto
with \( \text{Ass} \ \langle x \in \bigcup T \rangle \langle y \in \bigcup T \rangle \) have \( \{x,y\} \) \( (is\_in\_the\_spectrum\_of) \) \( IsUConnected \)
unfolding antiProperty_def
by auto
then have \( (x,y) \subseteq 1 \) using UConn_spectrum by auto
moreover have \( x \in \{x,y\} \) by auto
ultimately have \( x = (x,y) \) using lepoll_1_is_sing[of \( \{x,y\} \) x] by auto
moreover
have \( y \in \{x,y\} \) by auto
ultimately have \( y \in \{x\} \) by auto
then have \( y = x \) by auto
then have False using \( \langle x \neq y \rangle \) by auto

1075
then have $\exists U \in T. x \in U \land y \notin U$ by auto

then show $T$ is $T_1$ unfolding $isT1_def$ by auto

qed

Is it natural that separation axioms and connection axioms are anti-properties of each other; as the concepts of connectedness and separation are opposite.

To end this section, let’s try to characterize anti-sober spaces.

**lemma sober_spectrum:**

shows $(A \text{ is in the spectrum of } \text{IsSober}) \iff A \subsetneq 1$

**proof**

assume $A \text{ is in the spectrum of } \text{IsSober}$

{ assume $A = 0$
  then have $A \subsetneq 1$ using $empty_lepollI$ by auto
}

moreover

{ assume $A \neq 0$
  note $A$
  moreover have $\top: \{0, A\} \text{ is a topology}$ unfolding $\text{IsATopology_def}$ by auto
  moreover have $\bigcup \{0, A\} = A$ by auto
  then have $\bigcup \{0, A\} \approx A$ by auto
  ultimately have $\{0, A\} \text{ is sober}$ using $\text{Spec_def}$ by auto
  moreover have $\{0, A\} \text{ is hyperconnected}$ using $\text{Indiscrete_HConn}$ by auto
  moreover have $\{0, A\} \text{ restricted to } A = \{0, A\}$ unfolding $\text{RestrictedTo_def}$ by auto
  moreover have $A \text{ is closed in } \{0, A\}$ unfolding $\text{IsClosed_def}$ by auto
  note $\langle A \neq 0 \rangle$
  ultimately have $\exists x \in A. A = \text{Closure}(\{x\}, \{0, A\}) \land (\forall y \in \bigcup \{0, A\}. A = \text{Closure}(\{y\}, \{0, A\})) \implies y = x$ unfolding $\text{IsSober_def}$ by auto
  then obtain $x$ where $x \in A = \text{Closure}(\{x\}, \{0, A\})$ and $\forall y \in A. A = \text{Closure}(\{y\}, \{0, A\}) \implies y = x$ by auto
  { fix $y$ assume $y \in A$
    with $\top$ have $\text{Closure}(\{y\}, \{0, A\}) \text{ is closed in } \{0, A\}$ using $\text{topology0.cl_is_closed}$
    $\text{topology0_def}$ by auto
    moreover
    from $\langle y \in A \rangle$ top have $y \in \text{Closure}(\{y\}, \{0, A\})$ using $\text{topology0.cl_contains_set}$
    $\text{topology0_def}$ by auto
    ultimately have $A - \text{Closure}(\{y\}, \{0, A\}) \in \{0, A\} \setminus \{0, A\}$
    unfolding $\text{IsClosed_def}$
  }

}
by auto
then have $\text{A-Closure}\{y\},\{0,A\}\neq\text{A-Closure}\{y\},\{0,A\}$=0 by auto
moreover
from $\langle y\in A\rangle \langle y\in \text{Closure}\{y\},\{0,A\}\rangle$ have $y\in y\notin A$-Closure($\{y\},\{0,A\}$)
by auto
ultimately have $A\text{-Closure}\{y\},\{0,A\}\neq\text{A-Closure}\{y\},\{0,A\}=A$, simp, auto
moreover
from $\langle y\in A\rangle$ top have Closure($\{y\},\{0,A\})\subseteq A$ using topology0_def topology0.Top_3_L11(1)
by blast
then have $A\text{-Closure}\{y\},\{0,A\}=\text{A-Closure}\{y\},\{0,A\}$ by auto
ultimately have $A\text{-Closure}\{y\},\{0,A\}=\text{A-Closure}\{y\},\{0,A\}$ by auto

with reg have $\forall y\in A. x=y$ by auto
with $\langle x\in A\rangle$ have $A=\{x\}$ by blast
then have $A\approx1$ using singleton_eqpoll_1 by auto
then have $A\leq1$ using eqpoll_imp_lepoll by auto
ultimately have $A\approx1$ by auto

next
assume $A\leq1$

{ fix $T$ assume $T$\{is a topology}$\cup T\approx A$
  { assume $\cup T=0$
    then have $T$\{is sober} unfolding IsSober_def by auto
  }
mOREOVER
  { assume $\cup T\neq 0$
    then obtain $x$ where $x\in \cup T$ by blast
    moreover
    from $\langle \cup T\approx A\rangle \langle A\leq1\rangle$ have $\cup T\leq1$ using eq_lepoll_trans by auto
    ultimately have $\cup T=\{x\}$ using lepoll_i_is_sing by auto
    moreover
    have $T\subseteq \text{Pow}\{\cup T\}$ by auto
    ultimately have $T\subseteq \text{Pow}\{\{x\}\}$ by auto
    then have $T\subseteq\{0,\{x\}\}$ by blast
    moreover
    from $\langle T$\{is a topology}\rangle$ have $0\in T$ using empty_open by auto
    moreover
    from $\langle T$\{is a topology}\rangle$ have $\cup T\in T$ unfolding IsATopology_def by auto
    with $\langle \cup T=\{x\}\rangle$ have $\{x\}\in T$ by auto
    ultimately have $T$\_def:$T=\{0,\{x\}\}$ by auto
    then have $dd:\text{Pow}\{\cup T\}$-{0}$=\{\{x\}\}$ by auto
    { fix $B$ assume $B\in \text{Pow}\{\cup T\}$-{0}


with dd have B_def:B={x} by auto
from ‹T is a topology› have (⋃T){is closed in}T using topology0_def
topology0.Top_3_L1
  by auto
with ‹⋃T={x}› ‹T is a topology› have Closure({x},T)={x} using topology0_def
  unfolding topology0_def by auto
moreover
{  
  fix y assume y∈⋃T
  with ‹⋃T={x}› have y=x by auto
}
then have ( ∀ y∈⋃T. B = Closure({y}, T) → y = x ) by auto
moreover note ‹x∈⋃T›
ultimately have ( ∃ x∈⋃T. B = Closure({x}, T) ∧ ( ∀ y∈⋃T. B = Closure({y}, T) → y = x ) )
  by auto
} then have T{is sober} unfolding IsSober_def by auto
} ultimately have T{is sober} by blast
} then show A {is in the spectrum of} IsSober unfolding Spec_def by auto
qed

theorem (in topology0)anti_sober:
  shows ‹T{is anti-}IsSober› ↔ T={0, ⋃T}
proof
  assume T={0, ⋃T}
  {  
    fix A assume A∈Pow(⋃T)(T{restricted to}A){is sober}
    {  
      assume A=0
      then have A≤1 using empty_lepollI by auto
      then have A{is in the spectrum of}IsSober using sober_spectrum
    by auto
    }
  moreover
  {  
    assume A≠0
    have ⋃T∈{0, ⋃T}0∈{0, ⋃T} by auto
    with ‹T={0, ⋃T}› have (⋃T)∈T 0∈T by auto
    with ‹A∈Pow(⋃T)› have {0, A}⊆(T{restricted to}A) unfolding RestrictedTo_def
    by auto
    moreover
    have ∀ B∈{0, ⋃T}. B=0∨B=⋃T by auto
    with ‹T={0, ⋃T}› have ∀ B∈T. B=0∨B=⋃T by auto
    with ‹A∈Pow(⋃T)› have T{restricted to}A⊆{0, A} unfolding RestrictedTo_def
  }
ultimately have $T \langle \text{restricted to} \rangle A = \{0, A\}$ by auto

moreover
have $A$ (is closed in) $\{0, A\}$ unfolding $\text{IsClosed}_A$ by auto

moreover
have $\{0, A\}$ (is hyperconnected) using $\text{Discrete}_HConn$ by auto

moreover
from $\langle A \in \text{Pow}(\bigcup T) \rangle$ have $(T \langle \text{restricted to} \rangle A) \langle \text{restricted to} \rangle A = T \langle \text{restricted to} \rangle A$ using $\text{subspace_of_subspace} [\text{of } A A T]$

by auto

moreover
note $\langle A \neq 0 \rangle$ $\langle A \in \text{Pow}(\bigcup T) \rangle$
ultimately have $A \langle \text{Pow}(\bigcup (T \langle \text{restricted to} \rangle A)) \langle \text{restricted to} \rangle A \rangle (T \langle \text{restricted to} \rangle A) \langle \text{restricted to} \rangle A \rangle (T \langle \text{restricted to} \rangle A) \langle \text{restricted to} \rangle A \rangle \langle \text{hyperconnected} \rangle$

by auto

with $\langle (T \langle \text{restricted to} \rangle A) \langle \text{is sober} \rangle \rangle$ have $\exists x \in \bigcup (T \langle \text{restricted to} \rangle A)$. $A = \text{Closure}(\{x\}, T \langle \text{restricted to} \rangle A) \wedge (\forall y \in \bigcup (T \langle \text{restricted to} \rangle A)$. $A = \text{Closure}(\{y\}, T \langle \text{restricted to} \rangle A) \rightarrow y = x)$

unfolding $\text{IsSober}_A$ by auto

with top_def have $\exists x \in A$. $A = \text{Closure}(\{x\}, \{0, A\}) \wedge (\forall y \in A$. $A = \text{Closure}(\{y\}, \{0, A\}) \rightarrow y = x)$ by auto

then obtain $x$ where $x \in A \subseteq \text{Closure}(\{x\}, \{0, A\})$ and reg: $\forall y \in A$. $A = \text{Closure}(\{y\}, \{0, A\})$

$\rightarrow y = x$ by auto

{ fix $y$ assume $y \in A$
from $\langle A \neq 0 \rangle$ have top: $\langle 0, A \rangle$ (is a topology) using $\text{indiscrete}_A$ $\text{ptopology} [\text{of } A]$ $\text{indiscrete}_A$ $\text{partition} [\text{of } A]$. $\text{Ptopology}_A$ $\text{is_a_topology}(1) [\text{of } \{A\} A]$

by auto

with $\langle y \in A \rangle$ have $\text{Closure}(\{y\}, \{0, A\})$ (is closed in) $\{0, A\}$ using $\text{topology}_0$. $\text{cl_is_closed}$

by auto

moreover
from $\langle y \in A \rangle$ top have $y \in \text{Closure}(\{y\}, \{0, A\})$ using $\text{topology}_0$. $\text{cl_contains_set}$

by auto

ultimately have $A - \text{Closure}(\{y\}, \{0, A\}) \subseteq \{0, A\}$. $\text{Closure}(\{y\}, \{0, A\}) \cap A \neq 0$

unfolding $\text{IsClosed}_A$

by auto

then have $A - \text{Closure}(\{y\}, \{0, A\}) = A \wedge A - \text{Closure}(\{y\}, \{0, A\}) = 0$

by auto

moreover
from $\langle y \in A \rangle$ $\langle y \in \text{Closure}(\{y\}, \{0, A\}) \rangle$ have $y \in A \notin A - \text{Closure}(\{y\}, \{0, A\})$

by auto

ultimately have $A - \text{Closure}(\{y\}, \{0, A\}) = 0$ by (cases $A - \text{Closure}(\{y\}, \{0, A\}) = A$, simp, auto)

moreover
from $\langle y \in A \rangle$ top have $\text{Closure}(\{y\}, \{0, A\}) \subseteq A$ using $\text{topology}_0$. $\text{def}$

$\text{topology}_0$. $\text{Top}_3 [\text{L11}(1)]$ by blast

then have $A - (A - \text{Closure}(\{y\}, \{0, A\}) = \text{Closure}(\{y\}, \{0, A\})$ by auto

ultimately have $A = \text{Closure}(\{y\}, \{0, A\})$ by auto

1079
with reg \( \langle x \in A \rangle \) have \( A = \{x\} \) by blast
then have \( A \approx 1 \) using singleton_eqpoll_1 by auto
then have \( A \leq 1 \) using eqpoll_imp_lepoll by auto
then have \( A \) (is in the spectrum of) IsSober using sober_spectrum by auto
ultimately have \( A \) (is in the spectrum of) IsSober by auto
next
assume \( T \) (is anti-) IsSober using antiProperty_def by auto
fix \( A \)
assume \( A \in T \wedge A \neq \emptyset \)
then obtain \( x \ w y \) where \( x \in A \wedge y \in \bigcup T \wedge x \neq y \) by blast
with \( \langle A \in T \rangle \) have \( \{x\} \in T \{\text{restricted to}\} \{x, y\} \) unfolding RestrictedTo_def
ultimately have \( \{x\} \{\text{is T}_2\} \) unfolding isT2_def by auto
then have \( \{x, y\} \{\text{is sober}\} \) using topology0.T2_imp_anti_HConn[of \( T \{\text{restricted to}\} \{x, y\} \)]
Top_1_L4 topology0_def topology0.anti_HConn_iff_T1_sober[of \( T \{\text{restricted to}\} \{x, y\} \)]
by auto
moreover
assume \( \{x\} \not\in T \{\text{restricted to}\} \{x, y\} \)
moreover
from \( \langle x \in A \in T \rangle \) have \( T \{\text{restricted to}\} \{x, y\} \subseteq \{0, \{x\}, \{x, y\}\} \) by blast
moreover
note \( \langle x \in T \{\text{restricted to}\} \{x, y\} \rangle \) empty_open[OF Top_1_L4[of \( \{x, y\}\)]]
moreover
from \( \langle y \in \bigcup T \rangle \) have \( \text{tot}: \bigcup T \{\text{restricted to}\} \{x, y\} \) = \( \{x, y\} \)
ultimately have \( \text{tot} \{\text{restricted to}\} \{x, y\} \) = \( \{0, \{x\}, \{x, y\}\} \) by auto
auto
{
  fix B assume B∈Pow({x,y})-{0}B{is closed in}(T{restricted to}{x,y})
with top_d_def have (⋃(T{restricted to}{x,y}))-B∈{0,\{x\},\{x,y\}}
unfolding IsClosed_def by simp
  moreover have B∈\{x\},\{y\},\{x,y\} using B∈Pow({x,y})-{0} by blast
moreover note tot
ultimately have \{x,y\}-B∈\{0,\{x\},\{x,y\}\} by auto
have xin:x∈Closure({x},T{restricted to}{x,y}) using topology0.cl_contains_set[of T{restricted to}{x,y}]
Top_1_L4[of {x,y}] unfolding topology0_def[of (T {restricted to} {x, y})]
tot} {x, y})} using tot by auto
  \{ assume \{x\}{is closed in}(T{restricted to}{x,y})
  then have \{x\}-{x}∈(T{restricted to}{x,y}) unfolding IsClosed_def
  using tot
  by auto
moreover
from \langle x\neq y \rangle have \{x,y\}-{x}={y} by auto
ultimately have \{y\}∈(T{restricted to}{x,y}) by auto
then have False using \langle y\\notin(T{restricted to}{x,y}) \rangle by auto
\}
then have \neg(\{x\}{is closed in}(T{restricted to}{x,y})) by auto
moreover
from tot have (Closure({x},T{restricted to}{x,y}))\{is closed in\}(T{restricted to}{x,y})
  using topology0.cl_is_closed unfolding topology0_def using Top_1_L4[of 
{ x,y}]
tot by auto
ultimately have \neg(Closure({x},T{restricted to}{x,y})={x}) by auto
moreover note xin topology0.Top_3_L11(1)[of T{restricted to}{x,y}]{x}]
tot
ultimately have cl_x:Closure({x},T{restricted to}{x,y})={x,y}
unfolding topology0_def
  using Top_1_L4[of \{x,y\}] by auto
have \{y\}{is closed in}(T{restricted to}{x,y}) unfolding IsClosed_def
using tot
  top_d_def \langle x\neq y \rangle by auto
then have cl_y:Closure({y},T{restricted to}{x,y})={y} using topology0.Top_3_L8[of T{restricted to}{x,y}]
unfolding topology0_def using Top_1_L4[of \{x,y\}] tot by auto
\}
assume \{x,y\}-B=0
with \langle B∈Pow({x,y})-{0} \rangle have B:{x,y}=B by auto
\}
fix m
assume dis:m∈{x,y} and B_def:B=Closure({m},T{restricted to}
to\{x,y\})
{
   assume m=y
   with B_def have B=Closure({y},T\{restricted to\}{x,y}) by auto
   with cl_y have B=\{y\} by auto
   with B have \{x,y\}=\{y\} by auto
   moreover have x\in\{x,y\} by auto
   ultimately
   have x\in\{y\} by auto
   with \langle x\neq y \rangle have False by auto
}
   with dis have m=x by auto
}
then have (\forall m\in\{x,y\}. B=Closure(\{m\},T\{restricted to\}{x,y})\implies m=x
 ) by auto
moreover
   have B=Closure(\{x\},T\{restricted to\}{x,y}) using cl_x B by auto
   ultimately have \exists t\in\{x,y\}. B=Closure(\{t\},T\{restricted to\}{x,y})
\land (\forall m\in\{x,y\}. B=Closure(\{m\},T\{restricted to\}{x,y})\implies m=t )
   by auto
}
moreover
{
   assume \{x,y\}-B\neq 0
   with \langle \{x,y\}-B\in\{0,\{x\},\{x,y\}\} \rangle have or:\{x,y\}-B=\{x\}\lor \{x,y\}-B=\{x,y\}
by auto
{
   assume \{x,y\}-B=\{x\}
   then have x\in\{x,y\}-B by auto
   with \langle B\in\{x\},\{y\},\{x,y\}\rangle \langle x\neq y \rangle have B:B=\{y\} by blast
   
   fix m
   assume dis:m\in\{x,y\} and B_def:B=Closure(\{m\},T\{restricted to\}{x,y})
}
   assume m=x
   with B_def have B=Closure(\{x\},T\{restricted to\}{x,y}) by auto
   with cl_x have B=\{x,y\} by auto
   with B have \{x,y\}=\{y\} by auto
   moreover have x\in\{x,y\} by auto
   ultimately
   have x\in\{y\} by auto
   with \langle x\neq y \rangle have False by auto
}
   with dis have m=y by auto
}
moreover
have $B = \text{Closure}(\{y\}, T_{\text{restricted to}} \{x, y\})$ using cl_y B by auto
ultimately have $\exists t \in \{x, y\}. B = \text{Closure}(\{t\}, T_{\text{restricted to}} \{x, y\})$
$\land (\forall m \in \{x, y\}. B = \text{Closure}(\{m\}, T_{\text{restricted to}} \{x, y\}) \rightarrow m = t)$
by auto
}

moreover
{
assume $\{x, y\} - B \neq \{x\}$
with or have $\{x, y\} - B = \{x, y\}$ by auto
then have $x \in \{x, y\} - B \in \{x, y\} - B$ by auto
with $\langle B \in \{x\}, \{y\}, \{x, y\}\rangle \langle x \neq y \rangle$ have False by auto
}
ultimately have $\exists t \in \{x, y\}. B = \text{Closure}(\{t\}, T_{\text{restricted to}} \{x, y\})$
$\land (\forall m \in \{x, y\}. B = \text{Closure}(\{m\}, T_{\text{restricted to}} \{x, y\}) \rightarrow m = t)$
by auto
}

ultimately have $\exists t \in \{x, y\}. B = \text{Closure}(\{t\}, T_{\text{restricted to}} \{x, y\})$
$\land (\forall m \in \{x, y\}. B = \text{Closure}(\{m\}, T_{\text{restricted to}} \{x, y\}) \rightarrow m = t)$
by auto
}
then have $(T_{\text{restricted to}} \{x, y\})\{\text{is sober}\}$ unfolding IsSober_def using tot by auto
ultimately have $(T_{\text{restricted to}} \{x, y\})\{\text{is sober}\}$ by auto
with $\langle T\{\text{is anti-}\}\text{IsSober} \rangle$ have $\{x, y\}\{\text{is in the spectrum of}\}\text{IsSober}$
unfolding antiProperty_def
using $\langle x \in A \rightarrow A \in T \leftrightarrow y \in \bigcup T - A \rangle$ by auto
then have $\{x, y\} \subseteq 1$ using sober_spectrum by auto
moreover
have $x \in \{x, y\}$ by auto
ultimately have $\{x, y\} = \{x\}$ using lepoll_1_is_sing[of $\{x, y\}x$] by auto
moreover have $y \in \{x, y\}$ by auto
ultimately have $y \in \{x\}$ by auto
then have False using $\langle x \neq y \rangle$ by auto
}
then have $T \subseteq \{0, \bigcup T\}$ by auto
with empty_open[OF topSpaceAssum] topSpaceAssum show $T = \{0, \bigcup T\}$ unfolding IsATopology_def
by auto
qed
Suppose $T$ is a topology, $r$ is an equivalence relation on $X = \bigcup T$ and $P_r : X \to X/r$ maps an element of $X$ to its equivalence class $r\{x\}$. Then we can define a topology (on $X/r$) by taking the collection of those subsets $V$ of $X/r$ for which the inverse image by the projection $P_r$ is in $T$. This is the weakest topology on $X/r$ such that $P_r$ is continuous. In this theory we consider a seemingly more general situation where we start with a topology $T$ on $X = \bigcup T$ and a surjection $f : X \to Y$ and define a topology on $Y$ by taking those subsets $V$ of $Y$ for which the inverse image by the mapping $f$ is in $T$. Turns out that this construction is in a way equivalent to the previous one as the topology defined this way is homeomorphic to the topology defined by the equivalence relation $r_f$ on $X$ that relates two elements of $X$ if $f$ has the same value on them.

### 79.1 Definition of quotient topology

In this section we define the quotient topology generated by a topology $T$ and a surjection $f : \bigcup T \to Y$, and show its basic properties.

For a topological space $X = \bigcup T$ and a surjection $f : X \to Y$ we define the collection of subsets of $Y$ whose inverse images by $f$ are open.

**Definition (in topology0)**

\[
\text{QuotientTop} \ (\text{by})_\text{f} \equiv \{U \in \text{Pow}(Y) . f^{-1}(U) \in T\}
\]

Outside of the topology0 context we will indicate also the generating topology and write $\text{QuotientTop}_\text{T} \ (\text{by})_\text{f} \ (\text{from})_\text{X}$.

**Abbreviation QuotientTopTop (in topology0)**

\[
\text{QuotientTopTop} \ (\text{by}) \ (\text{from})_\text{X} \equiv \text{topology0.quotientTop}\text{T}\text{Y}\text{f}
\]

The quotient topology is indeed a topology.

**Theorem (in topology0)**

\[
\text{quotientTop_is_top} : \text{assumes f\in\text{surj}(\bigcup T,Y) shows \{\text{quotient topology in} Y \ (\text{by}) f\} \ (\text{is a topology})}
\]

**Proof**

\[
\text{have \{\text{quotient topology in} Y \ (\text{by}) f\}=(U \in \text{Pow}(Y) . f^{-1}(U) \in T) using QuotientTop_def assms by auto}
\]

**Moreover**

\[
\text{fix M x B assume M: M \subseteq \{U \in \text{Pow}(Y) . f^{-1}(U) \in T\} then have M \subseteq Y by blast moreover have A1: f^{-1}(\bigcup M) = (\bigcup y \in (\bigcup M). f^{-1}(y)) using vimage_eq_UN by blast moreover}
\]

1084
The quotient function is continuous.

lemma (in topology0) quotient_func_cont:
  assumes \( f \in \text{surj}(\bigcup \tau_1, Y) \)
  shows \( \text{IsContinuous}(T, (\{\text{quotient topology in} \} Y \{\text{by} \} f), f) \)
  unfolding \( \text{IsContinuous_def} \) using QuotientTop_def assms by auto

One of the important properties of this topology, is that a function from the quotient space is continuous iff the composition with the quotient function is continuous.

theorem (in two_top_spaces0) cont_quotient_top:
  assumes \( h \in \text{surj}(\bigcup \tau_1, Y) \) \( g : Y \to \bigcup \tau_2 \) \( \text{IsContinuous}(\tau_1, \tau_2, g \ 0 \ h) \)
  shows \( \text{IsContinuous}((\{\text{quotient topology in} \} Y \{\text{by} \} h \ {\text{from}} \ \tau_1), \tau_2, g) \)
proof-
  { fix \( U \) assume \( U \in \tau_2 \)
    with assms(3) have \( (g \ 0 \ h) - (U) \in \tau_1 \) unfolding \( \text{IsContinuous_def} \) by auto
    then have \( h - (g - (U)) \in \tau_1 \) using vimage_comp by auto
    with assms(1) have \( g - (U) \in (\{\text{quotient topology in} \} Y \{\text{by} \} h \ {\text{from}} \ \tau_1) \)
}

1085
The underlying set of the quotient topology is \( Y \).

**Lemma (in topology0) total_quo_func:**

Assumes \( f \in \text{surj}(\bigcup T, Y) \)

Shows \( \bigcup (\{\text{quotient topology in} Y \ \{\text{by}\} \ f\}) = Y \)

**Proof:**

From \( \text{assms} \) have \( f-Y = \bigcup T \) using \( \text{func1}_1 \_L4 \) unfolding \( \text{surj}_\text{def} \) by auto

Moreover have \( \bigcup T \in T \) using \( \text{topSpaceAssum} \) unfolding \( \text{IsATopology}_\text{def} \) by auto ultimately

Have \( Y \in (\{\text{quotient topology in} Y \ \{\text{by}\} f \ \{\text{from}\} T) \) using \( \text{QuotientTop}_\text{def} \) \( \text{assms} \) by auto

Then show thesis using \( \text{QuotientTop}_\text{def} \) \( \text{assms} \) by auto

**79.2 Quotient topologies from equivalence relations**

In this section we will show that the quotient topologies come from an equivalence relation.

The quotient projection \( b \mapsto r\{b\} \) is a function that maps the domain of the relation to the quotient. Note we do not need to assume that \( r \) is an equivalence relation.

**Lemma quotient_proj_fun:**

Shows \( \{\langle b, r\{b\} \rangle. \ b \in A\} : A \rightarrow A/\!/r \) unfolding \( \text{Pi}_\text{def} \) \( \text{function}_\text{def} \) \( \text{domain}_\text{def} \)

Unfolding \( \text{quotient}_\text{def} \) by auto

The quotient projection is a surjection. Again \( r \) does not need to be an equivalence relation here.

**Lemma quotient_proj_surj:**

Shows \( \{\langle b, r\{b\} \rangle. \ b \in A\} \in \text{surj}(A,A/\!/r) \)

**Proof:**

{ Fix \( y \) assume \( y \in A/\!/r \)

Then obtain \( x \) where \( x \in A \ y = r(x) \) unfolding \( \text{quotient}_\text{def} \) by auto

Then have \( \exists x \in A. \ \{\langle b, r\{b\} \rangle. \ b \in A\}(x) = y \) using \( ZF\text{fun_from_tot_val1} \)

By auto

} Then show thesis using \( \text{quotient_proj_fun} \) unfolding \( \text{surj}_\text{def} \) by auto

Qed

The inverse image of a subset \( U \) of the quotient by the quotient projection
is the union of $U$. Note since $U$ is a subset of $A/r$ it is a collection of equivalence classes.

**Lemma preim_equi_proj:**
assumes $U \subseteq A/r$ equiv(A,r)
shows $\{\langle b, r(b) \rangle . b \in A \} - (U) = \bigcup U$

**Proof**

- **Fix** $y$ assume $y \in \bigcup U$
  - then obtain $V$ where $y \in V$ and $V \in U$ by auto
  - with asms have $y \in \{\langle b, r(b) \rangle . b \in A \} - (U)$
  - using EquivClass_1_L1 EquivClass_1_L2 by blast
- thus $\bigcup U \subseteq \{\langle b, r(b) \rangle . b \in A \} - U$ by blast

**Outside of the topology0 context we need to indicate the original topology.**

**Abbreviation** EquivQuoTop (_{quotient by}_)
where $T \{\text{quotient by} r\} \equiv \text{topology0.EquivQuo}(T,r)$

First, another description of the topology (more intuitive):

**Theorem (in topology0) quotient_equiv_rel:**
assumes equiv((\bigcup T),r)
shows $\{\text{quotient by} r\} = \{U \in \text{Pow}((\bigcup T)/r) . \bigcup U \in T\}$

**Proof**

- have $\{\text{quotient topology in} \}((\bigcup T)/r\by\{\langle b, r\{b\} \rangle . b \in \bigcup T\}) = \{U \in \text{Pow}((\bigcup T)/r) . \{\langle b, r\{b\} \rangle . b \in \bigcup T\} \in U\}$
  - using QuotientTop_def quotient_proj_surj by auto
- moreover have $\{U \in \text{Pow}((\bigcup T)/r) . \{\langle b, r\{b\} \rangle . b \in \bigcup T\} \in U\} = \{U \in \text{Pow}((\bigcup T)/r) . \bigcup U \in T\}$
- **Proof**
  - **Fix** $U$ assume $U \in \text{Pow}((\bigcup T)/r) . \{\langle b, r\{b\} \rangle . b \in \bigcup T\} \in U\$ with asms have $U \in \text{Pow}((\bigcup T)/r) . \bigcup U \in T$ using preim_equi_proj
  - by auto
thus \( \{ U \in \mathcal{P}(\bigcup T) / r) : \{ b, r(b) \} : b \in \bigcup T \} \subseteq \{ U \in \mathcal{P}(\bigcup T) / r). \bigcup U \in T \} \) by auto

then \( \{ U \in \mathcal{P}(\bigcup T) / r) : \{ b, r(b) \} : b \in \bigcup T \} \subseteq \{ U \in \mathcal{P}(\bigcup T) / r). \bigcup U \in T \} \) by auto

We apply previous results to this topology.

**Theorem (in topology0) total_quo_equi:**
assumes \( \text{equiv}(\bigcup T, r) \)
shows \( \bigcup \{ \text{quotient by} r) = (\bigcup T) / r \)
using total_quo_func quotient_proj_surj EquivQuo_def assms by auto

The quotient by an equivalence relation is indeed a topology.

**Theorem (in topology0) equiv_quo_is_top:**
assumes \( \text{equiv}(\bigcup T, r) \)
shows \( \{ \text{quotient by} r) \} \) is a topology
using quotientTop_is_top quotient_proj_surj EquivQuo_def assms by auto

The next theorem is the main result of this section: all quotient topologies arise from an equivalence relation given by the quotient function \( f : X \to Y \).

This means that any quotient topology is homeomorphic to a topology given by an equivalence relation quotient.

**Theorem (in topology0) equiv_quotient_top:**
assumes \( f \in \text{surj}(\bigcup T, Y) \)
defines \( r = \{ (x, y) : x \in \bigcup T \times \bigcup T. f(x) = f(y) \} \)
defines \( g = \{ (y, f^{-y}) : y \in Y \} \)
shows \( \text{equiv}(\bigcup T, r) \) and
IsAhomeomorphism(\{\text{quotient topology in} Y by f), \{\text{quotient by} r), g)\)
proof-
from assms(1) have ff : \( f : \bigcup T \to Y \) unfolding surj_def by auto
from assms(2) show B : equiv(\bigcup T, r)
  unfolding equiv_def refl_def sym_def trans_def r_def by auto
have gg : g : \( Y \to (\bigcup T) / r) \)
proof -
  { fix B assume B\text{g}
    then obtain y where Y : y \in Y B = \{ y, f^{-y} \} unfolding g_def
    by auto
    then have f^{-y} \subseteq \bigcup T using func1_1_L3 ff by blast
    then have eq : f^{-y} = \{ x \in \bigcup T. (x, y) \in f \} using vimage_iff by auto
  }
from assms(1) Y obtain A where A1: A ∈ ∪ T f(A) = y unfolding surj_def

by blast
with ff have A: A ∈ f-{y} using func1_1_L15 by simp
{ fix t assume t ∈ f-{y}
  with A eq have t ∈ ∪ T A ∈ T ⟨t, y⟩ ∈ f by auto
  then have ft = fA using apply_equality assms(1) unfolding surj_def
  by auto
with assms(2) ⟨t ∈ ∪ T, A ∈ T⟩ have ⟨A, t⟩ ∈ r by auto
then have t ∈ r{A} by auto
}

moreover
{ fix t assume t ∈ r{A}
  with assms(2) have un: t ∈ ∪ T A ∈ T and eq2: f(t) = f(A)
  using image_iff by simp_all
  from ff un have ⟨t, f(t)⟩ ∈ g unfolding g_def by auto
  with eq2 A1 un eq have t ∈ f-{y} by simp
  hence r{A} ⊆ f-{y} by auto
}
ultimately have f-{y} = r{A} by auto
with A1(1) have f-{y} ∈ (∪ (T}//r by auto
ultimately have r{A} ⊆ f-{y} by auto
with Y have B ∈ Y × (∪ T}//r by auto
}
then show thesis unfolding Pi_def function_def domain_def g_def
  by auto
qed

then have gg2: g: Y → (∪ (Union by r)) using total_quo_equi B
by auto
{ fix s assume S: s ∈ (Union topology in Y by f)
  then have s ∈ Pow(Y) and P: f(s) ∈ T
  using QuotientTop_def topSpaceAssum assms(1) by auto
  have f-s = (∪ y ∈ s. f-{y}) using vimage_eq_UN by blast
  moreover
  from s ∈ Pow(Y)
  have ∀ y ∈ s. (y, f-{y}) ∈ g unfolding g_def by auto
  then have ∀ y ∈ s. gy = f-{y} using apply_equality gg by auto
  ultimately have f-s = (∪ y ∈ s. gy) by auto
  with P have (∪ y ∈ s. gy) ∈ T by auto
  moreover
  from s ∈ Pow(Y)
  have ∀ y ∈ s. gy ∈ (Union by r) using apply_type
  with s ∈ Pow(Y)
  have gy ∈ (Union by r) using quotient_equiv_rel B
  by auto
  with s ∈ Pow(Y)
  have gs ∈ (Union by r) using func_imagedef gg by auto
  } hence open: ∀ s ∈ (Union topology in Y by f). gs ∈ (Union by r)
by auto

have pr_fun: {⟨b, r(b)⟩. b ∈ T}→ (∪ T}//r
  using quotient_proj_fun by auto
{ fix b assume b: b ∈ T
have $b : f(b) \in Y$ using apply_funtype $ff$ by auto
with $b$ have $\text{com}$: $(g \circ f)(b) = g(f(b))$ using comp_fun_apply $ff$ by auto
from $b$ have $pg$: $(fb, f^{-1}(fb)) \in g$ unfolding $g$ def by auto
then have $g(fb) = f^{-1}(fb)$ using apply_equality $gg$ by auto
with $com$ have $\text{comeq}$: $(g \circ f)(b) = f^{-1}(fb)$ by auto
from $b$ have $A$: $f\{b\} = \{fb\} \subseteq \bigcup T$ using func_imagedef $ff$ by auto
moreover from $pg$ have $f^{-1}(fb) \in \bigcup (T\cap r)$ using $gg$ unfolding $\Pi$ def by auto
ultimately have $r\{b\} = f^{-1}(fb)$ using $EquivClass_1_L2$ $B$ by auto
then have $\text{reg}$: $\forall b \in \bigcup T. (g \circ f)(b) = \{⟨b, r\{b\}⟩. b \in \bigcup T\}$ by simp
moreover have $(g \circ f) : \bigcup T \to \bigcup (T\cap r)$ using comp_fun $ff$ $gg$ by auto
ultimately have $g \in \text{bij}(\bigcup (T\cap r))$ unfolding $bij$ def by auto
with $gcont$ $gopen$ show $\text{IsAhomeomorphism}(\bigcup (T\cap r), g)$ using bij_cont_open_homeo by auto
qed

The mapping $⟨b, c⟩ \mapsto ⟨r\{a\}, r\{b\}⟩$ is a function that maps the product of
the carrier by itself to the product of the quotients. Note $r$ does not have
to be an equivalence relation.

**lemma product_equiv_rel_fun:**

shows $\{(b,c),(r(b),r(c))\}$. ($b,c)\in (U \times U):(U \times U)\to ((U \times U) /r \times (U \times U) /r)$

proof

have $\{(b,r(b))\}$. $b(U \times \{b\})\in (U \times U) /r$ using quotient_proj_fun by auto

moreover have $\forall A \in (U \times U) . \{b,r(b)\} . b((U \times U) /r)$ by auto

ultimately have $\forall A \in (U \times U) . \{b,r(b)\} . b((U \times U) /r)$ using apply_equality

hence $\{b,c\} \in (U \times U) /r \times ((U \times U) /r)$

then show thesis using prod_fun quotient_proj_fun by auto

qed

The mapping $\langle b,c \rangle \mapsto \langle (r\{a\},r\{b\}) \rangle$ is a surjection of the product of the

carrier by itself onto the carrier of the product topology. Again $r$ does not

have to be an equivalence relation for this.

**lemma (in topology0) prod_equiv_rel_surj:**

shows $\{(b,c),(r(b),r(c))\}$. ($b,c)\in (U \times U):surj((U \times U) /r \times (U \times U) /r)$

proof

have $\forall b,c \in (U \times U). \{b,r(b)\} . b((U \times U) /r)$ using quotient_def by auto

moreover have $\forall b,c \in (U \times U). \{b,r(b)\} . b((U \times U) /r)$ by force

ultimately show thesis unfolding surj_def using Top_1_4_T1(3) topSpaceAssum

by auto

qed

The product quotient projection (i.e. the mapping the mapping $\langle b,c \rangle \mapsto

\langle (r\{a\},r\{b\}) \rangle$ is continuous.

**lemma (in topology0) product_quo_fun:**
assumes $\text{equiv}(U,T)$

shows

IsContinuous($T \times T, \{\{\text{quotient by}\} r\} \times \{\{\text{quotient by}\} r\} \times \{(b,c),(r\{b\},r\{c\})\}$)

(b,c) \in U \times U

proof-

have $\{\{\text{quotient by}\} r\} \cdot (U \rightarrow (U) / r)$ using quotient_proj_fun by auto moreover have $\forall A \in U. \langle A, r\{A\} \rangle \in \{\{\text{quotient by}\} r\} \cdot (b,r\{b\})$. $b \in U \cdot r\{A\} = r\{A\}$ using apply_equality by auto

hence IN: $\{\{\text{quotient by}\} r\} \cdot (U \times U) = \{\langle\langle b,c \rangle, \langle r\{b\}, r\{c\} \rangle \rangle. \langle b, c \rangle \in U \times U\}$

with assms have cont: IsContinuous($T, \{\{\text{quotient by}\} r\} \cdot (b,r\{b\}). b \in U\}$)

using quotient_func_cont quotient_proj_surj EquivQuo_def by auto

with assms have tot: $U \cdot (U) / r$ and top: $\{\{\text{quotient by}\} r\} \cdot \{\{\text{is a topology}\} \cdot \{\text{equiv_quo_is_top by auto

then have fun: $\{\{\text{quotient by}\} r\} \cdot (U \rightarrow (U) / r$ using quotient_proj_fun by auto

then have two_top_spaces0($T, \{\{\text{quotient by}\} r\} \cdot (b,r\{b\}). b \in U\}$)

unfolding two_top_spaces0_def using TopSpaceAssum top by auto

qed

The product of quotient topologies is a quotient topology given that the quotient map is open. This isn’t true in general.

theorem (in topology0) prod_quotient:

assumes $\text{equiv}(U,T,r) \forall A \in T. \{\{\text{quotient by}\} r\} \cdot (b,r\{b\}). b \in U \cdot \{\{\text{quotient by}\} r\}$

shows $\{\{\text{quotient topology in}\} (U_T) / r \times (U_T) / r\} \cdot \{\{\text{by}\} \cdot (b,c),(r\{b\},r\{c\})\}$

(b,c) \in U \times U

proof-

let $T \cdot \{\{\text{quotient by}\} r\} \cdot (b,c) \in U \times U$

{ fix $A$ assume $A \in T \cdot \times T_r$

with assms(1) have $\{\{\text{quotient by}\} r\} \cdot (b,c) \in U \times U \cdot (A) \in T \cdot T$

using product_quo_fun unfolding IsContinuous_def by auto

moreover

from $A$ have $\exists \in U \cdot (T_r) \times (T_r)$ by auto

with assms(1) have $\exists \in \text{Pow}(U \times U / r) \times (U_T) / r)$

using Top_1_4_T1(3) equiv_quo_is_top total_quo_equi by auto

ultimately have $\exists \in \exists \cdot$ unfolding topology0.QuotientTop_def Top_1_4_T1(1) TopSpaceAssum prod_equiv_rel_surj

topology0_def by auto
thus \((T_r)^*\times_2(T_r) \subseteq R\) by auto

{ fix A assume \(A \subseteq R\) with assms(1) have
  \[A : A \subseteq (\bigcup T)/r \times ((\bigcup T)/r) \{\{(b,c),(r(b),r(c))\} : (b,c) \in T \times T\} - (A)\]
  \(\in T \times T\)
  using topology0.QuotientTop_def Top_1_4_T1(1) topSpaceAssum prod_equiv_rel_surj
  unfolding topology0_def by auto

  unfolding \text{by auto}

  \(\{\text{fix C assume } C \subseteq A\}
  \text{with } A(1) \text{ obtain } C_1, C_2 \text{ where } C = \langle C_1, C_2 \rangle:
  C_1 \subseteq (\bigcup T)/r \text{ and } C_2 \subseteq (\bigcup T)/r\)
  \text{by auto}

  then obtain c_1, c_2 where \(C_1 : c_1 \subseteq T_c_2 \subseteq T\) and \(C_2 : c_1 = r(c_1) \text{ and } c_2 = r(c_2)\)

  unfolding \text{by auto}

  with \(A(2)\) have
  \(\exists V W. V \in T \land W \in T \land V \times W \subseteq \{(b,c),(r(b),r(c))\} : (b,c) \in T \times T\} - (A)\)
  \(\land \langle c_1, c_2 \rangle \in V \times W\)
  using prod_top_point_neighb topSpaceAssum by blast

  then obtain V W where
  \(V W : \forall V W. V \in T \land W \in T \land V \times W \subseteq \{(b,c),(r(b),r(c))\} : (b,c) \in T \times T\} - (A)\)
  \(c_1 \in V\)

  \(c_2 \in W\)
  by blast

  let \(V_r = \{(b,r(b)) : b \in T\}(V)\)
  let \(W_r = \{(b,r(b)) : b \in T\}(W)\)
  from \(V W\) assms have \(P : (V_r \times W_r) \in (T_r)^*\times T_r\)
  using prod_open_open_prod equiv_quo_is_top by auto

  \(\{\text{fix S assume } S \subseteq (V_r \times W_r)\}
  \text{then obtain } s_1, s_2 \text{ where } S = \langle s_1, s_2 \rangle : s_1 \in V_r \land s_2 \in W_r\)
  \text{by blast}

  then obtain \(t_1, t_2\) where
  \(T : \langle t_1, s_1 \rangle \in \{(b,r(b)) : b \in T\} \land \{t_2, s_2 \rangle \in \{(b,r(b)) : b \in T\} : t_1 \in V \land t_2 \in W\)

  using vimage_iff by auto

  with \(V W(3)\) have \(\exists S_0 \subseteq A. \langle \langle t_1, t_2 \rangle, S_0 \rangle \in \{(b,c),(r(b),r(c))\} : (b,c) \in T \times T\}\)

  using vimage_iff by auto

  then obtain \(S_0\) where \(S_0 \subseteq A\) and
  \(\langle \langle t_1, t_2 \rangle, S_0 \rangle \in \{(b,c),(r(b),r(c))\} : (b,c) \in T \times T\}\)
  \text{by auto}

  moreover from \(S(1)\) \(V W(1,2)\) have
  \(\langle \langle t_1, t_2 \rangle, S \rangle \in \{(b,c),(r(b),r(c))\} : (b,c) \in T \times T\}\)
  \text{by auto}

  ultimately have \(S \subseteq A\)
  using product_equiv_rel_fun unfolding Pi_def function_def
  \text{by auto}

  } \text{hence sub: } V_r \times W_r \subseteq A \text{ by blast}

  from \(CC CC2 CC1 \langle c_1 \in V \rangle \langle c_2 \in W \rangle\) have \(C \in V_r \times W_r\).
using image_iff by auto
with P sub have \( \exists U \in (T_r) \times_t (T_r). \ U \subseteq A \land C \subseteq U \)
  by (rule witness_exists1)
} hence \( \forall C \in A. \ \exists U \in (T_r) \times_t (T_r). \ C \subseteq U \)
  by blast
with assms(1) have \( A \in (T_r) \times_t (T_r) \)
  using topology0.open_neigh_open Top_1_4_T1 equiv_quo_is_top assms
unfolding topology0_def by auto
} thus \( R \subseteq (T_r) \times_t (T_r) \)
  by auto
qed

end

80 Topology 9

theory Topology_ZF_9
imports Topology_ZF_2 Group_ZF_2 Topology_ZF_7 Topology_ZF_8
begin

80.1 Group of homeomorphisms

This theory file deals with the fact the set homeomorphisms of a topological
space into itself forms a group.

First, we define the set of homeomorphisms.

definition
  HomeoG(T) \equiv \{ f: \bigcup T \to \bigcup T. \ IsAhomeomorphism(T,T,f) \}\n
The homeomorphisms are closed by composition.

lemma (in topology0) homeo_composition:
  assumes f \in HomeoG(T) g \in HomeoG(T)
  shows Composition(\bigcup T)(f, g) \in HomeoG(T)
proof-
  from assms have \( f: \bigcup T \to \bigcup T \)
    and \( g: \bigcup T \to \bigcup T \)
  unfolding HomeoG_def
  by auto
  from fun have f 0 g: \( \bigcup T \to \bigcup T \)
    using comp_fun by auto
  moreover
  from homeo have \( f: \bigcup T \to \bigcup T \)
    and \( g: \bigcup T \to \bigcup T \)
  unfolding IsAhomeomorphism_def
  by auto
  from bij have \( f: \bigcup T \to \bigcup T \)
    using comp_bij by auto
  moreover
  from cont have \( IsContinuous(T,T,f 0 g) \)
    using comp_cont by auto
  moreover
  have converse(f 0 g)=converse(g) 0 converse(f) using converse_comp by auto
  with contconv have \( IsContinuous(T,T,converse(f 0 g)) \)
    using comp_cont by auto ultimately

1094
The identity function is a homeomorphism.

**Lemma (in topology0) homeo_id:**
- **shows**: $\text{id}(\bigcup T) \in \text{HomeoG}(T)$
- **proof-**
  - **have**: converse($\text{id}(\bigcup T)$) $0 \text{id}(\bigcup T) = \text{id}(\bigcup T)$ using left_comp_inverse id_bij
  - **by auto**
  - **then have**: converse($\text{id}(\bigcup T)$) $= \text{id}(\bigcup T)$ using right_comp_id by auto
  - **then show**: thesis unfolding HomeoG_def IsAhomeomorphism_def using id_cont id_type id_bij
  - **by auto**
- **qed**

The homeomorphisms form a monoid and its neutral element is the identity.

**Theorem (in topology0) homeo_submonoid:**
- **shows**: $\text{IsAmonoid}(\text{HomeoG}(T),\text{restrict}(\text{Composition}(\bigcup T),\text{HomeoG}(T) \times \text{HomeoG}(T)))$
- **proof-**
  - **have cl:** $\text{HomeoG}(T)$ {is closed under} Composition($\bigcup T$) unfolding IsOpClosed_def
  - **using**: homeo_composition by auto
  - **moreover have**: sub:$\text{HomeoG}(T) \subseteq \bigcup T \rightarrow \bigcup T$ unfolding HomeoG_def by auto
  - **moreover**
    - **have ne:** $\text{TheNeutralElement}(\bigcup T \rightarrow \bigcup T, \text{Composition}(\bigcup T)) \in \text{HomeoG}(T)$ using homeo_id Group_ZF_2_5_L2(2) by auto
    - **ultimately show**: $\text{IsAmonoid}(\text{HomeoG}(T),\text{restrict}(\text{Composition}(\bigcup T),\text{HomeoG}(T) \times \text{HomeoG}(T)))$
      using Group_ZF_2_5_L2(1) monoid0.group0_1_T1 unfolding monoid0_def by force
      from cl sub ne have $\text{TheNeutralElement}(\text{HomeoG}(T),\text{restrict}(\text{Composition}(\bigcup T),\text{HomeoG}(T) \times \text{HomeoG}(T)))$
      using Group_ZF_2_5_L2(1) group0_1_L6 by blast
    - **moreover**
      - **have id($\bigcup T$)$=\text{TheNeutralElement}(\bigcup T \rightarrow \bigcup T, \text{Composition}(\bigcup T))$ using Group_ZF_2_5_L2(2) by auto
      - **ultimately show**: $\text{TheNeutralElement}(\text{HomeoG}(T),\text{restrict}(\text{Composition}(\bigcup T),\text{HomeoG}(T) \times \text{HomeoG}(T)))$
        by auto
  - **qed**

The homeomorphisms form a group, with the composition.

**Theorem (in topology0) homeo_group:**
- **shows**: $\text{IsAgroup}(\text{HomeoG}(T),\text{restrict}(\text{Composition}(\bigcup T),\text{HomeoG}(T) \times \text{HomeoG}(T)))$
- **proof-**
  - **fix x assume AS:** $x \in \text{HomeoG}(T)$
  - **then have**: surj:$x \in \text{surj}(\bigcup T, \bigcup T)$ and bij:$x \in \text{bij}(\bigcup T, \bigcup T)$ unfolding HomeoG_def IsAhomeomorphism_def bij_def by auto

1095
from bij have converse(x)∈bij(∪T,∪T) using bij_converse_bij by auto
with bij have conx_fun:converse(x)∈∪T→∪T using bij_def by auto
  from surj have id:x 0 converse(x)=id(∪T) using right_comp_inverse by auto
  from conx_fun have Composition(∪T){x,converse(x)}=x 0 converse(x) using func_ZF_5_L2 by auto
  with id have Composition(∪T){x,converse(x)}=id(∪T) by auto
moreover have converse(x)∈HomeoG(T) unfolding HomeoG_def using conx_fun(1)
homeo_inv AS unfolding HomeoG_def by auto
ultimately have ∃M∈HomeoG(T). Composition(∪T){x,M}=id(∪T) by auto
then have ∀x∈HomeoG(T). ∃M∈HomeoG(T). Composition(∪T){x,M}=id(∪T) by auto
then show thesis using homeo_submonoid definition_of_group by auto qed

80.2 Examples computed
As a first example, we show that the group of homeomorphisms of the co-cardinal topology is the group of bijective functions.

theorem homeo_cocardinal:
  assumes InfCard(Q)
  shows HomeoG(CoCardinal(X,Q))=bij(X,X)
proof
  from assms have n:Q≠0 unfolding InfCard_def by auto
  then show HomeoG(CoCardinal(X,Q))⊆bij(X,X) unfolding HomeoG_def
    by auto
  \{fix f assume a:f∈bij(X,X)
    then have converse(f)∈bij(X,X) using bij_converse_bij by auto
    then have cinj:converse(f)∈inj(X,X) unfolding bij_def by auto
    from a have fun:f∈X→X unfolding bij_def inj_def by auto
    then have two:two_top_spaces0((CoCardinal(X,Q)),(CoCardinal(X,Q)),f)
      unfolding two_top_spaces0_def
        by auto
      \{fix N assume N∈Pow(X)
        then have N_def:N=X ∨ (N∈Pow(X) ∧ N≺Q) using closed_sets_cocardinal by auto
        then have restrict(converse(f),N)∈bij(N,converse(f)N) using cinj
          restrict_bij by auto
        then have N≈f-N unfolding vimage_def eqpoll_def by auto
        then have f-N≈N using eqpoll_sym by auto
          with N_def have N=X ∨ (f-N≺Q ∧ N∈Pow(X)) using eq_lesspoll_trans by auto
      \}
  \}
\}
with fun have \( f-N \in X \lor (f-N \not\in Q \land (f-N) \in \text{Pow}(X)) \) using \( \text{func1}_1\_L3 \) \( \text{func1}_1\_L4 \) by auto
then have \( f-N \{\text{is closed in}\}(\text{CoCardinal}(X,Q)) \) using \( \text{closed}\_\text{sets}\_\text{cocardinal} \) \( n \) by auto
\}
then have \( \forall N. N\{\text{is closed in}\}(\text{CoCardinal}(X,Q)) \rightarrow f-N \{\text{is closed in}\}(\text{CoCardinal}(X,Q)) \) using \( \text{two}\_\text{top}\_\text{spaces0}\. \text{Top}_2\_1\_L3 \) \( \text{two}\_\text{top}\_\text{spaces0}\. \text{Top}_2\_1\_L4 \) \( \text{two}\_\text{top}\_\text{spaces0}\. \text{Top}_2\_1\_L2 \) two by auto
\}
then have \( \forall f \in \text{bij}(X,X). \text{IsContinuous}((\text{CoCardinal}(X,Q)),(\text{CoCardinal}(X,Q)),f) \) using \( \text{bij}\_\text{converse}\_\text{bij} \) by auto
\}
then have \( \forall f \in \text{bij}(X,X). \text{IsAhomeomorphism}((\text{CoCardinal}(X,Q)),(\text{CoCardinal}(X,Q)),f) \) unfolding \( \text{IsAhomeomorphism}\_\text{def} \) \( \text{n}\_\text{union}\_\text{cocardinal} \) by auto
then show \( \text{bij}(X,X) \subseteq \text{HomeoG}((\text{CoCardinal}(X,Q))) \) unfolding \( \text{HomeoG}\_\text{def} \) \( \text{bij}\_\text{def} \) \( \text{inj}\_\text{def} \) using \( \text{n}\_\text{union}\_\text{cocardinal} \) by auto
qed

The group of homeomorphism of the excluded set is a direct product of the bijections on \( X \ \backslash T \) and the bijections on \( X \cap T \).

\textbf{theorem homeo_excluded:}
shows \( \text{HomeoG}(\text{ExcludedSet}(X,T))=\{f\in\text{bij}(X,X). \ f(X-T)=(X-T)\} \)
proof
have \( \text{sub1}:X-T \subseteq X \) by auto
\{
\fix g assume \( g\in\text{HomeoG}(\text{ExcludedSet}(X,T)) \)
then have \( \text{fun}:g:X\rightarrow X \) and \( \text{bij}:g\in\text{bij}(X,X) \) and hom: \( \text{IsAhomeomorphism}((\text{ExcludedSet}(X,T)),(X,T)) \) unfolding \( \text{HomeoG}\_\text{def} \) using \( \text{union}\_\text{excludedset} \) unfolding \( \text{IsAhomeomorphism}\_\text{def} \) by auto
\{
assume A: g(X-T)=X and B: X\cap T\neq 0
have rfun: \( \text{restrict}(g,X-T):X-T \rightarrow X \) using fun \( \text{restrict}\_\text{fun} \) \( \text{sub1} \) by auto
moreover
from A fun have \( \{\text{gaa. aa}\in X-T\}=X \) using \( \text{func}\_\text{imagedef} \) \( \text{sub1} \) by auto
then have \( \forall x\in X. x\in\{\text{gaa. aa}\in X-T\} \) by auto
then have \( \forall x\in X. \exists aa\in X-T. x=gaa \) by auto
then have \( \forall x\in X. \exists aa\in X-T. x=\text{restrict}(g,X-T)aa \) by auto
with A have surj: \( \text{restrict}(g,X-T)\in\text{surj}(X,T,X) \) using rfun unfolding surj\_def by auto
from B obtain d where d\in Xd\in T by auto
with bij have gd\in X using apply\_funttype unfolding \( \text{bij}\_\text{def} \) \( \text{inj}\_\text{def} \) by auto
\}
then obtain \( s \) where \( \text{restrict}(g,X-T)s = gds \in X-T \) using \textit{surj unfolding} surj_def by blast
then have \( gs = gd \) by auto
with \( \langle d \in X \rangle \langle s \in X-T \rangle \) have \( s = d \) using \textit{bij unfolding} bij_def inj_def by auto
then have \( False \) using \( \langle s \in X-T \rangle \langle d \in T \rangle \) by auto
}
then have \( g(X-T) = X \rightarrow X\cap T = 0 \) by auto
then have \( \text{reg}(g)(X-T) = X \rightarrow X = X \) by auto
then have \( g(X-T) = X \rightarrow g(X-T) = X-T \) by auto
then have \( g(X-T) = X \rightarrow g \in \{ f \in \text{bij}(X,X). f(X-T) = (X-T) \} \) using \textit{bij} by auto
moreover
\{
fix \( gg \)
assume \( A: gg(X-T) \neq X \) and \( \text{hom2}(X,T) = \text{IsAhomeomorphism}((\text{ExcludedSet}(X,T)),(\text{ExcludedSet}(X,T)),g) \)
from \( \text{hom2} \) have \( \text{fun}: gg \in X \rightarrow X \) and \( \text{bij}: gg \in \text{bij}(X,X) \) unfolding \textit{IsAhomeomorphism_def bij_def inj_def} using union_excludedset by auto
have \( \text{sub}: X-T \subseteq \bigcup (\text{ExcludedSet}(X,T)) \) using union_excludedset by auto
with \( \text{hom2} \) have \( gg(\text{Interior}(X-T,(\text{ExcludedSet}(X,T)))) = \text{Interior}(gg(X-T),(\text{ExcludedSet}(X,T))) \)
using \textit{int_top_invariant} by auto
moreover
from \( \text{sub} \) have \( \text{Interior}(X-T,(\text{ExcludedSet}(X,T))) = X-T \) using \textit{interior_set_excludedset} by auto
ultimately have \( gg(X-T) = \text{Interior}(gg(X-T),(\text{ExcludedSet}(X,T))) \) by auto
moreover
have \( ss: gg(X-T) \subseteq X \) using \textit{fun func1_1_L6(2)} by auto
then have \( \text{Interior}(gg(X-T),(\text{ExcludedSet}(X,T))) = (gg(X-T))-T \) using \textit{interior_set_excludedset} A
by auto
ultimately have \( eq: gg(X-T) = (gg(X-T))-T \) by auto
\{
assume \( (gg(X-T)) \cap T \neq 0 \)
then obtain \( t \) where \( t \in T \) and \( \text{im}: t \in gg(X-T) \) by blast
then have \( t \notin (gg(X-T))-T \) by auto
then have \( False \) using \( eq \) \( \text{im} \) by auto
\}
then have \( (gg(X-T)) \cap T = 0 \) by auto
then have \( gg(X-T) \subseteq X-T \) using \( ss \) by blast
\}
then have \( \forall gg. gg(X-T) \neq X \land \text{IsAhomeomorphism}(\text{ExcludedSet}(X,T),\text{ExcludedSet}(X,T),gg) \rightarrow gg(X-T) \subseteq X-T \) by auto
moreover
from \( \text{bij} \) have \( \text{conbij}(X,T) = \text{bij}(X,X) \) using \textit{bij_converse_bij} by auto
then have \( \text{confun}(X,T) = \text{bij_def inj_def} \) by auto
\{
assume \( A: \text{converse}(g)(X-T) = X \) and \( B: X \cap T \neq 0 \)
have \( \text{rfun} = \text{restrict}(\text{converse}(g),X-T): X-T \rightarrow X \) using \textit{confun restrict_fun} sub1 by auto
moreover
from \( A \) \( \text{confun} \) have \( \{ \text{converse}(g)aa. aa \in X-T \} = X \) using \textit{func_imagedef} sub1 by auto
\}
1098
then have $\forall x \in X. x \in \{\text{converse}(g) aa. aa \in X-T\}$ by auto
then have $\forall x \in X. \exists aa \in X-T. x = \text{converse}(g) aa$ by auto
then have $\forall x \in X. \exists aa \in X-T. x = \text{restrict}(\text{converse}(g), X-T) aa$ by auto
with $A$ have $\text{surj} : \text{restrict}(\text{converse}(g), X-T) \subseteq \text{surj}(X-T, X)$ using $\text{rfun}$ unfolding $\text{surj_def}$ by auto
from $B$ obtain $d$ where $d \in X d \in T$ by auto
with $\text{conbij}$ have $\text{converse}(g) d \in X$ using $\text{apply_funtype}$ unfolding $\text{bij_def}$ $\text{inj_def}$ by auto
then obtain $s$ where $s \in X d \in T$ by auto
with $\text{conbij}$ unfolding $\text{bij_def}$ $\text{inj_def}$ by auto
then have $\text{False}$ using $s \in X d \in T$ by auto
then have $\text{converse}(g)(X-T) = X$ unfolding $\text{vimage_def}$ by auto
with $\text{G}$ have $\text{converse}(g)(X-T) = X$ unfolding $\text{bij_def}$ by auto
with $G$ have $\text{converse}(g)(X-T) = X$ unfolding $\text{bij_def}$ by auto
moreover have $\text{IsAhomeomorphism}(\text{ExcludedSet}(X,T), \text{ExcludedSet}(X,T), \text{converse}(g))$ using $\text{homeo_inv}$ by auto
moreover note $\text{hom}$ ultimately have $g \in \{f \in \text{bij}(X,X). f(X-T) = (X-T)\} \lor (g(X-T) \subseteq X-T \land \text{converse}(g)(X-T) \subseteq X-T)$ unfolding $\text{bij_def}$ by auto
then have $g \in \{f \in \text{bij}(X,X). f(X-T) = (X-T)\} \lor (g(X-T) \subseteq X-T \land \text{converse}(g)(X-T) \subseteq X-T)$ unfolding $\text{bij_def}$ by auto
then show $\text{HomeoG}(\text{ExcludedSet}(X,T)) \subseteq \{f \in \text{bij}(X,X). f(X-T) = (X-T)\}$ by auto
{ fix $g$ assume as $g \in \text{bij}(X,X) g(X-T) = X-T$
 then have $\text{inj} : g \in \text{inj}(X,X)$ and $\text{im} : g(X-T) = (X-T)$ unfolding $\text{bij_def}$ by auto
 from $\text{inj}$ have $g \in \text{bij}(X,X) g(X-T) = X-T$ using $\text{inj_vimage_image}$ $\text{sub1}$ by force
 with $\text{im}$ have as $3 : g(X-T) = X-T$ by auto
 { fix $A$
 assume $A \in \text{ExcludedSet}(X,T))$
 then have $A = X \setminus A \cap T = 0 A \subseteq X$ unfolding $\text{ExcludedSet_def}$ by auto
}
then have \( A \subseteq X - T \lor A = X \) by auto

moreover

\[
\begin{align*}
\{ & \text{assume } A = X \\
& \text{with as(1) have } gA = X \text{ using surj_range_image_domain unfolding bij_def} \\
& \text{by auto} \\
& \}
\end{align*}
\]

moreover

\[
\begin{align*}
\{ & \text{assume } A \subseteq X - T \\
& \text{then have } gA \subseteq (X - T) \text{ using func1_1_L8 by auto} \\
& \text{then have } gA \subseteq (X - T) \text{ using as(2) by auto} \\
& \}
\end{align*}
\]

ultimately have \( gA \subseteq (X - T) \lor gA = X \) by auto

then have \( gA \in (\text{ExcludedSet}(X, T)) \) unfolding ExcludedSet_def by auto

then have \( \forall A \in (\text{ExcludedSet}(X, T)). gA \in (\text{ExcludedSet}(X, T)) \) by auto

moreover

\[
\begin{align*}
\{ & \text{fix } A \text{ assume } A \in (\text{ExcludedSet}(X, T)) \\
& \text{then have } A = X \lor A \cap T = 0 A \subseteq X \text{ unfolding ExcludedSet_def by auto} \\
& \text{then have } A \subseteq X - T \lor A = X \text{ by auto moreover} \\
& \}
\end{align*}
\]

moreover

\[
\begin{align*}
\{ & \text{assume } A = X \\
& \text{with as(1) have } g - A = X \text{ using func1_1_L4 unfolding bij_def inj_def} \\
& \text{by auto} \\
& \}
\end{align*}
\]

moreover

\[
\begin{align*}
\{ & \text{assume } A \subseteq X - T \\
& \text{then have } g - A \subseteq (X - T) \text{ using func1_1_L8 by auto} \\
& \text{then have } g - A \subseteq (X - T) \text{ using as(3) by auto} \\
& \}
\end{align*}
\]

ultimately have \( g - A \subseteq (X - T) \lor g - A = X \) by auto

then have \( g - A \in (\text{ExcludedSet}(X, T)) \) unfolding ExcludedSet_def by auto

then have \( \text{IsContinuous}(\text{ExcludedSet}(X, T), \text{ExcludedSet}(X, T), g) \) unfolding IsContinuous_def by auto

note as(1) ultimately have \( \text{IsAhomeomorphism}(\text{ExcludedSet}(X, T), \text{ExcludedSet}(X, T), g) \)

using union_excludedset bij_cont_open_homeo by auto

with as(1) have \( g \in \text{HomeoG}(\text{ExcludedSet}(X, T)) \) unfolding bij_def inj_def

HomeoG_def using union_excludedset by auto

then show \( \{ f \in \text{bij}(X, X) . f \ (X - T) = X - T \} \subseteq \text{HomeoG}(\text{ExcludedSet}(X, T)) \) by auto

qed

We now give some lemmas that will help us compute \( \text{HomeoG}(\text{IncludedSet}(X, T)) \).

lemma cont_in_cont_ex:
assumes IsContinuous(IncludedSet(X,T),IncludedSet(X,T),f) f:X→X T⊆X
shows IsContinuous(ExcludedSet(X,T),ExcludedSet(X,T),f)
proof-
  from assms(2,3) have two:two_top_spaces0(IncludedSet(X,T),IncludedSet(X,T),f)
using union_includedset includedset_is_topology
  unfolding two_top_spaces0_def by auto
  {  
    fix A assume A∈(ExcludedSet(X,T))
    then have A∩T=0 ∨ A=X∩X unfolding ExcludedSet_def by auto
    then have A{is closed in}(IncludedSet(X,T)) using closed_sets_includedset
assms by auto
    then have f-A{is closed in}(IncludedSet(X,T)) using two_top_spaces0.TopZF_2_1_L1
assms(1)
    two assms includedset_is_topology by auto
    then have (f-A)∩T=0 ∨ f-A=X∩A⊆X using closed_sets_includedset assms(1,3)
by auto
    then have f-A∈(ExcludedSet(X,T)) unfolding ExcludedSet_def by auto
  }
  then show IsContinuous(ExcludedSet(X,T),ExcludedSet(X,T),f) unfolding IsContinuous_def by auto
qed

lemma cont_ex_cont_in:
assumes IsContinuous(ExcludedSet(X,T),ExcludedSet(X,T),f) f:X→X T⊆X
shows IsContinuous(IncludedSet(X,T),IncludedSet(X,T),f)
proof-
  from assms(2) have two:two_top_spaces0(ExcludedSet(X,T),ExcludedSet(X,T),f)
using union_excludedset excludedset_is_topology
  unfolding two_top_spaces0_def by auto
  {  
    fix A assume A∈(IncludedSet(X,T))
    then have T⊆A ∨ A=0⊆X unfolding IncludedSet_def by auto
    then have A{is closed in}(ExcludedSet(X,T)) using closed_sets_excludedset
assms by auto
    then have f-A{is closed in}(ExcludedSet(X,T)) using two_top_spaces0.TopZF_2_1_L1
assms(1)
    two assms excludedset_is_topology by auto
    then have T⊆(f-A) ∨ f-A=0∩A⊆X using closed_sets_excludedset assms(1,3)
by auto
    then have f-A∈(IncludedSet(X,T)) unfolding IncludedSet_def by auto
  }
  then show IsContinuous(IncludedSet(X,T),IncludedSet(X,T),f) unfolding IsContinuous_def by auto
qed

The previous lemmas imply that the group of homeomorphisms of the included set topology is the same as the one of the excluded set topology.

lemma homeo_included:
assumes T⊆X
shows $\text{HomeoG}(\text{IncludedSet}(X,T)) = \{ f \in \text{bij}(X, X) . f(X - T) = X - T \}$

proof-

{ fix $f$ assume $f \in \text{HomeoG}(\text{IncludedSet}(X,T))$
    then have $\text{hom} : \text{IsAHomeomorphism}(\text{IncludedSet}(X,T), \text{IncludedSet}(X,T), f)$
    and fun: $f : X \to X$ and
        $\text{bij} : f \in \text{bij}(X, X)$ unfolding $\text{HomeoG}\_\text{def}$ $\text{IsAHomeomorphism}\_\text{def}$ using $\text{union}\_\text{includedset}$
    assms by auto
    then have $\text{cont} : \text{IsContinuous}(\text{IncludedSet}(X,T), \text{IncludedSet}(X,T), f)$
    unfolding $\text{IsAHomeomorphism}\_\text{def}$ by auto
    then have $\text{IsContinuous}(\text{ExcludedSet}(X,T), \text{ExcludedSet}(X,T), f)$ using $\text{cont} \in \text{cont}\_\text{ex}$ fun assms by auto

    moreover
    { from $\text{hom}$ have $\text{cont} - 1 : \text{IsContinuous}(\text{IncludedSet}(X,T), \text{IncludedSet}(X,T), \text{converse}(f))$
        unfolding $\text{IsAHomeomorphism}\_\text{def}$ by auto
        have $\text{converse}(f) : X \to X$ using $\text{bij}\_\text{converse} \in \text{bij}$ $\text{bij}\_\text{def}$
        inj\_def by auto moreover
        note assms ultimately
        have $\text{IsContinuous}(\text{ExcludedSet}(X,T), \text{ExcludedSet}(X,T), \text{converse}(f))$
        using $\text{cont}\_\text{in}\_\text{cont}\_\text{ex}$ assms by auto
    }
    then have $\text{IsContinuous}(\text{ExcludedSet}(X,T), \text{ExcludedSet}(X,T), \text{converse}(f))$
    by auto

    moreover note $\text{bij}$ ultimately
    have $\text{IsAHomeomorphism}(\text{ExcludedSet}(X,T), \text{ExcludedSet}(X,T), f)$ unfolding $\text{IsAHomeomorphism}\_\text{def}$
    using $\text{union}\_\text{excludedset}$ by auto
    with fun have $f \in \text{HomeoG}(\text{ExcludedSet}(X,T))$ unfolding $\text{HomeoG}\_\text{def}$ using $\text{union}\_\text{excludedset}$ by auto
    }
    then have $\text{HomeoG}(\text{IncludedSet}(X,T)) \subseteq \text{HomeoG}(\text{ExcludedSet}(X,T))$ by auto

    moreover
    { fix $f$ assume $f \in \text{HomeoG}(\text{ExcludedSet}(X,T))$
        then have $\text{hom} : \text{IsAHomeomorphism}(\text{ExcludedSet}(X,T), \text{ExcludedSet}(X,T), f)$
        and fun: $f : X \to X$ and
            $\text{bij} : f \in \text{bij}(X, X)$ unfolding $\text{HomeoG}\_\text{def}$ $\text{IsAHomeomorphism}\_\text{def}$ using $\text{union}\_\text{excludedset}$
        assms by auto
        then have $\text{cont} : \text{IsContinuous}(\text{ExcludedSet}(X,T), \text{ExcludedSet}(X,T), f)$
        unfolding $\text{IsAHomeomorphism}\_\text{def}$ by auto
        then have $\text{IsContinuous}(\text{IncludedSet}(X,T), \text{IncludedSet}(X,T), f)$ using $\text{cont}\_\text{ex}\_\text{cont}\_\text{in}$ fun assms by auto

        moreover
        { from $\text{hom}$ have $\text{cont} - 1 : \text{IsContinuous}(\text{ExcludedSet}(X,T), \text{ExcludedSet}(X,T), \text{converse}(f))$
            unfolding $\text{IsAHomeomorphism}\_\text{def}$ by auto
            have $\text{converse}(f) : X \to X$ using $\text{bij}\_\text{converse} \in \text{bij}$ $\text{bij}\_\text{def}$
            inj\_def by auto moreover
            note assms ultimately
            have $\text{IsContinuous}(\text{IncludedSet}(X,T), \text{IncludedSet}(X,T), \text{converse}(f))$
        }
    }

}
using cont_ex_cont_in assms by auto
}
then have IsContinuous(IncludedSet(X,T),IncludedSet(X,T),converse(f))
by auto
moreover note bij ultimately
have IsAhomeomorphism(IncludedSet(X,T),IncludedSet(X,T),f) unfolding IsAhomeomorphism_def
using union_includedset assms by auto
with fun have f∈HomeoG(IncludedSet(X,T)) unfolding HomeoG_def using union_includedset assms by auto
}
then have HomeoG(ExcludedSet(X,T))⊆HomeoG(IncludedSet(X,T)) by auto
ultimately
show thesis using homeo_excluded by auto
qed

Finally, let's compute part of the group of homeomorphisms of an order topology.

lemma homeo_order:
assumes IsLinOrder(X,r)∃ x y. x≠y∧x∈X∧y∈X
shows ord_iso(X,r,X,r)⊆HomeoG(OrdTopology X r)
proof
fix f assume f∈ord_iso(X,r,X,r)
then have bij:f∈bij(X,X) and ord:∀x∈X. ∀y∈X. ⟨x, y⟩ ∈ r ↔ ⟨f x, f y⟩ ∈ r
unfolding ord_iso_def by auto
have twoSpac:two_top_spaces0(OrdTopology X r,OrdTopology X r,f) unfolding two_top_spaces0_def
using bij unfolding bij_def inj_def using union_ordtopology[OF assms]
Ordtopology_is_a_topology(1)[OF assms(1)]
by auto
{ fix c d assume A:c∈Xd∈X

{ fix x assume AA:x∈Xx≠cx≠d(c,x)∈r(x,d)∈r
then have ⟨fx,fx⟩∈r⟨fx,fd⟩∈r using A(2,1) ord by auto moreover
{ assume fx=fc ∨ fx=fd
then have x=c∨x=d using bij unfolding bij_def inj_def using A(2,1)
AA(1) by auto
then have False using AA(2,3) by auto
}
then have fx≠fcfx≠fd by auto moreover
have fx∈X using bij unfolding bij_def inj_def using apply_type AA(1)
by auto
ultimately have fx∈IntervalX(X,r,fc,fd) unfolding IntervalX_def
Interval_def by auto
}
then have {fx. x∈IntervalX(X,r,c,d)}⊆IntervalX(X,r,fc,fd) unfolding

by auto
moreover
{ fix y assume y∈IntervalX(X,r,fc,fd)
  then have y:y∈X y̸=fc y̸=fd ⟨fc,y⟩∈r ⟨y,fd⟩∈r unfolding IntervalX_def
  by auto
then obtain s where s:s∈X y=:fs using bij unfolding bij_def surj_def
by auto
  { assume s=c∨s=d
    then have fs=fc∨fs=fd by auto
    then have False using y(2,3) by auto
  }
then have s≠c≠d by auto moreover
have ⟨c,s⟩∈r ⟨s,d⟩∈r using y(4,5) s ord A(2,1) by auto moreover
note s(1) ultimately have s∈IntervalX(X,r,c,d) unfolding IntervalX_def
Interval_def by auto
then have y∈{fx. x∈IntervalX(X,r,c,d)} using s(2) by auto
} ultimately have {fx. x∈IntervalX(X,r,c,d)}=IntervalX(X,r,fc,fd) by auto
moreover
have IntervalX(X,r,c,d)⊆X unfolding IntervalX_def by auto
  have f:X→X using bij unfolding bij_def surj_def by auto ultimately
  have fintervalX(X,r,c,d)=IntervalX(X,r,fc,fd) using func_imagedef
by auto
then have inter:∀c∈X. ∀d∈X. fIntervalX(X,r,c,d)=IntervalX(X,r,fc,fd)
∧ fc∈X ∧ fd∈X using bij unfolding bij_def inj_def by auto
{ fix c assume A:c∈X
  { fix x assume AA:x∈X x≠c(c,x)∈r
    then have ⟨fc,fx⟩∈r using A ord by auto moreover
    { assume fx=fc
      then have x=c using bij unfolding bij_def inj_def using A AA(1)
by auto
      then have False using AA(2) by auto
    }
then have fx≠fc by auto moreover
  }
then have fx∈X using bij unfolding bij_def inj_def using apply_type AA(1)
by auto
  ultimately have fx∈RightRayX(X,r,fc) unfolding RightRayX_def by auto
then have {fx. x∈RightRayX(X,r,c)}⊆RightRayX(X,r,fc) unfolding RightRayX_def
by auto

1104
moreover
\{  
  \text{fix } y \text{ assume } y \in \text{RightRayX}(X,r,fc)  
  \text{then have } y : y \in X \neq fc \langle fc, y \rangle \in r \text{ unfolding RightRayX_def by auto}  
  \text{then obtain } s \text{ where } s : s \in X = fs \text{ using bij unfolding bij_def surj_def by auto}  
\}

\{  
  \text{assume } s = c  
  \text{then have } fs = fc \text{ by auto}  
  \text{then have False using } s(2) y(2) \text{ by auto}  
\}

then have \langle c, s \rangle \in r \text{ using } y(3) s \text{ ord } A \text{ by auto moreover}

\text{note } s(1) \text{ ultimately have } s \in \text{RightRayX}(X,r,c) \text{ unfolding RightRayX_def by auto}

then have \langle c, s \rangle \in r \text{ using } y(3) s \text{ ord } A \text{ by auto moreover}

\text{ultimately have } \{ fx, x \in \text{RightRayX}(X,r,c) \} = \text{RightRayX}(X,r,fc) \text{ by auto}

moreover

\text{have } \text{RightRayX}(X,r,c) \subseteq X \text{ unfolding RightRayX_def by auto moreover}

\text{have } f : X \rightarrow X \text{ using bij unfolding bij_def surj_def by auto ultimately}

\text{have } f \text{RightRayX}(X,r,c) = \text{RightRayX}(X,r,fc) \text{ using func_imagedef by auto}

\text{then have } \text{rray} : \forall c \in X. f \text{RightRayX}(X,r,c) = \text{RightRayX}(X,r,fc) \land fc \in X \text{ using bij}

\text{unfolding bij_def inj_def by auto}
\}

\{  
  \text{fix } c \text{ assume } A : c \in X  
  \{  
    \text{fix } x \text{ assume } AA : x \in X \neq c \langle x, c \rangle \in r  
    \text{then have } \langle fx, fc \rangle \in r \text{ using } A \text{ ord } A \text{ by auto moreover}  
    \{  
      \text{assume } fx = fc  
      \text{then have } x = c \text{ using bij unfolding bij_def inj_def using } A \text{ AA}(1) \text{ by auto}  
    \}  
    \text{then have False using } AA(2) \text{ by auto}  
  \}  
  \text{then have } fx \neq fc \text{ by auto moreover}

\text{have } fx \in X \text{ using bij unfolding bij_def inj_def using apply_type } AA(1) \text{ by auto ultimately}

\text{have } fx \in \text{LeftRayX}(X,r,fc) \text{ unfolding LeftRayX_def by auto}
\}

\text{then have } \{ fx, x \in \text{LeftRayX}(X,r,c) \} \subseteq \text{LeftRayX}(X,r,fc) \text{ unfolding LeftRayX_def by auto moreover}

\{  
  \text{fix } y \text{ assume } y \in \text{LeftRayX}(X,r,fc)  
  \text{then have } y : y \in X \neq fc \langle y, fc \rangle \in r \text{ unfolding LeftRayX_def by auto}  
  \text{then obtain } s \text{ where } s : s \in X = fs \text{ using bij unfolding bij_def surj_def}
\}

1105
by auto
{
  assume s=c
  then have fs=fc by auto
  then have False using s(2) y(2) by auto
}
then have s≠c by auto moreover
have ⟨s,c⟩∈r using y(3) s ord A by auto moreover
note s(1) ultimately have s∈LeftRayX(X,r,c) unfolding LeftRayX_def by auto
then have s̸=c by auto
moreover
have ⟨s,c⟩∈r using y(3) s ord A by auto
moreover
note s(1) ultimately have s∈LeftRayX(X,r,c)
unfolding LeftRayX_def by auto
then have y∈{fx. x∈LeftRayX(X,r,c)} using s(2) by auto
}
ultimately have {fx. x∈LeftRayX(X,r,c)}=LeftRayX(X,r,fc)
by auto
moreover
have LeftRayX(X,r,c)⊆X unfolding LeftRayX_def by auto
moreover
have f:X→X using bij unfolding bij_def surj_def by auto
ultimately
have f(LeftRayX(X,r,c))=LeftRayX(X,r,fc)
using func_imagedef by auto
}
then have lray:∀c∈X. f(LeftRayX(X,r,c))=LeftRayX(X,r,fc) ∧ fc∈X using bij
unfolding bij_def inj_def by auto
have r1:∀U∈{IntervalX(X, r, b, c) . ⟨b,c⟩∈X × X} ∪ {LeftRayX(X, r, b) . b∈X}
∪ {RightRayX(X, r, b) . b∈X}. fU∈({IntervalX(X, r, b, c) . ⟨b,c⟩∈X × X} 
∪ {LeftRayX(X, r, b) . b∈X} 
∪ {RightRayX(X, r, b) . b∈X})
apply safe prefer 3 using rray apply blast
prefer 2 using lray apply blast
using inter apply auto
proof-
  fix xa y assume xa∈X ya∈X
  then have fxax∈Xfy∈X using bij unfolding bij_def inj_def by auto
  then show ∃x∈X. ∃ya∈X. IntervalX(X, r, f xa, f y) = IntervalX(X, r, x, ya)
by auto
qed
have r2:{IntervalX(X, r, b, c) . ⟨b,c⟩∈X × X} ∪ {LeftRayX(X, r, b) . b∈X}
∪ {RightRayX(X, r, b) . b∈X} ⊆ (OrdTopology X r)
using base_sets_open[OF Ordtopology_is_a_topology(2)[OF assms(1)]]
by blast
{
  fix U assume U∈{IntervalX(X, r, b, c) . ⟨b,c⟩∈X × X} ∪ {LeftRayX(X, r, b) . b∈X}
  with r1 have fU∈{IntervalX(X, r, b, c) . ⟨b,c⟩∈X × X} ∪ {LeftRayX(X, r, b) . b∈X}
  by auto
  with r2 have fU∈(OrdTopology X r) by blast
}
then have ∀U∈{IntervalX(X, r, b, c) . ⟨b,c⟩∈X × X} ∪ {LeftRayX(X, r, b) . b∈X} ∪
{RightRayX(X, r, b) . b∈X}. fU∈(OrdTopology X r) by blast

1106
then have \( f\text{\_open}: \forall U \in (\text{OrdTopology } X \ r). fU \in (\text{OrdTopology } X \ r) \) using two_top_spaces0.base_image_open[OF twoSpac Ordtopology_is_a_topology(2)[OF assms(1)]] by auto

\[
\begin{align*}
\{ & \text{fix } c \text{ d assume } A:c \in X \in X \\
& \text{then obtain } cc \text{ dd where pre: } f:\text{IntervalX}(X, r, cc, dd) = \text{IntervalX}(X, r, c, d) \text{ by auto} \\
& \text{then have } f-(f\text{IntervalX}(X, r, cc, dd)) = f-(\text{IntervalX}(X, r, c, d)) \text{ by auto} \\
& \text{moreover} \\
& \text{have } f\inj(X, X) \text{ using bij unfolding bij_def by auto ultimately} \\
& \text{have } \text{IntervalX}(X, r, cc, dd) = f-\text{IntervalX}(X, r, c, d) \text{ using inj_vimage_image by auto} \\
& \text{moreover} \\
& \text{from pre(3,4) have } \text{IntervalX}(X, r, cc, dd) \in \{\text{IntervalX}(X, r, e1, e2). (e1, e2) \in X \times X\} \text{ by auto} \\
& \text{ultimately have } f-\text{IntervalX}(X, r, c, d) \in (\text{OrdTopology } X \ r) \text{ using base_sets_open[OF Ordtopology_is_a_topology(2)[OF assms(1)]]} \text{ by auto} \\
& \text{then have } \text{inter} : \forall c \in X. \forall d \in X. f-\text{IntervalX}(X, r, c, d) \in (\text{OrdTopology } X \ r) \text{ by auto} \\
\end{align*}
\]

\[
\begin{align*}
\{ & \text{fix } c \text{ assume } A:c \in X \\
& \text{then obtain } cc \text{ where pre: } f:\text{RightRayX}(X, r, cc) = \text{RightRayX}(X, r, c) \text{ by auto} \\
& \text{then have } f-(f\text{RightRayX}(X, r, cc)) = f-(\text{RightRayX}(X, r, c)) \text{ by auto} \\
& \text{moreover} \\
& \text{have } \text{RightRayX}(X, r, cc) \subseteq X \text{ unfolding RightRayX_def by auto} \\
& \text{ultimately have } f-\text{RightRayX}(X, r, c) \in (\text{OrdTopology } X \ r) \text{ using base_sets_open[OF Ordtopology_is_a_topology(2)[OF assms(1)]]} \text{ by auto} \\
& \text{then have } \text{rray} : \forall c \in X. f-\text{RightRayX}(X, r, c) \in (\text{OrdTopology } X \ r) \text{ by auto} \\
\end{align*}
\]

1107
then obtain cc where pre: fcc = ccc ∈ X using bij unfolding bij_def surj_def by blast

with lray have f · LeftRayX(X, r, cc) = LeftRayX(X, r, c) by auto
then have f · (f · LeftRayX(X, r, cc)) = f · (LeftRayX(X, r, c)) by auto

moreover
have LeftRayX(X, r, cc) ⊆ X unfolding LeftRayX_def by auto
moreover
have f · ∈ inj(X, X) using bij unfolding bij_def by auto
ultimately
have LeftRayX(X, r, cc) = f · LeftRayX(X, r, c) using inj_vimage_image by auto
moreover
from pre(2) have LeftRayX(X, r, cc) ∈ {LeftRayX(X, r, e2). e2 ∈ X} by auto
ultimately have f · LeftRayX(X, r, c) ∈ (OrdTopology X r) using base_sets_open[OF Ordtopology_is_a_topology(2)[OF assms(1)] by auto

then have ∀ c ∈ X. f · LeftRayX(X, r, c) ∈ (OrdTopology X r) by auto
fix U assume U ∈ {IntervalX(X, r, b, c) . ⟨b, c⟩ ∈ X × X} ∪ {LeftRayX(X, r, b) . b ∈ X} ∪ {RightRayX(X, r, b) . b ∈ X}
with lray inter rray have f · U ∈ (OrdTopology X r) by auto

then have ∀ U ∈ {IntervalX(X, r, b, c) . ⟨b, c⟩ ∈ X × X} ∪ {LeftRayX(X, r, b) . b ∈ X} ∪ {RightRayX(X, r, b) . b ∈ X}.
    f · U ∈ (OrdTopology X r) by blast
then have fcont: IsContinuous(OrdTopology X r, OrdTopology X r, f) using two_top_spaces0.Top_ZF_2_1_L5[OF twoSpac Ordtopology_is_a_topology(2)[OF assms(1)] by auto
from fcont f · open bij have IsAhomeomorphism(OrdTopology X r, OrdTopology X r, f) using bij_cont_open_homeo
union_ordtopology[OF assms] by auto
then show f ∈ HomeoG(OrdTopology X r) unfolding HomeoG_def using bij unfoldng bij_def inj_def by auto
qed

This last example shows that order isomorphic sets give homeomorphic topological spaces.

80.3 Properties preserved by functions

The continuous image of a connected space is connected.

theorem (in two_top_spaces0) cont_image_conn:
  assumes IsContinuous(τ₁, τ₂, f) f ∈ surj(X₁, X₂) τ₁{is connected}
  shows τ₂{is connected}
proof-

Every continuous function from a space which has some property $P$ and a space which has the property $\text{anti}(P)$, given that this property is preserved by continuous functions, it follows that the range of the function is in the spectrum. Applied to connectedness, it follows that continuous functions from a connected space to a totally-disconnected one are constant.

corollary (in two_top_spaces0) cont_conn_tot_disc:
  assumes IsContinuous($\tau_1, \tau_2, f$) $\tau_1$ {is connected} $\tau_2$ {is totally-disconnected}
  f:X_1→X_2 $X_1\neq\emptyset$
  shows $\exists q\in X_2. \ \forall w\in X_1. \ f(w)=q$

proof-
  from assms(4) have surj:f∈surj(X_1,range(f)) using fun_is_surj by auto
  have sub:range(f)⊆X_2 using func1_1_L5B assms(4) by auto
  from assms(1) have cont:IsContinuous($\tau_1, \tau_2$ {restricted to}range(f),f) using restr_image_cont range_image_domain
  assms(4) by auto
have union:$\bigcup (\tau_2 \{\text{restricted to}\} \text{range}(f)) = \text{range}(f)$ unfolding RestrictedTo_def using sub by auto
then have two_top_spaces0$(\tau_1, \tau_2 \{\text{restricted to}\} \text{range}(f), f)$ unfolding two_top_spaces0_def using surj unfolding surj_def using tau1_is_top topology0.Top_1_L4 unfolding topology0_def using tau2_is_top by auto
then have conn$(\tau_2 \{\text{restricted to}\} \text{range}(f)) \{\text{is connected}\}$ using two_top_spaces0.cont_image_conn surj assms(2) cont union by auto
then have range$(f) \{\text{is in the spectrum of}\} \text{IsConnected}$ using assms(3) sub unfolding IsTotDis_def antiProperty_def using union by auto
then have range$(f) \{\text{is in the spectrum of}\} \text{IsConnected}$ using assms(3) sub unfolding IsTotDis_def antiProperty_def using union by auto
then have range$(f) \subseteq 1$ using conn_spectrum by auto moreover from assms(5) have $fX_1 \neq 0$ using func1_1_L15A assms(4) by auto
then have range$(f) \neq 0$ using range_image_domain assms(4) by auto ultimately obtain q where uniq:range$(f) = \{q\}$ using lepoll_1_is_sing by blast
\begin{enumerate}
\item fix w assume $w \in X_1$
\then have $fw \in \text{range}(f)$ using func1_1_L5A(2) assms(4) by auto
\with uniq have $fw = q$ by auto
\end{enumerate}
then have $\forall w \in X_1. \ f_w = q$ by auto
then show thesis using uniq sub by auto
qed

The continuous image of a compact space is compact.

**theorem (in two_top_spaces0) cont_image_com:**
assumes IsContinuous$(\tau_1, \tau_2, f) f \in \text{surj}(X_1, X_2) X_1 \{\text{is compact of cardinal}\} K \{\text{in}\} \tau_1$
shows $X_2 \{\text{is compact of cardinal}\} K \{\text{in}\} \tau_2$

**proof:**
have $X_2 \subseteq \bigcup \tau_2$ by auto moreover
\begin{enumerate}
\item fix U assume as:$X_2 \subseteq \bigcup U U \subseteq \tau_2$
\then have $P : (f - V. V \in U) \subseteq \tau_1$ using assms(1) unfolding IsContinuous_def by auto
\from as(1) have $f - X_2 \subseteq f - (\bigcup U)$ by blast
\then have $f - X_2 \subseteq \text{converse}(f)(\bigcup U)$ unfolding vimage_def by auto moreover
\then have converse$(f)(\bigcup U) = (\bigcup V \in U. \text{converse}(f) V)$ using image_UN by force
ultimately
\then have $f - X_2 \subseteq (\bigcup V \in U. \text{converse}(f) V)$ by auto
\then have $f - X_2 \subseteq (\bigcup V \in U. f^{-V})$ unfolding vimage_def by auto
\then have $X_1 \subseteq (\bigcup V \in U. f^{-V})$ using func1_1_L4 assms(2) unfolding surj_def by force
ultimately
\then have $X_1 \subseteq \bigcup \{f - V. V \in U\}$ by auto
\with P assms(3) have $\exists N \in \text{Pow}(\{f - V. V \in U\}). X_1 \subseteq \bigcup N \wedge N \prec K$ unfolding IsCompactOfCard_def by auto

1110
then obtain $N$ where $N \in \text{Pow}\{\{f \cdot V. V \in U\}\} \cap X_1 \subseteq \bigcup N \blacktriangleleft K$ by auto
then have $\text{fin}:N \blacktriangleleft K$ and $\text{sub}:N \subseteq \{f \cdot V. V \in U\}$ and $\text{cov}:X_1 \subseteq \bigcup N$ unfolding $\text{FinPow}\_\text{def}$ by auto
from $\text{sub}$ have $\{fR. R \in N\} \subseteq \bigcup N$ using $\text{surj}\_\text{image_vimage}$ assms(2) by auto moreover have $\{fR. R \in N\} \subseteq U$ using $\text{surj}\_\text{image_vimage}$ assms(2) by auto moreover

let $\text{FN}=\{\langle R, fR \rangle. R \in N\}$

have $\text{FN}:\text{FN}:N \rightarrow \{fR. R \in N\}$ unfolding Pi$/_\text{def}$ function$/_\text{def}$ domain$/_\text{def}$ by auto

{ fix $S$ assume $S \in \{fR. R \in N\}$
  then obtain $R$ where $R\_\text{def}:R \in N \Rightarrow s$ by auto
  then have $\langle R, fR \rangle \in \text{FN}$ by auto
  then have $\exists R \in N. FNR=fR$ using $\text{FN}$ apply$/_\text{equality}$ by auto
  then have $\exists R \in N. FNR=fR$ using $\text{FN}$ apply$/_\text{equality}$ by auto
}

then have $\text{surj}:\text{FN} \in \text{surj}(N, \{fR. R \in N\})$ unfolding $\text{surj}\_\text{def}$ using $\text{FN}$ by force
from $\text{fin}$ have $N:N \blacktriangleleft K \blacktriangleleft K$ using assms(3) lesspoll$/_\text{imp} \blacktriangleleft$ lesspoll unfolding $\text{IsCompactOfCard}\_\text{def}$ using Card$/_\text{is}_\text{Ord}$ by auto
then have $\{fR. R \in N\} \subseteq N$ using $\text{surj}\_\text{fun_inv}_2 \text{surj}$ by auto
then have $\{fR. R \in N\} \blacktriangleleft K$ using $\text{fin}$ lesspoll$/_\text{trans}$ by blast moreover have $\bigcup \{fR. R \in N\}=f(\bigcup N)$ using image$/_\text{un}$ by auto
then have $X_1 \subseteq \bigcup \{fR. R \in N\}$ using $\text{cov}$ by blast
then have $X_2 \subseteq \bigcup \{fR. R \in N\}$ using assms(2) surj$/_\text{range_image_domain}$ by auto
ultimately have $\exists NN \in \text{Pow}(U). X_2 \subseteq \bigcup NN \land NN \blacktriangleleft K$ by auto
}
then have $\forall U \in \text{Pow}(\tau_2). X_2 \subseteq \bigcup U \longrightarrow (\exists NN \in \text{Pow}(U). X_2 \subseteq \bigcup NN \land NN \blacktriangleleft K)$ by auto
ultimately show thesis using assms(3) unfolding $\text{IsCompactOfCard}\_\text{def}$ by auto
qed

As it happens to connected spaces, a continuous function from a compact space to an anti-compact space has finite range.

corollary (in two_top_spaces0) cont_comp_anti_comp:
  assumes IsContinuous($\tau_1, \tau_2, f$) $X_1$ is compact in $\tau_1$ $\tau_2$ is anti-compact
  shows Finite(range(f)) and range(f)$\neq 0$
proof-
  from assms(4) have $\text{surj}:f \in \text{surj}(X_1, \text{range}(f))$ using $\text{fun_is_surj}$ by auto
  have $\text{sub}:\text{range}(f) \subseteq X_2$ using $\text{func1}_1\_\text{L5}$ by auto
  from assms(1) have $\text{cont}:\text{IsContinuous}(\tau_1, \tau_2$ restricted to)$\text{range}(f),f)$ using $\text{restr_image_cont}$ by auto
  assume $\text{assms}(4)$ by auto

1111
have union:\(\bigcup (\tau_2 \text{restricted to} \text{range}(f)) = \text{range}(f)\) unfolding RestrictedTo_def using sub by auto
then have two_top_spaces0(\tau_1,\tau_2 \text{restricted to} \text{range}(f), f) unfolding two_top_spaces0_def using surj unfolding surj_def using tau1_is_top topology0.Top_1_L4 unfolding topology0_def using tau2_is_top by auto
then have two_top_spaces0(\tau_1, \tau_2 \text{restricted to} \text{range}(f), f) unfolding two_top_spaces0_def using surj unfolding surj_def using tau1_is_top topology0.Top_1_L4 unfolding topology0_def using tau2_is_top by auto
then have range(f) \{is compact in\} (\tau_2 \text{restricted to} \text{range}(f)) unfolding two_top_spaces0_def using surj unfolding surj_def using tau1_is_top topology0.Top_1_L4 unfolding topology0_def using tau2_is_top by auto
then have range(f) \{is compact in\} (\tau_2 \text{restricted to} \text{range}(f)) using surj unfolding two_top_spaces0.cont_image_com cont union unfolding topology0_def using tau2_is_top by auto
then show Finite(range(f)) using compact_spectrum by auto moreover from \(\text{assms(5)}\) have \(fX \neq 0\) using func1_1_L15A \(\text{assms(4)}\) by auto then show range(f) \neq 0 using range_image_domain \(\text{assms(4)}\) by auto qed

As a consequence, it follows that quotient topological spaces of compact (connected) spaces are compact (connected).

corollary (in topology0) compQuot:
assumes \((\bigcup T) \{\text{is compact in}\} T \equiv (\bigcup T, r)\)
shows \((\bigcup T) / r \{\text{is compact in}\} \{\text{quotient by}\} r\)
proof-
  have surj:\{\langle b, r\{b\}\rangle. b \in (\bigcup T)\} \in \text{surj}(\bigcup T, (\bigcup T) / r) using quotient_proj_surj by auto
  moreover have tot:\(\bigcup \{\text{quotient by}\} r = (\bigcup T) / r\) using total_quo_equi \(\text{assms(2)}\) by auto
  ultimately have cont:isContinuous(T, \{\text{quotient by}\} r, \{\langle b, r\{b\}\rangle. b \in (\bigcup T)\}) using quotient_func_cont
    unfolding equivQuo_def \(\text{assms(2)}\) by auto
  from surj tot have two_top_spaces0(T, \{\text{quotient by}\} r, \{\langle b, r\{b\}\rangle. b \in (\bigcup T)\}) unfolding two_top_spaces0_def using topSpaceAssum equiv_quo_is_top \(\text{assms(2)}\) unfolding surj_def by auto
  with surj tot have two_top_spaces0.cont_image_com \(\text{Compact_is_card_nat}\) by force qed

corollary (in topology0) ConnQuot:
assumes T{is connected} equiv(\bigcup T, r)
shows (\{\text{quotient by}\} r\{is connected\}
proof-
  have surj:\{\langle b, r\{b\}\rangle. b \in (\bigcup T)\} \in \text{surj}(\bigcup T, (\bigcup T) / r) using quotient_proj_surj by auto
  moreover have tot:\(\bigcup \{\text{quotient by}\} r = (\bigcup T) / r\) using total_quo_equi \(\text{assms(2)}\) by auto
  ultimately have cont:isContinuous(T, \{\text{quotient by}\} r, \{\langle b, r\{b\}\rangle. b \in (\bigcup T)\}) using quotient_func_cont

EquivQuo_def assms(2) by auto
from surj tot have two_top_spaces0(T,{quotient by}r,{⟨b,r(b)⟩. b∈∪T}) unfolding two_top_spaces0_def
  using topSpaceAssum equiv_quo_is_top assms(2) unfolding surj_def by auto
with surj cont tot assms(1) show thesis using two_top_spaces0.cont_image_conn by force qed
end

81 Topology 10
theory Topology_ZF_10 imports Topology_ZF_7 begin
This file deals with properties of product spaces. We only consider product of two spaces, and most of this proofs, can be used to prove the results in product of a finite number of spaces.

81.1 Closure and closed sets in product space
The closure of a product, is the product of the closures.
lemma cl_product:
  assumes T{is a topology} S{is a topology} A⊆∪T B⊆∪S
  shows Closure(A×B,ProductTopology(T,S))=Closure(A,T)×Closure(B,S)
proof
  have A×B⊆∪T×∪S using assms(3,4) by auto
  then have sub:A×B⊆∪ProductTopology(T,S) using Top_1_4_T1(3) assms(1,2) by auto
  have top:ProductTopology(T,S){is a topology} using Top_1_4_T1(1) assms(1,2) by auto
  { fix x assume asx:x∈Closure(A×B,ProductTopology(T,S))
    then have reg:∀U∈ProductTopology(T,S). x∈U → U∩(A×B)≠0 using topology0.cl_inter_neigh
      sub top unfolding topology0_def by blast
    from asx have x∈∪ProductTopology(T,S) using topology0.Top_3_L11(1)
      top unfolding topology0_def
      using sub by blast
    then have xSigma:x∈∪T×∪S using Top_1_4_T1(3) assms(1,2) by auto
    then have ⟨fst(x),snd(x)⟩∈∪T×∪S using Pair_fst_snd_eq by auto
    then xT:fst(x)∈∪T and xS:snd(x)∈∪S by auto
    { fix U V assume as:U∈T fst(x)∈U
      have U⊆S using assms(2) unfolding IsATopology_def by auto
      with as have U×(∪S)∈ProductCollection(T,S) unfolding ProductCollection_def
      …
    }
by auto
then have P:U×(⋃S)∈ProductTopology(T,S) using Top_1_4_T1(2) assms(1,2)
base_sets_open by blast
with xS as(2) have ⟨fst(x),snd(x)⟩∈U×(⋃S) by auto
then have x∈U×(⋃S) using Pair_fst_snd_eq xS by auto
then have noEm:U∩A≠0 by auto

then have ∀U∈T. fst(x)∈U→U∩A≠0 by auto moreover
{
  fix U V assume as:U∈S snd(x)∈U
  have ∪T∈T using assms(1) unfolding IsATopology_def by auto
  with as have (∪T)×U∈ProductCollection(T,S) unfolding ProductCollection_def
  by auto
  then have P:(∪T)×U∈ProductTopology(T,S) using Top_1_4_T1(2) assms(1,2)
  base_sets_open by blast
  with xT as(2)
  have ⟨fst(x),snd(x)⟩∈(∪T)×U by auto
  then have x∈(∪T)×U using Pair_fst_snd_eq xSigma by auto
  then have noEm:(∪T)∩B≠0 by auto
}
then have ∀U∈S. snd(x)∈U→U∩B≠0 by auto
ultimately have fst(x)∈Closure(A,T) snd(x)∈Closure(B,S) using
  topology0.inter_neigh_cl assms(3,4) unfolding topology0_def
  using assms(1,2) xT xS by auto
from x assms(3,4) have x∈∪T×∪S using topology0.Top_3_L11(1) unfolding
  topology0_def
  using assms(1,2) by blast
then have xtot:x∈∪ProductTopology(T,S) using Top_1_4_T1(3) assms(1,2)
by auto
{
  fix P0 assume as:P0∈ProductTopology(T,S) x∈P0
  then obtain P0B where base:P0B∈ProductCollection(T,S) x∈P0B P0B⊆PO
  using point_open_base_neigh
  Top_1_4_T1(2) assms(1,2) base_sets_open by blast
  then obtain VT VS where V:VT∈T VS∈S x∈VT×VS P0B=VT×VS unfolding
  ProductCollection_def

1114
by auto
from V(3) have x:fst(x)∈VT snd(x)∈VS by auto
from V(1) have VT∩A≠0 by auto moreover
from V(2) have VS∩B≠0 by auto ultimately
have VT×VS∩A×B≠0 by auto

then have ∀P∈ProductTopology(T,S). x∈P→P∩A×B≠0 by auto
then have x∈Closure(A×B,ProductTopology(T,S)) using topology0.inter_neigh_cl
ultimately have VT×VS∩A×B≠0 by auto

by auto
qed

The product of closed sets, is closed in the product topology.

corollary closed_product:
assumes T{is a topology} S{is a topology} A{is closed in}TB{is closed in}
shows (A×B) {is closed in}ProductTopology(T,S)
proof-
\{ fix x y assume x∈∪ProductTopology(T,S)y∈∪ProductTopology(T,S)x≠y then have tot:x∈∪T×∪Sy∈∪T×∪Sx≠y using Top_1_4_T1(3) assms(1,2) by auto
then have Closure(A,T)=AClosure(B,S)=B using topology0.Top_3_L8 unfolding topology0_def by auto
then have Closure(A×B,ProductTopology(T,S))=A×B using cl_product assms(1,2) sub by auto
then show thesis using topology0.Top_3_L8 unfolding topology0_def using sub1 Top_1_4_T1(1) assms(1,2) by auto
qed

81.2 Separation properties in product space

The product of $T_0$ spaces is $T_0$.

theorem T0_product:
assumes T{is a topology} S{is a topology} T{is $T_0$} S{is $T_0$}
shows ProductTopology(T,S) {is $T_0$}
proof-
fix x y assume x∈∪ProductTopology(T,S)y∈∪ProductTopology(T,S)x≠y then have tot:x∈∪T×∪Sy∈∪T×∪Sx≠y using Top_1_4_T1(3) assms(1,2) by auto
then have (fst(x),snd(x))∈∪T×∪S(fst(y),snd(y))∈∪T×∪S and disj:fst(x)≠fst(y)∨snd(x)≠snd(y)
	using Pair_fst_snd_eq by auto
then have T:fst(x)∈∪Tfst(y)∈∪T and S:snd(y)∈∪Ssnd(x)∈∪S and p:fst(x)≠fst(y)∨snd(x)≠snd(y)
by auto 

\{ 
  assume \( \text{fst}(x) \neq \text{fst}(y) \) 
  with \( T \) \text{assms(3)} have \( \exists U \in T. (\text{fst}(x) \in U \land \text{fst}(y) \notin U) \lor (\text{fst}(y) \in U \land \text{fst}(x) \notin U) \) 
}\ 

unfolding \( \text{isT0_def} \) by auto 
then obtain \( U \) where \( U \in T (\text{fst}(x) \in U \land \text{fst}(y) /\in U) \lor (\text{fst}(y) \in U \land \text{fst}(x) /\in U) \) 
by auto 
with \( S \) have \( (\text{fst}(x),\text{snd}(x)) \in U \times (\bigcup S) \land (\text{fst}(y),\text{snd}(y)) \notin U \times (\bigcup S) \lor (\text{fst}(y),\text{snd}(y)) \in U \times (\bigcup S) \land (\text{fst}(x),\text{snd}(x)) /\in U \times (\bigcup S) \) 
using \( \text{Pair_fst_snd_eq tot(1,2)} \) by auto 
moreover have \( ((\bigcup S) \in S \text{ using } \text{assms(2)} \text{ unfolding } \text{IsATopology_def} \) 
by auto 
with \( \langle U \in T \rangle \text{ have } U \times (\bigcup S) \in \text{ProductTopology}(T,S) \text{ using } \text{prod_open_open_prod} \text{ assms(1,2)} \) by auto 
ultimately have \( \exists V \in \text{ProductTopology}(T,S). (x \in V \land y /\in V) \lor (y \in V \land x /\in V) \) proof qed 
} 
moreover 

\{ 
  assume \( \text{snd}(x) \neq \text{snd}(y) \) 
  with \( S \) \text{assms(4)} have \( \exists U \in S. (\text{snd}(x) \in U \land \text{snd}(y) \notin U) \lor (\text{snd}(y) \in U \land \text{snd}(x) \notin U) \) 
\} 
unfolding \( \text{isT0_def} \) by auto 
then obtain \( U \) where \( U \in S (\text{snd}(x) \in U \land \text{snd}(y) \notin U) \lor (\text{snd}(y) \in U \land \text{snd}(x) \notin U) \) 
by auto 
with \( T \) have \( (\text{fst}(x),\text{snd}(x)) \in (\bigcup T) \times U \land (\text{fst}(y),\text{snd}(y)) \notin (\bigcup T) \times U \lor (\text{fst}(y),\text{snd}(y)) \in (\bigcup T) \times U \land (\text{fst}(x),\text{snd}(x)) /\in (\bigcup T) \times U \) 
by auto 
then have \( (x \in (\bigcup T) \times U \land y /\in (\bigcup T) \times U) \lor (y \in (\bigcup T) \times U \land x /\in (\bigcup T) \times U) \) 
using \( \text{Pair_fst_snd_eq tot(1,2)} \) by auto 
moreover have \( (\bigcup T) \in T \text{ using } \text{assms(1)} \text{ unfolding } \text{IsATopology_def} \) 
by auto 
with \( \langle U \in S \rangle \text{ have } (\bigcup T) \times U \in \text{ProductTopology}(T,S) \text{ using } \text{prod_open_open_prod} \text{ assms(1,2)} \) by auto 
ultimately have \( \exists V \in \text{ProductTopology}(T,S). (x \in V \land y /\in V) \lor (y \in V \land x /\in V) \) proof qed 
} 
moreover 

note \( \text{disj} \) 
ultimately have \( \exists V \in \text{ProductTopology}(T,S). (x \in V \land y /\in V) \lor (y \in V \land x /\in V) \) 
by auto 

then show thesis unfolding \( \text{isT0_def} \) by auto 

qed 

The product of \( T_1 \) spaces is \( T_1 \). 

\textbf{theorem T1_product:} 
assumes \( T \{\text{is a topology}\} S \{\text{is a topology}\} T \{ T_1 \} S \{ T_1 \} \)
shows \( \text{ProductTopology}(T,S) \{ \text{is } T \_1 \} \)

proof -

\{ 
  fix \( x, y \) assume \( x \in \bigcup \text{ProductTopology}(T,S) y \in \bigcup \text{ProductTopology}(T,S) x \neq y \)
  then have \( \text{tot} : x \in \bigcup T \times \bigcup S y \in \bigcup T \times \bigcup S x \neq y \) using Top_1_4_T1(3) assms(1,2)
  by auto
  then have \( \langle \text{fst}(x), \text{snd}(x) \rangle \in \bigcup T \times \bigcup S \langle \text{fst}(y), \text{snd}(y) \rangle \in \bigcup T \times \bigcup S \) and disj: \( \text{fst}(x) \neq \text{fst}(y) \wedge \text{snd}(x) \neq \text{snd}(y) \)
  using Pair_fst_snd_eq by auto
  then have \( \text{fst}(x) \in \bigcup T \) \( \text{snd}(y) \in \bigcup S \) and \( \text{fst}(x) \neq \text{fst}(y) \) \( \text{snd}(x) \neq \text{snd}(y) \)
  using assms(3) unfolding isT1_def
  by auto
  moreover have \( (U : T) \in \bigcup T \times (U : S) \) \( (f : T) \neq f : U \times (U : S) \) using Pair_fst_snd_eq tot(1,2)
  by auto
  moreover have \( (U : S) \in S \) using assms(2) unfolding IsATopology_def
  by auto
  with \( U : T \) have \( U \times (U : S) \in \text{ProductTopology}(T,S) \) using prod_open_open_prod assms(1,2) by auto
  ultimately have \( \exists V : \text{ProductTopology}(T,S). (x \in V \wedge y \notin V) \) proof qed
  moreover have \( \exists V : \text{ProductTopology}(T,S). (x \in V \wedge y \notin V) \) proof qed
  \}

moreover have \( \exists V : \text{ProductTopology}(T,S). (x \in V \wedge y \notin V) \) proof qed
moreover note disj
ultimately have \( \exists V : \text{ProductTopology}(T,S). (x \in V \wedge y \notin V) \) by auto

then show thesis unfolding isT1_def by auto
qed
The product of $T_2$ spaces is $T_2$.

**Theorem T2_product:**

- **Assumes:** $T$ (is a topology) $S$ (is a topology) $T$ (is $T_2$) $S$ (is $T_2$)
- **Shows:** ProductTopology$(T,S)$ (is $T_2$)

**Proof:**

```ml
fix x y assume x∈∪ProductTopology(T,S)y∈∪ProductTopology(T,S)x≠y
then have tot:x∈∪T×∪Sy∈∪T×∪Sx≠y using Top_1_4_T1(3) assms(1,2)
by auto
then have (fst(x),snd(x))∈∪T×∪S(fst(y),snd(y))∈∪T×∪S and disj:fst(x)≠fst(y)∧snd(x)≠snd(y)
using Pair_fst_snd_eq by auto
then have T:fst(x)∈∪Tfst(y)∈∪T and S:snd(y)∈∪Ssnd(x)∈∪S and
p:fst(x)≠fst(y)∧snd(x)≠snd(y)
by auto
{ assume fst(x)≠fst(y)
with T assms(3) have (∃U∈T. ∃V∈T. (fst(x)∈U∧fst(y)∈V) ∧ U∩V=0)
    unfolding isT2_def by auto
then obtain U V where U∈T V∈T fst(x)∈U fst(y)∈V U∩V=0 by auto
with S have (fst(x),snd(x))∈∪(S) (fst(y),snd(y))∈∪(S) and
disjoint:((∪∪)∩(V×∪S)=0) by auto
then have x∈∪∪(S)y∈∪(S) using Pair_fst_snd_eq tot(1,2) by auto
moreover have (∪∪)∈S using assms(2) unfolding IsATopology_def
by auto
with ⟨U∈∪V∈∪⟩ have P:∪∪∈∪S∈ProductTopology(T,S) ∪∪∈∪S∈ProductTopology(T,S)
using prod_open_open_prod assms(1,2) by auto
note disjoint ultimately
have x∈∪∪(S) ∧ y∈∪∪(S) ∧ (∪∪)∩(∪∪)=0 by auto
with P(2) have ∃UU∈ProductTopology(T,S). (x∈∪∪(S) ∧ y∈UU ∧
(∪×(∪∪))∩UU=0)
using exI[where x=V×(∪∪) and P=λt. t∈ProductTopology(T,S) ∧
(x∈∪∪(S) ∧ y∈∪∪(S))∩(∪∪)=0] by auto
with P(1) have ∃VV∈ProductTopology(T,S). ∃UU∈ProductTopology(T,S).
(x∈VV ∧ y∈UU ∧ VV∩UU=0)
using exI[where x=V×(∪∪) and P=λt. t∈ProductTopology(T,S) ∧
(∃UU∈ProductTopology(T,S). (x∈t ∧ y∈UU ∧ (t)∩UU=0))] by auto
} moreover
{ assume snd(x)≠snd(y)
with S assms(4) have (∃U∈S. ∃V∈S. (snd(x)∈U∧snd(y)∈V) ∧ U∩V=0)
unfolding isT2_def by auto
then obtain U V where U∈S V∈S snd(x)∈U snd(y)∈V U∩V=0 by auto
with T have (fst(x),snd(x))∈∪T×∪U (fst(y),snd(y))∈∪T×∪V and
disjoint:(∪T×∪)∩(∪T×∪)=0 by auto
```

1118
then have \( x \in (\bigcup T) \times U \) \( y \in (\bigcup T) \times V \) using Pair_fst_snd_eq tot(1,2) by auto

moreover have \( (\bigcup T) \in T \) using assms(1) unfolding IsATopology_def by auto

with \( \langle U, V \rangle \in S \) have \( P: (\bigcup T) \times U \in \text{ProductTopology}(T, S) \) \( (\bigcup T) \times V \in \text{ProductTopology}(T, S) \)

using prod_open_open_prod assms(1,2) by auto

note disjoint ultimately

have \( x \in (\bigcup T) \times U \) \( y \in (\bigcup T) \times V \) \( ((\bigcup T) \times U) \cap ((\bigcup T) \times V)=0 \) by auto

with \( P(2) \) have \( \exists U \in \text{ProductTopology}(T, S) \) \( \exists V \in \text{ProductTopology}(T, S) \)

using exI [where \( x = (\bigcup T) \times U \) \( P = \lambda t. t \in \text{ProductTopology}(T, S) \) \( x \in (\bigcup T) \times U \) \( y \in (\bigcup T) \times V \) \( (\bigcup T) \times U \) \( \cap \) \( (\bigcup T) \times V \) \( = 0 \) ] by auto

moreover note disjoint ultimately

have \( \exists V \in \text{ProductTopology}(T, S) \) \( \exists U \in \text{ProductTopology}(T, S) \) \( x \in V \) \( y \in U \) \( V \cap U = 0 \) by auto

then show \( \text{thesis} \) unfolding isT2_def by auto
qed

The product of regular spaces is regular.

**Theorem regular_product:**

assumes \( T \) is a topology \( S \) is a topology \( T \) is regular \( S \) is regular

shows \( \text{ProductTopology}(T, S) \) is regular

**Proof:**

- \( \{ \) fix \( x \) \( U \) assume \( x \in \bigcup\text{ProductTopology}(T, S) \) \( U \in \text{ProductTopology}(T, S) \) \( x \in U \)
  - then obtain \( V \) \( W \) where \( VW: V \in TW \subseteq U \) and \( x \in V \times W \) using prod_top_point_neighb
    - asms(1,2) by blast
    - then have \( P: \text{fst}(x) \in V \) \( \text{snd}(x) \in W \) by auto
      - from \( P(1) \) \( \langle V \in T \rangle \) obtain \( V V \) where \( VW: V \in T \) \( \subseteq V \) \( V V \in T \)
        - using asms(1,3) topology0.regular_imp_exist_clos_neig unfolding topology0_def
          - by force
      - moreover from \( P(2) \) \( \langle W \in S \rangle \) obtain \( W W \) where \( WW: W \in S \) \( \subseteq W W \)
        - using asms(2,4) topology0.regular_imp_exist_clos_neig unfolding topology0_def
          - by force
      - ultimately have \( x \in V V \times W W \) using \( x \) by auto
      - moreover from \( \langle \text{Closure}(V V, T) \subseteq V \rangle \) \( \langle \text{Closure}(W W, S) \subseteq W \rangle \) have \( \text{Closure}(V V, T) \times \text{Closure}(W W, S) \subseteq V \times W \)
        - by auto
      - moreover from \( V V(3) \) \( W W(3) \) have \( VW \cup TW \subseteq U S \) by auto

1119
ultimately have $x \in V \times W$ \text{Closure}(V \times W) \subseteq V \times W$

using $c_{1 \text{ product}}$ assms(1,2)

by auto

moreover have $V \times W \in \text{ProductTopology}(T,S)$ using $\text{prod_open_open_prod}$

assms(1,2)

by auto

ultimately have $\exists Z \in \text{ProductTopology}(T,S). x \in Z$ \text{Closure}(Z,\text{ProductTopology}(T,S)) \subseteq V \times W$

by auto

with $\text{VV}(3) \text{ WW}(3)$ have $\exists Z \in \text{ProductTopology}(T,S). x \in Z$ \text{Closure}(Z,\text{ProductTopology}(T,S)) \subseteq U$

by auto

then have $\forall x \in \bigcup \text{ProductTopology}(T,S). x \in U \rightarrow \exists Z \in \text{ProductTopology}(T,S). x \in Z$ \text{Closure}(Z,\text{ProductTopology}(T,S)) \subseteq U$

by auto

then show thesis using $\text{topology0.exist_clos_neig_imp_regular}$ unfolding $\text{topology0_def}$

using assms(1,2) $\text{Top_1_4_T1}(1)$ by auto

qed

81.3 Connection properties in product space

First, we prove that the projection functions are open.

\textbf{lemma projection_open:}

assumes $T \{\text{is a topology}\} S \{\text{is a topology}\} B \in \text{ProductTopology}(T,S)$

shows $\{y \in \bigcup T. \exists x \in \bigcup S. \langle y,x \rangle \in B\} \in T$

proof-

{ fix z assume $z \in \{y \in \bigcup T. \exists x \in \bigcup S. \langle y,x \rangle \in B\}$

then obtain $x$ where $x:x \in \bigcup S$ and $z:z \in \bigcup T$ and $p:\langle z,x \rangle \in B$ by auto

then have $z \in \{y \in \bigcup T. \langle y,x \rangle \in B\} \subseteq \{y \in \bigcup T. \exists x \in \bigcup S. \langle y,x \rangle \in B\}$ by auto

moreover

from $x$ have $\forall U \in \bigcup \text{ProductTopology}(T,S). x \in U \rightarrow (\exists Z \in \text{ProductTopology}(T,S). x \in Z \wedge \text{Closure}(Z,\text{ProductTopology}(T,S)) \subseteq U)$ by auto

then show thesis using $\text{topology0.exist_clos_neig_imp_regular}$ unfolding $\text{topology0_def}$

using assms(1,2) $\text{Top_1_4_T1}(1)$ by auto

qed

\textbf{lemma projection_open2:}

assumes $T \{\text{is a topology}\} S \{\text{is a topology}\} B \in \text{ProductTopology}(T,S)$

shows $\{y \in \bigcup S. \exists x \in \bigcup T. \langle x,y \rangle \in B\} \in S$

proof-

{ fix z assume $z \in \{y \in \bigcup S. \exists x \in \bigcup T. \langle x,y \rangle \in B\}$

then obtain $x$ where $x:x \in \bigcup T$ and $z:z \in \bigcup S$ and $p:\langle x,z \rangle \in B$ by auto

then have $z \in \{y \in \bigcup S. \langle x,y \rangle \in B\} \subseteq \{y \in \bigcup S. \exists x \in \bigcup T. \langle x,y \rangle \in B\}$ by auto

moreover

1120
from x have \{y \in S. \langle x, y \rangle \in B\} \in S using prod_sec_open1 assms by auto
ultimately have \exists V \in S. z \in V \land V \subseteq \bigcup \{y \in S. \exists x \in T. \langle x, y \rangle \in B\} unfolding Bex_def by auto

ultimately have \exists V \in S. z \in V \land V \subseteq \bigcup \{y \in S. \exists x \in \bigcup T. \langle x, y \rangle \in B\} unfolding topology0.open_neigh_open using assms(2) by blast
qed

The product of connected spaces is connected.

theorem compact_product:
  assumes T{is a topology}S{is a topology}T{is connected}S{is connected}
  shows ProductTopology(T,S){is connected}
proof-
  { fix U assume U:U \in ProductTopology(T,S) U{is closed in}ProductTopology(T,S) then have P:U \in ProductTopology(T,S) \bigcup ProductTopology(T,S)-U \in ProductTopology(T,S) unfolding IsClosed_def by auto
    { fix s assume s:s \in \bigcup S with P(1) have p:\{x \in \bigcup T. \langle x, s \rangle \in U\} \in T using prod_sec_open2 assms(1,2) by auto
      from s P(2) have oop:\{y \in \bigcup T. \langle y, s \rangle \in (\bigcup ProductTopology(T,S)-U)\} \in T using prod_sec_open2 assms(1,2) by blast
      then have \bigcup T-(\bigcup T-{y \in \bigcup T. \langle y, s \rangle \in (\bigcup ProductTopology(T,S)-U)})={y \in \bigcup T. \langle y, s \rangle \in (\bigcup ProductTopology(T,S)-U)} by auto
    } moreover
    { fix t assume t:t \in T-{y \in \bigcup T. \langle y, s \rangle \in (\bigcup ProductTopology(T,S)-U)} then have tt:t \in \bigcup T t \notin \{y \in \bigcup T. \langle y, s \rangle \in (\bigcup ProductTopology(T,S)-U)\} by auto
      then have (t,s)\notin (\bigcup ProductTopology(T,S)-U) by auto
      then have (t,s)\in U \lor (t,s)\notin \bigcup ProductTopology(T,S) by auto
      then have (t,s)\in U \lor (t,s)\notin \bigcup T \times \bigcup S using Top_1_4_T1(3) assms(1,2) by auto
    } moreover
    { fix t assume t:\{x \in \bigcup T. \langle x, s \rangle \in U\} then have tt:t \in \bigcup T \langle t, s \rangle \in U by auto
      then have (t,s)\notin \bigcup ProductTopology(T,S)-U by auto
      then have tt\notin \{y \in \bigcup T. \langle y, s \rangle \in (\bigcup ProductTopology(T,S)-U)\} by auto
      with tt(1) have t \in \bigcup T-{y \in \bigcup T. \langle y, s \rangle \in (\bigcup ProductTopology(T,S)-U)} by auto
    } moreover
    { fix t assume t:\{x \in \bigcup T. \langle x, s \rangle \in U\} then have tt:t \in \bigcup T \langle t, s \rangle \in U by auto
      then have (t,s)\notin \bigcup ProductTopology(T,S)-U by auto
      then have tt\notin \{y \in \bigcup T. \langle y, s \rangle \in (\bigcup ProductTopology(T,S)-U)\} by auto
      with tt(1) have t \in \bigcup T-{y \in \bigcup T. \langle y, s \rangle \in (\bigcup ProductTopology(T,S)-U)} by auto
    } moreover
    { fix t assume t:\{x \in \bigcup T. \langle x, s \rangle \in U\} then have tt:t \in \bigcup T \langle t, s \rangle \in U by auto
      then have (t,s)\notin \bigcup ProductTopology(T,S)-U by auto
      then have tt\notin \{y \in \bigcup T. \langle y, s \rangle \in (\bigcup ProductTopology(T,S)-U)\} by auto
      with tt(1) have t \in \bigcup T-{y \in \bigcup T. \langle y, s \rangle \in (\bigcup ProductTopology(T,S)-U)} by auto
    } ultimately have \{x \in \bigcup T. \langle x, s \rangle \in U=\bigcup T-{y \in \bigcup T. \langle y, s \rangle \in (\bigcup ProductTopology(T,S)-U)}

1121
by blast
  with cl have \(\{x\in\bigcup T. \langle x,s\rangle\in U\}\) \{is closed in\}\(T\) by auto
  with p assms(3) have \(\{x\in\bigcup T. \langle x,s\rangle\in U\}\) \(\lor\) \(\{x\in\bigcup T. \langle x,s\rangle\in U\}\) = \(\bigcup T\)
  unfolding IsConnected_def
  by auto moreover
  { assume \(\{x\in\bigcup T. \langle x,s\rangle\in U\}\) = 0
    then have \(\forall x\in\bigcup T. \langle x,s\rangle\notin U\) by auto }
  moreover
  { assume AA: \(\{x\in\bigcup T. \langle x,s\rangle\in U\}\) = \(\bigcup T\)
    fix \(x\) assume \(x\in\bigcup T. \langle x,s\rangle\in U\) by auto
    then have \(\langle x,s\rangle\in U\) by auto }
  ultimately have \(\forall x\in\bigcup T. \langle x,s\rangle\notin U\) \(\lor\) \(\forall x\in\bigcup T. \langle x,s\rangle\in U\) by blast
  then have reg: \(\forall s\in\bigcup S. \ (\forall x\in\bigcup T. \langle x,s\rangle\notin U) \lor (\forall x\in\bigcup T. \langle x,s\rangle\in U)\) by auto
  { fix \(q\) assume qU: \(q\in\bigcup T\times\{\text{snd}(qq). qq\in U\}\)
    then obtain \(t\) \(u\) where \(t:\in\bigcup T\) \(u\in U\) \(q=\langle t,\text{snd}(u)\rangle\) by auto
    with U(1) have \(u\in\bigcup\text{ProductTopology}(T,S)\) by auto
    then have \(u\in\bigcup T\times\bigcup S\) using Top_1_4_T1(3) assms(1,2) by auto
    moreover
    then have uu: \(u=\langle \text{fst}(u),\text{snd}(u)\rangle\) using Pair_fst_snd_eq by auto
    ultimately
    have fu: \(\text{fst}(u)\in\bigcup T\) \(\text{snd}(u)\in\bigcup S\) by (safe,auto)
    with reg have \(\forall tt\in\bigcup T. \ (tt,\text{snd}(u))\notin U \lor (\forall tt\in\bigcup T. \ (tt,\text{snd}(u))\in U)\) by auto
    with \(t(1,3)\) have \(q\in U\) by auto }
  then have \(\bigcup T\times\{\text{snd}(qq). qq\in U\}\) \(\subseteq\) \(U\) by auto
  { fix \(t\) assume t: \(t:\in\bigcup T\)
    with P(1) have p: \(\{x\in\bigcup S. \langle t,x\rangle\in U\}\) \(\subseteq\) \(S\) using prod_sec_open1 assms(1,2) by auto
    from t P(2) have oop: \(\{x\in\bigcup S. \langle t,x\rangle\in (\bigcup\text{ProductTopology}(T,S)-U)\}\) \(\subseteq\) \(S\)
    using prod_sec_open1
    assms(1,2) by blast
    then have \(\bigcup S-\bigcup S-\{x\in\bigcup S. \langle t,x\rangle\in (\bigcup\text{ProductTopology}(T,S)-U)\}\) \(\in\) \(S\)
    using \(t,y\) \(\in\) \(\bigcup\text{ProductTopology}(T,S)-U\) by auto
    with oop have cl: \(\bigcup S-\{y\in\bigcup S. \langle t,y\rangle\in (\bigcup\text{ProductTopology}(T,S)-U)\}\) \(\in\) \(S\) unfolding IsClosed_def by auto
    {
fix $s$ assume $s \in \bigcup S - \{y \in \bigcup S. \ (t, y) \in \bigcup \text{ProductTopology}(T, S) - U\}$
then have $t : t \in \bigcup S$ s $\notin \{y \in \bigcup S. \ (t, y) \in \bigcup \text{ProductTopology}(T, S) - U\}$

by auto
then have $t \in \bigcup \text{ProductTopology}(T, S) - U$ by auto
then have $t \in U \vee t \notin \bigcup \text{ProductTopology}(T, S) - U$ by auto
then have $t \notin \bigcup T \times \bigcup S$ using $\text{Top}_1._4._T1(3)$ assms(1,2)

by auto

with $t t(1) t$ have $t \in U$ by auto

with $t t(1)$ have $s \in \bigcup S - \{y \in \bigcup S. \ (t, y) \in \bigcup \text{ProductTopology}(T, S) - U\}$

by auto

moreover

fix $s$ assume $s \in \bigcup S. \ (t, x) \in U$
then have $t \in U \vee t \notin \bigcup S. \ (t, x) \in U$ by auto
then have $\forall x \in \bigcup S. \ (t, x) \in U$ by auto

with $t t(1)$ have $s \in \bigcup S - \{y \in \bigcup S. \ (t, y) \in \bigcup \text{ProductTopology}(T, S) - U\}$

by auto

ultimately have $\forall s \in \bigcup S. \ (t, x) \in U \vee \forall s \in \bigcup S. \ (t, x) \in U$ by $\text{blast}$

unfolding $\text{IsConnected}_\text{def}$
by auto
moreover

assumption $x \in \bigcup S. \ (t, x) \in U$ by $\text{auto}$

then have $\forall x \in \bigcup S. \ (t, x) \notin U$ by $\text{auto}$

moreover

assume $AA: \forall x \in \bigcup S. \ (t, x) \in U = \bigcup S$

by auto

ultimately have $(\forall x \in \bigcup S. \ (t, x) \notin U) \lor (\forall x \in \bigcup S. \ (t, x) \in U)$ by $\text{blast}$
then have $\forall s \in \bigcup T. \ (\forall x \in \bigcup S. \ (s, x) \notin U) \lor (\forall x \in \bigcup S. \ (s, x) \in U)$ by $\text{auto}$

fix $q$ assume $q \in \{\text{fst}(qq). \ qq \in \bigcup U\} \times \bigcup S$
then obtain $qq \ s$ where $t : q = (\text{fst}(qq), s)$ $qq \in \bigcup U$ $s \in \bigcup S$ by $\text{auto}$
with $U(1)$ have $qq \in \bigcup \text{ProductTopology}(T, S) \times \bigcup S$ by $\text{auto}$
then have $qq \in \bigcup T \times \bigcup S$ using $\text{Top}_1._4._T1(3)$ assms(1,2) by $\text{auto}$

ultimately
have \(fq \in \bigcup T\) by (safe,auto)
from \(fq(1)\) reg have \((\forall tt \in U. (fst(qq),tt) \notin U)\lor (\forall tt \in U. (fst(qq),tt) \in U)\)
by auto moreover
with \(qq \in U\) \(qq \in U\) (force)
\begin{align*}
\text{moreover have } t(1,3) \text{ have } q \in U \text{ by auto}
\end{align*}

ultimately
have \((t,s) \in \bigcup U\) by blast
with \(U1\) have \((t,s) \in U\) by auto
moreover have \(t=fst((t,s))\) by auto
moreover note \(s \in U\)

ultimately
have \((U=0)\lor (U=\bigcup T\times \bigcup S)\) by auto

then show thesis unfolding IsConnected_def using Top_1_4_T1(3) assms(1,2)
by auto
qed

end

82 Topology 11

theory Topology_ZF_11 imports Topology_ZF_7 Finite_ZF_1

begin

This file deals with order topologies. The order topology is already defined in Topology_ZF_examples_1.thy.

82.1 Order topologies

We will assume most of the time that the ordered set has more than one point. It is natural to think that the topological properties can be translated to properties of the order; since every order rises one and only one topology in a set.
82.2 Separation properties

Order topologies have a lot of separation properties.

Every order topology is Hausdorff.

**Theorem order_top_T2:**

**Assumes:**

IsLinOrder($X$, $r$) \( \exists x \ y. \ x \neq y \land x \in X \land y \in X \)

**Shows:**

$(\text{OrdTopology } X \ r)\{\text{is T}_2\}$

**Proof:**

- **Fix** $x \ y$ assume $A1: x \in \bigcup (\text{OrdTopology } X \ r) y \in \bigcup (\text{OrdTopology } X \ r) x \neq y$
  - then have $AS: x \in y \in X x \neq y$ using $\text{union_orptopology}[OF \ \text{assms}(1) \ \text{assms}(2)]$

  by auto

- **Assume** $A2: \exists z \in X - \{x, y\}$. $(x, y) \in r \rightarrow (x, z) \in r \land (z, y) \in r \land (y, z) \in r$\( \land (z, x) \in r \)
  - from $AS(1,2)$ $\text{assms}(1)$ have $\langle x, y \rangle \in r \land \langle y, x \rangle \in r$ unfolding $\text{IsLinOrder_def}$

  by auto moreover

  - **Assume** $\langle x, y \rangle \in r$
    - with $A2$ obtain $z$ where $z: \langle x, z \rangle \in r \land (z, y) \in r \land x \neq x \neq y$ by auto
    - with $A2(1,2)$ have $x \in \text{LeftRayX}(X, r, z) y \in \text{RightRayX}(X, r, z)$ unfolding $\text{LeftRayX_def RightRayX_def}$

      by auto

    - have $\text{LeftRayX}(X, r, z) \cap \text{RightRayX}(X, r, z) = 0$ using $\text{inter_lray_rray}[OF \ \text{assms}(3) \ \text{assms}(3)]$

      unfolding $\text{IntervalX_def}$ using $\text{Order_ZF_2_L4}[OF \ \text{total_is_refl} _ z(3)]$ $\text{assms}(1)$ unfolding $\text{IsLinOrder_def}$

      by auto

    - have $\text{LeftRayX}(X, r, z) \in (\text{OrdTopology } X \ r) \text{RightRayX}(X, r, z) \in (\text{OrdTopology } X \ r)$

      using $\text{z}(3)$ $\text{base_sets_open}[OF \ \text{Ordtopology_is_a_topology}(2)[OF \ \text{assms}(1)]]$ by auto

      ultimately have $\exists U \in (\text{OrdTopology } X \ r) . \ \exists V \in (\text{OrdTopology } X \ r). \ x \in U \land y \in V \land U \cap V = 0$ by auto
ultimately have \( \exists U \in (\text{OrdTopology} X r). \exists V \in (\text{OrdTopology} X r). x \in U \wedge y \in V \wedge U \cap V = \emptyset \) by auto

moreover

{ assume A2: \( \forall z \in X - \{x, y\}. ((x, y) \in r \wedge ((x, z) \notin r \wedge (z, y) \notin r)) \) \( \vee (y, x) \in r \wedge ((y, z) \notin r \wedge (z, x) \notin r) \) from AS(1,2) assms(1) have disj:\( (x, y) \in r \vee (y, x) \in r \) unfolding IsLinOrder_def IsTotal_def by auto moreover

{ assume TT: \( (x, y) \in r \) with AS assms(1) have T: \( (y, x) \notin r \) unfolding IsLinOrder_def antisym_def by auto from TT AS(1-3) have x\( \in \text{LeftRayX}(X,r,x) \) y\( \in \text{RightRayX}(X,r,x) \) unfolding LeftRayX_def RightRayX_def by auto moreover

{ fix z assume z\( \in \text{LeftRayX}(X,r,x) \cap \text{RightRayX}(X,r,x) \) then have \( (z, y) \in r \vee (x, z) \notin \emptyset \) unfolding RightRayX_def LeftRayX_def by auto with A2 T have False by auto

} then have LeftRayX(X,r,y)\( \cap \text{RightRayX}(X,r,x) = \emptyset \) by auto moreover have LeftRayX(X,r,y)\( \in (\text{OrdTopology} X r) \) \( \text{RightRayX}(X,r,x) \in (\text{OrdTopology} X r) \) using base_sets_open[OF Ordtopology_is_a_topology(2)[OF assms(1)]] AS by auto ultimately have \( \exists U \in (\text{OrdTopology} X r). \exists V \in (\text{OrdTopology} X r). x \in U \wedge y \in V \wedge U \cap V = \emptyset \) by auto

moreover

{ assume TT: \( (y, x) \in r \) with AS assms(1) have T: \( (x, y) \notin r \) unfolding IsLinOrder_def antisym_def by auto from TT AS(1-3) have y\( \in \text{LeftRayX}(X,r,y) \) x\( \in \text{RightRayX}(X,r,y) \) unfolding LeftRayX_def RightRayX_def by auto moreover

{ fix z assume z\( \in \text{LeftRayX}(X,r,x) \cap \text{RightRayX}(X,r,y) \) then have \( (z, x) \in r \vee (y, z) \notin \emptyset \) unfolding RightRayX_def LeftRayX_def by auto with A2 T have False by auto

}
then have \( \text{LeftRay}_{X}(x, r, x) \cap \text{RightRay}_{X}(X, r, y) = 0 \) by auto
moreover have \( \text{LeftRay}_{X}(X, r, x) \in (\text{OrdTopology } X, r) \)
\( \text{RightRay}_{X}(X, r, y) \in (\text{OrdTopology } X, r) \)
using base_sets_open[OF Ordtopology_is_a_topology(2)[OF assms(1)]]
AS by auto
ultimately have \( \exists U \in (\text{OrdTopology } X, r). \exists V \in (\text{OrdTopology } X, r). x \in U \wedge y \in V \wedge U \cap V = 0 \) by auto
ultimately have \( \exists U \in (\text{OrdTopology } X, r). \exists V \in (\text{OrdTopology } X, r). x \in U \wedge y \in V \wedge U \cap V = 0 \) by auto
ultimately have \( \exists U \in (\text{OrdTopology } X, r). \exists V \in (\text{OrdTopology } X, r). x \in U \wedge y \in V \wedge U \cap V = 0 \) by auto
then show thesis unfolding isT2_def by auto
qed

Every order topology is \( T_4 \), but the proof needs lots of machinery. At the end of the file, we will prove that every order topology is normal; sooner or later.

82.3 Connectedness properties

Connectedness is related to two properties of orders: completeness and density

Some order-dense properties:

definition
IsDenseSub (_ {is dense in}_{with respect to}_) where
A {is dense in}X{with respect to}r ≡
\( \forall x \in X. \forall y \in X. (x, y) \in r \land x \neq y \rightarrow (\exists z \in A - \{x, y\}. (x, z) \in r \land (z, y) \in r) \)
definition
IsDenseUnp (_ {is not-properly dense in}_{with respect to}_) where
A {is not-properly dense in}X{with respect to}r ≡
\( \forall x \in X. \forall y \in X. (x, y) \in r \land x \neq y \rightarrow (\exists z \in A. (x, z) \in r \land (z, y) \in r) \)
definition
IsWeaklyDenseSub (_ {is weakly dense in}_{with respect to}_) where
A {is weakly dense in}X{with respect to}r ≡
\( \forall x \in X. \forall y \in X. (x, y) \in r \land x \neq y \rightarrow ((\exists z \in A - \{x, y\}. (x, z) \in r \land (z, y) \in r) \lor \text{Interval}(X, r, x, y) = 0) \)
definition
IsDense (_ {is dense with respect to}_) where
X {is dense with respect to}r ≡
\( \forall x \in X. \forall y \in X. (x, y) \in r \land x \neq y \rightarrow (\exists z \in X - \{x, y\}. (x, z) \in r \land (z, y) \in r) \)

lemma dense_sub:
shows \((X \text{ is dense with respect to } r) \iff (X \text{ is dense in } X \text{ with respect to } r)\)

unfolding IsDenseSub_def IsDense_def by auto

**Lemma not_prop_dense_sub:**
shows \((A \text{ is dense in } X \text{ with respect to } r) \Rightarrow (A \text{ is not-properly dense in } X \text{ with respect to } r)\)

unfolding IsDenseSub_def IsDenseUnp_def by auto

In densely ordered sets, intervals are infinite.

**Theorem dense_order_inf_intervals:**
assumes \(\text{IsLinOrder}(X, r) \\& \\text{Interval}(X, r, b, c) \neq \emptyset \in X \in X \text{ is dense with respect to } r\)
shows \(\neg \text{Finite}(\text{Interval}(X, r, b, c))\)

proof
assume fin: \(\text{Finite}(\text{Interval}(X, r, b, c))\)

have sub: \(\text{Interval}(X, r, b, c) \subseteq X\)
unfolding IntervalX_def
by auto

have p: \(\text{Minimum}(r, \text{Interval}(X, r, b, c)) \in \text{Interval}(X, r, b, c)\)
using Finite_ZF_1_T2(2)[OF assms(1) Finite_Fin[OF fin sub] assms(2)]
by auto
then have \(\langle b, \text{Minimum}(r, \text{Interval}(X, r, b, c)) \rangle \in r\)
unfolding IntervalX_def
by auto

with assms(3,5) sub p
obtain z1 where z1: \(z1 \in X \neq b \neq c\)
\(\langle b, z1 \rangle \in r \langle z1, \text{Minimum}(r, \text{Interval}(X, r, b, c)) \rangle \in r\)
unfolding IsDense_def
by blast

from p have B: \(\text{Minimum}(r, \text{Interval}(X, r, b, c)), c \in r\)
unfolding IntervalX_def
by auto
moreover
have trans(r) using assms(1) unfolding IsLinOrder deflect by auto
moreover
note z1(5) ultimately have z1a: \(\langle z1, c \rangle \in r\)
unfolding trans_def by fast
\{ assume z1=c
with B have \(\text{Minimum}(r, \text{Interval}(X, r, b, c)), z1 \in r\)
by auto
with z1(5) have z1=Minim(um)(r, IntervalX(x, r b, c)) using assms(1)
unfolding IsLinOrder_def antisym_def by auto
then have False using z1(3) by auto
\}
then have z1\neq c by auto
with z1(1,2,4) z1a have z1\in IntervalX(x, r, b, c) unfolding IntervalX_def
using Order_ZF_2_L1 by auto
then have \(\text{Minimum}(r, \text{Interval}(X, r, b, c)), z1 \in r\)
using Finite_ZF_1_T2(4)[OF assms(1) Finite_Fin[OF fin sub] assms(2)]
by auto
with z1(5) have z1=Minimum(r, IntervalX(X, r, b, c)) using assms(1)
unfolding IsLinOrder_def antisym_def by auto
with z1(3) show False by auto
qed

Left rays are infinite.
theorem dense_order_inf_lrays:
assumes IsLinOrder(X,r) LeftRayX(X,r,c)≠0c∈X X{is dense with respect to}r
shows ¬Finite(LeftRayX(X,r,c))
proof-
from assms(2) obtain b where b∈X{b,c}∈rb≠c unfolding LeftRayX_def
by auto
with assms(3) obtain z where z∈X-{b,c}{b,z}∈r(z,c)∈r using assms(4)
unfolding IsDense_def by auto
then have IntervalX(X,r,b,c)≠0 unfolding IntervalX_def using Order_ZF_2_L1
by auto
then have nFIN:¬Finite(IntervalX(X,r,b,c)) using dense_order_inf_intervals[OF assms(1) _ assms(3,4)]
{b∈X} by auto
{ fix d assume d∈IntervalX(X,r,b,c)
then have (b,d)∈r(d,c)∈rd∈Xd≠bd≠c unfolding IntervalX_def using Order_ZF_2_L1
by auto
then have d∈LeftRayX(X,r,c) unfolding LeftRayX_def by auto
} then have IntervalX(X,r,b,c)⊆LeftRayX(X,r,c) by auto
with nFIN show thesis using subset_Finite by auto
qed

Right rays are infinite.

theorem dense_order_inf_rrays:
assumes IsLinOrder(X,r) RightRayX(X,r,b)≠0b∈X X{is dense with respect to}r
shows ¬Finite(RightRayX(X,r,b))
proof-
from assms(2) obtain c where c∈X{b,c}∈rb≠c unfolding RightRayX_def
by auto
with assms(3) obtain z where z∈X-{b,c}{b,z}∈r(z,c)∈r using assms(4)
unfolding IsDense_def by auto
then have IntervalX(X,r,b,c)≠0 unfolding IntervalX_def using Order_ZF_2_L1
by auto
then have nFIN:¬Finite(IntervalX(X,r,b,c)) using dense_order_inf_intervals[OF assms(1) _ assms(3,4)]
{c∈X} by auto
{ fix d assume d∈IntervalX(X,r,b,c)
then have (b,d)∈r(d,c)∈rd∈Xd≠bd≠c unfolding IntervalX_def using Order_ZF_2_L1
by auto
then have d∈RightRayX(X,r,b) unfolding RightRayX_def by auto
} then have IntervalX(X,r,b,c)⊆RightRayX(X,r,b) by auto
with nFIN show thesis using subset_Finite by auto
qed

The whole space in a densely ordered set is infinite.
corollary dense_order_infinite:
assumes IsLinOrder(X,r) X{is dense with respect to}r
∃x y. x≠y ∧ x∈X ∧ y∈X
shows ¬(X≺nat)
proof-
from assms(3) obtain b c where B:b∈X c∈X b≠c by auto
{ assume (b,c)∉r
with assms(1) have (c,b)∈r unfolding IsLinOrder_def IsTotal_def using b∈X c∈X by auto
with assms(2) B obtain z where z∈X¬(b,c) (c,z)∈r (z,b)∈r unfolding IsDense_def by auto
then have IntervalX(X,r,c,b)≠0 unfolding IntervalX_def using Order_ZF_2_L1 by auto
then have ¬(Finite(IntervalX(X,r,c,b))) using dense_order_inf_intervals[OF assms(1) _ (c,b)∈r assms(2)]
by auto moreover
have IntervalX(X,r,c,b)⊆X unfolding IntervalX_def by auto
ultimately have ¬(Finite(X)) using subset_Finite by auto
then have ¬(X≺nat) using lesspoll_nat_is_Finite by auto
}
moreover
{ assume (b,c)∈r
with assms(2) B obtain z where z∈X¬(b,c) (b,z)∈r (z,c)∈r unfolding IsDense_def by auto
then have IntervalX(X,r,b,c)≠0 unfolding IntervalX_def using Order_ZF_2_L1 by auto
then have ¬(Finite(IntervalX(X,r,b,c))) using dense_order_inf_intervals[OF assms(1) _ (b,c)∈r assms(2)]
by auto moreover
have IntervalX(X,r,b,c)⊆X unfolding IntervalX_def by auto
ultimately have ¬(Finite(X)) using subset_Finite by auto
then have ¬(X≺nat) using lesspoll_nat_is_Finite by auto
}
ultimately show thesis by auto
qed

If an order topology is connected, then the order is complete. It is equivalent
to assume that r ⊆ X × X or prove that r ∩ X × X is complete.

theorem conn_imp_complete:
assumes IsLinOrder(X,r) ∃x y. x≠y ∧ x∈X ∧ y∈X
(OrdTopology X r){is connected}
shows r{is complete}
proof-
{ assume ¬(r{is complete})
then obtain A where A:A≠∅IsBoundedAbove(A,r)¬(HasAminimum(r, ∩b∈A. r {b})) unfolding

1130
IsComplete_def by auto
from A(3) have r1:\forall m\in\bigcap b\in A. r \{b\}. \exists x\in\bigcap b\in A. r \{b\}. \langle m,x\rangle \not\in r unfolding HasAminimum_def
by force
from A(1,2) obtain b where r2:\forall x\in A. \langle x,b \rangle \in r unfolding IsBoundedAbove_def
by auto
with assms(3) A(1) have A \subseteq Xb \in X by auto
with assms(3) have r3:\forall x\in A. r \{c\} \subseteq X using image_iff by auto
from r2 have \forall x\in A. b \in r(x) using image_iff by auto
then have noE:b\in\bigcap b\in A. r \{b\} using A(1) by auto

fix x assume x\in(\bigcap b\in A. r \{b\})
then have \forall c\in A. x \in r\{c\} by auto
with r3 have x\in X by auto

fix x assume x: x\in(\bigcap b\in A. r \{b\})
with r1 have \exists z\in\bigcap b\in A. r \{b\}. \langle x,z \rangle \not\in r by auto
then obtain z where z:z\in(\bigcap b\in A. r \{b\}). \langle x,z \rangle \not\in r by auto
from x z(1) sub have x\in Xz \in X by auto
with z(2) have \langle z,x \rangle \in r using assms(1) unfolding IsLinOrder_def IsTotal_def
by auto
then have xx: x\in RightRayX(X,r,z) unfolding RightRayX_def using \langle x\in X, \langle x,z \rangle \not\in r \rangle
assms(1) unfolding IsLinOrder_def using total_is_refl unfolding refl_def by auto

fix m assume m\in RightRayX(X,r,z)
then have m: m \in X-{z}\langle z,m \rangle \in r unfolding RightRayX_def by auto

fix c assume c\in A
with z(1) have \langle c,z \rangle \in r using image_iff by auto
with m(2) have \langle c,m \rangle \in r using assms(1) unfolding IsLinOrder_def
trans_def by fast
then have m\in r\{c\} using image_iff by auto

with A(1) have m\in(\bigcap b\in A. r \{b\}) by auto

then have sub1: RightRayX(X,r,z) \subseteq(\bigcap b\in A. r \{b\}) by auto
have RightRayX(X,r,z) \subseteq(OrdTopology X r) using base_sets_open[OF Ordtopology_is_a_topology(2)[OF assms(1)]] \langle z\in X \rangle
by auto
with sub1 xx have \exists U\in(OrdTopology X r). x\in U \wedge U\subseteq(\bigcap b\in A. r \{b\})
by auto

then have (\bigcap b\in A. r \{b\})\in(OrdTopology X r) using topology0.open_neigh_open[OF topology0_ordtopology[OF assms(1)]]
by auto moreover
\{ 
  fix x assume \( x \in X - (\bigcap b \in A. r \{b\}) \)
  then have \( x \notin (\bigcap b \in A. r \{b\}) \) by auto
  with \( A(1) \) obtain \( b \) where \( x \notin r(b) \) by auto
  then have \( \langle b, x \rangle \notin r \) using image_iff by auto
  with \( \langle A \subseteq X \rangle \) \( b \in A \langle x \in X \rangle \) have \( \langle x, b \rangle \in r \) using \( \text{assms}(1) \) unfolding \( \text{IsLinOrder_def} \)
  \( \text{IsTotal_def} \) by auto
  then have \( xx : x \in \text{LeftRayX}(X, r, b) \) unfolding \( \text{LeftRayX_def} \) by auto
  \}
  \{ 
    fix \( y \) assume \( y \in \text{LeftRayX}(X, r, b) \cap (\bigcap b \in A. r \{b\}) \)
    then have \( y \in X - \{b\} \) by auto
    then have \( y \in X - \{b\} \langle y, b \rangle \in r \forall c \in A. y \in r(c) \) unfolding \( \text{IsLinOrder_def} \)
    using \( \text{assms}(1) \) unfolding \( \text{antisym_def} \) by auto
    then have \( y = b \) using \( \text{assms}(1) \) unfolding \( \text{antisym_def} \) by auto
    then have \( False \) using \( \langle y \in X - \{b\} \rangle \) by auto
  \}
  then have \( \text{sub1} : \text{LeftRayX}(X, r, b) \subseteq X - (\bigcap b \in A. r \{b\}) \) unfolding \( \text{IsClosed_def} \)
  \( \text{IsConnected_def} \) by auto
  moreover note \( \text{assms}(4) \) ultimately
  have \( (\bigcap b \in A. r \{b\}) = X \) using \( \text{union_ordtopology} \) \( \text{IsClosed_def} \) unfolding \( \text{IsConnected_def} \)
  by auto
  then have \( e1 : (\bigcap b \in A. r \{b\}) = X \) using \( \text{noE} \) by auto
  then have \( \forall x \in X. \forall b \in A. x \in r(b) \) by auto
  then have \( r4 : \forall x \in X. \forall b \in A. \langle b, x \rangle \in r \) using image_iff by auto
  \{ 
    fix \( a1, a2 \) assume \( aa : a1 \notin A2 \in A \neq a2 \)
    with \( \langle A \subseteq X \rangle \) \( a1 \notin X \) \( a2 \notin X \) by auto
    then have \( a1 = a2 \) unfolding \( \text{IsLinOrder_def} \) \( \text{antisym_def} \) by auto
  \}
with aA(3) have False by auto
}
moreover
from A(1) obtain t where t∈A by auto
ultimately have A={t} by auto
with r4 have ∃x∈X. (t,x)∈r(t, x) using A Sub X by auto
then have HasAminimum(r,X) unfolding HasAminimum_def by auto
with e1 have HasAminimum(r, ∩ b∈A. r {b}) by auto
with A(3) have False by auto
}
then show thesis by auto
qed

If an order topology is connected, then the order is dense.

theorem conn_imp_dense:
  assumes IsLinOrder(X,r) ∃ x y. x ≠ y ∧ x∈X ∧ y∈X
  (OrdTopology X r) {is connected}
shows X {is dense with respect to} r
proof-
  { assume ¬(X {is dense with respect to} r)
    then have ∃ x1∈X. ∃ x2∈X. (x1,x2)∈r ∧ x1≠x2 ∧ (∀ z∈X-{x1,x2}. (x1,z)∉ r ∨ (z,x2)∉ r)
      unfolding IsDense_def by auto
    then obtain x1 x2 where x:x1∈X x2∈X ⟨x1,x2⟩∈ r x1≠x2 (∀ z∈X-{x1,x2}. (x1,z)∉ r ∨ (z,x2)∉ r)
      by auto
    from x(1,2) have P: LeftRayX(X,r,x2) ∈ (OrdTopology X r) RightRayX(X,r,x1) ∈ (OrdTopology X r)
      using base_sets_open[OF Ordtopology_is_a_topology(2)[OF assms(1)]
    by auto
    { fix x assume x∈X-LeftRayX(X,r,x2)
      then have x∈X x∉LeftRayX(X,r,x2) by auto
      then have (x,x2)∉ r ∨ x=x2 unfolding LeftRayX_def by auto
      then have ⟨x2,x⟩∈r ∨ x=x2 using assms(1) ⟨x∈X⟩ ⟨x2∈X⟩ unfolding IsLinOrder_def IsTotal_def by auto
      then have s:⟨x2,x⟩∈r using assms(1) unfolding IsLinOrder_def using total_is_refl ⟨x2∈X⟩
        unfolding refl_def by auto
      with x(3) have (x1,x)∈r using assms(1) unfolding IsLinOrder_def trans_def by fast
      then have x=x1 ∨ x∈RightRayX(X,r,x1) unfolding RightRayX_def using ⟨x∈X⟩ by auto
      with s have ⟨x2,x1⟩∈r ∨ x∈RightRayX(X,r,x1) by auto
      with x(3) have x1=x2 ∨ x∈RightRayX(X,r,x1) using assms(1) unfolding IsLinOrder_def
        antisym_def by auto
      with x(4) have x∈RightRayX(X,r,x1) by auto
    }
    then have X-LeftRayX(X,r,x2) ⊆ RightRayX(X,r,x1) by auto
moreover
\{ 
fix x assume x∈RightRayX(X,r,x1) 
then have xr:x∈X-{x1}\{x1,x\}∈r unfolding RightRayX_def by auto 
\{ 
assume x∈LeftRayX(X,r,x2) 
then have xl:x∈X-{x2}\{x,x2\}∈r unfolding LeftRayX_def by auto 
from xl xr x(5) have False by auto 
\} 
ultimately have RightRayX(X,r,x1)=X-LeftRayX(X,r,x2) by auto 
then have LeftRayX(X,r,x2){is closed in}(OrdTopology X r) using P(2) 
union_ordtopology[OF assms(1,2)] unfolding IsClosed_def LeftRayX_def by auto 
with P(1) have LeftRayX(X,r,x2)=0⋃LeftRayX(X,r,x2)=X using union_ordtopology[OF assms(1,2)] unfolding IsConnected_def by auto 
then have x∈X-LeftRayX(X,r,x2) by auto 
\} 
ultimately have RightRayX(X,r,x1)=X-LeftRayX(X,r,x2) by auto 
then have LeftRayX(X,r,x2) is closed in (OrdTopology X r) using P(2) 
union_ordtopology[OF assms(1,2)] unfolding IsClosed_def LeftRayX_def by auto 
with P(1) have LeftRayX(X,r,x2)=0|LeftRayX(X,r,x2)=X using union_ordtopology[OF assms(1,2)] unfolding IsConnected_def by auto 
then have x∈X-LeftRayX(X,r,x2) by auto 
then have False unfolding LeftRayX_def by auto 
\} 
then show thesis by auto 
qed 

Actually a connected order topology is one that comes from a dense and complete order.

First a lemma. In a complete ordered set, every non-empty set bounded from below has a maximum lower bound.

lemma complete_order_bounded_below: 
assumes r {is complete} IsBoundedBelow(A,r) A≠0 r⊆X×X 
shows HasAminimum(r,⋂c∈A. r-{c}) 
proof-
let M=⋂c∈A. r-{c} 
from assms(3) obtain t where A:t∈A by auto 
{ 
fix m assume m∈M 
with A have m∈r-{t} by auto 
then have ⟨m,t⟩∈r by auto 
} 
then have (∀x∈⋂c∈A. r - {c}. ⟨x, t⟩ ∈ r) by auto 
then have IsBoundedAbove(M,r) unfolding IsBoundedAbove_def by auto 
moreover 
from assms(2,3) obtain l where ∄x∈A. (l, x) ∈ r unfolding IsBoundedBelow_def by auto 
then have l∈M by auto 
then have l∉0 by auto moreover note assms(1) 
ultimately have HasAminimum(r,⋂c∈M. r - {c}) unfolding IsComplete_def by auto 
\}
then obtain \( \mathbf{r} \) where \( \mathbf{r}:\mathbf{r}\in(\bigcap_{c\in M. r \{c\}}) \forall s\in(\bigcap_{c\in M. r \{c\}}). (\mathbf{r},s)\in r \)

unfolding HasAminimum_def
by auto

\{
  fix \mathbf{a} \text{ assume } A: \mathbf{a}\in A
  \{
    fix \mathbf{c} \text{ assume } M: \mathbf{c}\in M
    \text{ with } A \text{ have } (\mathbf{c},\mathbf{a})\in r \text{ by auto}
    \text{ then have } \mathbf{a}\in r\{\mathbf{c}\} \text{ by auto}
  \}
  \text{ then have } \mathbf{a}\in(\bigcap_{c\in M. r \{c\}}) \text{ using } \mathbf{r}(1) \text{ by auto}
\}

then have \( A\subseteq(\bigcap_{c\in M. r \{c\}}) \text{ by auto} \)
with \( \mathbf{r}(2) \text{ have } \forall s\in A. (\mathbf{r},s)\in r \text{ by auto} \)
then have \( \mathbf{r}\in M \) using assms(3) by auto
moreover
\{
  fix \mathbf{m} \text{ assume } \mathbf{m}\in M
  \text{ then have } \mathbf{r}\in r\{\mathbf{m}\} \text{ using } \mathbf{r}(1) \text{ by auto}
  \text{ then have } (\mathbf{m},\mathbf{r})\in r \text{ by auto}
\}

then have \( \forall \mathbf{m}\in M. (\mathbf{m},\mathbf{r})\in r \text{ by auto} \)
ultimately show thesis unfolding HasAmaximum_def by auto
qed

theorem comp_dense_imp_conn:
  assumes IsLinOrder(X,r) \( \exists \mathbf{x} \mathbf{y}. \mathbf{x}\neq \mathbf{y} \wedge \mathbf{x}\in X \wedge \mathbf{y}\in X \) r\(\subseteq X\times X \)
  \( X \) \( \text{ is dense with respect to} \) r r{complete}
shows (OrdTopology X r){connected}
proof-
\{
  assume \( \neg ((\text{OrdTopology } X \ r)\{\text{is connected}\}) \)
  then obtain \( U \) where \( U:U\neq\emptyset\neq X\in (\text{OrdTopology } X \ r)U\{\text{is closed in}\}(\text{OrdTopology} X \ r) \)
  unfolding IsConnected_def using union_ortdtopology[OF assms(1,2)]
by auto
  from U(4) have A: X\(\subseteq U\in (\text{OrdTopology } X \ r)\)\(\subseteq\text{ unifying } \) IsClosed_def
using union_ortdtopology[OF assms(1,2)] by auto
  from U(1) obtain \( u \) where u\(\in U\) by auto
  from A(2) U(1,2) have X\(\subseteq U\neq\emptyset\) by auto
  then obtain \( \mathbf{v} \) where v\(\in X\subseteq U\) by auto
  with \( \langle u\in U \rangle \langle U\subseteq X \rangle \) have \( \langle u,\mathbf{v}\rangle\in r\langle \mathbf{v},u\rangle\in r \) using assms(1) unfolding IsLinOrder_def
  IsTotal_def
by auto
\{
  assume \( \langle u,\mathbf{v}\rangle\in r \)
  \text{ have LeftRayX}(X,r,\mathbf{v})\(\in (\text{OrdTopology } X \ r) \) using base_sets_open[OF Ordtopology_is_a_topology(2)[OF assms(1)]]
  \langle \mathbf{v}\in X\subseteq U \rangle \text{ by auto}
then have \(U \cap \text{LeftRay}_X(X,r,v) \in (\text{OrdTopology}_X r)\) using \(\text{Ordtopology}_X\text{is_a_topology}(1)\)

\[
\begin{array}{l}
\text{OF assms(1)} \quad \text{unfolding IsATopology_def by auto} \\
\{ \\
\text{fix } b \quad \text{assume } b \in (U) \cap \text{LeftRay}_X(X,r,v) \\
\text{then have } (b,v) \in r \quad \text{unfolding LeftRayX_def by auto} \\
\} \\
\text{then have bound:}\text{IsBoundedAbove}(U \cap \text{LeftRay}_X(X,r,v),r) \quad \text{unfolding IsBoundedAbove_def by auto} \\
\end{array}
\]

moreover with \(\langle u,v \rangle \in r\) and \(u \in U\) and \(U \subseteq X\) and \(v \in X-U\) have 
\(nE: U \cap \text{LeftRay}_X(X,r,v) \neq 0\) using assms(5) unfolding IsComplete_def by auto

let \(\text{min}=\text{Supremum}(r,U \cap \text{LeftRay}_X(X,r,v))\)

\[
\begin{array}{l}
\{ \\
\text{fix } c \quad \text{assume } c \in U \cap \text{LeftRay}_X(X,r,v) \\
\text{then have } (c,v) \in r \quad \text{unfolding LeftRayX_def by auto} \\
\} \\
\text{then have a1: } \langle \text{min},v \rangle \in r \quad \text{using Order_ZF_5_L3[OF nE Hmin assms(1)] unfolding IsComplete_def by auto} \\
\end{array}
\]

\[
\begin{array}{l}
\{ \\
\text{assume ass: } \text{min} \in U \\
\text{then obtain } V \quad \text{where } V: \text{min} \in V \subseteq U \\
V \in \{\text{Interval}_X(X,r,b,c). \ (b,c) \in X \times X\} \cup \{\text{LeftRay}_X(X,r,b). \ b \in X\} \cup \{\text{RightRay}_X(X,r,b). \ b \in X\} \text{ using point_open_base_neigh} \\
\text{OF Ordtopology_is_a_topology}(2)[OF assms(1)] \quad V \in (\text{OrdTopology}_X r) \text{ ass] by blast} \\
\{ \\
\text{assume } V \in \{\text{RightRay}_X(X,r,b). \ b \in X\} \\
\text{then obtain } b \quad \text{where } b:bX \ V=\text{RightRay}_X(X,r,b) \text{ by auto} \\
\text{note a1 moreover} \\
\text{from } V(1) \text{ b(2) have a2: } \langle b,\text{min} \rangle \in r \text{min} \neq b \quad \text{unfolding RightRayX_def by auto} \\
\text{ultimately have } \langle b,v \rangle \in r \quad \text{using assms(1) unfolding IsLinOrder_def trans_def by blast moreover} \\
\{ \\
\text{assume } b=v \\
\text{with a1 a2(1) have } \text{b=min using assms(1) unfolding IsLinOrder_def antisym_def by auto} \\
\text{with a2(2) have False by auto} \\
\} \\
\text{ultimately have False using V(2) b(2) unfolding RightRayX_def using } \langle v \in X-U \rangle \text{ by auto} \\
\} \\
\text{moreover} \\
\{ \\
\text{assume } V \in \{\text{LeftRay}_X(X,r,b). \ b \in X\} \\
\}
\end{array}
\]
then obtain \( b \) where \( b:V = \text{LeftRayX}(X,r,b) \) \( b \in X \) by auto

\{
\begin{align*}
\text{assume } & \langle v,b \rangle \in r \\
\text{then have } & b = v \text{ using } \langle v \in X - U \rangle \text{ by auto}
\end{align*}
\}

then have \( b:v \in \text{LeftRayX}(X,r,b) \) unfolding \( \text{LeftRayX} \) def using \( \langle v \in X - U \rangle \) by auto

then have \( \text{bv}:\langle b,v \rangle \in r \) using assms(1) unfolding \( \text{IsLinOrder} \) def

\begin{align*}
\text{IsTotal def using } & b(1) \langle v \in X - U \rangle \text{ by auto}
\end{align*}

then have \( \text{bv}:\langle b,v \rangle \in r \) using assms(1) unfolding \( \text{IsLinOrder} \) def

\begin{align*}
\text{antisym def by } & \text{fast}
\end{align*}

moreover

\{
\begin{align*}
\text{assume } & z = v \\
\text{with } & b:v \in \text{LeftRayX}(X,r,b) \text{ unfolding } \text{IsLinOrder} \text{ def}
\end{align*}
\}

ultimately have \( z: \text{LeftRayX}(X,r,v) \) unfolding \( \text{LeftRayX} \) def using \( \langle v \in X \cup U \rangle \) by auto

with \( b:v \in \text{LeftRayX}(X,r,v) \) unfolding \( \text{IsLinOrder} \) def by auto

then have False using \( \langle v \in X \cup U \rangle \) by auto

moreover

\{
\begin{align*}
\text{assume } & \langle c,v \rangle \in r \\
\text{then obtain } & b: V = \text{IntervalX}(X,r,b,c) \\
\text{from } & b:V \in \text{IntervalX}(X,r,b,c) \text{ unfolding } \text{IntervalX} \text{ def by auto}
\end{align*}
\}

\begin{align*}
\text{assume } & A: \langle c,v \rangle \in r \\
\text{from } & m: \langle c,v \rangle \in r \text{ unfolding } \text{IsLinOrder} \text{ def using } \text{assms(4)}
\end{align*}

ultimately have \( z: \text{LeftRayX}(X,r,v) \) unfolding \( \text{LeftRayX} \) def using \( \langle v \in X \cup U \rangle \) by auto

with \( b:v \in \text{LeftRayX}(X,r,v) \) unfolding \( \text{IsLinOrder} \) def by auto

then have \( \langle v \in X \cup U \rangle \) by auto

then have \( \langle z,v \rangle \in r \) using assms(1) unfolding \( \text{IsLinOrder} \) def

\begin{align*}
\text{antisym def by auto}
\end{align*}

moreover

\{
\begin{align*}
\text{assume } & V \in \{ \text{IntervalX}(X,r,b,c) \} \langle b,c \rangle \in X \times X \\
\text{then obtain } & b,c \text{ where } b: V = \text{IntervalX}(X,r,b,c) \text{ unfolding } \text{IntervalX} \text{ def by auto}
\end{align*}
\}

\begin{align*}
\text{assume } & A: \langle c,v \rangle \in r \\
\text{from } & m: \langle c,v \rangle \in r \text{ unfolding } \text{IsLinOrder} \text{ def using } \text{assms(4)}
\end{align*}

ultimately have \( z: \text{LeftRayX}(X,r,v) \) unfolding \( \text{LeftRayX} \) def using \( \langle v \in X \cup U \rangle \) by auto

then have \( \langle z,v \rangle \in r \) using assms(1) unfolding \( \text{IsLinOrder} \) def

\begin{align*}
\text{antisym def by auto}
\end{align*}
trans_def
  by fast
  with $z(1)$ have $z \in \text{Interval}(X,r,b,c) \lor z = b$ using $z(3)$ unfolding IntervalX_def
  Interval_def by auto
  then have $z \in \text{Interval}(X,r,b,c)$ using $m(2)$ $z(2,3)$ using assms(1)
unfolding IsLinOrder_def
  antisym_def by auto
  with $b(1)$ $V(2)$ have $z \in U$ by auto moreover from $A$ $z(1)$ have $(z,v) \in r$ using assms(1) unfolding IsLinOrder_def
trans_def by fast
  moreover have $z \neq v$ using $A$ $z(1,3)$ assms(1) unfolding IsLinOrder_def
antisym_def by auto
  ultimately have $z \in U \cap \text{LeftRay}(X,r,v)$ unfolding LeftRayX_def
using $z(3)$ by auto
  then have $\min \in r \{ z \}$ using Order_ZF_4_L4(1)[OF _ Hmin] assms(1)
unfolding Supremum_def IsLinOrder_def
  by auto
  then have $\langle z, \min \rangle \in r$ by auto
  with $z(2,3)$ have False using assms(1) unfolding IsLinOrder_def
antisym_def by auto
  }
then have $vc : \langle v,c \rangle \in r v \neq c$ using assms(1) unfolding IsLinOrder_def
IsTotal_def using $\langle v \in X-U \rangle$
  $b(3)$ by auto
  {
assume $\min = v$
  with $V(2,1)$ $\langle v \in X-U \rangle$ have False by auto
  }
then have $\min \neq v$ by auto
  with $a1$ obtain $z$ where $z : (\min, z) \in r \langle z, v \rangle \in r z \in X - \{ \min, v \}$ using assms(4)
  unfolding IsDense_def
  using $V(1,2)$ $\langle U \subseteq X \rangle \langle v \in X-U \rangle$ by blast
  from $z(2)$ $vc(1)$ have $zc : (z, c) \in r$ using assms(1) unfolding IsLinOrder_def
trans_def
  by fast moreover from $m(2)$ $z(1)$ have $\langle b, z \rangle \in r$ using assms(1) unfolding IsLinOrder_def
trans_def
  by fast ultimately have $z \in \text{Interval}(r,b,c)$ using Order_ZF_2_LiB by auto moreover
  {
assume $z = c$
  then have False using $z(2)$ $vc$ using assms(1) unfolding IsLinOrder_def
antisym_def
  by fast
  }
then have $z \neq c$ by auto moreover
  {
assume $z = b$

1138
then have \( z = \min \) using \( m(2) \) \( z(1) \) using \( \text{assms}(1) \) unfolding \( \text{IsLinOrder_def} \)
with \( z(3) \) have False by auto
}
then have \( z \neq b \) by auto moreover
have \( z \in X \) using \( z(3) \) by auto ultimately
have \( z \in \text{IntervalX}(X, r, b, c) \) unfolding \( \text{IntervalX_def} \) by auto
then have \( z \in V \) using \( b(1) \) by auto
then have \( z \in U \) using \( V(2) \) by auto moreover
from \( z(2, 3) \) have \( z \in \text{LeftRayX}(X, r, v) \) unfolding \( \text{LeftRayX_def} \) by auto ultimately
have \( z \in U \) using \( b(1) \) by auto
then have \( \min \in r \{ z \} \) using \( \text{Order_ZF_4_L4}(1)[\text{OF } _\text{Hmin}] \) \( \text{assms}(1) \)
unfolding \( \text{Supremum_def} \) \( \text{IsLinOrder_def} \) by auto
ultimately have False using \( V(3) \) by auto
}
then have \( \text{ass:min} \in X - U \) using \( a1 \) \( \text{assms}(3) \) by auto
then obtain \( V \) where \( \text{V:min} \in V \subseteq X - U \)
\( V \in \{ \text{IntervalX}(X, r, b, c). \langle b, c \rangle \in X 	imes X \} \cup \{ \text{LeftRayX}(X, r, b). \ b \in X \} \cup \{ \text{RightRayX}(X, r, b). \ b \in X \} \)
unfolding \( \text{point_open_base_neigh} \) \( \text{OF } \text{Ordtopology_is_a_topology}(2)[\text{OF } \text{assms}(1)] \) \( X - U \in (\text{OrdTopology } \text{X } r) \rangle \) \( \text{ass} \) by blast
{ assume \( V \in \{ \text{IntervalX}(X, r, b, c). \langle b, c \rangle \in X \times X \} \)
then obtain \( b \ c \) where \( b : V = \text{IntervalX}(X, r, b, c) \) \( b \in X \) \( \text{c} \in X \) by auto
from \( b V(1) \) have \( m : (\text{min}, c) \in r \langle b, \text{min} \rangle \in r \text{min} \neq b \text{ min} \neq c \) unfolding \( \text{IntervalX_def} \) by auto
ultimately have \( \langle \text{b}, x \rangle \in r \langle \text{x}, \text{c} \rangle \rangle \) \( \text{X} \langle \text{X} \rangle \) by auto
then have \( \langle (x, b) \rangle \in r \langle (x, c) \rangle \rangle \) \( \text{X} \langle \text{X} \rangle \) by auto
then have \( \langle (x, b) \rangle \in r \langle (x, c) \rangle \rangle \) \( \text{X} \langle \text{X} \rangle \) by auto
then have \( \langle (x, b) \rangle \in r \langle (c, x) \rangle \rangle \) \( \text{X} \langle \text{X} \rangle \) by auto
then have \( \langle k, x \rangle \in r \langle (c, x) \rangle \rangle \) \( \text{X} \langle \text{X} \rangle \) by auto
ultimately have False using \( \text{assms}(1) \) unfolding \( \text{IsLinOrder_def} \) \( \text{IsTotal_def} \)
using \( \text{b}(2, 3) \) by auto
ultimately have \( \langle (x, b) \rangle \in r \langle (c, x) \rangle \rangle \) \( \text{X} \langle \text{X} \rangle \) by auto
ultimately have \( \langle (x, b) \rangle \in r \langle (c, x) \rangle \rangle \) \( \text{X} \langle \text{X} \rangle \) by auto
ultimately have False using \( \text{assms}(1) \) unfolding \( \text{IsLinOrder_def} \) using \( \text{total_is_refl} \)
ultimately have False using \( \text{assms}(1) \) unfolding \( \text{IsLinOrder_def} \) using \( \text{total_is_refl} \)
ultimately have False using \( \text{assms}(1) \) unfolding \( \text{IsLinOrder_def} \) using \( \text{total_is_refl} \)
ultimately have \((x,b)\in r \lor (c,\text{min})\in r\) using assms(1) unfolding IsLinOrder_def trans_def

by fast
  with m(1) have \((x,b)\in r \lor c=\text{min}\) using assms(1) unfolding IsLinOrder_def antisym_def by auto
  with m(4) have \((x,b)\in r\) by auto

} then have \((\text{min},b)\in r\) using Order_ZF_5_L3[OF _ nE Hmin] assms(1) unfolding IsLinOrder_def by auto

\begin{align*}
  \mbox{moreover} \quad \\
  \{ \\
  \mbox{assume } V\in \{\text{RightRayX}(X,r,b). b\in X\} \\
  \mbox{then obtain } b \mbox{ where } b: V=\text{RightRayX}(X,r,b) \mbox{ b}\in X \mbox{ by auto} \\
  \mbox{from b V(1) have } m: (b,\text{min})\in r \mbox{min}\neq b \mbox{ unfolding RightRayX_def by auto} \\
  \{ \\
  \mbox{fix } x \mbox{ assume } A: x\in U \cap \text{LeftRayX}(X,r,v) \\
  \mbox{then have } (x,v)\in r \mbox{x}\in U \mbox{ unfolding LeftRayX_def by auto} \\
  \mbox{then have } x\notin V \mbox{ using V(2) by auto} \\
  \mbox{then have } x\notin \text{RightRayX}(X,r,b) \mbox{ using b(1) by auto} \\
  \mbox{then have } ((b,x)\notin r \lor x=b) x\in X \mbox{ unfolding RightRayX_def using } x\in U \mbox{ U}\subseteq X \mbox{ by auto} \\
  \mbox{then have } (x,b)\in r \mbox{ using assms(1) unfolding IsLinOrder_def using total_is_refl unfolding refl_def unfolding IsTotal_def using b(2) by auto} \\
  \} \\
\end{align*}

\begin{align*}
  \mbox{moreover} \quad \\
  \{ \\
  \mbox{assume } V\in \{\text{LeftRayX}(X,r,b). b\in X\} \\
  \mbox{then obtain } b \mbox{ where } b: V=\text{LeftRayX}(X,r,b) \mbox{ b}\in X \mbox{ by auto} \\
  \mbox{from b V(1) have } m: (\text{min},b)\in r \mbox{min}\neq b \mbox{ unfolding LeftRayX_def by auto} \\
  \{ \\
  \mbox{fix } x \mbox{ assume } A: x\in U \cap \text{LeftRayX}(X,r,v) \\
  \mbox{then have } (x,v)\in r \mbox{x}\in U \mbox{ unfolding LeftRayX_def by auto} \\
  \mbox{then have } x\notin V \mbox{ using V(2) by auto} \\
  \mbox{then have } x\notin \text{LeftRayX}(X,r,b) \mbox{ using b(1) by auto} \\
  \mbox{then have } ((b,x)\notin r \lor x=b) x\in X \mbox{ unfolding LeftRayX_def using } x\in U \mbox{ U}\subseteq X \mbox{ by auto} \\
  \mbox{then have } (b,x)\in r \mbox{ using assms(1) unfolding IsLinOrder_def using total_is_refl unfolding refl_def unfolding IsTotal_def using b(2) by auto} \\
  \} \end{align*}
trans_def by fast
moreover
from bound A have ∃g. ∀y∈U∩LeftRayX(X,r,v). ⟨y,g⟩∈r using
nE
unfolding IsBoundedAbove_def by auto
then obtain g where g:∀y∈U∩LeftRayX(X,r,v). ⟨y,g⟩∈r by auto
with nE obtain t where t∈U∩LeftRayX(X,r,v) by auto
with g have ⟨t,g⟩∈r by auto
with g have boundX:∃g∈X. ∀y∈U∩LeftRayX(X,r,v). ⟨y,g⟩∈r by auto
ultimately have ⟨x,min⟩∈r using Order_ZF_5_L7(2)[OF assms(3) _ assms(5)
_ boundX]
assms(1) <U⊆X> A unfolding LeftRayX_def IsLinOrder_def by auto
ultimately have x=min using assms(1) unfolding IsLinOrder_def antigm_def by auto
} then have U∩LeftRayX(X,r,v)⊆{min} by auto moreover
{ assume min∈U∩LeftRayX(X,r,v)
then have min∈U by auto
then have False using V(1,2) by auto
}
ultimately have False using nE by auto
moreover note V(3)
ultimately have False by auto
} with assms(1) have ⟨v,u⟩∈r unfolding IsLinOrder_def IsTotal_def using
⟦∀v∈X. v∈U⟧ unfolding LeftRayX_def by auto
have RightRayX(X,r,v)∈(OrdTopology X r) using base_sets_open[OF Ordtopology_is_a_topology assms(1)]
⟦∀v∈X-U. v∈U⟧ unfolding IsATopology_def by auto
{ fix b assume b∈(U∩RightRayX(X,r,v)
then have ⟨v,b⟩∈r unfolding RightRayX_def by auto
}
then have bound:IsBoundedBelow(U∩RightRayX(X,r,v),r) unfolding IsBoundedBelow_def by auto
with ⟨⟨v,u⟩∈r⟩⟨u∈U⟩⟨U⊆X⟩⟨v∈X-U⟩ have nE: U∩RightRayX(X,r,v)≠0 unfolding RightRayX_def by auto
have Hmax:HasAmaximum(r,⋂c∈U∩RightRayX(X,r,v). r-{c}) using complete_order_bounded_below[assms(5) bound nE assms(3)].
let max=Infimum(r,U∩RightRayX(X,r,v))
fix c assume c∈U∩RightRayX(X,r,v)
then have ⟨v,c⟩∈r unfolding RightRayX_def by auto

} then have a1:⟨v,max⟩∈r using Order_ZF_5_L4[OF _ nE Hmax] assms(1)
unfolding IsLinOrder_def
by auto

{ assume ass:max∈U
then obtain V where V:max∈VV⊆U
V∈{IntervalX(X,r,b,c). ⟨b,c⟩∈X×X}∪{LeftRayX(X,r,b). b∈X}∪{RightRayX(X,r,b).
b∈X}
using point_open_base_neigh
[OF Ordtopology_is_a_topology(2)[OF assms(1)] U∈(OrdTopology X r)]
by blast

} assume V∈{RightRayX(X,r,b). b∈X}
then obtain b where b:bcX V=RightRayX(X,r,b) by auto
from V(1) b(2) have a2:⟨b,max⟩∈max≠b unfolding RightRayX_def
by auto

} then have bv:⟨v,b⟩∈r using assms(1) unfolding IsLinOrder_def IsTotal_def
using b(1)
⟨v∈X-U⟩ by auto
from a2 assms(4) obtain z where z:⟨b,z⟩∈r(z,max)∈rz∈X-{b,max}
unfolding IsDense_def
using b(1) V(1,2) ⟨v∈X-U⟩ by blast
then have rayb:z∈RightRayX(X,r,b) unfolding RightRayX_def by auto
from z(1) bv have (v,z)∈r using assms(1) unfolding IsLinOrder_def
trans_def by fast moreover

{ assume z=v
with bv have (z,b)∈r by auto
with z(1) have b=z using assms(1) unfolding IsLinOrder_def antisym_def
by auto
then have False using z(3) by auto

} ultimately have z∈RightRayX(X,r,v) unfolding RightRayX_def using z(3) by auto
with rayb have z∈U∩RightRayX(X,r,v) using V(2) b(2) by auto
then have max∈r-{z} using Order_ZF_4_L3(1)[OF _ Hmax] assms(1)
unfolding Infimum_def IsLinOrder_def
by auto
then have (max,z)∈r by auto
with z(2,3)have False using assms(1) unfolding IsLinOrder_def
antisym_def by auto
} moreover
{
  assume \( V \in \{ \text{LeftRay}_X(X, r, b) \mid b \in X \} \)
  then obtain \( b \) where \( b : V = \text{LeftRay}_X(X, r, b) \) \( b \in X \) by auto
  note a1 moreover
  from V(1) b(1) have \( a2 : (\text{max}, b) \in r \max \neq b \) unfolding LeftRayX_def by auto
  ultimately have \( \langle v, b \rangle \in r \) using assms(1) unfolding IsLinOrder_def trans_def by blast
  moreover have False by auto
  } ultimately have False using V(2) b(1) unfolding LeftRayX_def using \( \{ v \in X \cup X \} \) by auto
} moreover
{
  assume \( V \in \{ \text{Interval}_X(X, r, b, c) \mid (b, c) \in X \times X \} \)
  then obtain \( b \) \( c \) where \( b : V = \text{Interval}_X(X, r, b, c) \) \( b \in X \) \( c \in X \) by auto
  from b V(1) have m: \( (\text{max}, c) \in r (b, \text{max}) \neq b \) \( \text{max} \neq c \) unfolding IntervalX_def by auto
  
  { assume A: \( \langle v, b \rangle \in r \) from m obtain z where \( z : (z, \text{max}) \in r (b, z) \in r z \in X - \{ b, \text{max} \} \) using assms(4) unfolding IsDense_def
    using b(2) V(1,2) \( \{ U \subseteq X \} \) by blast
    from z(1) have \( (z, c) \in r \) using m(1) assms(1) unfolding IsLinOrder_def trans_def by fast
    with z(2) have \( z \in \text{Interval}_X(X, r, b, c) \) \( \lor z = c \) using z(3) unfolding IntervalX_def
    then have \( z \in \text{Interval}_X(X, r, b, c) \) using m(1) z(1,3) using assms(1) unfolding IsLinOrder_def
    antisym_def by auto
    then have \( z \in \text{Interval}_X(X, r, b, c) \) using m(1) z(1,3) using assms(1) unfolding IsLinOrder_def
    antisym_def by auto
    with b(1) V(2) have \( z \in U \) by auto moreover
    from A z(2) have \( \langle v, z \rangle \in r \) using assms(1) unfolding IsLinOrder_def trans_def by fast
    moreover have \( z \neq v \) using A z(2,3) assms(1) unfolding IsLinOrder_def antisym_def by auto
    ultimately have \( z \in U \) \( \cup \text{RightRay}_X(X, r, v) \) unfolding RightRayX_def using z(3) by auto
    then have \( \text{max} \in r - \{ z \} \) using Order_ZF_4_L3(1)[OF \_ Hmax] assms(1) unfolding Infimum_def IsLinOrder_def

1143
by auto
then have $(\text{max},z) \in r$ by auto
with $z(1,3)$ have False using assms(1) unfolding IsLinOrder_def
antisym_def by auto
}\}
then have $vc: (b,v) \in r \forall v \neq b$ using assms(1) unfolding IsLinOrder_def
IsTotal_def using $\forall v \in X - U$
\begin{align*}
&b(2) \text{ by auto} \\
&\{ \text{ assume max}=v \\
&\text{ with } V(2,1) \forall v \in X - U \text{ have False by auto} \}
\end{align*}
then have $v \neq \text{max}$ by auto moreover
note a1 moreover
have $\text{max} \in X$ using $V(1,2)$ $\forall U \subseteq X$ by auto
moreover have $v \in X$ using $\forall v \in X - U$ by auto
ultimately obtain $z$ where $z: (v,z) \in r \forall (z,\text{max}) \in r \in X - \{v, \text{max}\}$ using
assms(4) unfolding IsDense_def
by auto
from $z(1)$ $vc(1)$ have $zc: (b,z) \in r$ using assms(1) unfolding IsLinOrder_def
trans_def
by fast moreover
from $m(1)$ $z(2)$ have $(z,c) \in r$ using assms(1) unfolding IsLinOrder_def
trans_def
by fast ultimately
have $z \in \text{Interval}(r,b,c)$ using Order_ZF_2_L1B by auto moreover
\begin{align*}
&\{ \text{ assume } z=b \\
&\text{ then have } \text{False using } z(1) \text{ } vc \text{ using assms(1) } \text{unfolding IsLinOrder_def} \}
\end{align*}
antisym_def
by fast
\begin{align*}
&\{ \text{ assume } z=c \\
&\text{ then have } z=\text{max using } m(1) \text{ } z(2) \text{ using assms(1) } \text{unfolding IsLinOrder_def} \}
\end{align*}
antisym_def by auto
with $z(3)$ have False by auto
\begin{align*}
&\{ \text{ assume } z \neq c \text{ by auto moreover} \\
&\text{ have } z \in X \text{ using } z(3) \text{ by auto ultimately} \\
&\text{ have } z \in \text{Interval}(X,r,b,c) \text{ unfolding } \text{Interval}(X)_\text{def} \text{ by auto} \\
&\text{ then have } z \in V \text{ using } b(1) \text{ by auto} \\
&\text{ then have } z \in U \text{ using } V(2) \text{ by auto moreover} \\
&\text{ from } z(1,3) \text{ have } z \in \text{RightRay}(X,r,v) \text{ unfolding } \text{RightRay}(X)_\text{def} \text{ by auto ultimately} \\
&\text{ have } z \in U \cup \text{RightRay}(X,r,v) \text{ by auto} \\
&\text{ then have } \text{max} \in r - \{z\} \text{ using } \text{OrderZF}_4_L3(1)[OF } \_ \text{Hmax] } \text{assms(1) unfolding Infimum_def IsLinOrder_def}
\end{align*}
by auto
then have $(\max, z) \in r$ by auto
with $z(2,3)$ have False using assms(1) unfolding IsLinOrder_def
antisym_def by auto

ultimately have False using $V(3)$ by auto

then have $\max \in X - U$ using $a1$ assms(3) by auto
then obtain $V$ where $V : max \in V \subseteq X - U$
$V \in \{\text{IntervalX}(X, r, b, c). (b,c) \in X \times X\} \cup \{\text{LeftRayX}(X, r, b). b \in X\} \cup \{\text{RightRayX}(X, r, b). b \in X\}$ using point_open_base_neigh
[OF Ordtopology_is_a_topology(2) OF assms(1)] $\langle X - U \in (\text{OrdTopology X r}) \rangle$ unfolding IsLinOrder_def IsTot ordered_def by auto

\begin{align*}
\{ & \text{assume $V \in \{\text{IntervalX}(X, r, b, c). (b,c) \in X \times X\}$} \\
& \text{then obtain $b$ $c$ where $b : V = \text{IntervalX}(X, r, b, c)$ $b \in X$ $c \in X$ by auto} \\
& \text{from $b$ $V(1)$ have $m : (\max, c) \in r$ $\max \neq b$ $\max \neq c$ unfolding IntervalX_def} \\
& \text{Interval_def by auto} \\
& \{ & \text{fix $x$ assume $A : x \in U \cap \text{RightRayX}(X, r, v)$} \\
& \text{then have $\langle v, x \rangle \in r \subseteq U$ unfolding RightRayX_def by auto} \\
& \text{then have $x \notin V$ using $V(2)$ by auto} \\
& \text{then have $x \notin \text{Interval}(r, b, c) \cap X \forall x = b \forall x = c$ using $b(1)$ unfolding IntervalX_def by auto} \\
& \text{then have $\langle (b, x) \notin r \cup \langle x, c \rangle \notin r \rangle \forall x = b \forall x = c$ using Order_ZF_2_L1B $\langle x \in U \rangle \langle U \subseteq X \rangle$ by auto} \\
& \text{then have $\langle (x, b) \in r \cup \langle x, c \rangle \in r \rangle \forall x = b \forall x = c$ using assms(1) unfolding IsLinOrder_def IsTotal_def using $b(2,3)$ by auto} \\
& \text{then have $\langle (x, b) \in r \cup \langle x, c \rangle \in r \rangle$ using assms(1) unfolding IsLinOrder_def using total_is_refl unfolding refl_def using $b(2,3)$ by auto moreover} \\
& \text{from $A$ have $\langle \max, x \rangle \in r$ using Order_ZF_4_L3(1) OF _ $Hmax$} \text{assms(1) unfolding Infimum_def IsLinOrder_def by auto} \\
& \text{ultimately have $\langle (\max, b) \in r \cup \langle c, x \rangle \in r \rangle$ using assms(1) unfolding IsLinOrder_def trans_def} \\
& \text{by fast} \\
& \text{with $m(2)$ have $(\max = b \cup \langle c, x \rangle \in r)$ using assms(1) unfolding IsLinOrder_def antisym_def by auto} \\
& \text{with $m(3)$ have $\langle c, x \rangle \in r$ by auto} \\
& \} \text{then have $\langle c, \max \rangle \in r$ using Order_ZF_5_L4 OF _ $nE$ $Hmax$} \text{assms(1) unfolding IsLinOrder_def by auto} \\
& \text{with $m(1,4)$ have False using assms(1) unfolding IsLinOrder_def antisym_def by auto} \\
& \} \text{moreover} \\
& \}
\end{align*}
assume $V \in \{\text{RightRayX}(X,r,b). b \in X\}$
then obtain $b$ where $b:V=\text{RightRayX}(X,r,b)$ $b \in X$ by auto
from $b$ $V(1)$ have $m:(b,\max) \in rmax \neq b$ unfolding $\text{RightRayX}_{\text{def}}$ by auto

} 

by auto
then have $(x,b) \in r$ using $\text{assms}(1)$ unfolding $\text{IsLinOrder}_{\text{def}}$ using $\text{total}_\text{is_refl}$ unfolding $\text{refl}_{\text{def}}$ unfolding $\text{IsTotal}_{\text{def}}$ using $\text{b}(2)$ by auto moreover from $A$ have $(\max,x) \in r$ unfolding $\text{Order}_{\text{ZF}_{4}}_{\text{L3}}(1)[OF _{Hmax}]$ $\text{assms}(1)$ unfolding $\text{Infimum}_{\text{def}}$ $\text{IsLinOrder}_{\text{def}}$
by auto ultimately have $(\max,b) \in r$ using $\text{assms}(1)$ unfolding $\text{IsLinOrder}_{\text{def}}$ $\text{trans}_{\text{def}}$
by fast
with $m$ have False using $\text{assms}(1)$ unfolding $\text{IsLinOrder}_{\text{def}}$ $\text{antisym}_{\text{def}}$
by auto

} 

moreover 
{ 
assume $V \in \{\text{LeftRayX}(X,r,b). b \in X\}$
then obtain $b$ where $b:V=\text{LeftRayX}(X,r,b)$ $b \in X$ by auto
from $b$ $V(1)$ have $m:(\max,b) \in rmax \neq b$ unfolding $\text{LeftRayX}_{\text{def}}$ by auto

} 

by auto
then have $(b,x) \in r$ using $\text{assms}(1)$ unfolding $\text{IsLinOrder}_{\text{def}}$ using $\text{total}_\text{is_refl}$ unfolding $\text{refl}_{\text{def}}$ unfolding $\text{IsTotal}_{\text{def}}$ using $\text{b}(2)$ by auto

} 

with $\text{nE}$ have $b \in (\bigcap c \in U \cap \text{RightRayX}(X,r,v). r-(c))$ by auto
then have $(b,\max) \in r$ unfolding $\text{Infimum}_{\text{def}}$ using $\text{Order}_{\text{ZF}_{4}}_{\text{L3}}(2)[OF _{Hmax}]$ $\text{assms}(1)$ unfolding $\text{IsLinOrder}_{\text{def}}$ by auto
with $m$ have False using $\text{assms}(1)$ unfolding $\text{IsLinOrder}_{\text{def}}$ $\text{antisym}_{\text{def}}$
by auto

} 

moreover note $V(3)$
ultimately have False by auto

} 

1146
then show thesis by auto
qed

82.4 Numerability axioms

A $\kappa$-separable order topology is in relation with order density.

If an order topology has a subset $A$ which is topologically dense, then that subset is weakly order-dense in $X$.

**Lemma dense_top_imp_Wdense_ord:**

assumes IsLinOrder($X,r$) Closure($A$,OrdTopology $X$ $r$)=$X$ $A$$\subseteq X$ $\exists x \ y. \ x \neq y \land x \in X \land y \in X$

shows $A$ {is weakly dense in}$X$ {with respect to}$r$

**Proof:**

{ fix $r_1 \ r_2$ assume $r_1$$\in Xr_2$$\in Xr_1$$\neq r_2$ $(r_1,r_2)$$\in r$

then have $\text{Interval}_X(X,r,r_1,r_2)\subseteq \{\text{Interval}_X(X, r, b, c) . (b,c) \in X \times X\} \cup \{\text{LeftRay}_X(X, r, b) . b \in X\} \cup \{\text{RightRay}_X(X, r, b) . b \in X\}$

by auto

then have $P:\text{Interval}_X(X,r,r_1,r_2)\subseteq (\text{OrdTopology} \ X \ r)$ using base_sets_open[OF Ordtopology_is_a_topology(2)[OF assms(1)]]

by auto

have $\text{Interval}_X(X,r,r_1,r_2) \subseteq X$ unfolding IntervalX_def by auto

then have int:$\text{Closure}(A,\text{OrdTopology} \ X \ r) \cap \text{Interval}_X(X,r,r_1,r_2)=\text{Interval}_X(X,r,r_1,r_2)$ using assms(2) by auto

{ assume $\text{Interval}_X(X,r,r_1,r_2)\neq 0$

then have $A \cap (\text{Interval}_X(X,r,r_1,r_2))\neq 0$ using topology0.cl_inter_neigh[OF topology0_ordtopology[OF assms(1)] _ P , of $A$]

using assms(3) union_ordtopology[OF assms(1,4)] int by auto

} then have $(\exists z \in A-{r_1,r_2}). \ (r_1,z)\in r \land (z,r_2)\in r \ \forall \text{Interval}_X(X,r,r_1,r_2)=0$

unfolding IntervalX_def

Interval_def by auto

} then show thesis unfolding IsWeaklyDenseSub_def by auto
qed

Conversely, a weakly order-dense set is topologically dense if it is also considered that: if there is a maximum or a minimum elements whose singletons are open, this points have to be in $A$. In conclusion, weakly order-density is a property closed to topological density.

Another way to see this: Consider a weakly order-dense set $A$:

- If $X$ has a maximum and a minimum and $\{min, max\}$ is open: $A$ is topologically dense in $X \setminus \{min, max\}$, where $min$ is the minimum in $X$ and $max$ is the maximum in $X$. 

1147
• If \( X \) has a maximum, \( \{ \max \} \) is open and \( X \) has no minimum or \( \{ \min \} \) isn’t open: \( A \) is topologically dense in \( X \setminus \{ \max \} \), where \( \max \) is the maximum in \( X \).

• If \( X \) has a minimum, \( \{ \min \} \) is open and \( X \) has no maximum or \( \{ \max \} \) isn’t open: \( A \) is topologically dense in \( X \setminus \{ \min \} \), where \( \min \) is the minimum in \( X \).

• If \( X \) has no minimum or maximum, or \( \{ \min, \max \} \) has no proper open sets: \( A \) is topologically dense in \( X \).

**Lemma**: \( \text{wdense}_\text{ord}_\text{imp}_\text{dense}_\text{top} \):

\[
\text{Assumes } \text{IsLinOrder}(X, r) \land \text{A is weakly dense in } X \text{ with respect to } r \land A \subseteq X
\exists x. y. x \neq y \land x, y \in X
\text{HasMinimum}(r, X) \rightarrow \{ \text{Minimum}(r, X) \} \in (\text{OrdTopology } X, r) \rightarrow \text{Minimum}(r, X) \in A
\text{HasMaximum}(r, X) \rightarrow \{ \text{Maximum}(r, X) \} \in (\text{OrdTopology } X, r) \rightarrow \text{Maximum}(r, X) \in A
\text{Shows } \text{Closure}(A, \text{OrdTopology } X, r) = X
\]

**Proof**

- \{ fix \( x \) assume \( x \in X \)
- \{ fix \( U \) assume \( \exists x \in U \subseteq (\text{OrdTopology } X, r) \)
  then have \( \exists V \subseteq \{ \text{Interval}(X, r, b, c) . (b, c) \in X \times X \} \cup \{ \text{LeftRay}(X, r, b) . b \in X \} \cup \{ \text{RightRay}(X, r, b) . b \in X \} . V \subseteq U \land x \in V \)
    using point_open_base_neigh[OF Ordtopology_is_a_topology[OF assms(1)]]
    by auto
  then obtain \( V \)
    where \( V \subseteq \{ \text{Interval}(X, r, b, c) . (b, c) \in X \times X \} \cup \{ \text{LeftRay}(X, r, b) . b \in X \} \cup \{ \text{RightRay}(X, r, b) . b \in X \} \land x \in V \)
    by blast
    note \( V(1) \) moreover
    \{ assume \( V \subseteq \{ \text{Interval}(X, r, b, c) . (b, c) \in X \times X \} \)
    then obtain \( b, c \)
      where \( b, c \in X \times X \)
    by auto
      with \( V(3) \) have \( x \in X \land x \in A \land x \neq b \land x \neq c \)
    unfolding IntervalX_def
    Interval_def by auto
      then have \( \langle b, c \rangle \in r \) using assms(1) unfolding IsLinOrder_def trans_def
    by fast
    moreover from \( x(1-3) \) have \( b \neq c \) using assms(1) unfolding IsLinOrder_def
    antisym_def by fast
    moreover note \( \exists x \in A \land x \neq b \land x \neq c \)
    ultimately have \( \exists z \in A \land \langle z, c \rangle \in r \) unfolding IsWeaklyDenseSub_def
    by auto
    then obtain \( z \)
      where \( z \in A \land \langle z, c \rangle \in r \land (z, c) \in r \land z \neq b \land z \neq c \)
    unfolding IntervalX_def
    Interval_def by auto
    then have \( A \setminus U \neq 0 \) using \( V(2) \) b(3) by auto
  \}
  moreover
  

assume \( V \in \{ \text{RightRayX}(X, r, b) \mid b \in X \} \)
then obtain \( b \) where \( b : b \in V = \text{RightRayX}(X, r, b) \) by auto
with \( V(3) \) have \( x : (b, x) \in r \ (b \neq x) \) unfolding \( \text{RightRayX}\_\text{def} \) by auto moreover
note \( b(1) \) moreover
have \( U \subseteq \bigcup \langle \text{OrdTopology } X \mid r \rangle \) using \( \text{ass}(2) \) by auto
then have \( U \subseteq X \) using \( \text{union}\_\text{ordtopology} [\text{OF } \text{ass}(1, 4)] \) by auto
then have \( x \in X \) using \( \text{ass}(1) \) by auto moreover
note \( \text{assms}(2) \) ultimately
have \( \text{disj} : (\exists z \in A - \{ b, x \}. (b, z) \in r \land (z, x) \in r) \lor \text{IntervalX}(X, r, b, x) = 0 \) unfolding \( \text{IsWeaklyDenseSub}\_\text{def} \) by auto
\{
assume \( B : \text{IntervalX}(X, r, b, x) = 0 \)
\{
assume \( \exists y \in X. (x, y) \in r \land x \neq y \)
then obtain \( y \) where \( y : y \in X \) using \( \text{ass}(2) \) by auto
with \( x \) have \( x : x \in \text{IntervalX}(X, r, b, y) \) unfolding \( \text{IntervalX}\_\text{def} \) by auto
trans_def by fast
moreover have \( b \neq y \) using \( \text{assms}(1) \) unfolding \( \text{IsLinOrder}\_\text{def} \)
antisym_def by fast
ultimately
have \( \exists z \in A - \{ b, y \}. (b, z) \in r \land (z, y) \in r \) using \( \text{assms}(2) \) unfolding \( \text{IsWeaklyDenseSub}\_\text{def} \)
using \( \langle x \in X \rangle \) by auto
then obtain \( z \) where \( z \in A - b, z \neq b \) by auto
then have \( z : z \in X - V \) using \( \text{assms}(3) \) by auto
by auto
then have \( z : z \in X - U \) using \( V(2) \) by auto
then have \( A - U \neq 0 \) by auto
\}
moreover
\{
assume \( R : \forall y \in X. (x, y) \in r \rightarrow x = y \)
\{
fix \( y \) assume \( y : \text{RightRayX}(X, r, b) \)
then have \( y : (b, y) \in r \ y \in X - \{ b \} \) unfolding \( \text{RightRayX}\_\text{def} \) by auto
\{
assume \( A : y \neq x \)
then have \( (x, y) \notin r \) using \( \text{R } y(2) \) by auto
then have \( (y, x) \in r \) using \( \text{assms}(1) \) unfolding \( \text{IsLinOrder}\_\text{def} \)
\text{IsTotal}\_\text{def} \)
using \( \langle x \in X \rangle y(2) \) by auto
with \( A \ y \) have \( y \in \text{IntervalX}(X, r, b, x) \) unfolding \( \text{IntervalX}\_\text{def} \)
\text{Interval}\_\text{def} \)
by auto
then have False using \( B \) by auto
\}
then have \( y = x \) by auto

} then have \( \text{RightRayX}(X, r, b) = \{x\} \) using V(3), b(2) by blast

moreover

\{
  \text{fix} \ t \ \text{assume} \ T : t \in X \\
  \text{assume} \ t = x \\
  \text{then have} \ (t, x) \in r \ \text{using} \ \text{assms}(1) \ \text{unfolding} \ \text{IsLinOrder_def} \\
  \text{using} \ \text{OrderZF}_1.L1 \ \text{T by auto}
\}

moreover

\{
  \text{assume} \ t \neq x \\
  \text{then have} \ (x, t) \not\in r \ \text{by auto} \\
  \text{then have} \ (t, x) \in r \ \text{using} \ \text{assms}(1) \ \text{unfolding} \ \text{IsLinOrder_def} \ \text{IsTotal_def} \\
  \text{using} \ T < x \in X \ \text{by auto} \\
\}

ultimately have \( (t, x) \in r \) by auto

\}

with \( <x \in X> \) have \( \text{HM:HasAmaximum}(r, X) \) unfolding \( \text{HasAmaximum_def} \) by auto

then have \( \text{Maximum}(r, X) \in \forall t \in X. (t, \text{Maximum}(r, X)) \in r \) using \text{OrderZF}_4.L3

\( \text{assms}(1) \) unfolding \( \text{IsLinOrder_def} \) by auto

with \( R <x \in X> \) have \( \text{xm:x=Maximum}(r, X) \) by auto

moreover note \( b(2) \)

ultimately have \( V = \{\text{Maximum}(r, X)\} \) by auto

then have \( \{\text{Maximum}(r, X)\} \in (\text{OrdTopology X r}) \) using \text{base_sets_open[OF Ordtopology_is_a_topology(2)[OF assms(1)]]}

\( V(1) \) by auto

with \( \text{HM} \) have \( \text{Maximum}(r, X) \in A \) using \( \text{assms}(6) \) by auto

with \( \text{xm have} \ x \in A \) by auto

with \( V(2,3) \) have \( A \cap U \neq 0 \) by auto

\}

ultimately have \( A \cap U \neq 0 \) by auto

\}

moreover

\{
  \text{assume} \ \text{IntervalX}(X, r, b, x) \neq 0 \\
  \text{with} \ \text{disj have} \ \exists z \in A - \{b, x\}. (b, z) \in r \land (z, x) \in r \ \text{by auto} \\
  \text{then obtain} \ z \ \text{where} \ z \in A \neq b \in r \ \text{by auto} \\
  \text{then have} \ z \in A \in \text{RightRayX}(X, r, b) \ \text{unfolding} \ \text{RightRayX_def using} \ \text{assms}(3) \ \text{by auto} \\
  \text{then have} \ z \in A \cap U \ \text{using} \ V(2) \ \text{b(2) by auto} \\
  \text{then have} \ A \cap U \neq 0 \ \text{by auto}
\}

ultimately have \( A \cap U \neq 0 \) by auto

1150
moreover
{
assume \( \forall V \in \{ \text{LeftRayX}(X, r, b) \mid b \in X \} \)
then obtain \( b \) where \( b \in \text{LeftRayX}(X, r, b) \) by auto
with \( V(3) \) have \( x : (x, b) \in r \) \( b \neq x \) unfolding \text{LeftRayX_def} by auto
\}

moreover
note \( U \subseteq \bigcup (\text{OrdTopology} X \ r) \) using \( \text{ass}(2) \) by auto
then have \( U \subseteq X \) unfolding union_ordtopology[OF \( \text{ass}(1, 4) \)] by auto
then have \( x \in X \) using \( \text{union_ordtopology[OF assms(1,4)]} \) by auto
moreover
note \( \text{assms}(2) \) ultimately
have \( \text{disj} : (\exists z \in A - \{ b, x \}. (x, z) \in r \land (z, b) \in r) \lor \text{IntervalX}(X, r, x, b) = 0 \) unfolding \text{IsWeaklyDenseSub_def} by auto
\}

moreover
{
assume \( \forall B : \text{IntervalX}(X, r, x, b) = 0 \)
\}

assume \( \exists y \in X. (y, x) \in r \land x \neq y \)
then obtain \( y \) where \( y : y \in X \land (y, x) \in r \land x \neq y \) by auto
with \( x \) have \( x \in \text{IntervalX}(X, r, y, b) \) unfolding \text{IntervalX_def} by auto
moreover
have \( (y, b) \in r \) using \( \text{y(2)} \) \( \text{x(1)} \) \( \text{assms}(1) \) unfolding \text{IsLinOrder_def} trans_def by fast
moreover have \( b \neq y \) using \( \text{y(2,3)} \) \( \text{x(1)} \) \( \text{assms}(1) \) unfolding \text{IsLinOrder_def} antisym_def by fast
ultimately
have \( (\exists z \in A - \{ b, x \}. (y, z) \in r \land (z, b) \in r) \) using \( \text{assms}(2) \) unfolding \text{IsWeaklyDenseSub_def} using \( \text{y(1)} \) \( \text{b(1)} \) by auto
then obtain \( z \) where \( z \in A \land \overline{r} b \neq z \) by auto
then have \( z \in A \cap U \) unfolding \text{LeftRayX_def} by auto
moreover
{
assume \( \forall y \in X. (y, x) \in r \longrightarrow x = y \)
\}

fix \( y \) assume \( \exists y \in \text{LeftRayX}(X, r, b) \)
then have \( y : (y, b) \in r \) \( y \in X - \{ b \} \) unfolding \text{LeftRayX_def} by auto
moreover
note \( A : y \neq x \)
then have \( (y, x) \notin r \) using \( \text{R} \) \( y(2) \) by auto
then have \( (x, y) \in r \) using \( \text{assms}(1) \) unfolding \text{IsLinOrder_def IsTotal_def} using \( \langle x \in X \rangle \) \( y(2) \) by auto
with \( A \) \( y \) have \( y \in \text{IntervalX}(X, r, x, b) \) unfolding \text{IntervalX_def Interval_def}

1151
by auto
then have False using B by auto

then have \( y = x \) by auto

then have \( \text{LeftRayX}(X, r, b) = \{ x \} \) using \( V(3) \) \( b(2) \) by blast

moreover
\[
\begin{align*}
\text{fix } t &\text{ assume } T : t \in X \\
&\text{ assume } t = x \\
&\text{ then have } \langle x, t \rangle \in r \text{ using } \text{assms}(1) \text{ unfolding } \text{IsLinOrder_def} \\
&\text{ using } \text{Order_ZF_1_L1} T \text{ by auto}
\end{align*}
\]

moreover
\[
\begin{align*}
&\text{ assume } t \neq x \\
&\text{ then have } \langle t, x \rangle \notin r \text{ using } R T \text{ by auto} \\
&\text{ then have } \langle x, t \rangle \in r \text{ using assms(1) unfolding } \text{IsLinOrder_def} \\
&\text{ using } \text{IsTotal_def} \text{ unfolding } T \langle x \in X \rangle \text{ by auto}
\end{align*}
\]

ultimately have \( \langle x, t \rangle \in r \) by auto

with \( x \in X \) have \( \text{HM: HasAminimum}(r, X) \) unfolding \( \text{HasAminimum_def} \)

by auto
then have \( \text{Minimum}(r, X) \in \forall t \in X. (\text{Minimum}(r, X), t) \in r \) using \( \text{Order_ZF_4_L4} \) \( \text{assms}(1) \) unfolding \( \text{IsLinOrder_def} \)

by auto
with \( x \in X \) have \( \text{ xm: x = Minimum(r, X) by auto} \)

moreover note \( b(2) \)
ultimately have \( V = \{ \text{Minimum}(r, X) \} \) by auto
then have \( \{ \text{Minimum}(r, X) \} \in (\text{OrdTopology} X \ r) \) using \( \text{base_sets_open[OF \text{Ordtopology_is_a_topology}(2) [OF assms(1)]]} \) \( V(1) \) by auto

with \( \text{HM} \) have \( \text{Minimum}(r, X) \in A \) using \( \text{assms}(5) \) by auto
with \( \text{xm} \) have \( x \in A \) by auto
with \( V(2, 3) \) have \( A \cap U \neq 0 \) by auto

ultimately have \( A \cap U \neq 0 \) by auto

moreover
\[
\begin{align*}
&\text{ assume } \text{IntervalX}(X, r, x, b) \neq 0 \\
&\text{ with } \text{disj} \text{ have } \exists z \in A - \{ b, x \}. (x, z) \in r \land (z, b) \in r \text{ by auto} \\
&\text{ then obtain } z \text{ where } z \in A \land z \neq b \land (z, b) \in r \text{ by auto} \\
&\text{ then have } z \in A \land z \in \text{LeftRayX}(X, r, b) \text{ unfolding } \text{LeftRayX_def} \text{ using } \text{assms}(3) \\
&\text{ by auto}
\end{align*}
\]

then have \( z \in A \cap U \) using \( V(2) \) \( b(2) \) by auto

1152
then have \( A \cap U \neq 0 \) by auto

ultimately have \( A \cap U \neq 0 \) by auto

ultimately have \( A \cap U \neq 0 \) by auto

then have \( \forall U \in (\text{OrdTopology } X \ r). \ x \in U \implies U \cap A \neq 0 \) by auto

moreover note \( x \in X \) moreover

note \( \text{assms}(3) \) \( \text{topology0.inter_neigh_cl}[\text{OF topology0.ordtopology}[\text{OF assms(1)}]] \)

\( \text{union_orddtopology}[\text{OF assms(1,4)}] \) ultimately have \( x \in \text{Closure}(A, \text{OrdTopology } X \ r) \)

by auto

then have \( X \subseteq \text{Closure}(A, \text{OrdTopology } X \ r) \) by auto

with \( \text{topology0.Top_3_L11}(1) \) \( \text{OF topology0.ordtopology}[\text{OF assms(1)}] \)

\( \text{assms}(3) \) \( \text{union_orddtopology}[\text{OF assms(1,4)}] \) show thesis by auto

qed

The conclusion is that an order topology is \( \kappa \)-separable iff there is a set \( A \) with cardinality strictly less than \( \kappa \) which is weakly-dense in \( X \).

**Theorem separable_imp_wdense:**

assumes \( (\text{OrdTopology } X \ r) \) \( \text{is separable of cardinal} \) \( Q \) \( \exists x \ y. \ x \neq y \land x \in X \land y \in X \)

\( \text{IsLinOrder}(X, r) \)

shows \( \exists A \in \text{Pow}(X). \ A \prec Q \land (A \text{ is weakly dense in } X \text{ with respect to } r) \)

**Proof:**

from \( \text{assms} \) obtain \( U \) where \( U \in \text{Pow}(\bigcup (\text{OrdTopology } X \ r)) \)

\( \text{Closure}(U, \text{OrdTopology } X \ r) = \bigcup (\text{OrdTopology } X \ r) \cup Q \)

unfolding \( \text{IsSeparableOfCard_def} \) by auto

then have \( U \in \text{Pow}(X) \)

\( \text{Closure}(U, \text{OrdTopology } X \ r) = X \cup Q \)

using \( \text{union_orddtopology}[\text{OF assms(3,2)}] \)

by auto

with \( \text{dense_top_imp_Wdense_ord}[\text{OF assms(3)}] \) show thesis by auto

qed

**Theorem wdense_imp_separable:**

assumes \( \exists x \ y. \ x \neq y \land x \in X \land y \in X \) \( (A \text{ is weakly dense in } X \text{ with respect to } r) \)

\( \text{IsLinOrder}(X, r) \)

\( A \prec Q \)

\( \text{InfCard}(Q) \)

\( A \subseteq X \)

shows \( (\text{OrdTopology } X \ r) \) \( \text{is separable of cardinal} \) \( Q \)

**Proof:**

{ assume \( \text{Hmin:HasAmaximum}(r, X) \)

then have \( \text{MaxX:Maximum}(r, X) \in X \) using \( \text{Order_ZF_4_L3}(1) \)

unfolding \( \text{IsLinOrder_def} \)

by auto

}

{ assume \( \text{HMax:HasAminimum}(r, X) \)}
then have $\text{MinX} : \text{Minimum}(r,X) \in X$ using Order_ZF_4_L4(1) assms(3) unfolding IsLinOrder_def
by auto
let $A = A \cup \{\text{Maximum}(r,X), \text{Minimum}(r,X)\}$
have $\text{Finite}(\{\text{Maximum}(r,X), \text{Minimum}(r,X)\})$ by auto
then have $\{\text{Maximum}(r,X), \text{Minimum}(r,X)\} \prec \text{nat}$ using n_lesspoll_nat unfolding Finite_def using eq_lesspoll_trans by auto
moreover
from assms(5) have $\text{nat} \prec \text{Q} \lor \text{nat} = \text{Q}$ unfolding InfCard_def
using $\text{lt}_\text{Card}_\text{imp}_\text{lesspoll}[\text{of} \, \text{Qnat}]$ unfolding $\text{lt}_\text{def} \, \text{succ}_\text{def}$
using Card_is_Ord[of Q] by auto
ultimately have $\{\text{Maximum}(r,X), \text{Minimum}(r,X)\} \prec \text{Q}$ using lesspoll_trans by auto
moreover
assume $\text{nmin} : \neg \text{HasAminimum}(r,X)$
let $A = A \cup \{\text{Maximum}(r,X)\}$
have $\text{Finite}(\{\text{Maximum}(r,X)\})$ by auto
then have $\{\text{Maximum}(r,X)\} \prec \text{nat}$ using n_lesspoll_nat unfolding Finite_def using eq_lesspoll_trans by auto
moreover
from assms(5) have $\text{nat} \prec \text{Q} \lor \text{nat} = \text{Q}$ unfolding InfCard_def
using $\text{lt}_\text{Card}_\text{imp}_\text{lesspoll}[\text{of} \, \text{Qnat}]$ unfolding $\text{lt}_\text{def} \, \text{succ}_\text{def}$
using Card_is_Ord[of Q] by auto
ultimately have $\{\text{Maximum}(r,X)\} \prec \text{Q}$ using lesspoll_trans by auto
moreover
assume $\text{nmin} : \neg \text{HasAminimum}(r,X)$
let $A = A \cup \{\text{Maximum}(r,X)\}$
have $\text{Finite}(\{\text{Maximum}(r,X)\})$ by auto
then have $\{\text{Maximum}(r,X)\} \prec \text{nat}$ using n_lesspoll_nat unfolding Finite_def using eq_lesspoll_trans by auto
moreover
from assms(5) have $\text{nat} \prec \text{Q} \lor \text{nat} = \text{Q}$ unfolding InfCard_def
using $\text{lt}_\text{Card}_\text{imp}_\text{lesspoll}[\text{of} \, \text{Qnat}]$ unfolding $\text{lt}_\text{def} \, \text{succ}_\text{def}$
using Card_is_Ord[of Q] by auto
ultimately have $\{\text{Maximum}(r,X)\} \prec \text{Q}$ using lesspoll_trans by auto
moreover
ultimately have thesis by auto

moreover
{ assume nmax:\neg HasAmaximum(r,X)
  
  assume HMin:HasAminimum(r,X)
  then have MinX:Minimum(r,X)\in X using Order_ZF_4_L4(1) assms(3) unfolding IsLinOrder_def
  by auto
  let A=A \cup \{Minimum(r,X)\}
  have Finite({Minimum(r,X)}) by auto
  then have {Minimum(r,X)}\prec \text{nat} using n_lesspoll_nat
    unfolding Finite_def using eq_lesspoll_trans by auto
  moreover
  from assms(5) have \text{nat} \prec Q \lor \text{nat} = Q unfolding InfCard_def
    using lt_Card_imp_lesspoll[of Qnat] unfolding lt_def succ_def
    using Card_is_Ord[of Q] by auto
  ultimately have {Minimum(r,X)} \prec Q using lesspoll_trans by auto
  with assms(4,5) have C:A \prec Q using less_less_imp_un_less
    by auto
  have WeakDense:A\{is weakly dense in\}X\{with respect to\}r using assms(2) unfolding IsWeaklyDenseSub_def by auto
  from MinX assms(6) have S:A\subseteq X by auto
  then have Closure(A,OrdTopology X r)=X using Wdense_ord_imp_dense_top
    [OF assms(3) WeakDense _ assms(1)] nmax by auto
  then have thesis unfolding IsSeparableOfCard_def using union_ordtopology[OF assms(3,1)]
    S C by auto
  }
moreover
{ assume nmin:\neg HasAminimum(r,X)
  let A=A
  from assms(4,5) have C:A \prec Q by auto
  have WeakDense:A\{is weakly dense in\}X\{with respect to\}r using assms(2) unfolding IsWeaklyDenseSub_def by auto
  from assms(6) have S:A\subseteq X by auto
  then have Closure(A,OrdTopology X r)=X using Wdense_ord_imp_dense_top
    [OF assms(3) WeakDense _ assms(1)] nmin nmax by auto
  then have thesis unfolding IsSeparableOfCard_def using union_ordtopology[OF assms(3,1)]
    S C by auto
  }
ultimately have thesis by auto

ultimately show thesis by auto
83 Properties in topology 2

theory Topology_ZF_properties_2 imports Topology_ZF_7 Topology_ZF_1b
  Finite_ZF_1 Topology_ZF_11

begin

83.1 Local properties.

This theory file deals with local topological properties; and applies local
compactness to the one point compactification.

We will say that a topological space is locally "P" iff every point
has a neighbourhood basis of subsets that have the property "P" as
subspaces.

definition
  IsLocally (_{is locally}_ 90)
  where T{is a topology} = T{is locally}P ≡ (∀x∈∪T. ∀b∈T. x∈b →
     (∃c∈Pow(b). x∈Interior(c,T) ∧ P(c,T)))

83.2 First examples

Our first examples deal with the locally finite property. Finiteness is a
property of sets, and hence it is preserved by homeomorphisms; which are
in particular bijective.

The discrete topology is locally finite.

lemma discrete_locally_finite:
  shows Pow(A){is locally}(λA.λB. Finite(A)))
proof-
  have ∀b∈Pow(A). ∪(Pow(A){restricted to}b)=b unfolding RestrictedTo_def
  by blast
  then have ∀b∈{x}. x∈A}. Finite(b) by auto moreover
  have reg:∀S∈Pow(A). Interior(S,Pow(A))=S unfolding Interior_def by auto
  { fix x b assume x∈∪Pow(A) b∈Pow(A) x∈b
  then have {x}⊆b x∈Interior({x},Pow(A)) Finite({x}) using reg by auto
  then have ∃c∈Pow(b). x∈Interior(c,Pow(A))∧Finite(c) by blast
  } then have ∀x∈∪Pow(A). ∀b∈Pow(A). x∈b → (∃c∈Pow(b). x∈Interior(c,Pow(A))
     ∧ Finite(c))) by auto

qed

end
then show thesis using IsLocally_def[OF Pow_is_top] by auto
qed

The included set topology is locally finite when the set is finite.

**lemma included_finite_locally_finite:**
assumes Finite(A) and A ⊆ X
shows (IncludedSet(X,A)){is locally}(λA.(λB. Finite(A)))

**proof-**

have ∀ b∈Pow(X). b∩A⊆b by auto

moreover
note assms(1)
ultimately have rr: ∀ b∈{A∪{x}. x∈X}. Finite(b) by force

{ fix x b assume x∈⋃ (IncludedSet(X,A)) b∈(IncludedSet(X,A)) x∈b
then have A∪x⊆b A∪x∈{A∪x}. x∈X and sub: b⊆X unfolding IncludedSet_def by auto
moreover have A ∪ {x} ⊆ X using assms(2) sub ⟨x∈b⟩ by auto
then have x∈Interior(A∪{x},IncludedSet(X,A)) using interior_set_includedset[of A∪{x}X] by auto
ultimately have ∃ c∈Pow(b). x∈Interior(c,IncludedSet(X,A))∧ Finite(c) using rr by blast
}

then have ∀ x∈⋃ (IncludedSet(X,A)). ∀ b∈(IncludedSet(X,A)). x∈b → (∃ c∈Pow(b). x∈Interior(c,IncludedSet(X,A))∧ Finite(c)) by auto
then show thesis using IsLocally_def includedset_is_topology by auto
qed

### 83.3 Local compactness

**definition**

IsLocallyComp (_{is locally-compact} 70)
where T{is locally-compact}≡T(∀B. AT. B{is compact in}T)

We center ourselves in local compactness, because it is a very important tool in topological groups and compactifications.

If a subset is compact of some cardinal for a topological space, it is compact of the same cardinal in the subspace topology.

**lemma compact_imp_compact_subspace:**
assumes A{is compact of cardinal}K{in}T A⊆B
shows A{is compact of cardinal}K{in}(T{restricted to}B) unfolding IsCompactOfCard_def

**proof**
from assms show C:Card(K) unfolding IsCompactOfCard_def by auto
from assms have A⊆⋃ T unfolding IsCompactOfCard_def by auto
then have AA:A⊆⋃ (T{restricted to}B) using assms(2) unfolding RestrictedTo_def by auto
moreover
{ fix M assume M∈Pow(T{restricted to}B) A⊆M
let M={S∈T. B∩S∈M}
}
from \( \{ M \in \text{Pow}(T \{ \text{restricted to} B \}) \} \) have \( \bigcup M \subseteq \bigcup M \) unfolding RestrictedTo_def by auto

with \( \{ A \subseteq M \} \) have \( A \subseteq \bigcup M \subseteq \text{Pow}(T) \) by auto

with assms have \( \exists N \in \text{Pow}(M) \). \( A \subseteq \bigcup N \land N \prec K \) unfolding IsCompactOfCard_def by auto

then obtain \( N \) where \( N \in \text{Pow}(M) \) \( A \subseteq \bigcup N \) by auto

then have \( N \{ \text{restricted to} \} B \subseteq M \) unfolding RestrictedTo_def FinPow_def by auto

moreover

let \( f = \{ (B, B \cap B) \} _{B \in \mathbb{N}} \)

have \( f : N \to (N \{ \text{restricted to} \} B) \) unfolding Pi_def function_def domain_def RestrictedTo_def by auto

then have \( f \in \text{surj}(N, N \{ \text{restricted to} \} B) \) unfolding surj_def RestrictedTo_def using apply_equality by auto

from \( N \prec K \) have \( N \lessprec K \) unfolding lesspoll_def by auto

with \( f \in \text{surj}(N, N \{ \text{restricted to} \} B) \) have \( N \{ \text{restricted to} \} B \lessprec N \) using surj_fun_inv_2 Card_is_Ord C by auto

moreover

from \( A \subseteq \bigcup M \) have \( A \subseteq \bigcup (N \{ \text{restricted to} \} B) \) using assms(2) unfolding RestrictedTo_def by auto

ultimately have \( \exists N \in \text{Pow}(M) \). \( A \subseteq \bigcup N \land N \prec K \) by auto

\}

with \( A \subseteq \bigcup (T \{ \text{restricted to} \} B) \land (\forall M \in \text{Pow}(T \{ \text{restricted to} \} B). \ A \subseteq \bigcup M \longrightarrow (\exists N \in \text{Pow}(M) \). \ A \subseteq \bigcup N \land N \prec K) \) by auto

qed

The converse of the previous result is not always true. For compactness, it holds because the axiom of finite choice always holds.

lemma compact_subspace_imp_compact:

assumes \( A \{ \text{is compact in} \} (T \{ \text{restricted to} \} B) \ A \subseteq B \)

shows \( A \{ \text{is compact in} \} T \) unfolding IsCompact_def

proof

from assms show \( A \subseteq \bigcup T \) unfolding IsCompact_def RestrictedTo_def by auto

next

\{

fix \( M \) assume \( M \in \text{Pow}(T) \). \( A \subseteq M \)

let \( M = (M \{ \text{restricted to} \} B) \)

from \( M \in \text{Pow}(T) \) have \( M \in \text{Pow}(T \{ \text{restricted to} \} B) \) unfolding RestrictedTo_def by auto

from \( A \subseteq M \) have \( A \subseteq \bigcup M \) unfolding RestrictedTo_def using assms(2) by auto

by auto

with assms \( M \in \text{Pow}(T \{ \text{restricted to} \} B) \) obtain \( N \) where \( N \in \text{FinPow}(M) \)

\( A \subseteq N \) unfolding IsCompact_def by blast

from \( N \in \text{FinPow}(M) \) have \( N \prec \text{nat} \) unfolding FinPow_def Finite_def using n_lesspoll_nat eq_lesspoll_trans

by auto

then have \( \text{Finite}(N) \) using lesspoll_nat_is_Finite by auto

\}

1158
then obtain \( n \) where \( n \in \mathbb{N} \) unfolding Finite_def by auto
then have \( N \leq n \) using eqpoll_imp_lepoll by auto
moreover
\[
\begin{align*}
& \text{fix } BB \text{ assume } BB \in N \\
& \text{with } \{ N \in \text{FinPow}(M) \} \text{ have } BB \in M \text{ unfolding FinPow_def by auto} \\
& \text{then obtain } S \text{ where } S \in M \text{ and } BB = B \cap S \text{ unfolding RestrictedTo_def by auto}
\end{align*}
\]
then have \( S \subseteq \{ S \in M. B \cap S = BB \} \) by auto
then obtain \( \{ S \in M. B \cap S = BB \} \neq 0 \) by auto
moreover
\[
\begin{align*}
& \text{then have } \forall BB \in N. ((\forall \in \mathbb{N}. \{ S \in M. B \cap S = W \}) BB) \neq 0 \text{ by auto moreover}
& \text{from } \forall n \in \mathbb{N}. \text{ have } (N \leq n \land (\forall t \in N. (\forall \in \mathbb{N}. \{ S \in M. B \cap S = W \}) t) \neq 0) \\
& \text{using finite_choice unfolding AxiomCardinalChoiceGen_def by blast}
\end{align*}
\]
ultimately
obtain \( f \) where AA: \( f : \Pi(N, \lambda t. (\forall \in \mathbb{N}. \{ S \in M. B \cap S = W \}) t) \forall t \in N. ft \in (\forall \in \mathbb{N}. \{ S \in M. B \cap S = W \}) t \) by blast
from AA(2) have ss: \( \forall t \in N. ft \in \{ S \in M. B \cap S = t \} \) using beta_if by auto
then have \( \{ ft. t \in N \} \subseteq M \) by auto
fix t assume t \( \in N \\
with ss have \( ft \in \{ S \in M. B \cap S \in N \} \) by auto
\]
with AA(1) have FF: \( f : N \rightarrow \{ S \in M. B \cap S \in N \} \) unfolding Pi_def Sigma_def using beta_if by auto moreover
\[
\begin{align*}
& \text{fix } aa \ bb \text{ assume } AAA : aa \in N \ bb \in N \ faa = fbb \\
& \text{from AAA(1) ss have } B \cap (faa) = aa \text{ by auto} \\
& \text{with AAA(3) have } B \cap (fbb) = aa \text{ by auto} \\
& \text{with ss AAA(2) have } aa = bb \text{ by auto}
\end{align*}
\]
ultimately have \( f \in \text{inj}(N, \{ S \in M. B \cap S \in N \}) \) unfolding inj_def by auto
then have \( f \in \text{bij}(N, \text{range}(f)) \) using inj_bij_range by auto
then have \( f \in \text{bij}(N, fN) \) using range_image_domain FF by auto
then have \( f \in \text{bij}(N, \{ ft. t \in N \}) \) using func_image_def FF by auto
then have \( N \cong \{ ft. t \in N \} \) unfolding eqpoll_def by auto
with \( \{ N \cong n \} \) have \( ft. t \in N \) using eqpoll_sym eqpoll_trans by blast
with \( \forall n \in \mathbb{N}. \text{ have } \text{Finite}(\{ ft. t \in N \}) \) unfolding Finite_def by auto
with ss have \( \{ ft. t \in N \} \in \text{FinPow}(M) \) unfolding FinPow_def by auto moreover
\[
\begin{align*}
& \text{fix } aa \text{ assume } aa \in A \\
& \text{with } \{ A \subseteq N \} \text{ obtain } b \text{ where } b \in N \text{ and } aa \in b \text{ by auto} \\
& \text{with ss have } B \cap (fb) = b \text{ by auto} \\
& \text{with } \{ aa \in b \} \text{ have } aa \in B \cap (fb) \text{ by auto} \\
& \text{then have } aa \in fb \text{ by auto} \\
& \text{with } \{ b \in N \} \text{ have } aa \in \bigcup \{ ft. t \in N \} \text{ by auto}
\end{align*}
\]
then have $A \subseteq \bigcup \{ ft. t \in \mathbb{N} \}$ by auto ultimately
have $\exists R \in \text{FinPow}(M). A \subseteq R$ by auto

then show $\forall M \in \text{Pow}(T). A \subseteq \bigcup M \rightarrow (\exists N \in \text{FinPow}(M). A \subseteq \bigcup N)$ by auto

qed

If the axiom of choice holds for some cardinal, then we can drop the compact sets of that cardinal are compact of the same cardinal as subspaces of every superspace.

**Lemma** $\text{Kcompact_subspace_impl_Kcompact}:
  
  assumes $A$ (is compact of cardinal) $Q$ (in) $(T \text{(restricted to)} B) \ A \subseteq B$ ({the axiom of} $Q$ {choice holds})
  
  shows $A$ (is compact of cardinal) $Q$ (in) $T$
  
  proof -
  from assms(1) have $a1$ : $\text{Card}(Q)$ unfolding $\text{IsCompactOfCard_def RestrictedTo_def}$
  by auto
  
  from assms(1) have $a2$ : $A \subseteq \bigcup T$
  unfolding $\text{IsCompactOfCard_def RestrictedTo_def}$
  by auto

  { fix $M$ assume $M \in \text{Pow}(T) \ A \subseteq \bigcup M$
  
  let $M = M \text{(restricted to)} B$
  
  from $\langle M \in \text{Pow}(T) \rangle$ have $M \in \text{Pow}(T \text{restricted to} B)$ unfolding $\text{RestrictedTo_def}$
  by auto
  
  from $\langle A \subseteq \bigcup M \rangle$ have $A \subseteq \bigcup M$
  unfolding $\text{RestrictedTo_def using assms(2)}$
  by auto

  with assms $\langle M \in \text{Pow}(T \text{restricted to} B) \rangle$ obtain $N$ where $N : N \in \text{Pow}(M)$
  
  $A \subseteq \bigcup N \ N \prec Q$ unfolding $\text{IsCompactOfCard_def by blast}$

  from $N(3)$ have $N \prec Q$ using $\text{lesspoll_imp_lepoll by auto moreover}$

  { fix $BB$ assume $BB \in N$
  
  with $\langle N \in \text{Pow}(M) \rangle$ have $BB \in M$ unfolding $\text{FinPow_def by auto}$
  
  then obtain $S$ where $S : M$ and $BB = B \cap S$ unfolding $\text{RestrictedTo_def}$
  by auto

  then have $S : \{ S : M. B \cap S = BB \}$ by auto
  
  then obtain $\{ S : M. B \cap S = BB \} \neq 0$ by auto

  } then have $\forall BB \in N. ((\forall w \in N. \{ S \in M. B \cap S = w \}) BB) \neq 0$ by auto moreover

  have $(N \subseteq Q \wedge (\forall t \in N. (\forall w \in N. \{ S \in M. B \cap S = w \}) t \neq 0)) \rightarrow (\exists f. f \in \text{Pi}(N, \lambda t. (\forall w \in N. \{ S \in M. B \cap S = w \}) t) \wedge (\forall t \in N. f t \in (\forall w \in N. \{ S \in M. B \cap S = w \}) t)))$

  using assms(3) unfolding $\text{AxiomCardinalChoiceGen_def by blast}$
  ultimately

  obtain $f$ where $AA : f : \text{Pi}(N, \lambda t. (\forall w \in N. \{ S \in M. B \cap S = w \}) t) \forall t \in N. f t \in (\forall w \in N. \{ S \in M. B \cap S = w \}) t)$

  by blast

  from $AA(2)$ have $as : \forall t \in N. f t \in \{ S : M. B \cap S = t \}$ using $\text{beta_if by auto}$

  then have $\{ ft. t \in N \} \subseteq M$ by auto

  { fix $t$ assume $t \in N$

  1160
with \( ss \) have \( ft \in \{ S \in M. B \cap S \in N \} \) by auto

} with \( AA(1) \) have \( FF : f : N \rightarrow \{ S \in M. B \cap S \in N \} \) unfolding \( \text{Pi_def} \) \( \text{Sigma_def} \) using \( \text{beta_if} \) by auto moreover

\[
\begin{align*}
\text{fix} \ aa \ bb \ & \ \text{assume} \ AA : aa \in N \ bb \in N \ faa = fbb \\
\text{from} \ AA(1) \ & \ \text{ss have} \ B \cap (faa) = aa \ by \ auto \\
\text{with} \ AA(3) \ & \ \text{have} \ B \cap (fbb) = aa \ by \ auto \\
\text{with} \ ss \ AA(2) \ & \ \text{have} \ aa = bb \ by \ auto
\end{align*}
\]

ultimately have \( f \in \text{inj}(N, \{ S \in M. B \cap S \in N \}) \) unfolding \( \text{inj_def} \) by auto

then have \( f \in \text{bij}(N, \text{range}(f)) \) using \( \text{inj_bij_range} \) by auto

then have \( f \in \text{bij}(N, \{ \text{ft.} \ t \in N \}) \) using \( \text{func_imagedef} \) \( FF \) by auto

then have \( N = \{ \text{ft.} \ t \in N \} \) unfolding \( \text{eqpoll_def} \) by auto

moreover

\[
\begin{align*}
\text{fix} \ aa \ & \ \text{assume} \ aa \in A \\
\text{with} \ AA : A \subseteq \bigcup N \ & \ \text{obtain} \ b \ where \ b \in N \ and \ aa \in b \ by \ auto \\
\text{with} \ ss \ & \ \text{have} \ B \cap (fb) = b \ by \ auto \\
\text{with} \ ss \ & \ \text{have} \ B \cap (faa) = aa \ by \ auto \\
\text{with} \ ss \ AA(2) \ & \ \text{have} \ aa = bb \ by \ auto
\end{align*}
\]

then have \( A \subseteq \bigcup \{ \text{ft.} \ t \in N \} \) by auto ultimately have \( \exists R \in \text{Pow}(M). A \subseteq R \land R \prec Q \) by auto

then show thesis using \( a1 \) \( a2 \) unfolding \( \text{IsCompactOfCard_def} \) by auto

qed

Every set, with the cofinite topology is compact.

lemma cofinite_compact:

shows \( X \{ \text{is compact in}\}(\text{CoFinite} \ X) \) unfolding \( \text{IsCompact_def} \)

proof

show \( X \subseteq \bigcup (\text{CoFinite} \ X) \) using \( \text{union_cocardinal} \) unfolding \( \text{Cofinite_def} \)

next

\[
\begin{align*}
\text{fix} \ M \ & \ \text{assume} \ M \in \text{Pow}(\text{CoFinite} \ X) \ X \subseteq M \\
\text{assume} \ M = 0 \lor M = \{ 0 \} \\
\text{then have} \ M \in \text{FinPow}(M) \ & \ \text{unfolding} \ \text{FinPow_def} \ by \ auto \\
\text{with} \ & \ \text{have} \ \exists N \in \text{FinPow}(M). X \subseteq N \ by \ auto
\end{align*}
\]

moreover

\[
\begin{align*}
\end{align*}
\]
assume \( M \neq \emptyset \neq \{0\} \)
then obtain \( U \) where \( U \in MU \neq 0 \) by auto
with \( \{ M \in \text{Pow}(\text{CoFinite} X) \} \) have \( U \in \text{CoFinite} X \) by auto
with \( \{ U \neq 0 \} \) have \( U \subseteq X \) (\( X-U \)-nat unfolding) \( \text{Cofinite_def} \) \( \text{CoCardinal_def} \) by auto
then have \( \text{Finite}(X-U) \) using \( \text{lesspoll_nat_is_Finite} \) by auto
then have \( (X-U) \{ \text{is in the spectrum of} \} \) \( \{ \bigcup \} \{ \text{is compact in} \} T \) using \( \text{compact_spectrum} \)
by auto
then have \( ((X-U) \approx X-U) \rightarrow ((X-U) \{ \text{is compact in} \} \{ X-U \}) \) unfolding \( \text{Spec_def} \) using \( \text{InfCard_nat} \) \( \text{CoCar_is_topology} \) unfolding \( \text{Cofinite_def} \) by auto
then have \( \text{com:}(X-U) \{ \text{is compact in} \} \{ X-U \} \) unfolding \( \text{union_cocardinal} \) using \( \text{InfCard_nat} \) \( \text{CoCar_is_topology} \) unfolding \( \text{Cofinite_def} \) by auto
then obtain \( N \) where \( N \subseteq M \) \( \text{Finite}(N) \) \( X-U \subseteq \bigcup N \) unfolding \( \text{FinPow_def} \) by auto
with \( \{ U \in M \} \) have \( N \cup \{ U \} \subseteq M \) \( \text{Finite}(N \cup \{ U \}) \) (\( X \subseteq \bigcup \{ N \cup \{ U \} \} \) by auto
then have \( \exists N \in \text{FinPow}(M) . X \subseteq \bigcup \{ N \} \) unfolding \( \text{FinPow_def} \) by blast
ultimately have \( \exists N \in \text{FinPow}(M) . X \subseteq \bigcup N \) by auto
}
then show \( \forall M \in \text{Pow}(\text{CoFinite} X) . X \subseteq M \rightarrow (\exists N \in \text{FinPow}(M) . X \subseteq \bigcup N) \) by auto
qed

A corollary is then that the cofinite topology is locally compact; since every subspace of a cofinite space is cofinite.

corollary cofinite_locally_compact:
shows \( \{ \text{CoFinite} X \} \{ \text{is locally-compact} \} \)
proof-
have \( \text{cof:topology0} \{ \text{CoFinite} X \} \) and \( \text{cof1:} \{ \text{CoFinite} X \} \{ \text{is a topology} \} \)
using \( \text{CoCar_is_topology} \) \( \text{InfCard_nat} \) \( \text{CoFinite_def} \) unfolding \( \text{topology0_def} \) by auto
\{
fix \( x \) \( B \) assume \( x \subseteq \bigcup \{ \text{CoFinite} X \} B \subseteq \{ \text{CoFinite} X \} \) \( x \in B \)
then have \( x \in \text{Interior}(B,\text{CoFinite} X) \) using \( \text{topology0.Top_2_L3[of cof]} \) by auto
moreover
\}

1162
from \( B \in (\text{CoFinite } X) \) have \( B \subseteq X \) unfolding Cofinite_def CoCardinal_def by auto
then have \( B \cap X = B \) by auto
then have \( (\text{CoFinite } X) \{\text{restricted to}\} B = \text{CoFinite } B \) using subspace_cocardinal unfolding Cofinite_def by auto
then have \( B \{\text{is compact in}\} (\text{CoFinite } X) \{\text{restricted to}\} B \) using cofinite_compact union_cocardinal unfolding Cofinite_def by auto
then have \( B \{\text{is compact in}\} (\text{CoFinite } X) \) using compact_subspace_imp_compact by auto
ultimately have \( \exists c \in \text{Pow}(B). x \in \text{Interior}(c, \text{CoFinite } X) \wedge c \{\text{is compact in}\} (\text{CoFinite } X) \) by auto
then have \( (\forall x \in \bigcup (\text{CoFinite } X). \forall b \in (\text{CoFinite } X). x \in b \rightarrow (\exists c \in \text{Pow}(b). x \in \text{Interior}(c, \text{CoFinite } X) \wedge c \{\text{is compact in}\} (\text{CoFinite } X)) \) by auto
then show thesis unfolding IsLocallyComp_def IsLocally_def[OF cof1] by auto
qed

In every locally compact space, by definition, every point has a compact neighbourhood.

theorem (in topology0) locally_compact_exist_compact_neig:
assumes \( T \{\text{is locally-compact}\} \)
shows \( \forall x \in \bigcup T. \exists A \in \text{Pow}(\bigcup T). A \{\text{is compact in}\} T \wedge x \in \text{int}(A) \)
proof-
{ fix \( x \) assume \( x \in \bigcup T \) moreover
then have \( \bigcup T \neq \emptyset \) by auto
have \( \bigcup T \in T \) using union_open topSpaceAssum by auto
ultimately have \( \exists c \in \text{Pow}(\bigcup T). x \in \text{int}(c) \wedge c \{\text{is compact in}\} T \) using assms IsLocally_def topSpaceAssum unfolding IsLocallyComp_def by auto
then have \( \exists c \in \text{Pow}(\bigcup T). c \{\text{is compact in}\} T \wedge x \in \text{int}(c) \) by auto
}
then show thesis by auto
qed

In Hausdorff spaces, the previous result is an equivalence.

theorem (in topology0) exist_compact_neig_T2_imp_locally_compact:
assumes \( \forall x \in \bigcup T. \exists A \in \text{Pow}(\bigcup T). x \in \text{int}(A) \wedge A \{\text{is compact in}\} T \{\text{T}_2\} \)
sows \( T \{\text{is locally-compact}\} \)
proof-
{ fix \( x \) assume \( x \in \bigcup T \) with assms(1) obtain \( A \) where \( A \in \text{Pow}(\bigcup T) \wedge x \in \text{int}(A) \) and Acom: \( A \{\text{is compact in}\} T \) by blast
then have \( A \cup A \{\text{is closed in}\} T \) using in_t2_compact_is_cl assms(2) by auto
then have sub: \( A \subseteq T \) unfolding IsClosed_def by auto
}
fix U assume U∈T x∈U
let V=Int(A∩U)
from x∈U x∈Int(A) have x∈U∩(Int(A)) by auto
moreover from U∈T have U∩(Int(A))∈T using Top_2_L2 topSpaceAssum
unfolding IsATopology_def
by auto moreover
have U∩(Int(A))⊆A∩U using Top_2_L1 by auto
ultimately have x∈V using Top_2_L5 by blast
have V⊆A using Top_2_L4 by auto
then have cl(V)⊆A using Acl Top_3_L13 by auto
moreover have U∩cl(V)=cl(V) by auto moreover
have clcl:cl(V) is closed in T using cl_is_closed(1) \langle V⊆A \rangle \langle A⊆U T \rangle
by auto
ultimately have comp:cl(V) is compact in T using Acom compact_closed[of AnatTc1(V)] Compact_is_card_nat
by auto
{ then have cl(V) is compact in \langle \langle T{restricted to}cl(V) \rangle \langle T{restricted to}cl(V) \rangle \rangle
using compact_imp_compact_subspace unfolding cl(V)natT] Compact_is_card_nat
by auto moreover
have \bigcup(T{restricted to}cl(V))=cl(V) unfolding RestrictedTo_def
using clcl unfolding IsClosed_def by auto moreover
ultimately have \bigcup(T{restricted to}cl(V)) is compact in \langle \langle T{restricted to}cl(V) \rangle \langle T{restricted to}cl(V) \rangle \rangle by auto
} then have \bigcup(T{restricted to}cl(V)) is compact in \langle \langle T{restricted to}cl(V) \rangle \langle T{restricted to}cl(V) \rangle \rangle
by auto moreover
have \langle T{restricted to}cl(V)\rangle is T_2 using asms(2) T2_here clcl
unfolding IsClosed_def by auto
ultimately have \langle T{restricted to}cl(V)\rangle is T_2 using topology0.T2_compact_is_normal
by auto
then have clvreg: \langle T{restricted to}cl(V)\rangle is regular using topology0.T4_is_T3
by auto
have V⊆cl(V) using cl_contains_set \langle V⊆A \rangle \langle A⊆U T \rangle by auto
then have V∈\langle T{restricted to}cl(V)\rangle unfolding RestrictedTo_def
by auto
using Top_2_L4
then have clvreg: \langle T{restricted to}cl(V)\rangle is regular using topology0.T4_is_T3
by auto
have V⊆cl(V) using cl_contains_set \langle V⊆A \rangle \langle A⊆U T \rangle by auto
then have V∈\langle T{restricted to}cl(V)\rangle unfolding RestrictedTo_def
by auto
using Top_2_L4
with \langle x∈V \rangle obtain W where Wop: W∈\langle T{restricted to}cl(V)\rangle and clcont: Closure(W, \langle T{restricted to}cl(V)\rangle) ⊆ V and cinW: x∈W
using topology0.regular_imp_exist_clos_neig unfolding topology0_def
using Top_1_L4 clvreg
by blast
from clcont Wop have W∈V using topology0.cl_contains_set unfolding topology0_def using Top_1_L4 by auto
with Wop have W∈\langle T{restricted to}cl(V)\rangle \langle restricted to \rangle V unfolding RestrictedTo_def by auto
moreover from \langle V⊆A \rangle \langle A⊆U T \rangle have V⊆U T by auto
then have $V \subseteq \text{cl}(V) \subseteq \bigcup T$ using $\langle V \subseteq \text{cl}(V) \rangle$ Top_3_L11(1) by auto
then have $(T|_{\text{cl}(V)}) \subseteq V$ using subspace_of_subspace by auto
ultimately have $W \subseteq (T|_{\text{cl}(V)})$ by auto
then obtain $UU$ where $UU \subseteq U \cap V$ unfolding RestrictedTo_def by auto
then have $W \in T$ using Top_2_L2 topSpaceAssum unfolding IsATopology_def by auto
moreover
have $W \subseteq \text{cl}(W) \subseteq \bigcup T$ unfolding topology0_def
using Top_1_L4 Wop by auto
ultimately have $A1: x \in \text{int}(\text{cl}(W), (T|_{\text{cl}(V)}))$ using Top_2_L6 cinW by auto
from clcont have $A2: \text{Closure}(W, (T|_{\text{cl}(V)})) \subseteq U$ using Top_2_L1 by auto
ultimately have $W \subseteq \text{cl}(V)$ using topology0.cl_contains_set unfolding topology0_def
by auto
from comp have $\text{cl}(V) \subseteq T$ using compact_imp_compact_subspace[of $\text{cl}(V)\cap T$] Compact_is_card_nat
by auto
with $\text{cl}(V)$ have $(\text{cl}(V) \cap \text{cl}(W), (T|_{\text{cl}(V)}))$ using compact_closed Compact_is_card_nat by auto
moreover
from clcont have $\text{cont}: (\text{cl}(W), (T|_{\text{cl}(V)})) \subseteq \text{cl}(V)$ using cl_contains_set $\langle V \subseteq A \rangle \langle A \subseteq \bigcup T \rangle$
by blast
then have $(\text{cl}(V) \cap \text{cl}(W), (T|_{\text{cl}(V)})) = \text{cl}(W, (T|_{\text{cl}(V)}))$ by auto
ultimately have $\text{Closure}(W, (T|_{\text{cl}(V)})) \subseteq U$ using compact_subspace_imp_compact[of $\text{Closure}(W, (T|_{\text{cl}(V)}))]$
cont by auto
with $A1$ $A2$ have $\exists c \in \text{Pow}(U). \ x \in \text{int}(c) \land c \subseteq \text{cl}(V)$ by auto
then have $\forall U \in T. \ x \in U \longrightarrow (\exists c \in \text{Pow}(U). \ x \in \text{int}(c) \land c \subseteq \text{cl}(V))$
ultimately have $\text{cl}(V)$ by auto
then show thesis unfolding IsLocally_def[OF topSpaceAssum] IsLocallyComp_def by auto
qed

83.4 Compactification by one point

Given a topological space, we can always add one point to the space and get a new compact topology; as we will check in this section.

definition
OPCompactification ({one-point compactification of}_ 90)
where {one-point compactification of}T≡T∪({⋃T}∪((⋃T)-K). K∈{B∈Pow(⋃T). B{is compact in}T ∧ B{is closed in}T})

Firstly, we check that what we defined is indeed a topology.

**Theorem (in topology0) op_comp_is_top:**

shows ({one-point compactification of}T) is a topology unfolding IsATopology_def

**Proof (safe)**

fix M assume M≤{one-point compactification of}T
then have disj:M⊆T∪(⋃T)-K. K∈{B∈Pow(⋃T). B{is compact in}T ∧ B{is closed in}T}) unfolding OPCompactification_def by auto

let MT={A∈M. A∈T}

have MT⊆T by auto

then have c1:⋃MT∈T using topSpaceAssum unfolding IsATopology_def by auto

let MK={A∈M. A/∈T}

have ⋃M=⋃MK∪⋃MT by auto

moreover have N:⋃T/∈(⋃T) using mem_not_refl by auto

fix B assume B∈{B∈Pow(⋃T). B{is compact in}T ∧ B{is closed in}T}) by auto

moreover have N:⋃T/∈(⋃T) using mem_not_refl by auto

assume MK=0

then have ⋃M=⋃MT by auto

then have ⋃M∈T using c1 by auto

then have ⋃M≤{one-point compactification of}T unfolding OPCompactification_def by auto

moreover

assume MK≠0

then obtain A where A∈MK by auto

then obtain K1 where A=(⋃T)∪((⋃T)-K1) K1∈Pow(⋃T) K1{is compact in}T K1{is closed in}T using MK_def by auto

with <A∈MK> have ⋂KK≤K1 by auto

from <A∈MK> <A=(⋃T)∪((⋃T)-K1) <K1∈Pow(⋃T)> have KK≠0 by blast

1166
fix K assume K∈KK
then have (⋃T)∪((⋃T)-K)∈MK K∈⋃T by auto
then obtain K K where A: (⋃T)∪((⋃T)-K) = (⋃T)∪((⋃T)-K) K∈⋃T

K∈is compact in)T K∈is closed in)T using MK_def by auto

note A(1) moreover
have (⋃T)-K∈(⋃T)∪((⋃T)-K) (⋃T)-K∈(⋃T)∪((⋃T)-K) by auto
ultimately have (⋃T)-K∈(⋃T)∪((⋃T)-K) (⋃T)-K∈(⋃T)∪((⋃T)-K)
by auto moreover

from N have ⋃T∉(⋃T)-K ⋃T∉(⋃T)-KK by auto ultimately
have (⋃T)-K∈(⋃T)-KK (⋃T)-KK∈((⋃T)-K) by auto
then have (⋃T)-K= (⋃T)-KK by auto moreover
from ⋃T K have K= (⋃T)-((⋃T)-K) by auto ultimately
have K= (⋃T)-((⋃T)-KK) by auto
with ⋃T K have K=KK by auto
with A(4) have K∈is compact in)T by auto

then have ∀K∈KK. K∈is compact in)T by auto
with ⋃T K have ((KK)∈is closed in)T using Top_3.L4 by auto
with ⋃T K have K∈is compact in)T have (K1∩(⋃T))∈is compact in)T using
Compact_is_card_nat

compact_closed[of K1nat∩KK] by auto moreover
from ⋃KK∈K1 have K1∩(⋃KK)=(⋃KK) by auto ultimately
have (⋃KK)∈is compact in)T by auto
with ⋃KK K1 have (⋃KK)∈is closed in)T T have (K1∈Pow(⋃T)) have ((⋃T)∪((⋃T)-⋃KK))∈(one
compactification of)T

unfolding OPCompactification_def by blast
have t: ⋃MK=⋃{A∈M. A∈(⋃T)∪((⋃T)-K). K∈B∈Pow(⋃T). B∈is compact
in)T} ∪ B∈(is compact in)T})
using MK_def by auto

{ fix x assume x∈⋃MK
  with t have x∈⋃{A∈M. A∈(⋃T)∪((⋃T)-K). K∈B∈Pow(⋃T). B∈is compact
  in)T} ∪ B∈(is closed in)T}) by auto
  then have ∃A∈{A∈M. A∈(⋃T)∪((⋃T)-K)}. K∈B∈Pow(⋃T). B∈is compact
  in)T} ∪ B∈(is closed in)T})}. x∈AA
  using Union_iff by auto
  then obtain AA where AA= ⋃A∈{A∈M. A∈(⋃T)∪((⋃T)-K)}. K∈B∈Pow(⋃T). B∈is compact
  in)T} ∪ B∈(is closed in)T}) x∈AA by auto
  then obtain K2 where AA= (⋃T)∪((⋃T)-K2) K2∈Pow(⋃T)K2∈is compact
  in)T} K2∈(is closed in)T by auto
  with x∈AA have x=⋃T ∨ (x∈(⋃T) x∉K2) by auto
  from K2∈Pow(⋃T). A∈ (⋃T)∪((⋃T)-K2) AA= (⋃T)∪((⋃T)-K2) AA(1) MK_def have K2∈KK
  by auto
  then have ⋃KK∈K2 by auto
  with x∈(⋃T) x∉K2 have x=⋃T ∨ (x∈(⋃T) x∉K2) by auto
  then have x∈(⋃T)∪((⋃T)-(⋃KK)) by auto
}
then have ⋃MK∈(⋃T)∪((⋃T)-(⋃KK)) by auto

1167
moreover
{  fix x assume x∈(UT)∪((UT)-(∩KK))
     then have x∈UTV(x∈(UT)∧x∉(∩KK)) by auto
     with ⟨KK≠0⟩ obtain K2 where K2∈KK x∈UTV(x∈UT∧x∉K2) by auto
     then have {UT}∪((UT)-(K2))∈MK by auto
     with ⟨x=UTV(x∈UT∧x∉K2)⟩ have x∈MK by auto
}
then have {UT}∪((UT)-(∩KK))⊆MK by (safe,auto)
ultimately have ∪MK={UT}∪((UT)-(∩KK)) by blast
from ⟨U_MT∈T⟩ have UT-(UT-∪MT)=∪MT by auto
with ⟨∪MT∈T⟩ have (UT-∪MT){is closed in}T unfolding IsClosed_def
by auto
have ((UT)-(∩KK))∪(UT-(UT-∪MT))=(UT)-((∩KK)∩(UT-∪MT)) by auto
then have ((UT)∪((UT)-(∩KK)))∪(UT-((UT-∪MT))=(UT)∪((UT)-(∩KK)∩(UT-∪MT)))
by auto
with ⟨∪MK={UT}∪((UT)-(∩KK))⟩ have UT-∪MK=UT∪(UT-∪MT) by auto
by auto
with ⟨∪M=∪MK U∈∪MT⟩ have unm:M={UT}∪((UT)-(∩KK)∩(UT-∪MT))
by auto
have ((∩KK)∩(UT-∪MT)) {is closed in}T using (∩KK){is closed in}T>>> UT-{∪MT}){is closed in}T>
   Top_3_L5 by auto
moreover
note ⟨UT-{∪MT}){is closed in}T⟩ ⟨∩KK}{is compact in}T⟩
then have ((∩KK)∩(UT-∪MT)) {is compact of cardinal nat}(in)T unfolding compact_closed[of ∩KK nat T UT-UMT] Compact_is_card_nat
by auto
then have ((∩KK)∩(UT-∪MT)) {is compact in}T using Compact_is_card_nat
by auto
ultimately have {UT}∪(UT-((∩KK)∩(UT-∪MT)))∈{one-point compactification of}T
unfolding OPCompactification_def IsClosed_def by auto
with unm have ∪M∈{one-point compactification of}T by auto
}
ultimately show ∪M∈{one-point compactification of}T by auto
next
fix U V assume U∈{one-point compactification of}T and V∈{one-point compactification of}T
then have A:U∈TV(∃KU∈Pow(UT). U={UT}∪(UT-KU)∧KU{is closed in}T)∧KU{is compact in}T)
V∈TV(∃KV∈Pow(UT). V={UT}∪(UT-KV)∧KV{is closed in}T)∧KV{is compact in}T)
unfolding OPCompactification_def by auto
have N:UT∉(UT) using mem_not_refl by auto
{  assume U∈TV∈T
  then have UNW∈T using Top_3_L5 unfolding IsATopology_def by
auto
then have $U \cap V \in \{\text{one-point compactification of}\} T$ unfolding OPCompactification_def by auto
}
moreover
{
assume $U \in T \notin T$
then obtain $K_V$ where $V : K_V$ is closed in $T_K$ is compact in $T_V$
using A(2) by auto
with $N \langle U \in T \rangle$ have $U \notin V$ by auto
then have $U \notin V$ by auto
then have $U \cap V = \bigcup (U-K_V)$ using V(3) by auto
moreover have $U \cap V \in T$ unfolding IsClosed_def by auto
with $\langle U \in T \rangle$ have $U \cap (U-K_V) \in T$ using topSpaceAssum unfolding IsATopology_def by auto
with $\langle U \forall \cap (U-K_V) \rangle$ have $U \forall V$ by auto
then have $U \forall V \in \{\text{one-point compactification of}\} T$ unfolding OPCompactification_def by auto
}
moreover
{
assume $U \forall T \notin T$
then obtain $K_V$ where $V : K_V$ is closed in $T_K$ is compact in $T_V$
using A(1) by auto
with $V(3) U(3)$ have $U \forall V$ by auto
then have $U \forall V \in \{\text{one-point compactification of}\} T$ unfolding OPCompactification_def by auto
}
moreover
{
assume $U \forall T \notin T$
then obtain $K_V$ where $V : K_V$ is closed in $T_K$ is compact in $T_V$
and $U : K_U$ is closed in $T_K$ is compact in $T_U$
using A by auto
with $V(3) U(3)$ have $U \forall \cap V$ by auto
then have $U \forall = (U \forall \cup (U-K_V) \cap (U-K_U))$ using V(3) U(3) by auto
moreover have $U \forall \cap (U-K_U) \in T$ using V(1) U(1) unfolding IsClosed_def by auto
by auto
then have $(U-K_V) \cap (U-K_U) \in T$ unfolding topSpaceAssum unfolding IsATopology_def by auto
then have $(U-K_V) = (U \forall - (U-K_V) \cap (U-K_U))$ by auto
moreover with $\langle U \forall - (U-K_V) \cap (U-K_U) \rangle$ have $(U \forall - (U-K_V) \cap (U-K_U)) \in \{\text{closed}\}$
in $T$ unfolding IsClosed_def by auto moreover from V(1) U(1) have $(\bigcup T-(\bigcup T-KV)\cap(\bigcup T-KU))=$KV\cup KU unfolding IsClosed_def by auto

with V(2) U(2) have $(\bigcup T-(\bigcup T-KV)\cap(\bigcup T-KU))$ is compact in $T$ using union_compact[of KVnatTKU] Compact_is_card_nat InfCard_nat by auto ultimately have $U\cap V\in\{\text{one-point compactification of}\}T$ unfolding OPCompactification_def by auto

ultimately have $U\cap V\in\{\text{one-point compactification of}\}T$ unfolding OPCompactification_def RestrictedTo_def by auto

ultimately show $U\cap V\in\{\text{one-point compactification of}\}T$ by auto qed

The original topology is an open subspace of the new topology.

theorem (in topology0) open_subspace:
  shows $\bigcup T\in\{\text{one-point compactification of}\}T$ and $(\{\text{one-point compactification of}\}T)\{\text{restricted to}\}\bigcup T=T$

proof - have 0 is compact in $T$ unfolding IsCompact_def FinPow_def by auto moreover note Top_3_L2 ultimately have $TT:0\in\{A\in Pow(\bigcup T). A\{\text{is compact in}\}T\}=$KV\cup KU unfolding IsCompact_def FinPow_def by auto

moreover note $TT:0\in\{A\in Pow(\bigcup T). A\{\text{is compact in}\}T\}\{\text{restricted to}\}\bigcup T=T$

also have ...=$(\bigcup T\cup(\bigcup T-K))$ K\in\{B\in Pow(\bigcup T). B\{\text{is compact in}\}T\} unfolding OPCompactification_def by blast

also have ...=$(\bigcup T\cup(\bigcup T-K))$ K\in\{B\in Pow(\bigcup T). B\{\text{is compact in}\}T\} by blast

We added only one new point to the space.

lemma (in topology0) op_compact_total:
  shows $\bigcup(\{\text{one-point compactification of}\}T)\{\text{restricted to}\}\bigcup T=T$ by auto

qed
The one point compactification, gives indeed a compact topological space.

**Theorem (in topology0) compact_op:**

shows \((\bigcup T) \cup (\bigcup T)\)\{is compact in\}\{one-point compactification of\}T\) by auto

**Proof:**

have \(0\{is compact in\}T\) unfolding IsCompact_def FinPow_def by auto

moreover note Top_3_L2 ultimately have \(0 \in \{A \in \Pow(\bigcup T). A\{is compact in\}T\) \& \(A\{is closed in\}T\) by auto

then have \((\bigcup T) \cup (\bigcup T)\)\{one-point compactification of\}T\) unfolding OPCompactification_def by auto

then show \(\bigcup T \in \bigcup\{one-point compactification of\}T\) by auto

next

fix \(x \in B\) assume \(x \in BB \in T\)

then show \(x \in \bigcup\{one-point compactification of\}T\) using open_subspace by auto

next

fix \(M\) assume \(A:M \subseteq \{one-point compactification of\}T\) \{\(\bigcup T\) \& \(\bigcup T \subseteq \bigcup M\)

then obtain \(R\) where \(R \in \bigcup T\) using mem_not_refl by auto

with \(R \in M\) \(\bigcup T \subseteq R\) obtain \(K\) where \(K:R = \{\bigcup T\} \cup (\bigcup T - K)\) \{is compact in\}\{one-point compactification of\}T\) is closed in\}T\)

unfolding OPCompactification_def by auto

from \(K(1,2)\) have \(B:\{\bigcup T\} \cup (\bigcup T) = R \cup K\) unfolding IsCompact_def by auto

with \(A(2)\) have \(K \subseteq \bigcup M\) by auto

from \(K(2)\) have \(K\{is compact in\}\{one-point compactification of\}T\) restrict to\}\bigcup T\) using open_subspace(2)

by auto

then have \(K\{is compact in\}\{one-point compactification of\}T\) using compact_subspace_imp_compact

\(<K\{is closed in\}T\) unfolding IsClosed_def by auto

with \(<K \subseteq \bigcup M\) \(A(1)\) have \(\exists N \in \FinPow(M). K \subseteq \bigcup N\) unfolding IsCompact_def by auto

then obtain \(N\) where \(N \in \FinPow(M)\) \(K \subseteq \bigcup N\) by auto

with \(<R \in N\) have \(N \cup (R) \in \FinPow(M)\) by 

by auto

with \(B\) show \(\exists N \in \FinPow(M). \{\bigcup T\} \cup (\bigcup T) \subseteq \bigcup N\) by auto

qed

The one point compactification is Hausdorff iff the original space is also Hausdorff and locally compact.

**Lemma (in topology0) op_compact_T2_1:**

assumes \(\{one-point compactification of\}T\)\{is T2\}

shows \(T\{is T2\}

1171
lemma (in topology0) op_compact_T2_2:
  assumes \(\{\text{one-point compactification of} \}T\)\{is T\}_2
  shows \(T\)\{is locally-compact\}
doctorate{proof}{
  \{ \fix \(x\) assume \(x \in \bigcup T\)
  \let\text\text
  \text then have \(x \in (\bigcup T) \cup (\bigcup T)\) by auto
  \let\text\text
  \text moreover have \(\bigcup T \in (\bigcup T) \cup (\bigcup T)\) by auto moreover
  \text from \(\langle x \in T \rangle\) have \(x \neq \bigcup T\) using \text{mem_not_refl} by auto
  \text ultimately have \(\exists U \in \{\text{one-point compactification of}\}T. \exists V \in \{\text{one-point compactification of}\}T. x \in U \land (\bigcup T) \in V \land U \cap V = 0\)
  \text using assms \text{op_compact_total} unfolding \text{isT2_def} by auto
  \text then obtain \(U, V\) where \(U \in \{\text{one-point compactification of}\}T\) \(V \in \{\text{one-point compactification of}\}T\)
  \text \text{using} \text{op_compact_total} unfolding \text{RestrictedTo_def} by auto
  \text ultimately have \(U \in T\) using \text{open_subspace(2)} by auto
  \text with \(\langle U \neq 0 \rangle\) \(K\) have \(U \subseteq K \subseteq \bigcup T\) using \text{IsClosed_def} by auto
  \text then have \(U \cap (\bigcup T) \in (\{\text{one-point compactification of}\}T)\{\text{restricted to}\} (\bigcup T)\)
  \text using \text{op_compact_total} by auto
  \text ultimately have \(U \subseteq T\) using \text{open_subspace(2)} by auto
  \text with \(\langle x \in U \rangle\) \(x \in \bigcup T\) using \text{Top_2_L6} by auto
  \text with \(\langle V \subseteq K \rangle\) have \(x \in \text{int}(K)\) using \text{Top_2_L6} by auto
  \text from \(UV(1)\) have \((\langle \bigcup T \rangle \subseteq \bigcup T)\) \((\{\text{one-point compactification of}\}T)\{\text{restricted to}\} (\bigcup T)\)
  \text unfolding \text{RestrictedTo_def} by auto
  \text then have \(\forall x \in \bigcup T. \exists A \in \text{Pow}(\bigcup T). x \in \text{int}(A) \land A\{\text{is compact in}\}T\) by auto
  \text then show thesis using \text{op_compact_T2_1[OF assms] \text{exist_compact_neig_T2_imp_locally_compact}} by auto
\qed
with <x≠y> have ∃U∈T. ∃V∈T. x∈U∧y∈V∧V=0 using assms(2) unfolding isT2_def by auto
then have ∃U∈({one-point compactification of}T). ∃V∈({one-point compactification of}T). x∈U∧y∈V∧V=0 unfolding OPCompactification_def by auto

moreover

{ assume x∉∪T∀y∉∪T with S have x∉∪T∀y∉∪T by auto with <x≠y> have (x∉∪T∧y∉∪T)∨(y∉∪T∧x∉∪T) by auto with S have (x∉∪T∧y∉∪T)∨(y∉∪T∧x∉∪T) by auto then obtain Ky Kx where (x∉∪T ∧ Ky(is compact in)T∧y∈int(Ky))∨(y∉∪T ∧ Kx(is compact in)T∧x∈int(Kx)) using in_t2_compact_is_cl assms(2) auto

moreover have (∪T∪(∪T-K))∩int(K)=0 by auto ultimately have (∪T∪(∪T-K))∩int(K)=0 by auto then have (∪T∪(∪T-K))∩int(K)=0 using Top_2_L1 by auto moreover from A have (∪T∪(∪T-K))∈({one-point compactification of}T)
unfolding OPCompactification_def
IsClosed_def by auto moreover have int(K)∈({one-point compactification of}T) using Top_2_L2 unfolding OPCompactification_def by auto ultimately have int(K)∈({one-point compactification of}T)∧(∪T∪(∪T-K))∩int(K)=0 by auto }

ultimately have (∪T)∪(∪T - Ky)∈({one-point compactification of}T)∧x∈(∪T)∪(∪T - Ky)∧y∈int(Ky)∧(∂(∪T)∪(∪T - Kx))∩int(Kx)=0 by auto moreover

{ assume (∪T)∪(∪T - Ky)∈({one-point compactification of}T)∧int(Ky)∈({one-point compactification of}T)
compactification of $T) \lambda x \in \{\bigcup T \cup (\bigcup (T - Ky)) \cap \text{int}(Ky) = 0\}
then have $\exists U \in \{\text{(one-point compactification of) } T\}. \exists V \in \{\text{(one-point compactification of) } T\}. x \in U \land y \in V \land U \cap V = 0$ using exI[of _ int(Ky)],of
$\lambda U \ V. U \in \{\text{(one-point compactification of) } T\} \land x \in U \land y \in V \land U \cap V = 0 \{\bigcup T \cup (\bigcup T - Ky)\}$
by auto

ultimately have $\exists U \in \{\text{(one-point compactification of) } T\}. \exists V \in \{\text{(one-point compactification of) } T\}. x \in U \land y \in V \land U \cap V = 0$ by blast

ultimately have $\exists U \in \{\text{(one-point compactification of) } T\}. \exists V \in \{\text{(one-point compactification of) } T\}. x \in U \land y \in V \land U \cap V = 0$ by auto

then show thesis unfolding isT2_def by auto

qed

In conclusion, every locally compact Hausdorff topological space is regular; since this property is hereditary.

corollary (in topology0) locally_compact_T2_imp_regular:
assumes $T \{\text{is locally-compact}\} T \{\text{is T}_2\}$
shows $T \{\text{is regular}\}$

proof-
from assms have ( $\{\text{(one-point compactification of) } T\} \{\text{is T}_2\}$ using op_compact_T2_3
by auto
then have ( $\{\text{(one-point compactification of) } T\} \{\text{is T}_1\}$ unfolding isT4_def
using T2_is_T1 topology0.T2_compact_is_normal
op_comp_is_top unfolding topology0_def using op_compact_total compact_op
by auto
then have ( $\{\text{(one-point compactification of) } T\} \{\text{is T}_3\}$ using topology0.T4_is_T3
op_comp_is_top unfolding topology0_def
by auto
then have ( $\{\text{(one-point compactification of) } T\} \{\text{is regular}\}$ using isT3_def
by auto moreover
have $\bigcup T \subseteq \bigcup (\{\text{(one-point compactification of) } T\} \{\text{is regular}\}$ using op_compact_total
by auto
ultimately have ( $\{\text{(one-point compactification of) } T\} \{\text{restricted to}\} \bigcup T$ \{is regular\} using regular_here by auto
then show $T \{\text{is regular}\}$ using open_subspace(2) by auto

qed
This last corollary has an explanation: In Hausdorff spaces, compact sets are closed and regular spaces are exactly the "locally closed spaces" (those which have a neighbourhood basis of closed sets). So the neighbourhood basis of compact sets also works as the neighbourhood basis of closed sets we needed to find.

definition
IsLocallyClosed (_is locally-closed) where T is locally-closed ≡ T is locally (λ B T. B is closed in T T)

lemma (in topology0) regular_locally_closed: shows T is regular ←→ (T is locally-closed)
proof
  assume T is regular
  then have a: ∀ x ∈ ∪ T. ∀ U ∈ T. (x ∈ U) → (∃ V ∈ T. x ∈ V ∧ cl(V) ⊆ U)
  unfolding regular_imp_exist_clos_neig by auto
  { fix x b assume x ∈ ∪ T b ∈ T with a obtain V where V ∈ T x ∈ V cl(V) ⊆ b by blast
    note cl(V) ⊆ b moreover
    from ∀ V ∈ T have ∀ V ∈ T by auto
    then have V ⊆ cl(V) using cl_contains_set by auto
    with x ∈ V ∀ V ∈ T have x ∈ int(cl(V)) using Top_2_L6 by auto moreover
    from ∀ V ∈ T have cl(V) is closed in T using cl_is_closed by auto
    ultimately have x ∈ int(cl(V)) cl(V) ⊆ cl(V) is closed in T by auto
    then have ∃ K ∈ Pow(b). x ∈ int(K) ∧ K is closed in T by auto
  }
  then have ∀ x ∈ ∪ T. ∀ b ∈ T. x ∈ b → (∃ V ∈ T. x ∈ V ∧ cl(V) ⊆ b)
  unfolding IsLocally_def[OF topSpaceAssum] by auto
next
  assume T is locally-closed
  then have a: ∀ x ∈ ∪ T. ∀ b ∈ T. x ∈ b → (∃ K ∈ Pow(b). x ∈ int(K) ∧ K is closed in T)
  unfolding IsLocally_def[OF topSpaceAssum] by auto
  { fix x b assume x ∈ ∪ T b ∈ T with a obtain K where K ⊆ b x ∈ int(K) is closed in T by blast
    have int(K) ⊆ K using Top_2_L1 by auto
    with K(3) have cl(int(K)) ⊆ K using Top_3_L13 by auto
    with K(1) have cl(int(K)) ⊆ b by auto moreover
    have int(K) ∈ T using Top_2_L2 by auto moreover
    note x ∈ int(K) ultimately have ∃ V ∈ T. x ∈ V ∧ cl(V) ⊆ b by auto
  }
  then have ∀ x ∈ ∪ T. ∀ b ∈ T. x ∈ b → (∃ V ∈ T. x ∈ V ∧ cl(V) ⊆ b) by auto
  then show T is regular using exist_clos_neig_imp_regular by auto
qed
83.5 Hereditary properties and local properties

In this section, we prove a relation between a property and its local property for hereditary properties. Then we apply it to locally-Hausdorff or locally-$T_2$.

We also prove the relation between locally-$T_2$ and another property that appeared when considering anti-properties, the anti-hyperconnectness.

If a property is hereditary in open sets, then local properties are equivalent to find just one open neighbourhood with that property instead of a whole local basis.

lemma (in topology0) her_P_is_loc_P:
assumes \( \forall TT. \forall B \in \text{Pow}(\bigcup TT). \forall A \in TT. TT \text{(is a topology)} \land P(B,TT) \rightarrow P(B \cap A,TT) \)
shows \( (T \text{(is locally)}P) \iff (\forall x \in \bigcup TT. \exists A \in T. x \in A \land P(A,T)) \)

proof
assume A:T\{is locally\}P
{ fix x assume x:x\in\bigcup T
  with A have \( \forall b \in T. x \in b \rightarrow (\exists c \in \text{Pow}(b). x \in \text{int}(c) \land P(c,T)) \) unfolding IsLocally_def[OF topSpaceAssum]
  by auto moreover
note x moreover 
  have \( \bigcup T \in T \) using topSpaceAssum unfolding IsATopology_def by auto
  ultimately have \( \exists c \in \text{Pow}(\bigcup T). x \in \text{int}(c) \land P(c,T) \) by auto
  then obtain c where c_def:A \in TX \land P(c,T) by auto
  have P:int(c)\in T using Top_2_L2 by auto moreover
  from c(1,3) topSpaceAssum asms have \( \forall A \in T. P(c \cap A,T) \) by auto
  ultimately have P(c\cap int(c),T) by auto moreover
  from Top_2_L1[of c] have int(c)\subseteq c by auto
  then have c\cap int(c)=int(c) by auto
  ultimately have P(int(c),T) by auto
  with P(c(2)) have \( \exists V \in T. x \in V \land P(V,T) \) by auto
}
then show \( \forall x \in \bigcup T. \exists V \in T. x \in V \land P(V,T) \) by auto
next
assume A: \( \forall x \in \bigcup T. \exists A \in T. x \in A \land P(A, T) \)
{ fix x assume x:x\in\bigcup T
  { fix b assume b:x\in b\in T
    from x A obtain A where A_def:A=Tx \land P(A,T) by auto
    from A_def(1,3) asms topSpaceAssum have \( \forall G \in T. P(A \cap G,T) \) by auto
    with b(2) have P(A\cap b,T) by auto
    moreover from b(1) A_def(2) have x\in A\cap b by auto moreover
    have A\cap b\in T using b(2) A_def(1) topSpaceAssum IsATopology_def by auto
    then have int(A\cap b)=A\cap b using Top_2_L3 by auto
    ultimately have x\in int(A\cap b) \land P(A\cap b, T) by auto
    then have \( \exists c \in \text{Pow}(b). x \in \text{int}(c) \land P(c, T) \) by auto
  }
}
then have $\forall b \in T. x \in b \rightarrow (\exists c \in \text{Pow}(b). x \in \text{int}(c) \wedge P(c, T))$ by auto

then show $T \{\text{is locally}P\}$ unfolding IsLocally_def[OF topSpaceAssum] by auto

qed

definition IsLocallyT2 (_\{is locally-T\_2\} 70)
  where $T \{\text{is locally-T}_2\} \equiv T \{\text{is locally}\}(\lambda B. \lambda T. (T \{\text{restricted to}B\}) \{\text{is T}_2\})$

Since $T_2$ is an hereditary property, we can apply the previous lemma.

corollary (in topology0) loc_T2:
  shows $T \{\text{is locally-T}_2\} \iff (\forall x \in \bigcup T. \exists A \in T. x \in A \wedge (T \{\text{restricted to}A\}) \{\text{is T}_2\})$

proof -
  { fix $TT$ $B$ $A$ assume $TT$: $TT \{\text{is a topology}\} (TT \{\text{restricted to}B\}) \{\text{is T}_2\}$ $A \in TT \{\text{Pow}(\bigcup TT)\}$
    then have $s: B \cap A \subseteq BB \subseteq \bigcup TT$ by auto
    then have $(TT \{\text{restricted to}B\}) \{\text{restricted to}B \cap A\}$ unfolding RestrictedTo_def using $s(2)$ by auto
    moreover have $\bigcup (TT \{\text{restricted to}B\}) = B$ unfolding RestrictedTo_def using $s(1)$ by auto
    moreover note $TT(2)$ ultimately have $(TT \{\text{restricted to}B\}) \{\text{is T}_2\}$ using T2_here by auto
  }
  then have $\forall TT. \forall B \in \text{Pow}(\bigcup TT). \forall A \in TT. TT \{\text{is a topology}\} \wedge (TT \{\text{restricted to}B\}) \{\text{is T}_2\} \rightarrow (TT \{\text{restricted to}B \cap A\}) \{\text{is T}_2\}$
    by auto
    with her_P_is_loc_P[where $P=\lambda A. \lambda TT. (TT \{\text{restricted to}A\}) \{\text{is T}_2\}$] show thesis unfolding IsLocallyT2_def by auto

qed

First, we prove that a locally-$T_2$ space is anti-hyperconnected.

Before starting, let’s prove that an open subspace of an hyperconnected space is hyperconnected.

lemma (in topology0) open_subspace_hyperconn:
  assumes $T \{\text{is hyperconnected}\} U \in T$
  shows $(T \{\text{restricted to}U\}) \{\text{is hyperconnected}\}$

proof -
  { fix $A$ $B$ assume $Ac(T \{\text{restricted to}U\})B \in (T \{\text{restricted to}U\}) \cap B = 0$ then obtain $A \cup B$ where $A = U \cap (B \cup A) = B \cup BU \subseteq TBU \subseteq T$ unfolding RestrictedTo_def by auto

1177
then have $A \in T \cap B \in T$ using topSpaceAssum assms(2) unfolding IsATopology_def by auto
with $\langle A \cap B = 0 \rangle$ have $A = 0 \lor B = 0$ using assms(1) unfolding IsHConnected_def by auto
}
then show thesis unfolding IsHConnected_def by auto
qed

lemma (in topology0) locally_T2_is_antiHConn:
  assumes $T \{\text{is locally-T}_2\}$
  shows $T \{\text{is anti-}\text{IsHConnected}\}$
proof -
{ fix $A$ assume $A \in \text{Pow}(U \cup T \{\text{restricted to} A\}) \{\text{is hyperconnected}\}$
{ fix $x$ assume $x \in A$
  with $A(1)$ have $x \in U \cup T$ by auto moreover
  have $U \cup T \subseteq T$ using topSpaceAssum unfolding IsATopology_def by auto
ultimately
  have $\exists c \in \text{Pow}(U \cup T). x \in \text{int}(c) \land (T \{\text{restricted to} c\} \{\text{is T}_2\})$ using assms unfolding IsLocallyT2_def IsLocally_def [OF topSpaceAssum] by auto
  then obtain $c$ where $c \in \text{Pow}(U \cup T)x \in \text{int}(c)(T \{\text{restricted to} c\} \{\text{is T}_2\})$ by auto
  have $\bigcup (T \{\text{restricted to} c\} \{T \{\text{restricted to} c\} \{\text{is T}_2\}\})$ by auto
  with $\langle c \in \text{Pow}(U \cup T) \rangle \langle U \cup T \rangle$ have tot:$\bigcup (T \{\text{restricted to} c\} = c$ by auto
  have $\text{int}(c) \subseteq T$ using Top_2_L2 by auto
  then have $A \cap (\text{int}(c)) \subseteq (T \{\text{restricted to} A\})$ unfolding RestrictedTo_def by auto
  with $A(2)$ have $((T \{\text{restricted to} A\}) \{\text{restricted to} (A \cap (\text{int}(c)))\} \{\text{is hyperconnected}\}$ using topology0.open_subspace_hyperconn unfolding topology0_def using Top_1_L4
  then have $\text{sub}_{A \cap (\text{int}(c))} \{\text{is hyperconnected}\}$ using subspace_of_subspace[of $A \cap (\text{int}(c))$] AT $A(1)$ by force moreover
  have $\text{int}(c) \subseteq c$ using Top_2_L1 by auto
  then have sub:$A \cap (\text{int}(c)) \subseteq c$ by auto
  then have $A \cap (\text{int}(c)) \subseteq (T \{\text{restricted to} c\}$ using tot by auto
  then have $((T \{\text{restricted to} c\} \{\text{restricted to} (A \cap (\text{int}(c)))\}) \{\text{is T}_2\}$ using T2_here[OF c(3)] by auto
  with $\langle c \in \text{Pow}(U \cup T) \rangle$ have $\langle c \in \text{Pow}(U \cup T) \rangle$ by auto
  ultimately have $((T \{\text{restricted to} c\} \{\text{restricted to} (A \cap (\text{int}(c)))\}) \{\text{is hyperconnected}\}$
}
{\text{restricted to}}(A \cap (\text{int}(c)))\{\text{is } T_2}\}

by auto

then have \((T\{\text{restricted to}}(A \cap (\text{int}(c)))\{\text{is hyperconnected}\}(T\{\text{restricted to}}(A \cap (\text{int}(c)))\{\text{is anti-} I\}_{\text{Connected}}\)

using topology0.\text{T2_imp_anti_HConn unfolding topology0_def using Top_1_L4 by auto}

moreover

have \(\bigcup(T\{\text{restricted to}}(A \cap (\text{int}(c))))=(\bigcup T) \cap A \cap (\text{int}(c))\) unfolding RestrictedTo_def

by auto

then have \(A \cap (\text{int}(c)) \subseteq \bigcup (T\{\text{restricted to}}(A \cap (\text{int}(c))))\) by auto

moreover

have \(A \cap (\text{int}(c)) \subseteq \bigcup T\) using A(1) Top_2_L2 by auto

then have \(T\{\text{restricted to}}(A \cap (\text{int}(c))))\{\text{restricted to}}(A \cap (\text{int}(c)))=(T\{\text{restricted to}}(A \cap (\text{int}(c))))\)

using subspace_of_subspace[of \(A \cap (\text{int}(c))\)A \cap (\text{int}(c))T] by auto

ultimately have \((A \cap (\text{int}(c)))\{\text{is in the spectrum of} I_{\text{Connected}}\})\) unfolding antiProperty_def

by auto

then have \(A \cap (\text{int}(c)) \subseteq 1\) using \(H_{\text{Connected}}\) by auto

then have \((A \cap (\text{int}(c))=\{x\})\) using \(\text{lepoll_1_is_sing} \langle x \in A \rangle \langle x \in \text{int}(c) \rangle\) by auto

then have \((x) \in (T\{\text{restricted to}}A)\) using \(\langle A \cap (\text{int}(c)) \in (T\{\text{restricted to}}A) \rangle\) by auto

then have \(\text{pointOpen} : \forall x \in A. \{x\} \in (T\{\text{restricted to}}A)\) by auto

{ fix \(x y\) assume \(x \neq y \in A y \in A\)

with \(\text{pointOpen}\) have \(\{x\} \in (T\{\text{restricted to}}A)\{y\} \in (T\{\text{restricted to}}A)\{x\} \cap \{y\}=0\)

by auto

with A(2) have \(\{x\}=0 \lor \{y\}=0\) unfolding \(I_{\text{Connected}}\) by auto

then have \(\text{False}\) by auto

} then have \(\text{uni} : \forall x \in A. \forall y \in A. x=y\) by auto

{ assume \(A \neq 0\)
 then obtain \(x\) where \(x \in A\) by auto
 with \(\text{uni}\) have \(A=\{x\}\) by auto
 then have \(A \approx 1\) using \(\text{singleton_eqpoll_1}\) by auto
 then have \(A \leq 1\) using \(\text{eqpoll_imp_lepoll}\) by auto
 }

moreover

{ assume \(A=0\)
 then have \(A \approx 0\) by auto
 then have \(A \leq 1\) using \(\text{empty_lepoll1 eq_lepoll_trans}\) by auto
 }

ultimately have \(A \leq 1\) by auto
then have $A$ (is in the spectrum of) $\text{IsHConnected}$ using $\text{HConn_spectrum}$ by auto

} then show thesis unfolding antiProperty_def by auto

qed

Now we find a counter-example for: Every anti-hyperconnected space is locally-Hausdorff.

The example we are going to consider is the following. Put in $X$ an anti-hyperconnected topology, where an infinite number of points don’t have finite sets as neighbourhoods. Then add a new point to the set, $p \notin X$. Consider the open sets on $X \cup p$ as the anti-hyperconnected topology and the open sets that contain $p$ are $p \cup A$ where $X \setminus A$ is finite.

This construction equals the one-point compactification iff $X$ is anti-compact; i.e., the only compact sets are the finite ones. In general this topology is contained in the one-point compactification topology, making it compact too.

It is easy to check that any open set containing $p$ meets infinite other nonempty open set. The question is if such a topology exists.

**Theorem (in topology0) COF_comp_is_top:**

assumes $T$ (is $T_1$) $\neg (\bigcup \mathcal{T} \prec \text{nat})$

shows $((\{\text{one-point compactification of}(\text{CoFinite } (\bigcup \mathcal{T})))\setminus \{\bigcup \mathcal{T}\}) \cup \mathcal{T})$

{is a topology}

**Proof**

- have $N : \bigcup \mathcal{T} \notin (\bigcup \mathcal{T})$ using mem_not_refl by auto
  
  { fix $M$ assume $M : M \subseteq ((\{\text{one-point compactification of}(\text{CoFinite } (\bigcup \mathcal{T})))\setminus \{\bigcup \mathcal{T}\}) \cup \mathcal{T}$
    let $MT = \{A \in M. A \in \mathcal{T}\}$
    let $MK = \{A \in M. A \notin \mathcal{T}\}$
    have $MM : (\bigcup MT) \cup (\bigcup MK) = \bigcup M$ by auto
    have $MN : \bigcup MT \subseteq \mathcal{T}$ unfolding $\text{topSpaceAssum}$ using IsATopology_def by auto
    then have $sub : MK \subseteq (\{\text{one-point compactification of}(\text{CoFinite } (\bigcup \mathcal{T})))\setminus \{\bigcup \mathcal{T}\})$
      using $N$ by auto
    then have $MK \subseteq ((\{\text{one-point compactification of}(\text{CoFinite } (\bigcup \mathcal{T})))$ by auto
    then have $CO : \bigcup MK \subseteq ((\{\text{one-point compactification of}(\text{CoFinite } (\bigcup \mathcal{T})))$
      using topology0.op_comp_is_top[OF topology0_CoCardinal[OF InfCard_nat]] unfolding $\text{Cofinite_def}$
      $\text{IsATopology_def}$ by auto
    
    assume $AS : \bigcup MK = \{\bigcup \mathcal{T}\}$
    moreover have $\forall R \in MK. R \subseteq \bigcup MK$ by auto
    ultimately have $\forall R \in MK. R \subseteq \{\bigcup \mathcal{T}\}$ by auto
    then have $\forall R \in MK. R = (\{\bigcup \mathcal{T}\}) \setminus R = 0$ by force
    moreover have $\forall R \in MK. R = 0$ by auto

1180
then have $\bigcup MK = 0$ by auto
  with $\text{AS}$ have False by auto 
\}
with $\text{CO}$ have $\text{CO2}: \bigcup MK \in \{\text{one-point compactification of} (\text{CoFinite} (\bigcup T)) \} - \{\bigcup T\}$ by auto 
\{ 
  assume $\bigcup MK \in (\text{CoFinite} (\bigcup T))$
  then have $\bigcup MK \in T$ using assms(1) $T1\text{cocardinal coarser}$ by auto 
  with $\text{MW}$ have $\bigcup MT, \bigcup MK \subseteq T$ by auto 
  then have $(\bigcup MT) \cup (\bigcup MK) \in T$ using union_open[OF topSpaceAssum, of $(\bigcup MT, \bigcup MK)$] by auto 
\}
moreover 
\{ 
  assume $\bigcup MK /\in \in (\text{CoFinite} (\bigcup T))$
  with $\text{CO}$ obtain $B$ where $B$ is compact in $(\text{CoFinite} (\bigcup T))$ $B$ is closed in $(\text{CoFinite} (\bigcup T))$
  $\bigcup MK = (\bigcup \text{CoFinite} (\bigcup T)) \cup (\bigcup (\text{CoFinite} (\bigcup T) - B))$ unfolding $\text{OpCompactification_def}$ by auto 
  then have $MK : (\bigcup MK = (\bigcup T) \cup (\bigcup T - B) \text{ is closed in } (\text{CoFinite} (\bigcup T)))$
  using union_cocardinal unfolding $\text{Cofinite_def}$ by auto 
\}
moreover 
\{ 
  assume $B = \bigcup T$
  with $\text{B}$ have $\bigcup MK = \{\bigcup T\}$ by auto 
  then have False using $\text{CO2}$ by auto 
\}
with $B$ have $B \subseteq \bigcup T$ and nat$B : \text{nat}$ by auto 
  have $(\bigcup T - (\bigcup MT)) \cap B \subseteq B$ by auto 
  then have $(\bigcup T - (\bigcup MT)) \cap B \subseteq B$ using subset_imp_lepoll by auto 
  then have $(\bigcup T - (\bigcup MT)) \cap B \text{ is closed in } (\text{CoFinite} (\bigcup T))$ using closed_sets_cocardinal unfolding $\text{Cofinite_def}$ by auto 
\{ 
  assume $B = \bigcup T$
  with $\text{MK}$ have $\bigcup MK = \{\bigcup T\}$ by auto 
  then have False using $\text{CO2}$ by auto 
\}
with $B$ have $B \subseteq \bigcup T$ and nat$B : \text{nat}$ by auto 
  have $(\bigcup T - (\bigcup MT)) \cap B \subseteq B$ by auto 
  then have $(\bigcup T - (\bigcup MT)) \cap B \subseteq B$ using subset_imp_lepoll by auto 
  then have $(\bigcup T - (\bigcup MT)) \cap B \text{ is closed in } (\text{CoFinite} (\bigcup T))$ using closed_sets_cocardinal unfolding $\text{Cofinite_def}$ by auto 

1181
by auto
  then have \((\bigcup T - ((\bigcup MT) \cup (\bigcup T-B)))\) is compact in \((\text{CoFinite } \bigcup T)\) using cofinite_compact
    union_cocardinal unfolding Cofinite_def by auto
  then have \((\bigcup T - ((\bigcup MT) \cup (\bigcup T-B)))\) is compact in \((\text{CoFinite } \bigcup T)\) using compact_subspace_imp_compact by auto
  ultimately have \((\bigcup T) \cup (\bigcup T - ((\bigcup MT) \cup (\bigcup T-B)))\) is compact in \((\text{CoFinite } (\bigcup T))\)
    unfolding OPCompactification_def using union_cocardinal unfolding Cofinite_def by auto
  ultimately have \((\bigcup T) \cup (\bigcup T - ((\bigcup MT) \cup (\bigcup T-B)))\) is compact in \((\text{CoFinite } (\bigcup T))\)
    by auto
  ultimately have \((\bigcup MT) \cup (\bigcup MK)\) is compact in \((\text{CoFinite } (\bigcup T))\)
    by auto
  assume AS: \((\bigcup MT) \cup (\bigcup MK) = \{\bigcup T\}\)
  from MN have T: \(\bigcup T \notin \bigcup MT\) using N by auto
  { fix x assume G: \(x \in \bigcup MT\)
    then have \(x \in (\bigcup MT) \cup (\bigcup MK)\) by auto
      with AS have \(x \in \{\bigcup T\}\) by auto
        then have \(x = \bigcup T\) by auto
          with T have False using G by auto
  }
  then have \(\bigcup MT = 0\) by auto
    with AS have \(\{\bigcup MK\} = \{\bigcup T\}\) by auto
      then have False using CD2 by auto
  with AA2 have \((\bigcup MT) \cup (\bigcup MK)\) is compact in \((\text{CoFinite } (\bigcup T))\) - \(\{\bigcup T\}\)
    by auto
  with MM have \(\bigcup M \in \{\text{one-point compactification of } (\text{CoFinite } (\bigcup T))\} - \{\bigcup T\}\)
    by auto
  ultimately have \(\bigcup M \in \{\text{one-point compactification of } (\text{CoFinite } (\bigcup T))\} - \{\bigcup T\}\) \(\bigcup T\)
    by auto
  then have \(\forall M \in \text{Pow}(\{\text{one-point compactification of } (\text{CoFinite } (\bigcup T))\} - \{\bigcup T\}) \cdot \bigcup M \in \{\text{one-point compactification of } (\text{CoFinite } (\bigcup T))\} - \{\bigcup T\}\) \(\bigcup T\)
    by auto
    moreover
  { fix U V assume U \(\in\) \(\{\text{one-point compactification of } (\text{CoFinite } (\bigcup T))\} - \{\bigcup T\}\) \(\bigcup TV \in \{\text{one-point compactification of } (\text{CoFinite } (\bigcup T))\} - \{\bigcup T\}\)
      moreover
    { assume U \(\in\) \(\bigcup TV\)
      then have \(\forall M \in \text{topSpaceAssum}\) unfolding IsATopology_def by auto
  } 1182
then have \( U \cap V \in \{\text{one-point compactification of} \ (\bigcup T)\} - \{\{\bigcup T\}\} \cup \bigcup T \)
by auto

moreover

\{
  \begin{align*}
  \text{assume } UV &\in \{\text{one-point compactification of} \ (\bigcup T)\} - \{\{\bigcup T\}\} \cup \{\text{one-point compactification of} \ (\bigcup T)\} \\
  \text{then have } 0 &\neq UV \in \{\text{one-point compactification of} \ (\bigcup T)\} \\
  \text{using } &\text{topology0.op_comp_is_top[OF topology0_CoCardinal[OF InfCard_nat]}} \\
  \text{unfolding } &\text{Cofinite_def} \\
  \text{IsATopology_def by auto} \\
  \text{then have } &\bigcup T \cap (U \cap V) \in \{\text{one-point compactification of} \ (\bigcup T)\} \\
  \text{using } &\text{topology0.open_subspace(2)[OF topology0_CoCardinal[OF InfCard_nat]]} \\
  \text{union_cocardinal unfolding } &\text{Cofinite_def by auto} \\
  \text{from } &\text{UV have } U \notin \{\{\bigcup T\}\} \cup \{\text{one-point compactification of} \ (\bigcup T)\} \\
  \text{of} &\{\text{CoFinite} \ (\bigcup T)\} \{\text{restricted to} \} \bigcup T \bigcup \{\{\bigcup T\}\} \\
  \text{unfolding } &\text{RestrictedTo_def by auto} \\
  \text{then have } &R(U \notin \{\bigcup T\} \cup \{\text{one-point compactification of} \ (\bigcup T)\} \cup \{\text{CoFinite} \ (\bigcup T)\} \cup (\bigcup T) \cup \{\{\bigcup T\}\} \cup \bigcup T) \\
  \text{using } &\text{topology0.open_subspace(2)[OF topology0_CoCardinal[OF InfCard_nat]]} \\
  \text{union_cocardinal unfolding } &\text{Cofinite_def by auto} \\
  \text{from } &\text{UV have } U \subseteq \bigcup T \{\text{one-point compactification of} \ (\bigcup T)\} \cup \{\text{CoFinite} \ (\bigcup T)\} \cup (\bigcup T) \cup \{\{\bigcup T\}\} \cup \bigcup T) \\
  \text{using } &\text{topology0.op_compact_total[OF topology0_CoCardinal[OF InfCard_nat]]} \\
  \text{union_cocardinal unfolding } &\text{Cofinite_def by auto} \\
  \text{then have } &E:U=(\bigcup T \cap V) \cup (\bigcup T \cap V) \cup (\{\{\bigcup T\}\} \cup \bigcup T) \\
  \text{by auto} \\
  \end{align*}
\}

\{
  \begin{align*}
  \text{assume } Q &\neq \bigcup T \\
  \text{then have } RR &U \cap V \cap (\bigcup T) = 0 \text{ using N by auto} \\
  \end{align*}
\}

\{
  \begin{align*}
  \text{assume } Q &\neq \bigcup T \\
  \text{with } E(1) &U=\{\{\bigcup T\}\} \cap U \text{ by auto} \\
  \text{also have } &\ldots \subseteq \{\{\bigcup T\}\} \text{ by auto} \\
  \text{ultimately have } &U \subseteq \{\{\bigcup T\}\} \text{ by auto} \\
  \text{then have } U &= 0 \text{ by auto} \\
  \text{with } R(1) &U \subseteq \{\{\bigcup T\}\} \text{ by auto} \\
  \text{then have } U \cap V &\neq 0 \text{ by auto} \\
  \text{then have } &\text{False using Q by auto} \\
  \end{align*}
\}

moreover

\{
  \begin{align*}
  \text{assume } &U \cap V = 0 \\
  \text{with } E(2) &V=\{\bigcup T\} \cap V \text{ by auto} \\
  \text{also have } &\ldots \subseteq \{\{\bigcup T\}\} \text{ by auto} \\
  \end{align*}
\}

1183
ultimately have $V \subseteq \{ \bigcup T \}$ by auto
then have $V = 0 \vee V = \{ \bigcup T \}$ by auto
with R(2) have $V = 0$ by auto
then have $U \cap V = 0$ by auto
then have False using Q by auto
}

moreover
{
assume $\bigcup T \cap U \neq 0 \bigcup T \cap V \neq 0$
with R(3,4) have $(\bigcup T \cap U) \cap (\bigcup T \cap V) \neq 0$ using Cofinite nat_HConn[OF assms(2)]
unfolding IsHConnected_def by auto
then have $\bigcup T \cap (U \cap V) \neq 0$ by auto
then have False using RR by auto
}
ultimately have False by auto
}

with $0$ have $U \cap V \in (\{\text{one-point compactification of } (\text{CoFinite } (\bigcup T))) - \{\bigcup T\}) \cup T$
by auto

moreover
{
assume $U \in T \in \{\text{one-point compactification of } (\text{CoFinite } (\bigcup T))) - \{\bigcup T\}$
from $UV(2)$ obtain $B$ where $V \in (\text{CoFinite } (\bigcup T)) \vee (V = \{ \bigcup T \} \cup (\bigcup T - B)) \land B$ is closed in $(\text{CoFinite } (\bigcup T)))$ unfolding OPCompactification_def
using T1_cocardinal_coarser by auto
then have $V \in (\bigcup T \cap U) \cap (\bigcup T - B) \land B$ is closed in $(\text{CoFinite } (\bigcup T))$
using UV(1) N by auto
then have $V \in (\bigcup T \cap U) \cap (\bigcup T - B) \land (\bigcup T - B) \in (\text{CoFinite } (\bigcup T)))$ unfolding IsClosed_def using union_cocardinal unfolding Cofinite_def by auto
then have $V \in (\bigcup T \cap U) \cap (\bigcup T - B) \land (\bigcup T - B) \in T$ using assms(1) T1_cocardinal_coarser by auto
with $UV(1)$ have $U \cap V \in T$ using topSpaceAssum unfolding IsATopology_def
by auto
then have $U \cap V \in (\{\text{one-point compactification of } (\text{CoFinite } (\bigcup T))) - \{\bigcup T\}) \cup T$
by auto
}

moreover
{
assume $U \in \{\text{one-point compactification of } (\text{CoFinite } (\bigcup T))) - \{\bigcup T\}$
from $UV(1)$ obtain $B$ where $U \in (\text{CoFinite } (\bigcup T)) \vee (U = \{ \bigcup T \} \cup (\bigcup T - B)) \land B$ is closed in $(\text{CoFinite } (\bigcup T)))$ unfolding OPCompactification_def
using T1_cocardinal_coarser unfolding Cofinite_def by auto
then have $U \in (\bigcup T \cap U) \cap (\bigcup T - B) \land B$ is closed in $(\text{CoFinite } (\bigcup T))$
using $UV(2)$ N by auto
then have $U \in (\bigcup T \cap U) \cap (\bigcup T - B) \land B$ is closed in $(\text{CoFinite } (\bigcup T))$
using UV(2) N by auto

1184
then have $U \in \mathcal{T} \lor (U \cap V = (\bigcup \mathcal{T}) \cap V)$ unfolding \texttt{IsClosed_def} using union_cocardinal unfolding Cofinite_def by auto
then have $U \in \mathcal{T} \lor (U \cap V = \bigcup \mathcal{T} \cap V) \land (\bigcup \mathcal{T}) \in \text{CoFinite} ((\bigcup \mathcal{T}))$ unfolding \texttt{Cofinite_def} by auto

ultimately have $U \cap V \in ((\text{one-point compactification of}) (\text{CoFinite} ((\bigcup \mathcal{T}))) - \{\{\bigcup \mathcal{T}\}\}) \cup \mathcal{T}$ by auto

ultimately show thesis unfolding \texttt{IsATopology_def} by auto
\end{proof}

The previous construction preserves anti-hyperconnectedness.

\begin{proof}
\begin{theorem}\texttt{COF_comp_antiHConn:}
assumes T\{is anti-\}IsHConnected \neg (\bigcup \mathcal{T} \prec \mathbb{N})
shows \((\text{one-point compactification of}) (\text{CoFinite} ((\bigcup \mathcal{T}))) - \{\{\bigcup \mathcal{T}\}\}) \cup \mathcal{T} \{is anti-\}IsHConnected
\end{theorem}\texttt{proof-}

have $N : \bigcup \mathcal{T} \notin (\bigcup \mathcal{T})$ using \texttt{mem_not_refl} by auto
from assms(1) have T1: T\{is $T_1$\} using anti_HConn_imp_T1 by auto
have tot1: $\bigcup \{\text{one-point compactification of}\} (\text{CoFinite} ((\bigcup \mathcal{T}))) = (\bigcup \mathcal{T}) \cup \bigcup \mathcal{T}$ using topology0.op_compact_total[OF topology0_CoCardinal[OF InfCard_nat], of $\bigcup \mathcal{T}$]

union_cocardinal[of nat $\bigcup \mathcal{T}$] unfolding Cofinite_def by auto
then have $(\bigcup \{\text{one-point compactification of}\} (\text{CoFinite} ((\bigcup \mathcal{T})))) \cup \mathcal{T} = (\bigcup \mathcal{T}) \cup \bigcup \mathcal{T}$ by auto

moreover have $(\bigcup \{\text{one-point compactification of}\} (\text{CoFinite} ((\bigcup \mathcal{T})))) \cup \mathcal{T} = (\bigcup \{\text{one-point compactification of}\} (\text{CoFinite} ((\bigcup \mathcal{T})))) \cup \bigcup \mathcal{T}$ by auto

ultimately have tot2: $\bigcup \{\text{one-point compactification of}\} (\text{CoFinite} ((\bigcup \mathcal{T}))) \cup \mathcal{T} = (\bigcup \mathcal{T}) \cup \bigcup \mathcal{T}$ by auto

have $(\bigcup \mathcal{T}) \cup \bigcup \mathcal{T} \in (\text{CoFinite} ((\bigcup \mathcal{T})))$ unfolding union_open[OF topology0.op_comp_is_top[OF topology0_CoCardinal[OF InfCard_nat]], of $\text{one-point compactification of}\} (\text{CoFinite} ((\bigcup \mathcal{T})))$]

tot1 unfolding Cofinite_def by auto

\end{proof}
have $\bigcup T \subseteq \{\bigcup T\}$ by auto ultimately
have $\bigcup T \subseteq \{\bigcup T\}$ by auto
with `Not have False by auto`
then have $\{\bigcup T\}$ by auto
ultimately have $\bigcup T \subseteq \{\bigcup T\}$ by auto

with `tot2`
ultimately have $\operatorname{TOT} : \bigcup (\{\bigcup \{\text{one-point compactification of}(\text{CoFinite } \bigcup T)\}\} - \{\{U\}\}) = \{\bigcup T\}$
by auto

\begin{verbatim}
{ fix A assume AS:A \subseteq \bigcup T (((\{\text{one-point compactification of}(\text{CoFinite } \bigcup T)\}) - \{\{U\}\})\{\text{restricted to}\}A) \{\text{is hyperconnected}\}
  from AS(1,2) have e0:((((\{\text{one-point compactification of}(\text{CoFinite } \bigcup T)\}) - \{\{U\}\})\{\text{restricted to}\}A) = (((\{\text{one-point compactification of}(\text{CoFinite } \bigcup T)\}) - \{\{U\}\})\{\text{restricted to}\}A
    using subspace_of_subspace[of A \bigcup T((\{\text{one-point compactification of}(\text{CoFinite } \bigcup T)\}) - \{\{U\}\})\{\text{restricted to}\}A]
  by auto
  have e1:(((\{\text{one-point compactification of}(\text{CoFinite } \bigcup T)\}) - \{\{U\}\})\{\text{restricted to}\}(\bigcup T) = (((\{\text{one-point compactification of}(\text{CoFinite } \bigcup T)\}) - \{\{U\}\})\{\text{restricted to}\}(\bigcup T)\{\text{restricted to}\}T)
    unfolding RestrictedTo_def by auto
  
  fix A assume A\in\bigcup T \{\text{restricted to}\} U T
  then obtain B where B\in A \cap \bigcup T\{\text{unfolding RestrictedTo_def by auto}
  then have A=B by auto
  with <A\in T> have A\in T by auto
}
then have T\{\text{restricted to}\} T \subseteq \bigcup T by auto moreover

  
  fix A assume \in T
  then have T \cap A=A by auto
  with <A\in \bigcup T> have A\in\bigcup T\{\text{restricted to}\} U\{\text{unfolding RestrictedTo_def by auto}
}
ultimately have T\{\text{restricted to}\} T = T by auto moreover

  fix A assume A\in(\{\text{one-point compactification of}(\text{CoFinite } \bigcup T)\}) - \{\{U\}\})\{\text{restricted to}\} T
  then obtain B where B\in(\{\text{one-point compactification of}(\text{CoFinite } \bigcup T)\}) - \{\{U\}\})\{\text{restricted to}\} T
  

\end{verbatim}
\{(\cup T))\} \cup \{\cup T\} \cap B = A \quad \text{unfolding} \quad \text{RestrictedTo_def by auto}

then have B \in \{(\text{one-point compactification of})(\text{CoFinite} (\cup T))\} \cup \{\cup T\} \cap B = A 

by auto

then have A \in \{(\text{one-point compactification of})(\text{CoFinite} (\cup T))\}\{\text{restricted to}\}\cup T \quad \text{unfolding} \quad \text{RestrictedTo_def by auto}

then have A \in (\text{CoFinite} (\cup T)) \text{ using topology0.open_subspace}(2)[\text{OF topology0_CoCardinal}[\text{OF InfCard_nat}]]

union_cocardinal unfolding \text{Cofinite_def by auto}

with T1 have A \in T \text{ using T1_cocardinal_coarser by auto}

then have A \in (\text{CoFinite} (\cup T)) \text{ using}\ topology0.open_subspace(2)[\text{OF assms(2)}] [\text{OF union_cocardinal}]

unfolding \text{Cofinite_def by auto}

with T1 have \cup T \in T \text{ using}\ topology0_Assum unfolding \text{IsATopology_def by auto}

then have P: \cup T \in \{(\text{one-point compactification of})(\text{CoFinite} (\cup T))\} \cup \{\cup T\} \text{ by auto}

\{ fix B assume sub:B \in \{(\text{one-point compactification of})(\text{CoFinite} (\cup T))\} \cup \{\cup T\} by auto

from P have sub: \cup T \in \{(\text{one-point compactification of})(\text{CoFinite} (\cup T))\} \cup \{\cup T\} \text{ by auto}

then have \{(\text{one-point compactification of})(\text{CoFinite} (\cup T))\} \cup \{\cup T\} \cap B = \{(\text{one-point compactification of})(\text{CoFinite} (\cup T))\} \cup \{\cup T\} \cap B \text{ by auto}

using \text{subspace_of_subspace[of \cup T \cap B]} [\text{OF assms(2)}] \text{ unfolding}\ \text{topology0_def by auto}

from sub have B \subseteq \{(\text{one-point compactification of})(\text{CoFinite} (\cup T))\} \cup \{\cup T\} \cap B \text{ by auto}

then have \{(\text{one-point compactification of})(\text{CoFinite} (\cup T))\} \cup \{\cup T\} \cap B = \{(\text{one-point compactification of})(\text{CoFinite} (\cup T))\} \cup \{\cup T\} \cap B 

by auto

using \text{subspace_of_subspace[of \cup T \cap B]} [\text{OF assms(2)}] \text{ unfolding}\ \text{topology0_def by auto}

with \text{hypSub have} \{(\text{one-point compactification of})(\text{CoFinite} (\cup T))\} \cup \{\cup T\} \cap B = \{(\text{one-point compactification of})(\text{CoFinite} (\cup T))\} \cup \{\cup T\} \cap B 

by auto

with \text{reg have} \{(\text{one-point compactification of})(\text{CoFinite} (\cup T))\} \cup \{\cup T\} \cap B = \{(\text{one-point compactification of})(\text{CoFinite} (\cup T))\} \cup \{\cup T\} \cap B 

by auto

then have \text{le:}\ \cup T \cap B \leq 1 \text{ using}\ \text{HConn_spectrum by auto}

\{ fix x assume x:x \in \cup T \cap B
with le have sing:⋃T∩B={x} using lepoll_1_is_sing by auto
{
  fix y assume y:y∈B
  then have y∈⋃T∪{⋃T} using sub by auto
  with y have y∈⋃T∩B by auto
  with sing have y=x using auto
}
then have B⊆{x,⋃T} by auto
with x have disj:B={x}∨B={(x,⋃T)} by auto
{
  assume ⋃T∈B
  with disj have B:B={x}∨B={x,⋃T} by auto
  from sing subop have singOp:{x}∈(({{one-point compactification of}(CoFinite ⋃T)})-{(⋃T)}∪{x}) using auto
  unfolding Cofinite_def by auto
  moreover have Finite({x}) by auto
  then have spec:{x} is in the spectrum of (AT, (⋃T) {is compact in} T) using compact_spectrum by auto
  have (((CoFinite ⋃T){restricted to}x){is a topology}∪((CoFinite ⋃T){restricted to}x))={x} using topology0.Top_1_L4[OF topology0_CoCardinal[OF InfCard_nat] cocardinal_is_T1[OF InfCard_nat]] unfolding RestrictedTo_def Cofinite_def
  using x union_cocardinal unfolding Cofinite_def by auto
  unfolding Spec_def by auto
  then have {x} {is compact in} (CoFinite ⋃T) using compact_subspace_imp_compact by auto
  moreover note x
  ultimately have {{⋃T}∪({⋃T}−{x})∈{one-point compactification of}(CoFinite ⋃T)} using union_cocardinal unfolding Cofinite_def by auto
}
{
  assume A:{{⋃T}∪({⋃T}−{x})}={⋃T}
  
  fix y assume P:y∈{⋃T}−{x}
  then have y∈{{⋃T}∪({⋃T}−{x})} by auto
  then have y={⋃T} using A by auto
  with N P have False by auto
}
then have {⋃T}−{x}=0 by auto
with x have {⋃T}=0 by auto
then have {⋃T}≺{nat} using singleton_eqpoll_1 by auto
moreover have 1≺nat using n_lesspoll_nat by auto
ultimately have {⋃T}≺{nat} using eq_lesspoll_trans by auto

1188
then have False using assms(2) by auto

ultimately have \{\bigcup T\}\cup (\bigcup T-\{x\})\in\{(one-point compactification of)(CoFinite (\bigcup T))\}-\{(\bigcup T)\}\cup T\) by auto
then have \bigcap \{\bigcup T\}\cup (\bigcup T-\{x\})\in\{(one-point compactification of)(CoFinite (\bigcup T))\}-\{(\bigcup T)\}\cup T\) by auto

moreover have \bigcap \{\bigcup T\}\cup (\bigcup T-\{x\})\in\{(one-point compactification of)(CoFinite (\bigcup T))\}-\{(\bigcup T)\}\cup T\) by auto

ultimately have \bigcup T\in\{(one-point compactification of)(CoFinite (\bigcup T))\}-\{(\bigcup T)\}\cup T\) by auto

then have \bigcup T\in\{(one-point compactification of)(CoFinite (\bigcup T))\}-\{(\bigcup T)\}\cup T\) by auto

then have \bigcup T\in\{(one-point compactification of)(CoFinite (\bigcup T))\}-\{(\bigcup T)\}\cup T\) by auto

ultimately have \bigcup T\in\{(one-point compactification of)(CoFinite (\bigcup T))\}-\{(\bigcup T)\}\cup T\) by auto

moreover have \bigcap \{\bigcup T\}\cup (\bigcup T-\{x\})\in\{(one-point compactification of)(CoFinite (\bigcup T))\}-\{(\bigcup T)\}\cup T\) by auto

ultimately have \bigcap \{\bigcup T\}\cup (\bigcup T-\{x\})\in\{(one-point compactification of)(CoFinite (\bigcup T))\}-\{(\bigcup T)\}\cup T\) by auto

moreover

\{ assume \bigcap \{\bigcup T\}\cup (\bigcup T-\{x\})\in\{(one-point compactification of)(CoFinite (\bigcup T))\}-\{(\bigcup T)\}\cup T\) by auto

ultimately have \bigcap \{\bigcup T\}\cup (\bigcup T-\{x\})\in\{(one-point compactification of)(CoFinite (\bigcup T))\}-\{(\bigcup T)\}\cup T\) by auto

then have B\in\{(one-point compactification of)(CoFinite (\bigcup T))\}-\{(\bigcup T)\}\cup T\) by auto

ultimately have \bigcap \{\bigcup T\}\cup (\bigcup T-\{x\})\in\{(one-point compactification of)(CoFinite (\bigcup T))\}-\{(\bigcup T)\}\cup T\) by auto

then have B\in\{(one-point compactification of)(CoFinite (\bigcup T))\}-\{(\bigcup T)\}\cup T\) by auto

then have B\in\{(one-point compactification of)(CoFinite (\bigcup T))\}-\{(\bigcup T)\}\cup T\) by auto

then have B\in\{(one-point compactification of)(CoFinite (\bigcup T))\}-\{(\bigcup T)\}\cup T\) by auto

then show thesis unfolding antiProperty_def using TOT by auto

qed

The previous construction, applied to a densely ordered topology, gives the desired counterexample. What happens is that every neighbourhood of \bigcup T is dense; because there are no finite open sets, and hence meets every non-empty open set. In conclusion, \bigcup T cannot be separated from other points by disjoint open sets.

Every open set that contains \bigcup T is dense, when considering the order topology in a densely ordered set with more than two points.

**Theorem neigh_infPoint_dense:**

fixes T X r defines T_def:T \equiv (OrdTopology X r)
assumes IsLinOrder(X,r) X(is dense with respect to)r
\exists x y. x\neq y\wedge x\in X\wedge y\in X \forall U\in\{(one-point compactification of)(CoFinite (\bigcup T))\}-\{(\bigcup T)\}\cup T
\forall V\in\{(one-point compactification of)(CoFinite (\bigcup T))\}-\{(\bigcup T)\}\cup T V\neq 0

\text{1189}
shows \( U \cap V \neq 0 \)

proof

have \( N : \{ U \} \subseteq (\{ U \}) \) using mem_not_refl by auto

have-tot1: \( \{ \text{(one-point compactification of) (CoFinite (\{ U \}))} \} = \{ U \} \cup \{ U \} \)

using topology0.op_comp_total[of CoFinite[0] CoCardinal[of InfCard_nat]]

by auto moreover

have \( \{ \text{(one-point compactification of) (CoFinite (\{ U \}))} \} = (\{ \text{(one-point compactification of) (CoFinite (\{ U \}))} \}) \cup \{ U \} \)

by auto

ultimately have tot2: \( \{ \text{(one-point compactification of) (CoFinite (\{ U \}))} \} = \{ U \} \cup \{ U \} \)

by auto

have \( \{ U \} : \{ U \} \subseteq (\{ U \}) \)

using union_open[of topology0.op_comp_is_top[of CoFinite[0] CoCardinal[of InfCard_nat]]]

of \{ \text{(one-point compactification of) (CoFinite (\{ U \}))} \}

by auto

ultimately have X=0 unfolding Cofinite_def by auto moreover

assume \( U = 0 \)

then have X=0 unfolding T_def using union_ordtopology[of assms(2)]

assms(4) by auto

then have False using assms(4) by auto

} then have \( U \) \neq 0 by auto

with \( N \) have \( \neg (\{ U \} \subseteq (\{ U \})) \) by auto

assume \( \{ U \} : \{ U \} \subseteq (\{ U \}) \)

moreover

have \( \{ U \} : \{ U \} \subseteq (\{ U \}) \)

by auto

ultimately have \( \{ U \} \subseteq (\{ U \}) \)

by auto

ultimately have \( \{ U \} \subseteq (\{ U \}) \)

by auto

ultimately have \( \{ U \} \subseteq (\{ U \}) \)

by auto

assume A: \( U \cap V = 0 \)

with assms(6) have NN: \( U \cap V \) by auto

1190
with assms(7) have \( V \subseteq (\text{CoFinite } \bigcup T) \cup T \) unfolding OPCompactification_def
using union_cocardinal
unfolding Cofinite_def by auto

moreover have \( T \{\text{is T}_2\} \) unfolding T_def using order_top_T2[OF assms(2)]
assms(4) by auto
then have \( T_1 : \{\text{is T}_1\} \) unfolding T2_is_T1 by auto
ultimately have \( V \in T \) using topology0.T1_cocardinal_coarser[OF topology0_ordtopology(1) assms(2)]
unfolding T_def by auto
from A assms(7)
have \( V \subseteq \bigcup \left( \{\text{one-point compactification of}(\text{CoFinite } \bigcup T)\} - \{\bigcup T\} \right) \cup T - U \) by auto

then have \( V \subseteq \{\bigcup T\} \cup \bigcup T - U \) unfolding T_def using \( \text{is T}_2 \)
from \( N \) have \( U \notin T \) using assms(6) by auto
then have \( U \notin (\text{CoFinite } \bigcup T) \cup T \) unfolding T_def
by auto
with assms(5,6) obtain \( B \) where \( U = \{\bigcup T\} \cup \bigcup T - B \) B{is closed in}(\text{CoFinite } \bigcup T) B \neq \bigcup T 
unfolding OPCompactification_def using union_cocardinal unfolding Cofinite_def
by auto
then have \( U = \{\bigcup T\} \cup (\bigcup T - B) \) B{is a set of cardinality} \( \text{nat} \) \( B \neq \bigcup T \) unfolding Cofinite_def
by auto
with \( \text{assms}(5,6) \) obtain \( B \) where \( U = \{\bigcup T\} \cup (\bigcup T - B) \) B{is a set of cardinality} \( \text{nat} \) \( B \neq \bigcup T \) unfolding Cofinite_def
by auto
with \( \text{assms}(8) \) obtain \( \langle b = \text{nat} \rangle \) have \( \text{Finite}(\bigcup T - U) \) unfolding lesspoll_nat_is_Finite by auto
with \( \langle V \subseteq (\bigcup T) - U \rangle \) have \( \text{Finite}(V) \) using subset_Finite by auto
from \( \text{assms}(8) \) obtain \( v \) where \( v \in V \) by auto
with VopT have \( \exists R \in \{\text{IntervalX}(X, r, b, c) . \langle b,c \rangle \in X \times X\} \cup \{\text{LeftRayX}(X, r, b) . b \in X\} \cup \{\text{RightRayX}(X, r, b) . b \in X\}. R \subseteq V \wedge v \in R \) using
point_open_base_neigh[OF Ordtopology_is_a_topology(2)[OF assms(2)]
unfolding T_def by auto
then obtain \( R \) where \( R \in \{\text{IntervalX}(X, r, b, c) . \langle b,c \rangle \in X \times X\} \)
\cup \{\text{LeftRayX}(X, r, b) . b \in X\} \cup \{\text{RightRayX}(X, r, b) . b \in X\}. R \subseteq V \forall v \in R \)
by blast
moreover
assume \( R \in \{\text{IntervalX}(X, r, b, c) . \langle b,c \rangle \in X \times X\} \)
then obtain \( b,c \) where \( b \in X \) \( \chi = \text{IntervalX}(X, r, b, c) \) by auto
with \( \langle v \in R \rangle \) have \( \neg \text{Finite}(R) \) using dense_order_inf_intervals[OF assms(2)]
by auto
with \( R \in V \) \( \langle \text{Finite}(V) \rangle \) have False using subset_Finite by auto
moreover
A densely ordered set with more than one point gives an order topology. Applying the previous construction to this topology we get a non locally-
Hausdorff space.

theorem OPComp_cofinite_dense_order_not_loc_T2:
  fixes T X r
  defines T_def:T ≡ (OrdTopology X r)
  assumes IsLinOrder(X,r) X{is dense with respect to}r
  shows ¬((({one-point compactification of}(CoFinite (⋃T)))-{⋃T})∪T){is
locally-T2})
proof
  have N:⋃T∉(⋃T) using mem_not_refl by auto
  have tot1:⋃({one-point compactification of}(CoFinite (⋃T)))={⋃T}∪T
  using topology0.op_compact_total[OF topology0_CoCardinal[OF InfCard_nat],
of ⋃T]
  unfolding Cofinite_def by auto
  moreover
  have ⋃({one-point compactification of}(CoFinite (⋃T)))∪T=(⋃({one-point
  compactification of}(CoFinite (⋃T))))∪T
  by auto
  ultimately have tot2:⋃({one-point compactification of}(CoFinite (⋃T)))∪T=(⋃T)∪T
  by auto
  have {⋃T}:⋃T∈({one-point compactification of}(CoFinite (⋃T))) using
  union_open[OF topology0.op_comp_is_top[OF topology0_CoCardinal[OF
  InfCard_nat]],of {one-point compactification of}(CoFinite (⋃T))]
  unfolding Cofinite_def by auto
  moreover
  assume ⋃T=0
  then have X=0 unfolding T_def using union_ordtopology[OF assms(2)]
  assms(4) by auto
  then have False using assms(4) by auto
qed
then have $\bigcup T \neq 0$ by auto
with $N$ have $\neg (\bigcup T \subseteq \{\bigcup T\})$ by auto
{
  assume $\{\bigcup T\}\cup \bigcup T = \{\bigcup T\}$ moreover
  have $\bigcup T \subseteq \{\bigcup T\}\cup \bigcup T$ by auto ultimately
  have $\bigcup T \subseteq \{\bigcup T\}$ by auto
  with $\neg$ have False by auto
}
then have $\{\bigcup T\}\cup \bigcup T \neq \{\bigcup T\}$ by auto ultimately
have $\{\bigcup T\}\cup \bigcup T \in \{\text{one-point compactification of}(\text{CoFinite}\ (\bigcup T))\}-\{\bigcup T\}$
by auto
then have $\{\bigcup T\}\cup \bigcup T \in \{\text{one-point compactification of}(\text{CoFinite}\ (\bigcup T))\}-\{\bigcup T\}\cup T$
by auto
ultimately have $\{\bigcup T\}\cup \bigcup T \in \{\text{one-point compactification of}(\text{CoFinite}\ (\bigcup T))\}-\{\bigcup T\}\cup T$
ultimately have $\{\bigcup T\}\cup \bigcup T \subseteq \bigcup (\{\text{one-point compactification of}(\text{CoFinite}\ (\bigcup T))\}-\{\bigcup T\}\cup T)$
by auto
moreover have $\{\text{one-point compactification of}(\text{CoFinite}\ (\bigcup T))\}-\{\bigcup T\}\cup T \subseteq \{\text{one-point compactification of}(\text{CoFinite}\ (\bigcup T))\}\cup T$
by auto
ultimately have $\{\text{one-point compactification of}(\text{CoFinite}\ (\bigcup T))\}-\{\bigcup T\}\cup T = \{\bigcup T\}$
by auto
have $T_1:T\{\text{is T}_1\}$ using order_top_T2[OF assms(2,4)] T2_is_T1 unfolding T_def by auto
moreover from assms(4) obtain $b\ c$ where $B:b\in X\ c\in X\ b\neq c$ by auto
{
  assume $(b,c)\notin r$
  with assms(2) have $(c,b)\in r$ unfolding IsLinOrder_def IsTotal_def using $b\in X\ c\in X$ by auto
  with assms(3) $B$ obtain $z$ where $z\in X-(b,c)-(c,z)\in r(z,b)\in r$ unfolding IsDense_def by auto
  then have IntervalX$(X,r,b,c)\neq 0$ unfolding IntervalX_def using Order_ZF_2_L1 by auto
  then have $\neg (\text{Finite}(\text{IntervalX}(X,r,b,c)))$ using dense_order_inf_intervals[OF assms(2) _ $c\notin X$ $b\in X$ assms(3)]
  by auto moreover
  have IntervalX$(X,r,b,c)\subseteq X$ unfolding IntervalX_def by auto
  ultimately have $\neg (\text{Finite}(X))$ using subset_Finite by auto
  then have $\neg (X\neq \text{nat})$ using lesspoll_nat_is_Finite by auto
}
moreover
{
  assume $(b,c)\in r$
  with assms(3) $B$ obtain $z$ where $z\in X-(b,c)-(b,z)-(z,c)\in r$ unfolding IsDense_def by auto
  then have IntervalX$(X,r,b,c)\neq 0$ unfolding IntervalX_def using Order_ZF_2_L1 by auto
}
then have \( \neg (\text{Finite}(\text{IntervalX}(X,r,b,c))) \) using dense_order_inf_intervals[OF assms(2)] by auto

moreover have \( \text{IntervalX}(X,r,b,c) \subseteq X \) unfolding IntervalX_def by auto

ultimately have \( \neg (\text{Finite}(X)) \) using subset_Finite by auto

then have \( \neg (X \prec \text{nat}) \) using lesspoll_nat_is_Finite by auto

ultimately have \( \neg (X \prec \text{nat}) \) by auto

with \( T_1 \) have top: \( \{\text{one-point compactification of}(\text{CoFinite } (\bigcup T))\} - \{\{\bigcup T\}\cup T\} \) is a topology using topology0.COF_comp_is_top[OF topology0_ordtopology[OF assms(2)]] unfolding T_def

using union_ordtopology[OF assms(2,4)] by auto

assume \( \{\text{one-point compactification of}(\text{CoFinite } (\bigcup T))\} - \{\{\bigcup T\}\cup T\} \) {is locally-\( T_2 \)}

moreover have \( \bigcup T \in \{\text{one-point compactification of}(\text{CoFinite } (\bigcup T))\} - \{\{\bigcup T\}\cup T\} \) using TOT by auto

moreover have \( \{\text{one-point compactification of}(\text{CoFinite } (\bigcup T))\} - \{\{\bigcup T\}\cup T\} \) {restricted to} \( C \) {is \( T_2 \)} using topology0.IsLocallyT2_def

ultimately have \( \exists c \in \text{Pow}(\bigcup \{\text{one-point compactification of}(\text{CoFinite } (\bigcup T))\} - \{\{\bigcup T\}\cup T\}) \)

unfolding IsLocallyT2_def IsLocally_def[OF top] by auto

then obtain \( C \) where \( C:C \subseteq \bigcup \{\text{one-point compactification of}(\text{CoFinite } (\bigcup T))\} - \{\{\bigcup T\}\cup T\} \) and \( T_2: \{\text{one-point compactification of}(\text{CoFinite } (\bigcup T))\} - \{\{\bigcup T\}\cup T\} \) {restricted to} \( C \) {is \( T_2 \)}

by auto

have sub: \( \text{Interior}(C, \{\text{one-point compactification of}(\text{CoFinite } (\bigcup T))\} - \{\{\bigcup T\}\cup T\}) \) ∈ \( \text{Pow}(\bigcup \{\text{one-point compactification of}(\text{CoFinite } (\bigcup T))\} - \{\{\bigcup T\}\cup T\}) \)

top unfolding topology0.Top_2_L1

top unfolding topology0_def by auto

have \( \bigcup \{\text{one-point compactification of}(\text{CoFinite } (\bigcup T))\} - \{\{\bigcup T\}\cup T\} \) {restricted to} \( C \) {is \( T_2 \)} using topology0.IsLocallyT2_def

ultimately have \( \{\text{one-point compactification of}(\text{CoFinite } (\bigcup T))\} - \{\{\bigcup T\}\cup T\} \) {restricted to} \( C \) {is \( T_2 \)}

by auto

have \( \bigcup \{\text{one-point compactification of}(\text{CoFinite } (\bigcup T))\} - \{\{\bigcup T\}\cup T\} \) {restricted to} \( C \) {is \( T_2 \)}

using subspace_of_subspace[OF sub C(1)] by auto

moreover have \( \bigcup \{\text{one-point compactification of}(\text{CoFinite } (\bigcup T))\} - \{\{\bigcup T\}\cup T\} \) {restricted to} \( C \) {is \( T_2 \)}

by auto

ultimately have \( T_2 \cdot 2: \{\text{one-point compactification of}(\text{CoFinite } (\bigcup T))\} - \{\{\bigcup T\}\cup T\} \) {restricted to} \( \text{Interior}(C, \{\text{one-point compactification of}(\text{CoFinite } (\bigcup T))\} - \{\{\bigcup T\}\cup T\}) \) {is \( T_2 \)
using T2_here[OF T2_pp] by auto

have top2:((((one-point compactification of)(CoFinite \cup T)) - \{\{T\}\})
\cup T){restricted to}(Interior(C, (((one-point compactification of)(CoFinite
\cup T)) - \{\{T\}\}) \cup T))}{is a topology}

using topology0.Top_1_L4 top unfolding topology0_def by auto

from C(2) pp have p1:\cup T\in\bigcup(((one-point compactification of)(CoFinite
\cup T)) - \{\{T\}\}) \cup T){restricted to}(Interior(C, (((one-point compactification of)(CoFinite
\cup T)) - \{\{T\}\}) \cup T))

unfolding RestrictedTo_def by auto

from topology0.Top_2_L2 have intOP:(Interior(C, (((one-point
compactification of)(CoFinite \cup T)) - \{\{T\}\}) \cup T)
\in ((one-point compactification of)(CoFinite \cup T)) - \{\{T\}\} \cup T)
unfolding topology0_def by auto

{fix x assume x\notin\cup T x\in\bigcup(((one-point compactification of)(CoFinite
\cup T)) - \{\{T\}\}) \cup T){restricted to}(Interior(C, (((one-point compactification of)(CoFinite
\cup T)) - \{\{T\}\}) \cup T))

with p1 have \exists U \in (((one-point compactification of)(CoFinite
\cup T)) - \{\{T\}\}) \cup T){restricted to}(Interior(C, (((one-point compactification of)(CoFinite
\cup T)) - \{\{T\}\}) \cup T)). \exists V \in (((one-point compactification of)(CoFinite
\cup T)) - \{\{T\}\}) \cup T){restricted to}(Interior(C, (((one-point
compactification of)(CoFinite \cup T)) - \{\{T\}\}) \cup T)).

x\in\cup U \cup V \forall x=0 using T2_2 unfolding isT2_def by auto

then obtain U V where UV:U \in (((one-point compactification of)(CoFinite
\cup T)) - \{\{T\}\}) \cup T){restricted to}(Interior(C, (((one-point compactification of)(CoFinite
\cup T)) - \{\{T\}\}) \cup T))

V \in (((one-point compactification of)(CoFinite \cup T)) - \{\{T\}\}) \cup T){restricted to}(Interior(C, (((one-point compactification of)(CoFinite
\cup T)) - \{\{T\}\}) \cup T))

U \neq \emptyset \cup U \cup V \forall x=0 by auto

from UV(1) obtain UC where UV:U=\_\cup \emptyset (((one-point compactification of)(CoFinite
\cup T)) - \{\{T\}\}) \cup T) \cap UC \subseteq (((one-point compactification of)(CoFinite
\cup T)) - \{\{T\}\}) \cup T)

unfolding RestrictedTo_def by auto

with top intOP have Uop:U \in (((one-point compactification of)(CoFinite
\cup T)) - \{\{T\}\}) \cup T unfolding IsATopology_def by auto

from UV(2) obtain VC where V=\_\cup \emptyset (((one-point compactification of)(CoFinite
\cup T)) - \{\{T\}\}) \cup T) \cap VC \subseteq (((one-point compactification of)(CoFinite
\cup T)) - \{\{T\}\}) \cup T)

unfolding RestrictedTo_def by auto

with top intOP have V \in (((one-point compactification of)(CoFinite
\cup T)) - \{\{T\}\}) \cup T unfolding IsATopology_def by auto

with UV(3-5) Uop neigh_infPoint_dense[OF assms(2-4),of UV] union_ordtopology[OF assms(2,4)]

have False unfolding T_def by auto

}

then have \bigcup(((one-point compactification of)(CoFinite \cup T)) - \{\{T\}\})
\cup T){restricted to}(Interior(C, (((one-point compactification of)(CoFinite
\cup T)) - \{\{T\}\}) \cup T)) \subseteq \{T\}
by auto

1195
with p1 have \(\bigcup\left(\left(\left\{\text{one-point compactification of}(\text{CoFinite } \bigcup T)\right\} - \{\bigcup T\}\right)\cup T\right)\{\text{restricted to}(\text{Interior}(C, \left(\left\{\text{one-point compactification of}(\text{CoFinite } \bigcup T)\right\} - \{\bigcup T\}\right)\cup T))\})=\{\bigcup T\} \) by auto

with top2 have \(\{\bigcup T\}\in\left(\left(\left\{\text{one-point compactification of}(\text{CoFinite } \bigcup T)\right\} - \{\bigcup T\}\right)\cup T\right)\{\text{restricted to}(\text{Interior}(C, \left(\left\{\text{one-point compactification of}(\text{CoFinite } \bigcup T)\right\} - \{\bigcup T\}\right)\cup T))\} \) unfolding IsATopology_def by auto

then obtain \(W\) where UT:{\bigcup T}=(\text{Interior}(C, \left(\left\{\text{one-point compactification of}(\text{CoFinite } \bigcup T)\right\} - \{\bigcup T\}\right)\cup T))\} \cap WW \in (\text{CoFinite } \bigcup T) - \{\bigcup T\} \cup T\) unfolding RestrictedTo_def by auto

from this(2) have \((\text{Interior}(C, \left(\left\{\text{one-point compactification of}(\text{CoFinite } \bigcup T)\right\} - \{\bigcup T\}\right)\cup T))\} \cap W \in (\text{CoFinite } \bigcup T) - \{\bigcup T\} \cup T\) using intOP top unfolding IsATopology_def by auto

with UT(1) have \(\{\bigcup T\}\in(\text{CoFinite } \bigcup T)\) by auto

then have \(\{\bigcup T\}\in T\) by auto

with N show False by auto

qed

This topology, from the previous result, gives a counter-example for anti-hyperconnected implies locally-\(T_2\).

**theorem** antiHConn_not_imp_loc_T2:

fixes \(T\) \(X\) \(r\)
defines \(T\)_def: \(T \equiv \text{OrdTopology } X\) \(r\)
assumes IsLinOrder(X,r) \(X\{\text{is dense with respect to} r\}\)
exists \(x\) \(y\). \(x\neq y\) \(\land x\in X\land y\in X\)
shows \(\neg\left(\left(\left\{\text{one-point compactification of}(\text{CoFinite } \bigcup T)\right\} - \{\bigcup T\}\right)\cup T\right)\{\text{is locally-}T_2\}\)
and \(\left(\left\{\text{one-point compactification of}(\text{CoFinite } \bigcup T)\right\} - \{\bigcup T\}\right)\cup T\{\text{is anti-}IsHConnected\}\)
using OPComp_cofinite_dense_order_not_loc_T2[OF assms(2-4)] dense_order_infinite[OF assms(2-4)] union_ordtopology[OF assms(2,4)] topology0.COF_comp_antiHConn[OF topology0_ordtopology[OF assms(2)] topology0.T2_imp_anti_HConn[OF topology0_ordtopology[OF assms(2)] order_top_T2[OF assms(2,4),rule_format]]]
unfolding \(T\)_def by auto

Let’s prove that \(T_2\) spaces are locally-\(T_2\), but that there are locally-\(T_2\) spaces which aren’t \(T_2\). In conclusion \(T_2\Rightarrow\text{ locally-}T_2\Rightarrow\text{ anti-hyperconnected}\); all implications proper.

**theorem** (in topology0) T2_imp_loc_T2:

assumes \(T\{\text{is } T_2\}\)
shows \(T\{\text{is locally-}T_2\}\)

proof-

{ fix \(x\) assume \(x\in \bigcup T\)

}
fix b assume b: b ∈ T x ∈ b
then have (T restricted to) b is T₂ using T₂_here assms by auto
moreover
from b have x ∈ int(b) using Top_2_L3 by auto
ultimately have ∃ c ∈ Pow(b). x ∈ int(c) ∧ (T restricted to) c is T₂ by auto
then have ∀ b ∈ T. x ∈ b −→ (∃ c ∈ Pow(b). x ∈ int(c) ∧ (T restricted to) c is T₂) by auto

then show thesis unfolding IsLocallyT2_def IsLocally_def [OF topSpaceAssum] by auto
qed

If there is a closed singleton, then we can consider a topology that makes this point double.

theorem (in topology0) doble_point_top:
assumes {m} is closed in T shows (T ∪ {U-{m}} ∪ {∪T} ∪ W. ⟨U, W⟩ ∈ {V ∈ T. m ∈ V} × T) is a topology
proof -
{
fix M assume M: M ⊆ T ∪ {U-{m}} ∪ {∪T} ∪ W. ⟨U, W⟩ ∈ {V ∈ T. m ∈ V} × T
let M = {V ∈ M. V ∈ T}
let Mm = {V ∈ M. V /∈ T}
have unm: ∪ M = ∪ Mm by auto
have tt: ∪ M ∈ T using topSpaceAssum unfolding IsATopology_def by auto
{ assume Mm = 0
then have ∪ Mm = 0 by auto
with unm have ∪ M = (∪ M) by auto
with tt have ∪ M ∈ T by auto
then have ∪ M ∈ T ∪ {U-{m}} ∪ {∪T} ∪ W. ⟨U, W⟩ ∈ {V ∈ T. m ∈ V} × T} by auto
}
moreover
{ assume AS: Mm ≠ 0
then obtain V where V: V ∈ M V /∈ T by auto
with M have V ∈ {U - {m}} ∪ {∪T} ∪ W. ⟨U, W⟩ ∈ {V ∈ T. m ∈ V} × T} by blast
then obtain U W where U: V = (U-{m}) ∪ {∪T} ∪ W U ∈ T m ∈ U W ∈ T by auto
let U = {V, W} ∈ T × T. m ∈ V \ (V-{m}) ∪ {∪T} ∪ W ∈ Mm
let fU = {fst(B). B ∈ U}
let sU = {snd(B). B ∈ U}
have fU ⊆ TsU ⊆ T by auto
then have P: ∪ fU ∈ T ∪ sU ∈ T using topSpaceAssum unfolding IsATopology_def by auto
moreover
have ⟨U, W⟩ ∈ U using U V by auto
then have m ∈ fU by auto
ultimately have s: (∪ fU, ∪ sU) ∈ {V ∈ T. m ∈ V} × T by auto
moreover have r: ∀ S ∈ {V ∈ T. m ∈ V} → R ∈ T → (S-{m}) ∪ {∪T} ∪ R ∈ {U-{m}} ∪ {∪T} ∪ W. 1197
\[ \{U,W\} \in \{V \in T. m \in V\} \times T \]

by auto
ultimately have \( (\bigcup \{U-(m)\} \cup \{U\} \cup sU \in \{U-(m)\} \cup \{U\} \cup W. \ (U,W) \in \{V \in T. m \in V\} \times T \) by auto
\[
\begin{aligned}
&\text{fix } v \text{ assume } v \in \bigcup Mm \\
&\text{then obtain } V \text{ where } v : v \in \bigcup Mm \text{ by auto} \\
&\text{then have } V : V \in V \in Mm \text{ by auto} \\
&\text{with } M \text{ have } V \in \{U - \{m\} \cup \{U\} \cup W. \ (U,W) \in \{V \in T. m \in V\} \times T \} \text{ by blast} \\
&\text{then obtain } U \ W \text{ where } U : V = (U - \{m\}) \cup \{U\} \cup W \text{ U \in T} \text{ by auto} \\
&\text{with } v \ (1) \text{ have } v \in (U - \{m\}) \cup \{U\} \cup W \text{ by auto} \\
&\text{then have } v \in \{U - \{m\}\} \cup \{U\} \cup sU \text{ by auto} \\
&\text{moreover from } U \ V \text{ have } (U,W) \in U \text{ by auto} \\
&\text{ultimately have } v \in ((\bigcup fU) - \{m\}) \cup \{U\} \cup (\bigcup sU) \text{ by auto} \\
\end{aligned}
\]

then have \( \bigcup Mm \subseteq ((\bigcup fU) - \{m\}) \cup \{U\} \cup (\bigcup sU) \) by blast moreover
\[
\begin{aligned}
&\text{fix } v \text{ assume } v \in ((\bigcup fU) - \{m\}) \cup \{U\} \cup (\bigcup sU) \\
&\text{assume } v \in \{U\} \\
&\text{then have } v \in (U - \{m\}) \cup \{U\} \cup W \text{ by auto} \\
&\text{with } (U,W) \in U \text{ have } v \in \bigcup Mm \text{ by auto} \\
&\text{moreover } \\
&\text{assume } v \notin \{U\} \cup V \cup (\bigcup sU) \\
&\text{with } v \text{ have } v \in ((\bigcup fU) - \{m\}) \text{ by auto} \\
&\text{then have } v \in fU \land v \neq m \text{ by auto} \\
&\text{then obtain } W \text{ where } v : v \in (W - \{m\}) \cup \{U\} \cup W \text{ by auto} \\
&\text{then have } v : (W - \{m\}) \cup \{U\} \cup W \text{ by auto} \\
&\text{then obtain } B \text{ where } \text{fst}(B) = W \text{ B \in U} \text{ v \in (W - \{m\}) \cup \{U\} \text{ by blast} } \\
&\text{then have } v \in \bigcup Mm \text{ by auto} \\
&\text{ultimately have } v \in \bigcup Mm \text{ by auto} \\
&\text{then have } ((\bigcup fU) - \{m\}) \cup \{U\} \cup \{U\} \cup \{U\} \cup (\bigcup sU) \subseteq \bigcup Mm \text{ by auto} \\
&\text{ultimately have } \bigcup Mm = ((\bigcup fU) - \{m\}) \cup \{U\} \cup \{U\} \cup \{U\} \cup (\bigcup sU) \text{ by auto} \\
&\text{then have } \bigcup Mm = ((\bigcup fU) - \{m\}) \cup \{U\} \cup \{U\} \cup \{U\} \cup (\bigcup sU) \cup (\bigcup M) \text{ using unmo by auto} \\
&\text{moreover from } P \text{ t t have } (\bigcup sU) \cup \{U\} \cup (\bigcup M) \in T \text{ using topSpaceAssum} \\
&\text{union_open[OF topSpaceAssum, of } \{\bigcup sU, \bigcup M\}] \text{ by auto} \\
&\text{with } s \text{ have } (\bigcup fU, \{\bigcup sU\} \cup \{U\}) \in \{V \in T. m \in V\} \times T \text{ by auto} \\
&\text{then have } ((\bigcup fU) - \{m\}) \cup \{U\} \cup \{U\} \cup \{U\} \cup (\bigcup sU) \cup (\bigcup M) \in \{\{U -(m)\} \cup \{U\} \cup W. \ (U,W) \in \{V \in T. m \in V\} \times T \} \text{ using r} \\
&\text{by auto} \\
&\text{ultimately have } \bigcup M \in \{\{U -(m)\} \cup \{U\} \cup W. \ (U,W) \in \{V \in T. m \in V\} \times T \} \text{ by auto} \\
&\text{then have } \bigcup M \in \{\{U -(m)\} \cup \{U\} \cup W. \ (U,W) \in \{V \in T. m \in V\} \times T \} \text{ by auto} \\
\end{aligned}
\]
ultimately
have $\bigcup M \in T \cup \{ (U - \{ m \}) \cup (\bigcup T) \cup W. \langle U, W \rangle \in \{ V \in T. m \in V \} \times T \}$. 

  } 

then have $\forall M \in \mathrm{Pow}(T \cup \{ (U - \{ m \}) \cup (\bigcup T) \cup W. \langle U, W \rangle \in \{ V \in T. m \in V \} \times T \})$. 

  } 

by auto 

moreover 

  { 
  fix A B assume: A \in T 

    assume B \in T 

    with A have A \cap B \in T using topSpaceAssum unfolding IsATopology_def 

  by auto 

  } 

moreover 

  { 
  assume B \not\in T 

    with ass(2) have B \in \{ (U - \{ m \}) \cup (\bigcup T) \cup W. \langle U, W \rangle \in \{ V \in T. m \in V \} \times T \} by auto 

    then obtain U W where U: U \in T mem_not_refl have ((U - \{ m \}) \cup (\bigcup T) \cup W. \langle U, W \rangle \in \{ V \in T. m \in V \} \times T \} using topSpaceAssum unfolding IsClosed_def 

    with U(1) have O: U \cap (\bigcup T - \{ m \}) \in T 

    using topSpaceAssum unfolding IsATopology_def 

    by auto 

    have U \cap (\bigcup T - \{ m \}) = U - \{ m \} using U(1) by auto 

    with 0 have U - \{ m \} \in T by auto 

    with A have (A \cap (U - \{ m \})) \in T using topSpaceAssum unfolding IsATopology_def 

    by auto 

    moreover 

    from A U(3) have A \cap W \in T using topSpaceAssum unfolding IsATopology_def 

    by auto 

    ultimately have (A \cap (U - \{ m \})) \cup (A \cap W) \in T using union_open[OF topSpaceAssum, of \{ A \cap (U - \{ m \}), A \cap W \}] by auto 

    with eq have A \cap B \in T by auto 

  } 

ultimately have A \cap B \in T by auto 

} 

moreover 

  { 
  assume A \not\in T 

    with ass(1) have A: A \in \{ (U - \{ m \}) \cup (\bigcup T) \cup W. \langle U, W \rangle \in \{ V \in T. m \in V \} \times T \} by auto 

    } 

assume B: B \in T
from A obtain U W where U:U∈Tm∈UW∈TA=(U-{m})∪{∪T}∪W by auto
moreover
from B mem_not_refl have ∪T∈B by auto
ultimately have A∩B=((U-{m})∪W)∩B by auto
then have eq:A∩B=((U-{m})∩B)∪(W∩B) by auto
have ∪T-{m}∈T using assms unfolding IsClosed_def by auto
with U(1) have 0:U∩(∪T-{m})∈T using topSpaceAssum unfolding IsATopology_def

by auto

have U∩(∪T-{m})=U-{m} using U(1) by auto
with 0 have U-{m}∈T by auto
with B have (U-{m})∩B∈T using topSpaceAssum unfolding IsATopology_def
by auto
moreover
from B U(3) have W∩B∈T using topSpaceAssum unfolding IsATopology_def
by auto
ultimately have ((U-{m})∩B)∪(W∩B)∈T using union_open[OF topSpaceAssum, of {((U-{m})∩B),(W∩B)}] by auto
with eq have A∩B=((U-{m})∩B)∪(W∩B) by auto

moreover

assume B∉T
with ass(2) have B∈{(U-{m})∪{∪T}∪W. (U,W)∈{V∈T. m∈V}×T} by auto
then obtain U W where U:U∈Tm∈UW∈TA=(U-{m})∪{∪T}∪W by auto
moreover
from A obtain UA WA where UA:UA∈Tm∈UA∈T∈UA∈T∈WA
by auto
ultimately have A∩B=((UA-{m})∪WA)∩((U-{m})∪W)∪{∪T} by auto
then have eq:A∩B=((UA-{m})∩(U-{m}))∪(WA∩W)∪(W∩W)∪{∪T} by auto
have ∪T-{m}∈T using assms unfolding IsClosed_def by auto
with U(1) UA(1) have 0:U∩(∪T-{m})∈T using topSpaceAssum unfolding IsATopology_def
by auto

have U∩(∪T-{m})=U-{m}UA∩(∪T-{m})=UA-{m} using U(1) UA(1) by auto
with 0 have 0:U-{m}∈T∈U-{m} by auto
then have ((UA-{m})∩(U-{m}))=UA∩U-{m} by auto
moreover
have UA∩U∈Tm∈UA∩U using U(1,2) UA(1,2) topSpaceAssum unfolding IsATopology_def
by auto
moreover
from 00 U(3) UA(3) have TT:WA∩(U-{m})∈T(UA-{m})∩W∈WA∩W∈T using topSpaceAssum unfolding IsATopology_def
by auto
from TT(2,3) have ((UA-{m})∩W)∈(W∩W)∈T using union_open[OF
The previous topology is defined over a set with one more point.

lemma (in topology0) union_doublepoint_top:
  assumes \{m\}\{is closed in\}T
  shows \bigcup \{TU\{\{U\{m\}\}\} ∪ \{\bigcup U\{W\}\}\} \subseteq \bigcup \{\bigcup U\}
proof
{ fix x assume x ∈ \bigcup \{TU\{\{U\{m\}\}\} ∪ \{\bigcup U\{W\}\}\} ∪ \{\bigcup U\} \subseteq \bigcup \{\bigcup U\}
  then obtain R where x ∈ R \subseteq \bigcup \{TU\{\{U\{m\}\}\} ∪ \{\bigcup U\}\} \subseteq \bigcup \{\bigcup U\}
by blast
{ assume R ∈ T
  with x \subseteq \bigcup \{TU\{\{U\{m\}\}\} ∪ \{\bigcup U\}\} \subseteq \bigcup \{\bigcup U\}
  moreover
  { assume R ∉ T
    with x \subseteq \bigcup \{TU\{\{U\{m\}\}\} ∪ \{\bigcup U\}\} \subseteq \bigcup \{\bigcup U\}
    then obtain U W where R = \bigcup \{U\{m\}\} ∪ \{\bigcup U\}\} \subseteq \bigcup \{\bigcup U\}
    with x \subseteq \bigcup \{TU\{\{U\{m\}\}\} ∪ \{\bigcup U\}\} \subseteq \bigcup \{\bigcup U\}
    ultimately have x ∈ \bigcup \{TU\} \subseteq \bigcup \{\bigcup U\}
  }
  ultimately have x ∈ \bigcup \{TU\} \subseteq \bigcup \{\bigcup U\}
}
then show \bigcup \{TU\{\{U\{m\}\}\} ∪ \{\bigcup U\}\} \subseteq \bigcup \{\bigcup U\}
by auto
fix x assume x∈∪T∪{∪T} then have dis:x∈∪T by auto
{
  assume x∈∪T then have x∈(∪T∪{m})∪∪T∪W. ⟨U,W⟩∈{(∪T. m∈V)×T} by auto
}
moreover
{
  assume x∉∪T with dis have x∈∪T by auto
  moreover from assms have ∪T-{m}∈Tm∈∪T unfolding IsClosed_def by auto
  moreover have 0∈T using empty_open topSpaceAssum by auto
  ultimately have x∈(∪T-{m})∪∪T∪0 (∪T-{m})∪∪T∪0∈(∪T-{m})∪∪T∪W.
    ⟨U,W⟩∈{(∪T. m∈V)×T}
    using union_open[OF topSpaceAssum] by auto
    then have x∈(∪T-{m})∪∪T∪0 (∪T-{m})∪∪T∪0∈T (∪T-{m})∪∪T∪W.
    ⟨U,W⟩∈{(∪T. m∈V)×T}
    by auto
    then have x∈(∪T∪{(U-{m})∪∪T∪W. ⟨U,W⟩∈{(∪T. m∈V)×T}) by blast
  }
  ultimately have x∈∪T∪{(∪T-{m})∪∪T∪W. ⟨U,W⟩∈{(∪T. m∈V)×T}) by auto
  then show ∪T∪{∪T}≤∪T∪{(∪T-{m})∪∪T∪W. ⟨U,W⟩∈{(∪T. m∈V)×T})
    by auto
  qed

In this topology, the previous topological space is an open subspace.

```
theorem (in topology0) open_subspace_double_point:
  assumes {m} {is closed in} T
  shows (∪T-{m})∪∪T∪W. ⟨U,W⟩∈{(∪T. m∈V)×T}){restricted to}∪T=T
and ∪T∈(∪T-{m})∪∪T∪W. ⟨U,W⟩∈{(∪T. m∈V)×T})
proof-
  have N:∪T∈∪T using mem_not_refl by auto
  { fix x assume x∈(∪T∪{(U-{m})∪∪T)}{restricted to}∪T
    then obtain U where U:∪T∪{(U-{m})∪∪T}W. ⟨U,W⟩∈{(∪T. m∈V)×T})x∈∪T U
    unfolding RestrictedTo_def by blast
    { assume U(1) have U∈{U-{m})∪∪T}W. ⟨U,W⟩∈{(∪T. m∈V)×T}) by auto
      then obtain V W where VW:U=(V-{m})∪∪T\Wm∈VW∈T by auto
      with N U(2) have x:x=(V-{m})∪W by auto
      have ∪T-{m}∈T using assms unfolding IsClosed_def by auto
      then have ∪T-{m}∈T using VW(2) topSpaceAssum unfolding IsATopology_def
        by auto moreover
      have V-{m}=V∩(∪T-{m}) using VW(2,3) by auto ultimately
```

1202
have \( V \setminus \{m\} \in T \) by auto
with \( VW(4) \) have \((V \setminus \{m\}) \cup W \in T \) using union_open[of topSpaceAssum, of \{V \setminus \{m\}, W\}]
by auto
with \( x \) have \( x \in T \) by auto

moreover

{ assume \( A : U \in T \) with \( U(2) \) have \( x = U \) by auto
with \( A \) have \( x \in T \) by auto
}
ultimately have \( x \in T \) by auto

then have \((T \cup \{U \setminus \{m\}\} \cup \{\bigcup T\})\{\text{restricted to}\} \cup T \subseteq T \) by auto
moreover

{ fix \( x \) assume \( x : x \in T \)
then have \( x \in (T \cup \{U \setminus \{m\}\} \cup \{\bigcup T\})\{\text{restricted to}\} \cup T \subseteq T \) by auto
moreover
from \( x \) have \( \bigcup T \setminus x = x \) by auto ultimately
have \( \exists M \in (T \cup \{U \setminus \{m\}\} \cup \{\bigcup T\}). \bigcup T \setminus M = x \) by blast
then have \( x \in (T \cup \{U \setminus \{m\}\} \cup \{\bigcup T\})\{\text{restricted to}\} \cup T \subseteq T \)
by auto
}
ultimately show \((T \cup \{U \setminus \{m\}\} \cup \{\bigcup T\}). \bigcup T \subseteq T \subseteq T \) by auto
have \( P : \bigcup T \in T \) using topSpaceAssum unfolding IsATopology_def by auto
then show \( \bigcup T \in (T \cup \{U \setminus \{m\}\} \cup \{\bigcup T\}). \bigcup T \subseteq T \) by auto
qed

The previous topology construction applied to a \( T_2 \) non-discrete space topology, gives a counter-example to: Every locally-\( T_2 \) space is \( T_2 \).

If there is a singleton which is not open, but closed; then the construction on that point is not \( T_2 \).

**Theorem (in topology0) loc_T2_imp_T2_counter_1:**
assumes \( \{m\} \notin T \) \{is closed in\( T \)\}
shows \(-((T \cup \{U \setminus \{m\}\} \cup \{\bigcup T\}). (U, W) \in (V \in T. m \in V) \times T)\) \{is \( T_2 \)\}
**Proof**
assume ass: \((T \cup \{U \setminus \{m\}\} \cup \{\bigcup T\}). (U, W) \in (V \in T. m \in V) \times T)\) \{is \( T_2 \)\}
then have tot1: \((T \cup \{U \setminus \{m\}\} \cup \{\bigcup T\}). (U, W) \in (V \in T. m \in V) \times T)=\bigcup T \cup \{\bigcup T\}
using union_doublepoint_top
assms(2) by auto
have \( m \neq \bigcup T \) using mem_not_refl assms(2) unfolding IsClosed_def by auto
moreover
from \( \text{ass tot1} \) have \( \forall x \ y. x \in \bigcup T \cup \{\bigcup T\} \land y \in \bigcup T \cup \{\bigcup T\} \land x \neq y \rightarrow (\exists M \in (T \cup \{U \setminus \{m\}\} \cup \{\bigcup T\}). (U, W))\)
\(\{U, W\}\in\{V\in T. m\in V\times T\}\).
\[\exists W\in T. (U-m)\cup U \subseteq W.\] \(\{U, W\}\in\{V\in T. m\in V\times T\}\).
\(x\in U \land y\in W \land x\neq y=0\)
unfolding \(\text{isT2}_\text{def by auto}
\)
moreover
from \text{assms}(2) have \(m\in U \cup T\) unfolding \(\text{IsClosed}_\text{def by auto more-over}
\)
have \(\bigcup U \subseteq T\) by auto ultimately
have \(\exists U\in T. (U-m)\cup T\subseteq W.\) \(\{U, W\}\in\{V\in T. m\in V\times T\}\).
\(\{U, W\}\in\{V\in T. m\in V\times T\}\). \(m\in U \cup T \\ (T \subseteq W)\)
by auto
then obtain \(U \subseteq W\) where \(U, W\in T \cup ((U-m)\cup U \subseteq W.\) \(\{U, W\}\in\{V\in T. m\in V\times T\}\))
\(\exists U\in T. (T \cup ((U-m)\cup U \subseteq W.\) \(\{U, W\}\in\{V\in T. m\in V\times T\}\)) m \in U \cup T \subseteq W \cup \emptyset = 0 \) using \(\text{tot1 by blast}
\)
then have \(U \subseteq W\) by auto
with \(UV(1)\) have \(P: U \subseteq T\) by auto
\{ assume \(W\in T
\)
then have \(U \subseteq T\) by auto
with \(UV(4)\) have \(U \subseteq T\) using \(\text{tot1 by auto}
\)
then have \(\text{False using mem_not_refl by auto}
\}
with \(UV(2)\) have \(U\in\{((U-m)\cup T)\subseteq W.\) \(\{U, W\}\in\{V\in T. m\in V\times T\}\) by auto
then obtain \(U \subseteq W\) where \(V: U = (U-m)\cup T \subseteq W.\) \(m \in U \subseteq W \subseteq T\) by auto
from \text{V}(2,3) \(P\) have \(\text{int:}\ U \subseteq T \subseteq W\) using \(\text{UV}(3)\) \(\text{topSpaceAssum}
\)
unfolding \(\text{IsATopology}_\text{def by auto}
\)
have \(U \subseteq (U-m)\subseteq W.\) \(\{U, W\}\in\{V\in T. m\in V\times T\}\) by auto
then have \(U \subseteq (U-m)\subseteq 0 \) using \(\text{UV}(5)\) by auto
with \text{int}(2) have \(U \subseteq m\) by auto
with \text{int}(1) \text{assms}(1) show \(\text{False by auto}
\)
qed

This topology is locally-\(T_2\).

\text{theorem (in topological0) locT2_imp_T2_counter_2:}
assumes \(\{m\}\subseteq T \subseteq m \subseteq T\) \(\text{is T2}\)
shows \(\text{T}\subseteq ((U-m)\cup U \subseteq W.\) \(\{U, W\}\in\{V\in T. m\in V\times T\})\) \{is locally-T2\}
proof
from \text{assms}(3) have \(T\subseteq T\) using \(\text{T2_is_T1 by auto}
\)
with \text{assms}(2) have \(m: (m)\) \{is closed in\} \(T\) using \(\text{T1_iff_singleton_closed}
\) by auto
have \(N: U \subseteq T\) using \(\text{mem_not_refl by auto}
\)
have \(\text{res: (T \subseteq ((U-m)\cup T) \subseteq W.\} \{restricted to\} \subseteq T}\)
and \(P: U \subseteq T\) and \(Q: U \subseteq (T \subseteq ((U-m)\cup T) \subseteq W.\) \(\{U, W\}\in\{V\in T. m\in V\times T\})\)
using \(\text{open_subspace_double_point mc}
\)
\text{topSpaceAssum unfolding \(\text{IsATopology}_\text{def by auto}
\)
\{ fix \(A\) assume \(\text{ass: A}\subseteq T \cup (U)\)
\{ assume \(A \neq U\)
with \text{ass} have \(A \subseteq U\) by auto
\}

1204
with Q res assms(3) have \( \bigcup T \in (\bigcup T \cup (U - \{ m \}) \cup (U \cap T) \cup W. (U, W) \in \{ V \in T. m \in V \} \times T) \) \& \( A \in T \land \exists \bigcup (T \cup (U - \{ m \}) \cup (U \cap T) \cup W. (U, W) \in \{ V \in T. m \in V \} \times T) \) \{ restricted to \( T \) \} (is \( T_2 \)) by auto
then have \( \exists Z \in (T \cup (U - \{ m \}) \cup (U \cap T) \cup W. (U, W) \in \{ V \in T. m \in V \} \times T) \). \( A \in Z \land (T \cup (U - \{ m \}) \cup (U \cap T) \cup W. (U, W) \in \{ V \in T. m \in V \} \times T) \) \{ restricted to \( Z \) \} (is \( T_2 \))
by blast
moreover
{
assume \( A : A = T \)
have \( \bigcup T \in T \cap T \in T \) using assms(2) empty_open[OF topSpaceAssum]
unfolding IsClosed_def using P by auto
then have \( \bigcup \{ T \} \cup (U - \{ m \}) \cup (U \cap T) \cup W. (U, W) \in \{ V \in T. m \in V \} \times T \)
by auto
then have \( \exists opp : (T - \{ m \}) \cup (U \cap T) \cup W. (U, W) \in \{ V \in T. m \in V \} \times T \)
by auto

fix \( A_1 A_2 \) assume points : \( A_1, A_2 : (U - \{ m \}) \cup (U \cap T) A_1 \neq A_2 \)
from points(1,2) have notm : \( A_1 \neq m \neq A_2 \) using assms(2) unfolding IsClosed_def
using mem_not_refl by auto
{
assume \( or : A_1 \in T A_2 \in T \)
with points(3) assms(3) obtain U V where \( UV : U \in V \in T \in T A_1 \in U A_2 \in V \)
\( U \cap W = 0 \) unfolding isT2_def by blast
from UV(1,2) have \( \forall U \in V = \{ (U - \{ m \}) \cup (U \cap T) \} \in (T \cup (U - \{ m \}) \cup (U \cap T) \cup W. (U, W) \in \{ V \in T. m \in V \} \times T) \)
\( \forall (U - \{ m \}) \cup (U \cap T) \in (T \cup (U - \{ m \}) \cup (U \cap T) \cup W. (U, W) \in \{ V \in T. m \in V \} \times T) \) \{ restricted to \( \bigcup T \) \}
unfolding RestrictedTo_def by auto moreover
then have \( \forall U \cap (U - \{ m \}) \cap (U \cap T) \cap (U \cap W) = 0 \) using UV(5) by auto
moreover
from UV(3) or(1) have notm(1) have \( A_1 \in U \cap (U - \{ m \}) \) by auto moreover
from UV(4) or(2) notm(2) have \( A_2 \in U \cap (U - \{ m \}) \) by auto ultimately
have \( \exists U : U \in (T \cup (U - \{ m \}) \cup (U \cap T) \cup W. (U, W) \in \{ V \in T. m \in V \} \times T) \) \{ restricted to \( \bigcup T \) \} \& \( A_1 \in U \in (U - \{ m \}) \cup (U \cap T) \cup W \) \& \( A_2 \in V \in (U \cap T) \cup W \) \& \( (U, W) \in \{ V \in T. m \in V \} \times T \) \{ restricted to \( \bigcup T \) \}
unfolding RestrictedTo_def by auto
then have \( \exists U : U \in (T \cup (U - \{ m \}) \cup (U \cap T) \cup W. (U, W) \in \{ V \in T. m \in V \} \times T) \)

1205
to}\{(\bigcup\{T\}-m)\cup\{T\}\wedge(\exists V. \ V \in T\cup((U\setminus m)\cup\{T\}\cup W. (U,W)\in\{(V\in T. m\in V)\times T\})\}^{\text{restricted to}}\{(\bigcup\{T\}-m)\cup\{T\}\wedge A1\in U\wedge A2\in V\wedge W=0\} \text{ using } \text{ex1[where } x=U\cap(\bigcup\{T\}-m) \text{ and } P=\lambda W.\\ W\in T\cup((U\setminus m)\cup\{T\}\cup W. (U,W)\in\{(V\in T. m\in V)\times T\})\}^{\text{restricted to}}\{(\bigcup\{T\}-m)\cup\{T\}\wedge(\exists V. \ V \in T\cup((U\setminus m)\cup\{T\}\cup W. (U,W)\in\{(V\in T. m\in V)\times T\})\}^{\text{restricted to}}\{(\bigcup\{T\}-m)\cup\{T\}\wedge A1\in U\wedge A2\in V\wedge W=0\} \text{ by blast},\\ \text{then have } \exists U\in T:\{(U\setminus m)\cup\{T\}\cup W. (U,W)\in\{(V\in T. m\in V)\times T\})\}^{\text{restricted to}}\{(\bigcup\{T\}-m)\cup\{T\}\wedge(\exists V. \ V \in T\cup((U\setminus m)\cup\{T\}\cup W. (U,W)\in\{(V\in T. m\in V)\times T\})\}^{\text{restricted to}}\{(\bigcup\{T\}-m)\cup\{T\}\wedge A1\in U\wedge A2\in V\wedge W=0\} \text{ by blast}},\\ \text{moreover},\\ \{\text{assume } A1\not\in\bigcup T, \\
\text{then have } \text{ig} : A1=\bigcup T \text{ using points(1) by auto, }\\ \{\text{assume } A2\not\in\bigcup T, \\
\text{then have } A2=\bigcup T \text{ using points(2) by auto, }\\ \text{with points(3) } \text{ig have False by auto}},\\ \text{then have } \text{ig}_2 : A2\in\bigcup T \text{ by auto moreover, }\\ \text{have } m\in\bigcup T \text{ using asms(2) unfolding IsClosed_def by auto, }\\ \text{moreover note notm(2) asms(3) ultimately obtain } U \ V \text{ where } U\neq\emptyset, V\neq\emptyset \text{ unfolding isT2_def by blast, }\\ \text{from } U\neq\emptyset, V\neq\emptyset \text{ by auto moreover, }\\ \text{have } 0\in T \text{ using empty_open topSpaceAssum by auto ultimately, }\\ \text{have } (U\setminus m)\cup\{T\}\in (\bigcup\{T\}-m)\cup\{T\}\cup W. (U,W)\in\{(V\in T. m\in V)\times T\}) \text{ by auto, }\\ \text{then have } \text{Uop} : (U\setminus m)\cup\{T\}\subseteq (U\setminus m)\cup\{T\}\cup W. (U,W)\in\{(V\in T. m\in V)\times T\}) \text{ by auto, }\\ \text{from } U\neq\emptyset, V\neq\emptyset \text{ have } V\in T \text{ unfolding RestrictedTo_def, }\\ \text{using Vop by blast moreover, }\\ \text{from sub(2) have } (U\setminus m)\cup\{T\}=(U\setminus m)\cup\{T\}\cap((U\setminus m)\cup\{T\}) \text{ by auto, }\\ \text{then have } \text{Uu} : (U\setminus m)\cup\{T\}\subseteq (U\setminus m)\cup\{T\}\cup W. (U,W)\in\{(V\in T. m\in V)\times T\}) \text{ unfolding RestrictedTo_def, }\\ \text{using Uop by blast moreover, }\\ \text{from } U\neq\emptyset, V\neq\emptyset \text{ have } (U\setminus m)\cup\{T\}\cap W=(U\setminus m)\cap W \text{ using mem_not_refl by auto, }\\ 1206
then have \((U-{m}) \cup \{(T)\} \cap V = 0\) using UV(5) by auto

with UV(4) \(V\) ig \(igA2\) have \(\exists V \in (T \cup \{(U-{m})\} \cup \{(T)\}) \cap m = V\) \(V \in T\)\{restricted to\}(\(\{(T-{m})\} \cup \{(T)\})\).

\(A1 \in (U-{m}) \cup \{(T)\} \cup A2 \in V\{((U-{m}) \cup \{(T)\}) \cap 0 = V\} \) by auto

with UV ig have \(\exists U. \ U \in ((U-{m}) \cup \{(T)\}) \cup \{(U-{m})\} \cup \{(T)\} \cup (U, W) \in (V \cap T)\}\{restricted to\}(\(\{(T-{m})\} \cup \{(T)\})\).

\(A1 \in U \cap A2 \in V \cup U = 0\) using exI[where \(x = ((U-{m}) \cup \{(T)\})\)] and

\(P = A1. \ U \in (T \cup ((U-{m}) \cup \{(T)\}) \cup \{(U-{m})\} \cup \{(T)\} \cup W. \ (V, W) \in (V \cap T)\}(\{(U-{m})\} \cup \{(T)\}) \cap (V \cup (U-{m}) \cup \{(T)\}) \cup (U \cup W) \in (V \cap T)\}\{restricted to\}(\(\{(T-{m})\} \cup \{(T)\})\).

\(A1 \in U \cap A2 \in V \cup U = 0\) by auto

then have \(\exists V \in (T \cup ((U-{m}) \cup \{(T)\}) \cup \{(U-{m})\} \cup \{(T)\} \cup W. \ (V, W) \in (V \cap T)\} \{restricted to\}(\(\{(T-{m})\} \cup \{(T)\})\).

\(\exists V \in (T \cup ((U-{m}) \cup \{(T)\}) \cup \{(U-{m})\} \cup \{(T)\} \cup W. \ (V, W) \in (V \cap T)\} \{restricted to\}(\(\{(T-{m})\} \cup \{(T)\})\).

\(A1 \in U \cup A2 \in V \cup U = 0\) by blast

moreover

assume \(A2 \subseteq \{T\}\)

then have \(igA2 = \{T\}\) using points(2) by auto

assume \(A1 \subseteq \{T\}\)

then have \(A1 = \{T\}\) using points(1) by auto

with points(3) \(ig\) have \(False\) by auto

then have \(igA2 : A1 \subseteq \{T\}\) by auto

moreover

have \(m \cup \{T\}\) using assms(2) unfolding IsClosed_def by auto

moreover note \(notm(1)\) assms(3) ultimately obtain \(U \cup \{V\}\) where

\(UV : U \cap V = 0\) unfolding iseT2_def by blast

from UV(1, 3) have \(U \in \{V \cap T. \ m \in \{V\}\}\) by auto

moreover

have \(0 \subseteq \{V\}\) using empty_open topSpaceAssum by auto

ultimately have \((U-{m}) \cup \{(T)\} \in ((U-{m}) \cup \{(T)\} \cup W. \ (V, W) \in (V \cap T)\}\{restricted to\}(\(\{(T-{m})\} \cup \{(T)\})\).

\(m \in U \cap A1 \in V \cup U = 0\) unfolding iseT2_def by blast

from UV(1, 2) have \(U \cap V = 0\) using points(3) by auto

by auto

from UV(1-3, 5) have \(sub: \{V \subseteq \{(T-{m})\} \cup \{(T)\} \subseteq ((U-{m}) \cup \{(T)\}) \subseteq \{(T-{m})\} \cup \{(T)\}\}

by auto

from sub(1) have \(V = ((U-{m}) \cup \{(T)\} \cap W\) by auto

then have \(V V : W \subseteq ((U-{m}) \cup \{(T)\} \cup W. \ (V, W) \in (V \cap T)\}\{restricted to\}(\(\{(T-{m})\} \cup \{(T)\})\).

\(U \cup \{V\}\) unfolding RestrictedTo_def

using Vop by blast moreover

from sub(2) have \((U-{m}) \cup \{(T)\}) \cap (\{(T-{m})\} \cup \{(T)\})\)

by auto

then have \(U U : ((U-{m}) \cup \{(T)\} \in T \cup ((U-{m}) \cup \{(T)\} \cup W. \ (V, W) \in (V \cap T)\}\{restricted to\}(\(\{(T-{m})\} \cup \{(T)\})\), unfolding RestrictedTo_def

using Uop by blast moreover

1207
then have $V \cap (\bigcup (U \setminus \{m\}) \cup \{T\}) = 0$ using $\text{mem_not_refl}$ by auto

with $UU \, \text{UV} (4)$ ig $\text{igA2}$ have $\exists U \in (T \cup (U \setminus \{m\}) \cup \{T\}) \cup W. \, (U, W) \in (V \in T. \, m \in V \times T) \{\text{restricted to} \, (\bigcup (T \setminus \{m\}) \cup \{T\}) \}$

$A1 \in V \cap A2 \in U \setminus V = 0$ by auto

with $VV \, \text{igA2}$ have $\exists U. \, U \in (T \cup (U \setminus \{m\}) \cup \{T\}) \cup W. \, (U, W) \in (V \in T. \, m \in V \times T) \{\text{restricted to} \, (\bigcup (T \setminus \{m\}) \cup \{T\}) \}$

$A1 \cap A2 \in V \setminus U = 0$ using $\text{ext \{where} \, x = V \text{and} \, P = \lambda U. \, U \in (T \cup (U \setminus \{m\}) \cup \{T\}) \cup W. \, (U, W) \in (V \in T. \, m \in V \times T) \{\text{restricted to} \, (\bigcup (T \setminus \{m\}) \cup \{T\}) \}$

$A1 \land A2 \in V \setminus U = 0$ by auto

then have $\exists U \in (T \cup (U \setminus \{m\}) \cup \{T\}) \cup W. \, (U, W) \in (V \in T. \, m \in V \times T) \{\text{restricted to} \, (\bigcup (T \setminus \{m\}) \cup \{T\}) \}$

$\exists V \in (T \cup (U \setminus \{m\}) \cup \{T\}) \cup W. \, (V, W) \in (V \in T. \, m \in V \times T) \{\text{restricted to} \, (\bigcup (T \setminus \{m\}) \cup \{T\}) \}$

$A1 \land A2 \in V \setminus U = 0$ by blast

unfolding $\text{RestrictedTo_def}$ by auto

then have $\bigcup (T \cup (U \setminus \{m\}) \cup \{T\}) \cup W. \, (U, W) \in (V \in T. \, m \in V \times T) \{\text{restricted to} \, (\bigcup (T \setminus \{m\}) \cup \{T\}) \}$

$\bigcup (T \cup (U \setminus \{m\}) \cup \{T\}) \cup W. \, (V, W) \in (V \in T. \, m \in V \times T) \{\text{restricted to} \, (\bigcup (T \setminus \{m\}) \cup \{T\}) \}$

$A1 \land A2 \in V \setminus U = 0$ by auto

ultimately have $\forall A \in (T \cup (U \setminus \{m\}) \cup \{T\}) \cup W. \, (U, W) \in (V \in T. \, m \in V \times T) \{\text{restricted to} \, (\bigcup (T \setminus \{m\}) \cup \{T\}) \}$

by force

with $\text{opp} \, A$ have $\exists Z \in (T \cup (U \setminus \{m\}) \cup \{T\}) \cup W. \, (U, W) \in (V \in T. \, m \in V \times T) \{\text{restricted to} \, Z \} \{\text{is} \, T_2 \}$

by blast

}
ultimately
have \( \exists Z \in (T \cup \{U - \{m\}\} \cup \{U\cup \{V\} \times T\}) \cup \{ \langle U,W \rangle \in \langle V \in T . m \in V \rangle \times T \} \) \{restricted to\} \{is T_2\} by blast

} then have \( \forall A \in (T \cup \{U - \{m\}\} \cup \{U\cup \{V\} \times T\}) \cup \{ \langle U,W \rangle \in \langle V \in T . m \in V \rangle \times T \} \) \{restricted to\} \{is T_2\}
using union_doublepoint_top mc by auto
with topology0.loc_T2 show \( (T \cup \{U - \{m\}\} \cup \{U\cup \{V\} \times T\}) \cup \{ \langle U,W \rangle \in \langle V \in T . m \in V \rangle \times T \} \) \{is locally-T_2\}
unfolding topology0_def using doble_point_top mc by auto
qed

There can be considered many more local properties, which; as happens with locally-T_2; can distinguish between spaces other properties cannot.

end

84 Properties in Topology 3

theory Topology_ZF_properties_3 imports Topology_ZF_7 Finite_ZF_1 Topology_ZF_1b
Topology_ZF_9
Topology_ZF_properties_2 FinOrd_ZF
begin

This theory file deals with more topological properties and the relation with the previous ones in other theory files.

84.1 More anti-properties

In this section we study more anti-properties.

84.2 First examples

A first example of an anti-compact space is the discrete space.

lemma pow_compact_imp_finite:
assumes B{is compact in}Pow(A)
shows Finite(B)
proof-
from assms have B:B\subseteq\forall M\subseteq Pow(Pow(A)). B\subseteq\bigcup M \rightarrow (\\exists N\in FinPow(M). B\subseteq\bigcup N)
unfolding IsCompact_def by auto
from B(1) have \( \{\{x\}. x\in B\} \subseteq Pow(Pow(A)) \) \( B\subseteq\bigcup \{\{x\}. x\in B\} \) by auto
with B(2) have \( \exists N \in FinPow(\{\{x\}. x\in B\}) \) \( B\subseteq\bigcup N \) by auto
then obtain N where \( N \subseteq FinPow(\{\{x\}. x\in B\}) \) \( B\subseteq\bigcup N \) by auto
then have Finite(N) \( N\subseteq\{\{x\}. x\in B\} \) \( B\subseteq\bigcup N \) unfolding FinPow_def by auto
then have Finite(N) \( \forall b\in N. \) Finite(b) \( B\subseteq\bigcup N \) by auto

1209
then have \( B \subseteq \bigcup N \) Finite(\( \bigcup N \)) using Finite_Union[of N] by auto
then show Finite(B) using subset_Finite by auto
qed

**Theorem** \( \text{pow\_anti\_compact} \):

shows \( \text{Pow}(A) \) {is anti-compact}

**Proof**

fix \( B \) assume as: \( B \subseteq \bigcup \text{Pow}(A) \) \( (\bigcup (\text{Pow}(A) \{\text{restricted to} B\})) \) {is compact in \( (\text{Pow}(A) \{\text{restricted to} B\}) \)}
then have sub: \( B \subseteq A \) by auto
then have \( \text{Pow}(B) = \text{Pow}(A) \{\text{restricted to} B\} \) unfolding RestrictedTo_def
by blast
with as(2) have \( (\bigcup \text{Pow}(B)) \) {is compact in \( \text{Pow}(B) \)} by auto
then have \( \text{Finite}(B) \) using pow_compact_imp_finite by auto
then have \( B \) {is in the spectrum of} \( (\lambda T. \ (\bigcup T) \) {is compact in} \( T \)) using compact_spectrum by auto
}
then show thesis unfolding IsAntiComp_def antiProperty_def by auto
qed

In a previous file, Topology_ZF_5.thy, we proved that the spectrum of the lindelöf property depends on the axiom of countable choice on subsets of the power set of the natural number.

In this context, the examples depend on whether this choice principle holds or not. This is the reason that the examples of anti-lindelöf topologies are left for the next section.

### 84.3 Structural results

We first differentiate the spectrum of the lindelöf property depending on some axiom of choice.

**Lemma** \( \text{lindelöf\_spec1} \):

assumes \( \{\text{the axiom of} \ \text{nat} \} \) {choice holds for subsets}(\( \text{Pow}(\text{nat}) \))
shows \( (\lambda \ (\text{is in the spectrum of}) \ (\lambda T. \ (\bigcup T) \) {is lindelöf in} \( T \)) \) \( (\lambda \) \( \text{nat} \)) \( \longleftrightarrow \)

**Lemma** \( \text{lindelöf\_spec2} \):

assumes \( \neg\ (\{\text{the axiom of} \ \text{nat} \} \) {choice holds for subsets}(\( \text{Pow}(\text{nat}) \))
shows \( (\lambda \ (\text{is in the spectrum of}) \ (\lambda T. \ (\bigcup T) \) {is lindelöf in} \( T \)) \) \( \longleftrightarrow \)
\( \text{Finite}(A) \)
proof
assume \( \text{Finite}(A) \)
then have \( A: A \) {is in the spectrum of} \( (\lambda T. \ (\bigcup T) \) {is compact in} \( T \))
using compact_spectrum by auto

1210
have s:nat≤csucc(nat) using le_imp_lesspoll[OF Card_csucc[OF Ord_nat]]

lt_csucc[OF Ord_nat] le_iff by auto

{ fix T assume T{is a topology} (∪T){is compact in}T
  then have (∪T){is compact of cardinal}nat{in}T using Compact_is_card_nat by auto
  then have (∪T){is compact of cardinal}csucc(nat){in}T using s compact_greater_card Card_csucc[OF Ord_nat] by auto
  then have (∪T){is lindeloef in}T unfolding IsLindeloef_def by auto
}
then have ∀T. T{is a topology} −→ (∪T){is compact in}T −→ (∪T){is lindeloef in}T by auto
with A show A {is in the spectrum of} (λT. (∪T){is lindeloef in}T) by auto

next assume A:A {is in the spectrum of} (λT. (∪T){is lindeloef in}T)
then have reg:∀T. T{is a topology} ∧ ∪T≈A −→ (∪T){is compact of cardinal} csucc(nat){in}T using Spec_def unfolding IsCompactOfCard_def by auto
moreover have {x}. x∈A∈Pow(Pow(A)) by auto
moreover have A=∪{x}. x∈A by auto
ultimately have ∃N∈Pow({x}. x∈A). A⊆∪N ∧ N<csucc(nat) by auto
then obtain N where N∈Pow({x}. x∈A) A⊆∪N ∧ N<csucc(nat) by auto
then have N⊆{x}. x∈A) ∧ N<csucc(nat) A⊆∪N using FinPow_def by auto
 { fix t assume t∈{x}. x∈A
  then obtain x where x∈At={x} by auto
  with ∅≤∪N have x∈∪N by auto
  then obtain B where B∈Nx∈B by auto
  with {x}. x∈A> have B={x} by auto
  with t={x}>B∈N have t∈N by auto
  } with {x}. x∈A> have N={x}. x∈A by auto
let B=(x,y). x∈A
from {x}. x∈A> have B:A→N unfolding Pi_def function_def by auto
with {x}. x∈A> have B:inj(A,N) unfolding inj_def using applyEquality by auto
then have A⊆N using lepoll_def by auto
with {x}. x∈A> have A=csucc(nat) using lesspoll_trans1 by auto
then have A⊆nat using Card_less_csucc_eq_le Card_nat by auto

1211
then have $A \prec \text{nat}$ using lepoll_iff_leqpoll by auto moreover

\{
  assume $A \approx \text{nat}$
  then have $\text{nat} \approx A$ using eqpoll_sym by auto
  with $A$ have $\text{nat}$ \{is in the spectrum of\} $(\lambda T. ((\bigcup T) \{\text{is lindeloef in} T\}))$ using equipollent_spect[
    where $P = (\lambda T. ((\bigcup T) \{\text{is lindeloef in} T\}))$] by auto
  moreover
  have $\bigcup \text{Pow}(\text{nat}) = \text{nat}$ by auto
  moreover
  have $\bigcup \text{Pow}(\text{nat}) \approx \text{nat}$ using eqpoll_refl by auto
  ultimately
  have $\text{nat}$ \{is compact of cardinal\} $\text{csucc}(\text{nat}) \{\text{in} \text{Pow}(\text{nat})\}$ using Spec_def unfolding IsLindeloef_def by auto
\}
ultimately have $A \prec \text{nat}$ by auto
then show $\text{Finite}(A)$ using lesspoll_nat_is_Finite by auto
qed

If the axiom of countable choice on subsets of the power set of the natural numbers doesn’t hold, then anti-lindeloef spaces are anti-compact.

\begin{proof}
  have $s : \text{nat} \leq \text{csucc}(\text{nat})$ using le_imp_lesspoll[OF Card_csucc[OF Ord_nat]]
  have $\text{lt}_\text{csucc}(\text{OF Ord_nat}) \leq \text{iff}$ by auto
  \{
    fix $T$ assume $T$ \{is a topology\} $(\bigcup T) \{\text{is compact in} T\}$
    then have $(\bigcup T) \{\text{is compact of cardinal} \text{nat} \{\text{in} T\}$ using Compact_is_card_nat by auto
    then have $(\bigcup T) \{\text{is compact of cardinal} \text{csucc}(\text{nat}) \{\text{in} T\}$ using $s$ compact_greater_card Card_csucc[OF Ord_nat] by auto
    then have $(\bigcup T) \{\text{is lindeloef in} T\}$ unfolding IsLindeloef_def by auto
  }
  then have $\forall T. T$ \{is a topology\} $\rightarrow ((\bigcup T) \{\text{is compact in} T\}) \rightarrow ((\bigcup T) \{\text{is lindeloef in} T\})$ by auto
  from eq_spect_rev_imp_anti[OF this] lindeloef_spec2[OF assms(1)] compact_spectrum show thesis using assms(2) unfolding IsAntiLin_def IsAntiComp_def by auto
\end{proof}

If the axiom of countable choice holds for subsets of the power set of the natural numbers, then there exists a topological space that is anti-lindeloef but no anti-compact.
theorem no_choice_imp_anti_lindeloef_is_anti_comp:
  assumes "{the axiom of} nat {choice holds for subsets}(Pow(nat))"
  shows "{one-point compactification of}Pow(nat){is anti-lindeloef}"
proof-
  have t:="{one-point compactification of}Pow(nat)"=nat\cup nat using topology0.op_compact_total
  unfolding topology0_def using Pow_is_top by auto
  have {nat}¬1 using singleton_eqpoll_1 by auto
  then have {nat}¬nat using n_lesspoll_nat eq_lesspoll_trans by auto
  moreover have s:nat¬csucc(nat) using lt_Card_imp_lesspoll[OF Card_csucc] lt_csucc[OF Ord_nat] by auto
  ultimately have {nat}¬csucc(nat) using lesspoll_trans by blast
  with s have {nat}¬csucc(nat) using less_less_imp_un_less[OF _ _ InfCard_csucc[of InfCard_nat]]
  by auto
  then have {nat}¬nat≤nat using Card_less_csucc_eq_le[OF Card_nat] by auto
  with t have r:"{one-point compactification of}Pow(nat)"≤nat by auto
  { fix A assume A:A"{one-point compactification of}Pow(nat)"
    (("{one-point compactification of}Pow(nat)"{restricted to}A))"{is lindeloef in}(("{one-point compactification of}Pow(nat)"{restricted to}A)
    from A(1) have A"{one-point compactification of}Pow(nat)" by auto
    with s have A≤nat using subset_imp_lepoll lepoll_trans by blast
    then have A"{is in the spectrum of}"(λT. (("{one-point compactification of}Pow(nat)"{restricted to}T))
    using assms lindeloef_spec1 by auto
  }
  then show thesis unfolding IsAntiLin_def antiProperty_def by auto
qed

theorem op_comp_pow_nat_no_anti_comp:
  shows ¬"{one-point compactification of}Pow(nat)"{is anti-compact}"
proof
  let T="{one-point compactification of}Pow(nat)"{restricted to}(nat ∪ nat)
  assume antiComp:"{one-point compactification of}Pow(nat)"{is anti-compact}
  have "(nat ∪ nat){is compact in}"("{one-point compactification of}Pow(nat)"
  unfolding topology0.compact_op[of Pow(nat)] Pow_is_top[of nat] unfolding topology0_def by auto
  then have "(nat ∪ nat){is compact in}T" using compact_imp_compact_subspace
  Compact_is_card_nat by auto
  moreover have T="(nat ∪ nat){is compact in}"("{one-point compactification of}Pow(nat)"
  unfolding RestrictedTo_def by auto
  then have T="(nat ∪ nat){is compact in}T" using topology0.op_compact_total unfolding
  topology0_def using Pow_is_top by auto
  ultimately have "T"{is compact in}T by auto

  1213
with antiComp have \( (\{nat\} \cup nat)\{is in the spectrum of\}(\lambda T. (\bigcup T)\{is compact in\} T) \) unfolding IsAntiComp_def antiProperty_def using topology0.op_compact_total unfolding topology0_def using Pow_is_top by auto
then have Finite(\{nat\} \cup nat) using compact_spectrum by auto
then have Finite(nat) using subset_Finite by auto
then show False using nat_not_Finite by auto
qed

In conclusion, we reached another equivalence of this choice principle.

The axiom of countable choice holds for subsets of the power set of the natural numbers if and only if there exists a topological space which is anti-lindelof but not anti-compact; this space can be chosen as the one-point compactification of the discrete topology on \( \mathbb{N} \).

**Theorem acc_pow_nat_equiv1:**

\[
\{\text{the axiom of}\ \{\text{choice holds for subsets}\}(\mathcal{P}(nat))\} \iff (\{\text{one-point compactification of}\}\mathcal{P}(nat)\{\text{is anti-lindelof}\})
\]

using \( \text{op comp pow nat no anti comp no choice imp anti lindelof is anti comp topology0 no choice imp anti lindelof is anti comp topology0 op comp is top} \) Pow_is_top[of nat] unfolding topology0_def by auto

**Theorem acc_pow_nat_equiv2:**

\[
(\exists T. T\{\text{is a topology}\} \land (T\{\text{is anti-lindelof}\}) \land \neg (T\{\text{is anti-compact}\}))
\]

using \( \text{op comp pow nat no anti comp no choice imp anti lindelof is anti comp topology0 no choice imp anti lindelof is anti comp topology0 op comp is top} \) Pow_is_top[of nat] unfolding topology0_def by auto

In the file Topology_ZF_properties.thy, it is proven that \( \mathbb{N} \) is lindelof if and only if the axiom of countable choice holds for subsets of \( \mathcal{P}(\mathbb{N}) \). Now we check that, in ZF, this space is always anti-lindelof.

**Theorem nat_anti_lindelof:**

\[
\text{shows } \mathcal{P}(nat)\{\text{is anti-lindelof}\}
\]

**Proof:**

\[
\{\text{fix } A \text{ assume } A \in \mathcal{P}(\bigcup \mathcal{P}(nat)) \} (\bigcup (\mathcal{P}(nat)\{\text{restricted to}\} A))\{\text{is lindelof in}\} (\mathcal{P}(nat)\{\text{restricted to}\} A)
\]

from \( A(1) \) have \( A \subseteq nat \) by auto
then have \( \mathcal{P}(nat)\{\text{restricted to}\} A = \mathcal{P}(A) \) unfolding RestrictedTo_def by blast
with \( A(2) \) have \( \text{lin} : A\{\text{is lindelof in}\} \mathcal{P}(A) \) using subset_imp_lepoll by auto
\[
\{\text{fix } T \text{ assume } T\{\text{is a topology}\} \cup T = A
\]
then have \( A = \bigcup T \) using eqpoll_sym by auto
then obtain \( f \) where \( f : f \in \text{bij}(A, \bigcup T) \) unfolding eqpoll_def by auto
then have \( f \in \text{surj}(A, \bigcup T) \) unfolding bij_def by auto

1214
moreover then have \( \text{IsContinuous}(\Pow(A), T, f) \) unfolding \( \text{IsContinuous} \_\text{def} \) surj\_def using func1.1\_L3 by blast

moreover have \( \text{two_top_spaces0}(\Pow(A), T, f) \) unfolding \( \text{two_top_spaces0} \_\text{def} \)
using \( f \_{T(1)} \) Pow\_is\_top unfolding \( \text{bij} \_\text{def} \) \( \text{inj} \_\text{def} \) by auto

ultimately have \( (\bigcup T)(\text{is Lindeloef in} T) \) using \( \text{two_top_spaces0} \_\text{cont_image_com} \)
lin unfolding \( \text{IsLindeloef} \_\text{def} \) by auto

} 
then have \( A \{\text{is in the spectrum of}\} (\lambda T. ((\bigcup T)(\text{is Lindeloef in} T)) \)
unfolding Spec\_def by auto

} 
then show thesis unfolding \( \text{IsAntiLin} \_\text{def} \) \( \text{antiProperty} \_\text{def} \) by auto 
qed

This result is interesting because depending on the different axioms we add to ZF, it means two different things:

- Every subspace of \( \mathbb{N} \) is Lindeloef.
- Only the compact subspaces of \( \mathbb{N} \) are Lindeloef.

Now, we could wonder if the class of compact spaces and the class of Lindeloef spaces being equal is consistent in ZF. Let’s find a topological space which is Lindeloef and no compact without assuming any axiom of choice or any negation of one. This will prove that the class of Lindeloef spaces and the class of compact spaces cannot be equal in any model of ZF.

\text{theorem lord_nat:}
\text{shows \((\text{LOrdTopology nat Le})=\{\text{LeftRayX}(\text{nat,Le,n}). n\in\text{nat}\} \cup \{\text{nat}\} \cup \{0\}\)}
proof-
{ 
  fix \( U \) assume \( U:U \subseteq \{\text{LeftRayX}(\text{nat,Le,n}). n\in\text{nat}\} \cup \{\text{nat}\} \cup \{0\} \neq 0 \)
  { 
    assume \( n\in\text{nat} \)
    with \( U \) have \( \bigcup U=n \) unfolding LeftRayX\_def by auto
    then have \( \bigcup U:U \subseteq \{\text{LeftRayX}(\text{nat,Le,n}). n\in\text{nat}\} \cup \{\text{nat}\} \cup \{0\} \) by auto 
  }
  moreover 
  { 
    assume \( n\notin U \)
    with \( U \) have \( U:U \subseteq \{\text{LeftRayX}(\text{nat,Le,n}). n\in\text{nat}\} \cup \{0\} \) by auto 
    { 
      assume \( A: \exists i. i\in\text{nat} \land \bigcup U \subseteq \text{LeftRayX}(\text{nat,Le,i}) \)
      let \( M=\mu i. i\in\text{nat} \land \bigcup U \subseteq \text{LeftRayX}(\text{nat,Le,i}) \)
      from \( A \) have \( M: M\in\text{nat} \bigcup U \subseteq \text{LeftRayX}(\text{nat,Le,M}) \) using LeastI[OF _ nat_into_Ord, where \( P=\lambda i. i\in\text{nat} \land \bigcup U \subseteq \text{LeftRayX}(\text{nat,Le,i}) \)]
      by auto
      { 
        fix \( y \) assume \( V: y\in\text{LeftRayX}(\text{nat,Le,M}) \)
        then have \( y:y\in\text{nat} \) unfolding LeftRayX\_def by auto
      }
  }
}

1215
{ assume \( \forall V \in U. \ y \notin V \)
then have \( \forall m \in \{ n \in \text{nat} . \ \text{LeftRayX}(\text{nat}, \text{Le}, n) \in U \}. \ y \notin \text{LeftRayX}(\text{nat}, \text{Le}, m) \)
using \text{UU} by \text{auto}
then have \( \forall m \in \{ n \in \text{nat} . \ \text{LeftRayX}(\text{nat}, \text{Le}, n) \in U \}. \ (y, m) \notin \text{Le} \forall y = m \)
unfolding \text{LeftRayX_def} using \( y \)
by \text{auto}
then have \( \forall m \in \{ n \in \text{nat} . \ \text{LeftRayX}(\text{nat}, \text{Le}, n) \in U \}. \ \langle y, m \rangle \notin \text{Le} \lor y = m \)
unfolding \text{LeftRayX_def} using \text{y}
by \text{auto}
then have \( \text{RR}: \forall m \in \{ n \in \text{nat} . \ \text{LeftRayX}(\text{nat}, \text{Le}, n) \in U \}. \ \langle m, y \rangle \in \text{Le} \)
using \text{Le_directs_nat(1)} \text{y}
unfolding \text{IsLinOrder_def \text{IsTotal_def} by blast}

\{
fix \( rr \ \text{V} \) assume \( \text{rr} \in \bigcup U \)
then obtain \( V \) where \( V: V \in U \ \text{rr} \in V \) by \text{auto}
with \text{UU obtain} \( m \) where \( m: V = \text{LeftRayX}(\text{nat}, \text{Le}, m) \ m \in \text{nat} \) by \text{auto}
with \text{V(1) \text{RR}} have \a: \langle m, y \rangle \in \text{Le} \text{a(1)} have \( \langle rr, m \rangle \in \text{Le} \) \text{rr} \in \text{nat} - \{m\} unfolding \text{LeftRayX_def}
by \text{auto}
from \text{a \text{b(1)}} have \( \langle rr, y \rangle \in \text{Le} \) unfolding \text{IsLinOrder_def \text{trans_def by blast}}
moreover
\{
assume \( \text{rr} = y \) 
with \text{a \text{b}} have \text{False} using \text{Le_directs_nat(1) unfolding \text{IsLinOrder_def \text{antisym_def by blast}}}
\}
ultimately have \( \text{rr} \in \text{LeftRayX}(\text{nat}, \text{Le}, y) \) unfolding \text{LeftRayX_def}
using \text{b(2)} by \text{auto}
\}
then have \( \bigcup U \subseteq \text{LeftRayX}(\text{nat}, \text{Le}, y) \) by \text{auto}
with \text{y M(1)} have \( \langle M, y \rangle \in \text{Le} \) using \text{Least_le by auto}
with \text{V have \text{False unfolding \text{LeftRayX_def using \text{Le_directs_nat(1)}}}
unfolding \text{IsLinOrder_def \text{antisym_def by blast}}
\}
then have \( y \in \bigcup U \) by \text{auto}
\}
then have \( \text{LeftRayX}(\text{nat}, \text{Le}, M) \subseteq \bigcup U \) by \text{auto}
with \text{M(2)} have \( \bigcup U = \text{LeftRayX}(\text{nat}, \text{Le}, M) \) by \text{auto}
with \text{M(1)} have \( \bigcup U \subseteq \{ \text{LeftRayX}(\text{nat}, \text{Le}, n). \ n \in \text{nat} \} \cup \{ \text{nat} \} \) by \text{auto}
\}
moreover
\{
assume \( \neg \exists i. i \in \text{nat} \land \bigcup U \subseteq \text{LeftRayX}(\text{nat}, \text{Le}, i) \)
then have \a: \forall i. i \in \text{nat} \rightarrow \neg (\bigcup U \subseteq \text{LeftRayX}(\text{nat}, \text{Le}, i)) by \text{auto}
\{
fix \( i \) assume \( i:i \in \text{nat} \)
with \( \text{A have AA: \neg (\bigcup U \subseteq \text{LeftRayX}(\text{nat}, \text{Le}, i)) by auto}
\{
assume i \notin \bigcup U
then have \( \forall V \in U. i \notin V \) by \text{auto}
\}

1216
then have \( \forall m \in \{n \in \text{nat. LeftRayX(nat, Le, n)} \in \mathcal{U} \}. i \notin \text{LeftRayX(nat, Le, m)} \) by auto

with \( i \) have \( \forall m \in \{n \in \text{nat. LeftRayX(nat, Le, n)} \in \mathcal{U} \}. \neg (i \leq m) \) by auto

unfolding \text{LeftRayX_def} by auto

with \( i \) have \( \forall m \in \{n \in \text{nat. LeftRayX(nat, Le, n)} \in \mathcal{U} \}. \langle i, m \rangle \in \text{Le} \lor i = m \)

unfolding \text{Le_def} by auto

then have \( \forall m \in \{n \in \text{nat. LeftRayX(nat, Le, n)} \in \mathcal{U} \}. m < i \lor m = i \)

using \text{not_le_iff_lt[OF nat_into_Ord[OF \( i \) nat_into_Ord]} by auto

then have \( M: \forall m \in \{n \in \text{nat. LeftRayX(nat, Le, n)} \in \mathcal{U} \}. m \leq i \)

using \text{le_iff[OF nat_into_Ord[OF \( i \)]]} by auto

\{ fix \( s \) assume \( s \in \bigcup \mathcal{U} \)
then obtain \( n \) where \( n:n \in \text{nat} \) \( s \in \text{LeftRayX(nat, Le, n)} \) \( \text{LeftRayX(nat, Le, n)} \in \mathcal{U} \)
using \( \text{UU} \) by auto
with \( M \) have \( n(2) \in \mathcal{U} \)
then have \( i \in \bigcup \mathcal{U} \)
by auto
\}

ultimately have \( \bigcup \mathcal{U} = \{n \in \text{nat} \} \cup \{\text{nat}\} \cup \{0\} \)
by auto

ultimately have \( \bigcup \{\text{LeftRayX(nat, Le, n)}. n \in \text{nat}\} \cup \{\text{nat}\} \cup \{0\} \)
by auto

moreover
\{ fix \( U \) assume \( U = 0 \)
then have \( \bigcup U = \{\text{LeftRayX(nat, Le, n)}. n \in \text{nat}\} \cup \{\text{nat}\} \cup \{0\} \)
by auto
\}

ultimately have \( \forall U. U \subseteq \{\text{LeftRayX(nat, Le, n)}. n \in \text{nat}\} \cup \{\text{nat}\} 
\rightarrow \bigcup U = \{\text{LeftRayX(nat, Le, n)}. n \in \text{nat}\} \cup \{\text{nat}\} \cup \{0\} \)
by auto

then have \( \{\text{LeftRayX(nat, Le, n)}. n \in \text{nat}\} \cup \{\text{nat}\} \cup \{0\} = \{\bigcup U \in \text{Pow}(\{\text{LeftRayX(nat, Le, n)}. n \in \text{nat}\} \cup \{\text{nat}\})\} \)
by blast
then show thesis using \text{LOrdtopology_ROrdtopology_are_topologies(2)[OF Le_directs_nat(1)]}
unfolding \text{IsAbaseFor_def} by auto
lemma countable_lord_nat:
  shows {LeftRayX(nat,Le,n). n∈nat} ∪{nat} ∪{0}≺csucc(nat)
proof-
  { fix e
    have {e}≺1 using singleton_eqpoll_1 by auto
    then have {e}≺nat using n_lesspoll_nat eq_lesspoll_trans by auto
    moreover have s:nat≺csucc(nat) using lt_Card_imp_lesspoll[OF Card_csucc] lt_csucc[OF Ord_nat] by auto
    ultimately have {e}≺csucc(nat) using lesspoll_trans by blast
  } then have {nat} ∪{0}≺csucc(nat) using less_less_imp_un_less[OF _ _ InfCard_csucc[OF InfCard_nat], of {nat} {0}]
    by auto moreover let FF=\{⟨n,LeftRayX(nat,Le,n)⟩. n∈nat\}
    have ff:FF:nat→{LeftRayX(nat,Le,n). n∈nat} unfolding Pi_def domain_def
    function_def by auto
    then have su:FF∈surj(nat,{LeftRayX(nat,Le,n). n∈nat}) unfolding surj_def
    using apply_equality[OF _ ff] by auto
    then have {LeftRayX(nat,Le,n). n∈nat}≺nat using surj_fun_inv_2[OF su lepoll_refl[of nat]] Ord_nat
    by auto
    then have {LeftRayX(nat,Le,n). n∈nat}≺csucc(nat) using Card_less_csucc_eq_le[OF Card_nat] by auto
    ultimately have {LeftRayX(nat, Le, n) . n ∈ nat} ∪ {nat} ∪ {0} ≺ csucc(nat) using less_less_imp_un_less[OF _ _ InfCard_csucc[OF InfCard_nat]] by auto
    moreover have {LeftRayX(nat, Le, n) . n ∈ nat} ∪ {nat} ∪ {0} = {LeftRayX(nat, Le, n) . n ∈ nat} ∪ {nat} ∪ {0} by auto
    ultimately show thesis by auto
qed

corollary lindelof_lord_nat:
  shows nat(is lindeloef in)(LOrdTopology nat Le)
  unfolding IsLindeloef_def using countable_lord_nat lord_nat card_top_comp[OF Card_csucc[OF Ord_nat]]
    union_lordtopology_rordtopology(1)[OF Le_directs_nat(1)] by auto

theorem not_comp_lord_nat:
  shows ¬(nat(is compact in)(LOrdTopology nat Le))
proof
  assume nat(is compact in)(LOrdTopology nat Le)
  with lord_nat have nat(is compact in)\{{LeftRayX(nat,Le,n). n∈nat} ∪\{nat\} ∪\{0\}} by auto
  then have ∀M∈Pow\{{LeftRayX(nat,Le,n). n∈nat} ∪\{nat\} ∪\{0\}}. nat⊆\bigcup M
→ (∃N ∈ FinPow(N). nat ≤ {N})
unfolding IsCompact_def by auto moreover
{
fix n assume n:n∈nat then have n<succ(n) by auto then have (n,succ(n))∈Le n≠succ(n) using n nat_succ_iff by auto then have n∈LeftRayX(nat,Le,succ(n)) unfolding LeftRayX_def using n by auto then have n∈∪({LeftRayX(nat,Le,n). n∈nat}) using n nat_succ_iff by auto
}
ultimately have ∃N ∈ FinPow({LeftRayX(nat,Le,n). n ∈ nat}). nat ≤ {N} unfolding FinPow_def by blast then obtain N where N ∈ FinPow({LeftRayX(nat,Le,n). n ∈ nat}). nat ≤ {N} by auto then have N:N ∈ FinPow({LeftRayX(nat,Le,n). n ∈ nat}). Finite(N) nat ≤ {N} unfolding FinPow_def by auto let F = {⟨n,LeftRayX(nat,Le,n)⟩. n ∈ nat. LeftRayX(nat,Le,n) ∈ N} have ff:F:{m ∈ nat. LeftRayX(nat,Le,m) ∈ N} → N unfolding Pi_def function_def by auto then have F ∈ surj({m ∈ nat. LeftRayX(nat,Le,m) ∈ N}, N) unfolding surj_def using N(1) apply_equality[OF _ ff] by blast moreover
{
fix x y assume xyF:x∈{m∈nat. LeftRayX(nat,Le,m)∈N} y∈{m∈nat. LeftRayX(nat,Le,m)∈N} Fx=Fy then have Fx=LeftRayX(nat,Le,x) Fy=LeftRayX(nat,Le,y) using apply_equality[OF _ ff] by auto with xyF(3) have lxy:LeftRayX(nat,Le,x)=LeftRayX(nat,Le,y) by auto
{
fix r assume r<x then have r≤x r≠x using leI by auto with xyF(1) have r∈LeftRayX(nat,Le,x) unfolding LeftRayX_def using le_in_nat by auto then have r∈LeftRayX(nat,Le,y) using lxy by auto then have r≤y r≠y unfolding LeftRayX_def by auto then have r<y using le_iff by auto
}
then have ∀r. r<x → r<y by auto then have r:¬(y<x) by auto
{
fix r assume r<y then have r≤y r≠y using leI by auto with xyF(2) have r∈LeftRayX(nat,Le,y) unfolding LeftRayX_def using le_in_nat by auto then have r∈LeftRayX(nat,Le,x) using lxy by auto then have r≤x r≠x unfolding LeftRayX_def by auto then have r<x using le_iff by auto
}
then have \( \neg (x < y) \) by auto
with \( r \) have \( x = y \) using not_lt_iff_le[OF nat_into_Ord nat_into_Ord]

\[ xyF(1,2) \]

\[ \text{le_anti_sym} \] by auto

\{ then have \( F \in \text{inj}(\{m \in \text{n}at. \text{LeftRayX}(\text{n}at, \text{Le}, m) \in \text{N}\}, \text{N}) \) unfolding \text{inj_def}

using \( ff \) by auto
ultimately have \( F \in \text{bij}(\{m \in \text{n}at. \text{LeftRayX}(\text{n}at, \text{Le}, m) \in \text{N}\}, \text{N}) \) unfolding \text{bij_def}
by auto

then have \( \{m \in \text{n}at. \text{LeftRayX}(\text{n}at, \text{Le}, m) \in \text{N}\} \approx \text{N} \) unfolding \text{eqpoll_def}
by auto

unfolding \( \text{FinPow_def} \) by auto

\}

\{ fix \( V \) \ and \( r \) assume \( V : \forall V \in \text{N} \ \exists V \in \text{N} \)
then obtain \( m \) where \( m : V = \text{LeftRayX}(\text{n}at, \text{Le}, m) \)
\( \text{LeftRayX}(\text{n}at, \text{Le}, m) \in \text{N} \) \( m \in \text{n}at \)
using \( N(1) \) by auto

with \( V(2) \) have \( xx : \langle r, m \rangle \in \text{Le} \) \( r \neq m \) unfolding \text{LeftRayX_def}
by auto

from \( m(2,3) \) have \( m : \forall m \in \text{n}at. \text{LeftRayX}(\text{n}at, \text{Le}, m) \in \text{N} \)
by auto
then have \( m : \forall m \in \text{n}at. \text{LeftRayX}(\text{n}at, \text{Le}, m) \in \text{N} \) by auto

then have \( xx : \langle r, m \rangle \in \text{Le} \) unfolding \text{le_trans}
by auto

moreover
\{
assume \( r = M \)
with \( xx \) have \( \text{False} \) using \text{le_anti_sym}
by auto
\}
ultimately have \( \exists L \subseteq \text{LeftRayX}(\text{n}at, \text{Le}, M) \)
unfolding \text{LeftRayX_def}
by auto
\}

then have \( \bigcup \text{N} \subseteq \text{LeftRayX}(\text{n}at, \text{Le}, M) \)
by auto

with \( M(2) \) have \( \bigcup \text{N} = \text{LeftRayX}(\text{n}at, \text{Le}, M) \)
by auto

with \( N(3) \) have \( \text{n}at \subseteq \text{LeftRayX}(\text{n}at, \text{Le}, M) \)
by auto

moreover from \( M(1) \) have \( \text{succ}(M) \in \text{n}at \)
using \text{nat_succI}
by auto
ultimately have \( \text{succ}(M) = \text{LeftRayX}(\text{n}at, \text{Le}, M) \)
by auto

then have \( \langle \text{succ}(M), M \rangle \in \text{Le} \)
unfolding \text{LeftRayX_def}
by blast
then show \( \text{False} \) by auto
qed

84.4 More Separation properties

In this section we study more separation properties.
84.5 Definitions

We start with a property that has already appeared in Topology_ZF_1b.thy. A KC-space is a space where compact sets are closed.

definition
IsKC (_ {is KC}) where
T{is KC} ≡ ∀A∈Pow(∪T). A{is compact in}T −→ A{is closed in}T

Another type of space is an US-space; those where sequences have at most one limit.

definition
IsUS (_ {is US}) where
T{is US} ≡ ∀N x y. (N:nat→∪T) ∧ NetConvTop(⟨N,Le⟩,x,T) ∧ NetConvTop(⟨N,Le⟩,y,T) −→ y=x

84.6 First results

The proof in Topology_ZF_1b.thy shows that a Hausdorff space is KC.

corollary (in topology0) T2_imp_KC:
  assumes T{is T2}
  shows T{is KC}
proof-
  { fix A assume A{is compact in}T
    then have A{is closed in}T using in_t2_compact_is_cl assms by auto
  }
then show thesis unfolding IsKC_def by auto
qed

From the spectrum of compactness, it follows that any KC-space is $T_1$.

lemma (in topology0) KC_imp_T1:
  assumes T{is KC}
  shows T{is T1}
proof-
  { fix x assume A:x∈∪T
    have Finite({x}) by auto
    then have {x}{is in the spectrum of}(∪T{is compact in}T)
      using compact_spectrum by auto moreover
    have (T{restricted to}{x}){is a topology} using Top_1_L4 by auto
    moreover have ∪(T{restricted to}{x})={x} using A unfolding RestrictedTo_def by auto
    ultimately have com:{x}{is compact in}(T{restricted to}{x}) unfolding Spec_def
      by auto
    then have {x}{is compact in}T using compact_subspace_imp_compact A by auto
  }

then have \(\{x\}\) closed in \(T\) using assms unfolding IsKC_def using A by auto

then show thesis using T1_iff_singleton_closed by auto

qed

Even more, if a space is KC, then it is US. We already know that for \(T_2\) spaces, any net or filter has at most one limit; and that this property is equivalent with \(T_2\). The US property is much weaker because we don’t know what happens with other nets that are not directed by the order on the natural numbers.

**Theorem (in topology0) KC_imp_US:**

assumes \(T\{\text{is KC}\}\)
show \(T\{\text{is US}\}\)

proof-

\[
\begin{align*}
\text{fix} & \quad N \quad x \quad y \quad \text{assume} \quad A: N : \text{nat} \to \bigcup T \langle N, \text{Le} \rangle \to N \quad x \neq y \\
\text{have} & \quad \text{dir:Le directs nat using Le_directs_nat by auto moreover} \\
\text{moreover} & \quad \text{from A(1) have dom:domain(N)=nat using func1_1_L1 by auto} \\
\text{moreover note} & \quad A(1) \quad \text{ultimately have Net:\langle N, \text{Le} \rangle\{is a net on\}\bigcup T\text{ unfolding IsNet_def by auto} \\
\text{from} & \quad A(3) \quad \text{have y:y}\in\bigcup T\text{ unfolding NetConverges_def[of Net] by auto} \\
\text{from} & \quad A(2) \quad \text{have x:x}\in\bigcup T\text{ unfolding NetConverges_def[of Net] by auto} \\
\text{from} & \quad A(2) \quad \text{have o1:}\forall U\in\text{Pow}(\bigcup T). \quad x\in\text{int}(U) \to (\exists r\in\text{nat.} \quad \forall s\in\text{nat.} \quad \langle r, s \rangle \in \text{Le} \\
\text{by auto} \\
\text{from} & \quad A(4) \quad \text{have o3:}\forall U\in\text{Pow}(\bigcup T). \quad x\in\text{int}(U) \to (\exists r\in\text{nat.} \quad \forall s\in\text{nat.} \quad \langle r, s \rangle \in \text{Le} \\
\text{by auto} \\
\text{from} & \quad B \quad \text{obtain n where n:n} \in\text{nat.} \quad \forall m\in\text{nat.} \quad \langle n, m \rangle \in \text{Le} \to Nm=y \\
\text{have} & \quad \{y\}\{\text{is closed in}\}T \text{ using } y \ T_1\text{ iff singleton closed KC_imp_T1} \\
\text{assms by auto} \\
\text{then have o2:}\bigcup T\text{ unfolding IsClosed_def by auto} \\
\text{then have int(}\bigcup T\text{ unfolding Top_2_L3 by auto} \\
\text{with A(4) x have o3:}\forall U\in\text{Pow}(\bigcup T). \quad x\in\text{int}(U) \to (\exists r\in\text{nat.} \quad \forall s\in\text{nat.} \quad \langle r, s \rangle \in \text{Le} \\
\text{by auto} \\
\text{with o1 o3 obtain r where r:r}\in\text{nat.} \quad \forall s\in\text{nat.} \quad \langle r, s \rangle \in \text{Le} \to Ns\in\bigcup T\text{ unfold-} \\
\text{ing IsDirectedSet_def by auto} \\
\text{with r(2) n(2) have Nz}\in\bigcup T\text{ unfolding } IsDirectedSet_def \text{ by auto} \\
\text{then have False by auto} \\
\end{align*}
\]

then have reg:\(\forall n\in\text{nat.} \quad \exists m\in\text{nat.} \quad Nm=y \ \wedge \ \langle n, m \rangle \in \text{Le} \) by auto

let \(NN=\{\langle n, N(\mu i. Ni=y \ \wedge \ \langle n, i \rangle \in \text{Le} \rangle. \quad n\in\text{nat}\}\) 

\[
\begin{align*}
& \text{fix x z assume A1:} \langle x, z \rangle \in NN \\
& \text{fix y' assume A2:} \langle x, y' \rangle \in NN
\end{align*}
\]
with A1 have z=y' by auto

} then have \( \forall y'. (x,y') \in \mathbb{N}N \rightarrow z=y' \) by auto

} then have \( \forall x \ z. (x, z) \in \mathbb{N} \rightarrow (\forall y'. (x,y') \in \mathbb{N}N \rightarrow z=y') \) by auto

moreover

\{ fix n assume as: n \in \mathbb{N}nat
with reg obtain m where N\(\neq y \land (n,m) \in \mathbb{N} \rightarrow \mathbb{N}nat \) by auto
then have LI: N(\mu i. N\(\neq y \land (n,i) \in \mathbb{N} \rightarrow \mathbb{N}nat \) by auto
\(\neq y \land (n,m) \in \mathbb{N}nat \) by auto
using LeastI[of \(\lambda m. N\(\neq y \land (n,m) \in \mathbb{N}nat \) by auto
\) nat_into_Ord[of m] by auto
then have \(\lambda i. N\(\neq y \land (n,i) \in \mathbb{N}nat \) by auto
then have N(\mu i. N\(\neq y \land (n,i) \in \mathbb{N}nat \) by auto
using apply_type[of A(1)]
by auto
with as have \(\langle n,N(\mu i. N\(\neq y \land (n,i) \in \mathbb{N}nat \) \rangle \in \mathbb{N}nat \times J \) by auto

\} then have \(\forall y \in \mathbb{N}nat. \ N\(\neq y \land \langle n,i \rangle \in \mathbb{N}nat \) by auto
have dom2: domain(\(\mathbb{N}nat \rightarrow J \) unfolding Pi_def function_def domain_def
by auto
\{ fix n assume as: n \in \mathbb{N}nat
with reg obtain m where N\(\neq y \land (n,m) \in \mathbb{N} \rightarrow \mathbb{N}nat \) by auto
then have LI: N(\mu i. N\(\neq y \land (n,i) \in \mathbb{N}nat \) \(\neq y \land \langle n,m \rangle \in \mathbb{N}nat \) \) nat_into_Ord[of m] by auto
then have N\(\neq y \land (n,m) \in \mathbb{N}nat \) using apply_type[of A _ NFUn] by auto
\} then have noy: \(\forall n \in \mathbb{N}nat. \ N\(\neq y \land \langle n,i \rangle \in \mathbb{N}nat \) by auto
have dom2: domain(\(\mathbb{N}nat \rightarrow J \) unfolding Pi_def function_def domain_def
by auto
\{ fix U assume U \in Pow(J \) x \in int(U)
then have \(\exists r \in \mathbb{N}nat. \ \forall s \in \mathbb{N}nat. \ (r,s) \in \mathbb{N}nat \rightarrow Ns \in U \) using o1 by auto
then obtain r where r_def: r \in \mathbb{N}nat \ \forall s \in \mathbb{N}nat. \ (r,s) \in \mathbb{N}nat \rightarrow Ns \in U \) by auto
\{ fix s assume AA: \(r,s) \in \mathbb{N}nat \) with reg obtain m where N\(\neq y \land \langle s,m \rangle \in \mathbb{N}nat \) by auto
then have \(\langle s, \mu \rangle. N\(\neq y \land \langle s,i \rangle \in \mathbb{N}nat \) using LeastI[of \(\lambda m. N\(\neq y \land \langle s,m \rangle \in \mathbb{N}nat \) \) nat_into_Ord by auto
with AA have \(\langle r, \mu \rangle. N\(\neq y \land \langle s,i \rangle \in \mathbb{N}nat \) using le_trans by auto
with r_def(2) have N(\mu i. N\(\neq y \land \langle s,i \rangle \in \mathbb{N}nat \) \(\neq y \land (s,m) \in \mathbb{N}nat \) \) nat_into_Ord by auto
then have NNs \(\neq y \land \langle s,m \rangle \in \mathbb{N}nat \) \) nat_into_Ord by auto
\) using apply_type[of A _ NFUn] AA by auto
\} then have \(\forall s \in \mathbb{N}nat. \ (r,s) \in \mathbb{N}nat \rightarrow NNs \in U \) by auto
with r_def(1) have \(\exists r \in \mathbb{N}nat. \ \forall s \in \mathbb{N}nat. \ (r,s) \in \mathbb{N}nat \rightarrow NNs \in U \) by auto

1223
then have \(\text{conv2} : (\NN, \Le) \to \undermap N\times \text{unfolding NetConverges_def[OF net2]}\) using \(x \in \text{dom2}\) by auto
let \(A = \{x\} \cup \NN\text{nat}\)

\[
\begin{align*}
\text{fix } M & \quad \text{assume } \text{Acov}: A \subseteq \bigcup M \subseteq T \\
\text{then have } x \in \bigcup M & \quad \text{by auto} \\
\text{then obtain } U & \quad \text{where } U \subseteq \text{int}(U) \text{ by auto} \\
\text{with } \text{Acov}(2) & \quad \text{have } U = \text{int}(U) \text{ by auto} \\
\text{with } \text{conv2 obtain } r & \quad \text{where } r \in \text{nat} \quad \forall s \in \text{nat}. \quad \langle r, s \rangle \in \Le \to \NN s \in U \\
\text{unfolding NetConverges_def[OF net2]} & \quad \text{using } \text{dom2 } UT \text{ by auto} \\
\text{have } N\text{resFun}: \text{restrict}(\NN, \{n \in \text{nat}. \quad \langle n, r \rangle \in \Le\}): \{n \in \text{nat}. \quad \langle n, r \rangle \in \Le\} \to \bigcup T & \quad \text{using restrict_fun} \\
\text{[OF NFun, of } \{n \in \text{nat}. \quad \langle n, r \rangle \in \Le\}\] & \quad \text{by auto} \\
\text{then have } \text{limit}(\NN, \{n \in \text{nat}. \quad \langle n, r \rangle \in \Le\}) & \quad \text{using } \text{fun_is_surj} \text{ by auto} \\
\text{moreover have } \{n \in \text{nat}. \quad \langle n, 0 \rangle \in \Le\} & \quad \{0\} \text{ by auto} \\
\text{then have } \text{Finite}(\{n \in \text{nat}. \quad \langle n, 0 \rangle \in \Le\}) & \quad \text{by auto} \\
\text{moreover } \{n \in \text{nat}. \quad \langle n, j \rangle \in \Le\} & \quad \text{by auto} \\
\text{fix } j & \quad \text{assume } \text{as}: j \in \text{nat} \text{ Finite}(\{n \in \text{nat}. \quad \langle n, j \rangle \in \Le\}) \\
\text{fix } t & \quad \text{assume } t \in \{n \in \text{nat}. \quad \langle n, \text{succ}(j) \rangle \in \Le\} \\
\text{then have } t \in \text{nat} \quad \{t, \text{succ}(j) \rangle \in \Le \text{ by auto} \\
\text{then have } t \leq \text{succ}(j) \text{ by auto} \\
\text{then have } t \subseteq \text{succ}(j) \text{ using le_imp_subset by auto} \\
\text{then have } t \subseteq j \cup \{j\} \text{ using succ_explained by auto} \\
\text{then have } j \in \text{nat} \quad \text{by auto} \\
\text{then have } t \leq j \text{ using subset_imp_le } \langle t \in \text{nat} \quad \langle j \in \text{nat} \quad \text{nat_into_Ord} \text{ by auto} \\
\text{then have } j \cup \{j\} \subseteq t \leq j \text{ using } \langle t \in \text{nat} \quad \langle j \in \text{nat} \quad \text{nat_into_Ord} \text{ unfolding Ord_def} \\
\text{Transset_def by auto} \\
\text{then have } \text{succ}(j) \subseteq t \leq j \text{ using succ_explained by auto} \\
\text{with } \langle t \subseteq \text{succ}(j) \rangle & \quad \text{have } t = \text{succ}(j) \quad \forall t \leq j \text{ by auto} \\
\text{with } \langle t \in \text{nat} \quad \langle j \in \text{nat} \rangle & \quad \text{have } t \in \{n \in \text{nat}. \quad \langle n, j \rangle \in \Le\} \cup \{\text{succ}(j)\} \\
\text{by auto} \\
\text{then have } \{n \in \text{nat}. \quad \langle n, \text{succ}(j) \rangle \in \Le\} \subseteq \{n \in \text{nat}. \quad \langle n, j \rangle \in \Le\} \cup \{\text{succ}(j)\} \text{ by auto} \\
\text{moreover have } \text{Finite}(\{n \in \text{nat}. \quad \langle n, j \rangle \in \Le\} \cup \{\text{succ}(j)\}) & \quad \text{using as(2)} \\
\text{Finite_cons} \text{ by auto} \end{align*}
\]

1224
ultimately have Finite({n ∈ nat. ⟨n, succ(j)⟩ ∈ Le})
by auto
}
then have ∀ j ∈ nat. Finite({n ∈ nat. ⟨n, j⟩ ∈ Le})
ultimately have Finite(range(restrict(NN, {n ∈ nat. ⟨n, r⟩ ∈ Le})))
ultimately have ∀ j ∈ nat. Finite({n ∈ nat. ⟨n, j⟩ ∈ Le})
−→ Finite({n ∈ nat. ⟨n, succ(j)⟩ ∈ Le})
by auto
ultimately have Finite((restrict(NN, {n ∈ nat. ⟨n, r⟩ ∈ Le})){n ∈ nat. ⟨n, r⟩ ∈ Le})
using range_image_domain[of NFun]
by auto
ultimately have (NN{n ∈ nat. ⟨n, r⟩ ∈ Le}){is in the spectrum of}(T{restricted to}NN{n ∈ nat . ⟨n, r⟩ ∈ Le})
ultimately have (NN{n ∈ nat . ⟨n, r⟩ ∈ Le}){is compact in}(T{restricted to}NN{n ∈ nat . ⟨n, r⟩ ∈ Le})
ultimately have (NN{n ∈ nat . ⟨n, r⟩ ∈ Le}){is compact in}T using compact_subspace_imp_compact
moreover have (NN{n ∈ nat . ⟨n, r⟩ ∈ Le})⊆∪M by auto
moreover note Acov(2) ultimately
obtain M where M:ℜ∈FinPow(M) (NN{n ∈ nat . ⟨n, r⟩ ∈ Le})⊆∪M
ultimately have M ∪ U∈FinPow(M) using U(2) unfolding FinPow_def
by auto
moreover

{ fix s assume s:s∈A s∉U
with U(1) have s∈NNnat by auto
then have s∈{NNn. n∈nat} using func_imagedef[of NFun] by auto
then obtain n where n:n∈nat s=NNn by auto

{ assume ⟨r,n⟩∈Le
with rr have NNNn∈U by auto
with n(2) s(2) have False by auto
}
then have ⟨r,n⟩∉Le by auto

1225
with rr(1) n(1) have ¬(r≤n) by auto
then have n≤r using Ord_linear_le[where thesis=(n,r)∈Le] nat_into_Ord[OF 
rr(1)]

    nat_into_Ord[OF n(1)] by auto
with rr(1) n(1) have (n,r)∈Le by auto
with n(2) have s∈{(NNt. t∈(n∈nat. (n,r)∈Le))} by auto moreover
have {n∈nat. (n,r)∈Le}⊆nat by auto
ultimately have s∈NN{n∈nat. (n,r)∈Le} using func_imagedef[OF NFun]
by auto
with M(2) have s∈∪M by auto
}
then have A⊆∪M ∪ U by auto
then have A⊆(∪M ∪ {U}) by auto ultimately
have ∃M∈FinPow(M). A⊆∪M by auto
}
then have ∀M∈Pow(T). A⊆∪M → (∃M∈FinPow(M). A⊆∪M) by auto moreover
have A⊆∪T using func_1_L6(2)[OF NFun] x by blast ultimately
have A is compact in)T unfolding IsCompact_def by auto
with assms have A is closed in)T unfolding IsKc_def IsCompact_def
by auto
then have ∪T-A∈T unfolding IsClosed_def by auto
then have ∪T-A=int(∪T-A) using Top_2_L3 by auto moreover
{
    assume y∈A
    with A(4) have y∈NNnat by auto
    then have y∈(NNn. n∈nat) using func_imagedef[OF NFun] by auto
    then obtain n where n∈natNNn=y by auto
    with n oy have False by auto
}
with y have y∈∪T-A by force ultimately
have y∈int(∪T-A) ∪ T-A Pow(∪T) by auto moreover
have (∀U∈Pow(∪T). y ∈ int(U) → (∃t∈nat. ∀m∈nat. (t, m) ∈ Le
→ N m ∈ U ))
using A(3) dom unfolding NetConverges_def[OF Net] by auto
ultimately have ∃t∈nat. ∀m∈nat. (t, m) ∈ Le → N m ∈ ∪T-A by blast
then obtain r where r_def:r∈nat ∀s∈nat. (r,s)∈Le → Ns∈∪T-A by auto
{
fix s assume AA:(r,s)∈Le
    with reg obtain m where Nm≠y (s,m)∈Le by auto
    then have (s,μ i. N≠y ∧ (s,i)∈Le)∈Le using LeastI[of μm.] Nm≠y
∧ (s,m)∈Le m]
nat_into_Ord by auto
with AA have (r,μ i. N≠y ∧ (s,i)∈Le)∈Le using le_trans by auto
with r_def(2) have N(μ i. N≠y ∧ (s,i)∈Le)∈∪T-A by force
then have NNN∈∪T-A using apply_equal[OF _ NFun] AA by auto
moreover have NNN∈(NNt. t∈nat} using AA by auto
then have \( \text{NNs} \in \text{NNnat} \) using func_imagedef[OF NFun] by auto
then have \( \text{NNs} \in A \) by auto
ultimately have \( \text{False} \) by auto

moreover have \( r \subseteq \text{succ}(r) \) using succ_explained by auto
then have \( r \leq \text{succ}(r) \) using subset_imp_le nat_into_Ord \( \langle r \in \text{nat} \rangle \) nat_succI by auto
ultimately have \( \text{False} \) by auto

then have \( \forall N \ x \ y. (N : \text{nat} \rightarrow \bigcup T) \land (\langle N, \text{Le} \rangle \rightarrow_N x \ {\in} T) \land (\langle N, \text{Le} \rangle \rightarrow_N y \ {\in} T) \rightarrow x = y \) by auto
then show thesis unfolding IsUS_def by auto
qed

US spaces are also \( T_1 \).

**Theorem** (in topology0) US_imp_T1:
assumes \( T \{\text{is US}\} \)
shows \( T \{\text{is T}_1\} \)
proof-

- \{ fix \( x \) assume \( x : x \in \bigcup T \)
then have \( \{x\} \subseteq \bigcup T \) by auto

  - \{ fix \( y \) assume \( y : y \neq x \ y \in \text{cl}({\{x\}}) \)
then have \( r: \forall U \in T. y \in U \rightarrow x \in U \) using cl_inter_neigh[OF \( \langle \{x\} \subseteq \bigcup T \rangle \)] by auto

  let \( N = \text{ConstantFunction}(\text{nat},x) \)
  have fun: \( N : \text{nat} \rightarrow \bigcup T \) using x func1_3_L1 by auto
  then have dom: domain\( (N) = \text{nat} \) using func1_1_L1 by auto
  with fun have Net: \( \langle N, \text{Le} \rangle \{\text{is a net on}\} \bigcup T \) using Le_directs_nat unfolding IsNet_def by auto

  \{ fix \( U \) assume \( U \in \text{Pow}(\bigcup T) \ x \in \text{int}(U) \)
then have \( x \in U \) using Top_2_L1 by auto
then have \( \forall n : \text{nat}. \ Nn \in U \) using func1_3_L2 by auto
then have \( \forall n : \text{nat}. \ (0,n) \in \text{Le} \rightarrow Nn \in U \) by auto
then have \( \exists r : \text{nat}. \ \forall n : \text{nat}. \ (r,n) \in \text{Le} \rightarrow Nn \in U \) by auto
\}
then have \( \langle N, \text{Le} \rangle \rightarrow_N x \) unfolding NetConverges_def[OF Net] using x dom by auto
moreover

- \{ fix \( U \) assume \( U \in \text{Pow}(\bigcup T) \ y \in \text{int}(U) \)
then have \( x \in \text{int}(U) \) using r Top_2_L2 by auto
then have \( x \in U \) using Top_2_L1 by auto
then have \( \forall n : \text{nat}. \ Nn \in U \) using func1_3_L2 by auto
then have \( \forall n : \text{nat}. \ (0,n) \in \text{Le} \rightarrow Nn \in U \) by auto
\}

1227
then have $\exists r \in \text{nat. } \forall n \in \text{nat. } (r,n) \in \text{Le} \rightarrow N_n \in U$ by auto

} then have $(N,\text{Le}) \rightarrow N$ y unfolding NetConverges_def[OF Net] using y(2) dom

Top_3_L11(1)[OF $\langle x \rangle \subseteq \bigcup T$] by auto

ultimately have $x = y$ using assms unfolding IsUS_def using fun by auto

with y(1) have False by auto

} then show thesis using T1_iff_singleton_closed by auto

qed

84.7 Counter-examples

We need to find counter-examples that prove that this properties are new ones.

We know that $T_2 \Rightarrow \text{loc.} T_2 \Rightarrow \text{anti-hyperconnected} \Rightarrow T_1$ and $T_2 \Rightarrow \text{KC} \Rightarrow \text{US} \Rightarrow T_1$. The question is: What is the relation between KC or US and, loc.T_2 or anti-hyperconnected?

In the file Topology_ZF_properties_2.thy we built a topological space which is locally-T_2 but no T_2. It happens actually that this space is not even US given the appropriate topology T.

lemma (in topology0) locT2_not_US_1:
assumes $\{m\} \notin T$ $\{m\}$ is closed in $T$ $\exists N \in \text{nat} \rightarrow \bigcup T. (\langle N,\text{Le} \rangle \rightarrow N m \m / \in N_{\text{nat}}$ shows $\exists N \in \text{nat} \rightarrow \bigcup T \{m\} \subseteq \{m\}$ by auto

} then have $\{m\}$ is closed in $T$ using Top_3_L8 x by auto

with y(1) have False by auto

} then show thesis using T1_iff_singleton_closed by auto

qed

then have $\exists r \in \text{nat. } \forall n \in \text{nat. } (r,n) \in \text{Le} \rightarrow N_n \in U$ by auto

} then have $(N,\text{Le}) \rightarrow N$ y unfolding NetConverges_def[OF Net] using y(2) dom

Top_3_L11(1)[OF $\langle x \rangle \subseteq \bigcup T$] by auto

ultimately have $x = y$ using assms unfolding IsUS_def using fun by auto

with y(1) have False by auto

} then show thesis using T1_iff_singleton_closed by auto

qed

84.7 Counter-examples

We need to find counter-examples that prove that this properties are new ones.

We know that $T_2 \Rightarrow \text{loc.} T_2 \Rightarrow \text{anti-hyperconnected} \Rightarrow T_1$ and $T_2 \Rightarrow \text{KC} \Rightarrow \text{US} \Rightarrow T_1$. The question is: What is the relation between KC or US and, loc.T_2 or anti-hyperconnected?

In the file Topology_ZF_properties_2.thy we built a topological space which is locally-T_2 but no T_2. It happens actually that this space is not even US given the appropriate topology T.
unfolding NetConverges_def[OF Net2] using dom by auto
{
    fix U assume \( U \subseteq \text{Pow}(\bigcup \{ \text{U}(U-\{m\}) \cup \bigcup T \cup T. m \in \{\forall V. m \in V \times T\}) \)
    \( m \in \text{Interior}(U,T) \cup \bigcup T \cup T. m \in \{\forall V. m \in V \times T\} \)
    let \( I = \text{Interior}(U,T) \cup \bigcup T \cup T. m \in \{\forall V. m \in V \times T\} \)
    have \( I \in \text{U}(U-\{m\}) \cup \bigcup T \cup T. m \in \{\forall V. m \in V \times T\} \) using topology0.Top_2_L2
    assms(2) doble_point_top unfolding topology0_def by blast
    then have \( (\bigcup T) \cap I \subseteq (\bigcup T) \cap I \) using Top_2_L3 by auto
    with \( U(2) \) assms(2) have \( m \in \text{int}(\bigcup T \cap I \cap I) \) unfolding isClosed_def by auto
    moreover note \( R \) ultimately have \( \exists r \in \text{nat}. \ \forall s \in \text{nat}. \ (r,s) \in \text{Le} \rightarrow \text{Ns} \in (\bigcup T) \cap I \)
    by blast
    then have \( \exists r \in \text{nat}. \ \forall s \in \text{nat}. \ (r,s) \in \text{Le} \rightarrow \text{Ns} \in U \) by auto
    moreover have \( \text{tt}_0_\text{topology0.}\text{Top}_2_\text{L1}\{\text{of} \}
    \( T \cup \{\text{U}(U-\{m\}) \cup \bigcup T \cup T. m \in \{\forall V. m \in V \times T\}) \) double_point_top assms(2)
    unfolding topology0_def by auto
}

then have \( \forall U \in \text{Pow}(\bigcup \{ \text{U}(U-\{m\}) \cup \bigcup T \cup T. m \in \{\forall V. m \in V \times T\}) \)
    \( m \in \text{Interior}(U,T) \cup \bigcup T \cup T. m \in \{\forall V. m \in V \times T\} \)
    using double_point_top[of assms(2)] unfolding topology0_def.
    have \( m \in (\bigcup T \cup T. m \in \{\forall V. m \in V \times T\}) \) using assms(2)
    unfolding double_point_top unfolding topology0_def by auto ultimately
    have \( \text{con1} : \langle N, \text{Le}_0 \rangle \rightarrow \text{N} \ \forall n \ \{\text{in} \} \ (\bigcup T \cup \{\text{U}(U-\{m\}) \cup \bigcup T \cup T. m \in \{\forall V. m \in V \times T\}) \) unfolding topology0.NetConverges_def[OF \( tt \text{ Net} \]
    using dom by auto
{
    fix U assume \( U \subseteq \text{Pow}(\bigcup \{ \text{U}(U-\{m\}) \cup \bigcup T \cup T. m \in \{\forall V. m \in V \times T\}) \)
    \( U \in T \subseteq \text{Interior}(U,T) \cup \bigcup T \cup T. m \in \{\forall V. m \in V \times T\} \)
    let \( I = \text{Interior}(U,T) \cup \bigcup T \cup T. m \in \{\forall V. m \in V \times T\} \)
    have \( I \in \text{U}(U-\{m\}) \cup \bigcup T \cup T. m \in \{\forall V. m \in V \times T\} \) using topology0.Top_2_L2
    assms(2) doble_point_top unfolding topology0_def by blast
    with \( U(2) \) mem_not_ref1 have \( I \in \{U-\{m\}) \cup \bigcup T \cup T. m \in \{\forall V. m \in V \times T\} \)
    by auto
    then obtain \( V \ \forall W \in \text{VW} \) where \( \forall W : I = (V-\{m\}) \cup \bigcup T \cup T. m \in \{\forall V. m \in V \times T\} \) by auto
    from \( (3,4) \) have \( m \in \text{int}(V) \) using Top_2_L3 by auto moreover
    have \( V \in \text{Pow}(\bigcup T) \) using \( \forall W \) by auto moreover
    note \( R \) ultimately
    have \( \exists r \in \text{nat}. \ \forall s \in \text{nat}. \ (r,s) \in \text{Le} \rightarrow \text{Ns} \in V \) by blast moreover
    from \( (3) \) have \( \forall s \in \text{nat}. \ (r,s) \in \text{Le} \rightarrow \text{Ns} \neq m \) using func_imagedef[OF \( N(1) \)] by auto
    ultimately
    have \( \exists r \in \text{nat}. \ \forall s \in \text{nat}. \ (r,s) \in \text{Le} \rightarrow \text{Ns} \in V-\{m\} \) by blast
    then have \( \exists r \in \text{nat}. \ (r,s) \in \text{Le} \rightarrow \text{Ns} \in W(1) \) by auto
    then have \( \exists r \in \text{nat}. \ (r,s) \in \text{Le} \rightarrow \text{Ns} \in U \) using topology0.Top_2_L1
    assms(2) doble_point_top unfolding topology0_def by blast

1229
then have \( \forall \mathcal{U} \in \text{Pow}(\bigcup \mathcal{T} \cup \{\mathcal{U} - \{m\}\} \cup \{\mathcal{U} \cup \mathcal{W} \mid (\mathcal{U}, \mathcal{W}) \in \{V \in \mathcal{T} \mid m \in V\} \times \mathcal{T})\} \). 

Moreover have \( \bigcup \mathcal{T} \in \text{Interior}(\mathcal{U}, \mathcal{T} \cup \{\mathcal{U} - \{m\}\} \cup \{\mathcal{U} \cup \mathcal{W} \mid (\mathcal{U}, \mathcal{W}) \in \{V \in \mathcal{T} \mid m \in V\} \times \mathcal{T})\} \) using assms(2) union_doublepoint_top by auto ultimately

have \( (\langle N, \text{Le} \rangle \to N \bigcup \mathcal{T} \in \text{in}(\mathcal{T} \cup \{\mathcal{U} - \{m\}\} \cup \{\mathcal{U} \cup \mathcal{W} \mid (\mathcal{U}, \mathcal{W}) \in \{V \in \mathcal{T} \mid m \in V\} \times \mathcal{T})\}) \) unfolding topology0.NetConverges_def[OF tt Net] using dom by auto

with con1 fun show thesis by auto qed

Corollary (in topology0) locT2_not_US_2: 

assumes \( \{m\} \not\in \mathcal{T} \{\text{is closed in}\} \mathcal{T} \exists N \in \text{nat} \to \mathcal{T} . (\langle N, \text{Le} \rangle \to N \mathcal{m} \wedge \mathcal{m} \not\in \mathcal{Nnat}) \)

shows \( \neg ((\mathcal{T} \cup \{\mathcal{U} - \{m\}\} \cup \{\mathcal{U} \cup \mathcal{W} \mid (\mathcal{U}, \mathcal{W}) \in \{V \in \mathcal{T} \mid m \in V\} \times \mathcal{T})\} \{\text{is US}\}) \)

proof-

have \( \mathcal{m} \not\in \bigcup \mathcal{T} \) using assms(2) mem_not_refl unfolding IsClosed_def by auto 

then show thesis using locT2_not_US_1 assms unfolding IsUS_def by blast qed

In particular, we also know that a locally-\( T_2 \) space doesn’t need to be KC; since KC \( \Rightarrow \) US. Also we know that anti-hyperconnected spaces don’t need to be KC or US, since locally-\( T_2 \) \( \Rightarrow \) anti-hyperconnected.

Let’s find a KC space that is not \( T_2 \), an US space which is not KC and a \( T_1 \) space which is not US.

First, let’s prove some lemmas about what relation is there between this properties under the influence of other ones. This will help us to find counter-examples.

Anti-compactness erases the differences between several properties.

Lemma (in topology0) anticompact_KC_equiv_T1: 

assumes \( \mathcal{T} \{\text{is anti-compact}\} \)

shows \( \mathcal{T} \{\text{is KC}\} \to \mathcal{T} \{\text{is T}_1\} \)

proof

assume \( \mathcal{T} \{\text{is KC}\} \)

then show \( \mathcal{T} \{\text{is T}_1\} \) using KC_imp_T1 by auto

next

assume AS: \( \mathcal{T} \{\text{is T}_1\} \)

{ 

fix \( A \) assume A:A\{is compact in\} \( \mathcal{T} \ A \in \text{Pow}(\bigcup \mathcal{T}) \)

then have \( A \{\text{is compact in}\} (\mathcal{T} \{\text{restricted to}\} A) \ A \in \text{Pow}(\bigcup \mathcal{T}) \) using compact_imp_compact_subspace Compact_is_card_nat by auto

moreover then have \( \bigcup (\mathcal{T} \{\text{restricted to}\} A) = A \) unfolding RestrictedTo_def by auto

ultimately have \( (\bigcup (\mathcal{T} \{\text{restricted to}\} A) \{\text{is compact in}\} (\mathcal{T} \{\text{restricted to}\} A) \ A \in \text{Pow}(\bigcup \mathcal{T}) \) by auto

with assms have Finite(A) unfolding IsAntiComp_def antiProperty_def using compact_spectrum by auto

1230
then obtain \( n \) where \( n \in \text{nat} \) unfolding Finite_def by auto
then have \( A < \text{nat} \) using eq_lesspoll_trans n_lesspoll_nat by auto moreover
  have \( \bigcup T - (\bigcup T - A) = A \) using A(2) by auto
  ultimately have \( \bigcup T - (\bigcup T - A) < \text{nat} \) by auto
then have \( \bigcup T - A \in \text{CoFinite} \) unfolding Cofinite_def CoCardinal_def
by auto
then have \( \bigcup T - A \in T \) unfolding IsClosed_def
by auto
moreover
  have \( \bigcup T - A \in T \) using AS T1_cocardinal_coarser by auto
  with A(2) have \( A \in \text{is closed in} T \) unfolding IsClosed_def by auto
then show \( T \in \text{is KC} \) unfolding IsKC_def by auto
qed

Then if we find an anti-compact and \( T_1 \) but no \( T_2 \) space, there is a counter-example for \( KC \Rightarrow T_2 \). A counter-example for US doesn't need to be KC mustn't be anti-compact.

The cocountable topology on \( \text{csucc}(\text{nat}) \) is such a topology.

The cocountable topology on \( \mathbb{N}^+ \) is hyperconnected.

**Lemma**: \( \text{c cocountable in csucc nat HConn} \):

shows \( (\text{CoCountable csucc(nat)}) \{\text{is hyperconnected}\} \)

**proof**-

\[
\begin{align*}
\{ & \text{fix } U \text{ V assume as:} U \in (\text{CoCountable csucc(nat)}) \land V \in (\text{CoCountable csucc(nat)}) \land U \cap V = 0 \\
& \text{then have csucc(nat) } \setminus U < \text{csucc(nat)} \cup U = 0 \text{ csucc(nat)} \setminus V < \text{csucc(nat)} \cup V = 0 \text{ unfolding CoCardinal_def by auto } \\
& \text{then have } (\text{csucc(nat)} \setminus U) \cup (\text{csucc(nat)} \setminus V) < \text{csucc(nat)} \cup U = 0 \text{ } \cup V = 0 \text{ using less_less_imp_un_less[} \\
& \text{OF } _ _ \text{ InfCard_csucc[OF InfCard_nat]] by auto more over } \\
& \{ & \text{assume } (\text{csucc(nat)} \setminus U) \cup (\text{csucc(nat)} \setminus V) < \text{csucc(nat)} \text{ more over } \\
& \text{have } (\text{csucc(nat)} \setminus U) \cup (\text{csucc(nat)} \setminus V) = \text{csucc(nat)} \setminus U \cap V \text{ by auto } \\
& \text{with as(3) have } (\text{csucc(nat)} \setminus U) \cup (\text{csucc(nat)} \setminus V) = \text{csucc(nat)} \text{ by auto } \\
& \text{ultimately have } \text{csucc(nat)} < \text{csucc(nat)} \text{ by auto } \\
& \text{then have } False \text{ by auto } \\
& \} \text{ ultimately have } U \cap V = 0 \text{ by auto } \\
& \} \text{ then show } (\text{CoCountable csucc(nat)}) \{\text{is hyperconnected}\} \text{ unfolding IsHConnected_def by auto } \\
\}
\]

Then show \( (\text{CoCountable csucc(nat)}) \{\text{is hyperconnected}\} \) unfolding IsHConnected_def
by auto

**qed**

The cocountable topology on \( \mathbb{N}^+ \) is not anti-hyperconnected.

**Corollary**: \( \text{c cocountable in csucc nat not Anti HConn} \):

shows \( \neg((\text{CoCountable csucc(nat)}) \{\text{is anti-}\text{IsHConnected}) \)

**proof**-

assume as: (CoCountable csucc(nat)) {is anti-} IsHConnected
have (CoCountable csucc(nat)) {is hyperconnected} using cocountable_in_csucc_nat_HConn
by auto more over
have \( csucc(nat) \neq 0 \) using \( \text{Ord}_0\lt\text{csucc}[OF \text{Ord_nat}] \) by auto

then have \( \bigcup (\text{CoCountable} \ \text{csucc(nat)}) = \text{csucc(nat)} \) using union_cocardinal

unfolding \( \text{Cocountable_def} \) by auto

have \( \forall A \in (\text{CoCountable} \ \text{csucc(nat)}). A \subseteq \bigcup (\text{CoCountable} \ \text{csucc(nat)}) \) by fast

with \( \text{uni} \) have \( \forall A \in (\text{CoCountable} \ \text{csucc(nat)}). A \subseteq \text{csucc(nat)} \) by auto

then have \( \forall A \in (\text{CoCountable} \ \text{csucc(nat)}). \ \text{csucc(nat)} \cap A = A \) by auto

ultimately have \((\text{CoCountable} \ \text{csucc(nat)})\{\text{restricted to}\} \text{csucc(nat)}\)\{\text{is hyperconnected}\}

unfolding \( \text{RestrictedTo_def} \) by auto

with \( \text{as} \) have \( (\text{csucc(nat)})\{\text{is in the spectrum of}\} \text{IsHConnected} \) unfolding antiProperty_def

using \( \text{uni} \) by auto

then have \( \text{csucc(nat)} \subseteq 1 \) using \( \text{HConn_spectrum} \) by auto

then have \( \text{csucc(nat)} \subseteq \text{nat} \) using \( \text{n_lesspoll_nat} \) lesspoll_trans1 by auto

then show False using \( \text{lt_csucc}[OF \text{Ord_nat}] \) \( \text{lt_Card_imp_lesspoll}[OF \text{Card_csucc}[OF \text{Ord_nat}]] \)

lesspoll_trans by auto

qed

The cocountable topology on \( \mathbb{N}^+ \) is not \( T_2 \).

theorem \( \text{cocountable_in_csucc_nat_noT2} \):

shows \( \neg (\text{CoCountable} \ \text{csucc(nat)})\{\text{is} \ T_2\} \)

proof

assume \( (\text{CoCountable} \ \text{csucc(nat)})\{\text{is} \ T_2\} \)

then have \( \neg \text{IsHConnected} \) using \( \text{topology0.T2_imp_anti_HConn}[OF \text{topology0.CoCardinal}[OF \text{InfCard_csucc}[OF \text{InfCard_nat}]]] \)

unfolding \( \text{Cocountable_def} \) by auto

then show False using \( \text{cocountable_in_csucc_nat_notAntiHConn by auto} \)

qed

The cocountable topology on \( \mathbb{N}^+ \) is \( T_1 \).

theorem \( \text{cocountable_in_csucc_nat_T1} \):

shows \( (\text{CoCountable} \ \text{csucc(nat)})\{\text{is} \ T_1\} \)

using \( \text{cocardinal_is_T1}[OF \text{InfCard_csucc}[OF \text{InfCard_nat}]] \) unfolding \( \text{Cocountable_def} \) by auto

The cocountable topology on \( \mathbb{N}^+ \) is anti-compact.

theorem \( \text{cocountable_in_csucc_nat_antiCompact} \):

shows \( (\text{CoCountable} \ \text{csucc(nat)})\{\text{is} \ \text{anti-compact}\} \)

proof

have \( \text{noE:csucc(nat)} \neq 0 \) using \( \text{Ord}_0\lt\text{csucc}[OF \text{Ord_nat}] \) by auto

{ \fix \ A assume as:\( A \subseteq \bigcup (\text{CoCountable} \ \text{csucc(nat)}) \ (\bigcup ((\text{CoCountable} \ \text{csucc(nat)})\{\text{restricted to}\} A))\{\text{is compact in}\}((\text{CoCountable} \ \text{csucc(nat)})\{\text{restricted to}\} A) \\
from \text{as}(1) \ \text{have as:} \ A \subseteq \text{csucc(nat)} \ \text{using union_cocardinal}[OF \text{noE}] \) unfolding \( \text{Cocountable_def} \) by auto

1232
have ((CoCountable csucc(nat))\{restricted to\}A)=CoCountable (A\setminus csucc(nat))
using subspace_cocardinal
unfolding Cocardinal_def by auto moreover
from as have A\setminus csucc(nat)=A by auto
ultimately have ((CoCountable csucc(nat))\{restricted to\}A)=CoCountable A by auto
with as(2) have comp:\{\bigcup (CoCountable A)\} is compact in (CoCountable A) by auto
with as(2)
have comp:\{\bigcup (CoCountable A)\} is compact in (CoCountable A) by auto
moreover note noE
ultimately have (A-{t}) is closed in (CoCountable A) using closed_sets_cocardinal[of csucc(nat)]
A-{t}A unfolding Cocardinal_def by auto
then have A-(A-{t})\in (CoCountable A) using union_cocardinal[of noE, of A]
unfolding Cocardinal_def by auto moreover
from t have A-(A-{t})={t} by auto ultimately have \{t\} \subseteq (CoCountable A) by auto
then have r:\forall t \in A. \{t\} \in (CoCountable A) by auto
{ fix U assume U:U \subseteq Pow(A)
  { fix t assume t \in U
    with U r have t \subseteq U \subseteq (CoCountable A) by auto
    then have \exists V \subseteq (CoCountable A). t \in V \land V \subseteq U by auto
  }
  then have U \subseteq (CoCountable A) using topology0.open_neigh_open[of topology0_CoCardinal[
    OF InfCard_csucc[of InfCard_nat]]] unfolding Cocardinal_def by auto
}
then have Pow(A) \subseteq (CoCountable A) by auto moreover
{ fix B assume B \subseteq (CoCountable A)
  then have B \subseteq Pow(\bigcup (CoCountable A)) by auto
  then have B \subseteq Pow(A) using union_cocardinal[of noE] unfolding Cocardinal_def by auto
}
ultimately have p:Pow(A)=(CoCountable A) by auto
then have (CoCountable A) is anti-compact using pow_anti_compact[of A] by auto moreover
from p have \bigcup (CoCountable A) = \bigcup Pow(A) by auto
then have \( \bigcup (\text{CoCountable } A) = A \) by auto
from comp have \( (\bigcup (\text{CoCountable } A)) \{\text{restricted to}\} (\bigcup (\text{CoCountable } A)) \{\text{is compact in}\} (\bigcup (\text{CoCountable } A)) \)
using compact_imp_compact_subspace
\( \text{Compact_is_card_nat tot unfolding RestrictedTo_def by auto} \)
ultimately have \( A \{\text{is in the spectrum of}\} (\bigcup T) \{\text{is compact in}\} T \)
using comp tot unfolding IsAntiComp_def antiProperty_def by auto
moreover have \( f \) where \( f \in \text{inj}(\text{nat}, A) \) unfolding lepoll_def by auto
moreover have \( \text{fun: } f : \text{nat} \to A \) unfolding inj_def by auto
then have \( e : \text{surj}(\text{nat}, \text{range}(f)) \) using fun_is_surj by auto
ultimately have \( f \in \text{bij}(\text{nat}, \text{range}(f)) \) unfolding bij_def inj_def surj_def
by auto
then have \( e : \text{eqpoll_def by auto} \)
then have \( e : \text{range}(f) \approx \text{nat} \) using eqpoll_sym
then have \( \text{as2: } \text{range}(f) \approx \text{csucc}(\text{nat}) \) using lt_Card_imp_lesspoll[OF Card_csucc[OF Ord_nat]
lc_succ[OF Ord_nat] eq_lesspoll_trans]
then have \( \text{range}(f) \{\text{is compact in}\} (\text{CoCountable } A) \)
using compact_imp_compact_subspace
\( \text{Compact_is_card_nat tot unfolding Ccocountable_def using func1_1_L5B[OF fun]} \)
oE by auto
then have \( \bigcup (\text{range}(f)) \{\text{is compact in}\} (\text{CoCountable } A) \) using compact_closed
\( \text{union_cocardinal[OF oE, of A]} \)
\( \text{comp Compact_is_card_nat unfolding Ccocountable_def by auto} \)
moreover have \( \text{int: } A \cap \text{range}(f) = \text{range}(f) \cap A = \text{range}(f) \)
using compact_imp_compact_subspace
\( \text{Compact_is_card_nat by auto} \)
moreover have \( (\text{range}(f) \cap A) = \text{CoCountable } (\text{range}(f) \cap A) \)
using subspace_cocardinal unfolding Ccocountable_def by auto
with \( \text{int}(2) \)
ultimately have \( \text{comp2: } \text{range}(f) \{\text{is compact in}\} (\text{CoCountable } \text{range}(f)) \)
fix t assume t:t∈range(f)
have range(f)-{t}⊆range(f) by auto
then have range(f)-{t}⊆range(f) using subset_imp_lepoll by auto
with as2 have range(f)-{t}⊆csucc(nat) using lesspoll_trans1 by auto
moreover note noE
ultimately have (range(f)-{t})(is closed in)(CoCountable range(f))
using closed_sets_cocardinal[of csucc(nat)
  range(f)-{t}range(f)] unfolding Ccountable_def by auto
then have range(f)-(range(f)-{t})∈(CoCountable range(f)) un-
folding IsClosed_def using union_cocardinal[OF noE, of range(f)]
unfolding Ccountable_def by auto moreover
from t have range(f)-(range(f)-{t})={t} by auto ultimately have \{t\}∈(CoCountable range(f)) by auto

then have r:∀t∈range(f). \{t\}∈(CoCountable range(f)) by auto
{ fix U assume U:U⊆Pow(range(f))

  { fix t assume t∈U
    with U r have t∈\{t\}⊆U\{t\}∈(CoCountable range(f)) by auto
    then have ∃V∈(CoCountable range(f)). t∈V ∧ V⊆U by auto
  }
  then have U∈(CoCountable range(f)) using topology0.open_neigh_open[OF topology0_CoCardinal[
    OF InfCard_csucc[OF InfCard_nat]]] unfolding Ccountable_def by auto
}
then have Pow(range(f))⊆(CoCountable range(f)) by auto moreover
{ fix B assume B∈(CoCountable range(f))
  then have B∈Pow(∪(CoCountable range(f))) by auto
  then have B∈Pow(range(f)) using union_cocardinal[OF noE] un-
folding Ccountable_def by auto
}
ultimately have p:Pow(range(f))=(CoCountable range(f)) by blast
then have (CoCountable range(f))(is anti-compact) using pow_anti_compact[of range(f)] by auto moreover
from p have ∪(CoCountable range(f))=∪Pow(range(f)) by auto
then have tot:∪(CoCountable range(f))=range(f) by auto
from comp2 have ∪(∪(CoCountable range(f))){restricted to}∪(CoCountable range(f))) (is compact in)(CoCountable range(f)) {restricted to}(∪(CoCountable range(f))) using compact_imp_compact_subspace
Compact_is_card_nat tot unfolding RestrictedTo_def by auto
ultimately have range(f)(is in the spectrum of)\λT. (∪T)(is compact
in)T
  using comp tot unfolding IsAntiComp_def antiProperty_def by auto
then have Finite(range(f)) using compact_spectrum by auto
then have Finite(nat) using e_eqpoll_impFinite_iff by auto
then have False using nat_not_Finite by auto

1235
ultimately have $A\{\text{is in the spectrum of}\}(\lambda T. (\bigcup T)\{\text{is compact in}\}T)$ by auto

then have $\forall A\in\text{Pow}(\bigcup (\text{CoCountable csucc(nat))})$. (((\bigcup (\text{CoCountable csucc(nat)}) {\text{restricted to}} A)) \{\text{is compact in}\} (\text{CoCountable csucc(nat)}) \{\text{restricted to} A\})

$\rightarrow (A\{\text{is in the spectrum of}\}(\lambda T. (\bigcup T)\{\text{is compact in}\}T))$ by auto

then show thesis unfolding IsAntiComp_def antiProperty_def by auto

qed

In conclusion, the cocountable topology defined on csucc(nat) is KC but not $T_2$. Also note that is KC but not anti-hyperconnected, hence KC or US spaces need not to be sober.

The cofinite topology on the natural numbers is $T_1$, but not US.

\textbf{Theorem cofinite_not_US:}
shows $\neg((\text{CoFinite nat})\{\text{is US}\})$
proof
assume $A:(\text{CoFinite nat})\{\text{is US}\}$
let $N=\text{id(nat)}$
have $f:N: \text{nat} \rightarrow \text{nat}$ using id_type by auto
then have $\text{fun}:N: \text{nat} \rightarrow \bigcup \text{(CoCardinal(nat,nat))}$ using union_cocardinal unfolding Cofinite_def by auto
then have $\text{dom:domain(N)=nat}$ using func1_1_L1 by auto
with $\text{fun}$ have $\text{NET}:\langle N,Le \rangle \{\text{is a net on}\} \bigcup \text{(CoCardinal(nat,nat))}$ unfolding IsNet_def using Le_directs_nat by auto
have $\text{tot:} \bigcup \text{(CoCardinal(nat,nat))=} \text{nat}$ using union_cocardinal by auto
{ fix $U\ n$ assume $U:U\in\text{Pow}(\bigcup (\text{CoFinite nat}))$ $n\in\text{Interior(U,(CoFinite nat))}$
  have $\text{Interior(U,(CoFinite nat))}\in\text{(CoFinite nat)}$ using topology0.Top_2_L2 topology0_CoCardinal[OF InfCard_nat] unfolding Cofinite_def by auto
  then have $\text{nat}\neg\text{-Interior(U,(CoFinite nat))}$ using $\text{nat-U}\subseteq\text{nat}\neg\text{-Interior(U,(CoFinite nat))}$ unfolding Cofinite_def by auto
  ultimately have $\text{r:nat-U=0} \lor (\forall r\in\text{nat-U}. \ (r,\text{Maximum(Le,nat-U)})\in\text{Le})$ using linord_max_props(3)[of nat-U Lenat-U] unfolding FinPow_def by auto
  { assume $\text{reg:}\forall s\in\text{nat}. \ \exists r\in\text{nat} \ (s,r)\subseteq\text{Le} \land \text{Nr}\subseteq\text{U}$
    
    1236
  }
}
with \( r \) have \( s : \forall r \in \text{o}. \langle r, \text{Maximum} \langle \text{Le}, \text{nat-U} \rangle \rangle \in \text{Le} \) \( \text{nat-U} \neq 0 \) using apply_type[of \( f \)] by auto

have \( \text{Maximum} \langle \text{Le}, \text{nat-U} \rangle \in \text{nat} \) using linord_max_props(2)[OF lin_s(2)]

then have \( \text{Maximum} \langle \text{Le}, \text{nat-U} \rangle \in \text{nat} \) using nat_succI by auto

then have \( \text{Maximum} \langle \text{Le}, \text{nat-U} \rangle \in \text{nat} \) using nat_SU by auto

then have \( \exists r \in \text{nat}. \langle r, \text{Maximum} \langle \text{Le}, \text{nat-U} \rangle \rangle \in \text{Le} \) ∧ \( \text{Nr} \notin \text{U} \) by auto

then obtain \( r \) where \( r \in \text{nat} \langle \text{Maximum} \langle \text{Le}, \text{nat-U} \rangle, r \rangle \in \text{Le} \) ∧ \( \text{Nr} \notin \text{U} \) by auto

from \( r \) where \( r \in \text{nat} \langle \text{Maximum} \langle \text{Le}, \text{nat-U} \rangle, r \rangle \in \text{Le} \) ∧ \( \text{Nr} \notin \text{U} \) by auto

then have \( \text{Maximum} \langle \text{Le}, \text{nat-U} \rangle \in \text{nat} \) using apply_type[of \( f \)] by auto

then have \( \text{Maximum} \langle \text{Le}, \text{nat-U} \rangle \in \text{nat} \) using linord_max_props(2)[OF lin_s(2)]

then have \( \exists s \in \text{nat}. \forall r \in \text{nat}. \langle s, r \rangle \in \text{Le} \rightarrow \text{Nr} \in \text{U} \) by auto

then have \( \forall n \in \text{nat}. \forall U \in \text{Pow}(\underbrace{\bigcup} \text{CoFinite nat}) \) \( n \in \text{Interior}(U, \text{CoFinite nat}) \rightarrow \) \( \exists s \in \text{nat}. \forall r \in \text{nat}. \langle s, r \rangle \in \text{Le} \rightarrow \text{Nr} \in \text{U} \) by auto

unfolding Cofinite_def by auto

then have \( \forall n \in \text{nat}. \forall U \in \text{Pow}(\bigcup \text{CoFinite nat}) \) \( n \in \text{Interior}(U, \text{CoFinite nat}) \rightarrow \) \( \exists s \in \text{nat}. \forall r \in \text{nat}. \langle s, r \rangle \in \text{Le} \rightarrow \text{Nr} \in \text{U} \) by auto

then have \( \langle N, \text{Le} \rangle \rightarrow N \) \( n \in \text{CoFinite nat} \) unfolding topology0.NetConverges_def[of \( \text{topology0_CoCardinal[of InfCard_nat] NET} \)]

using dom by auto

then have \( \langle N, \text{Le} \rangle \rightarrow N \) \( n \in \text{CoFinite nat} \) unfolding Cofinite_def by auto

then show False using A unfolding IsUS_def using fun unfolding Cofinite_def by auto

qed

To end, we need a space which is US but no KC. This example comes from the one point compactification of a \( T_2 \), anti-compact and non discrete space.

This \( T_2 \), anti-compact and non discrete space comes from a construction over the cardinal \( \mathbb{N}^+ \) or \( \text{csucc(nat)} \).

**Theorem extension_pow_top:**

shows \( \text{Pow}(\text{csucc(nat)}) \cup \{\text{csucc(nat)}\} \cup S. S \in (\text{CoCountable csucc(nat)})-{0}) \)\{is a topology\}

proof-

have \( \text{noE:csucc(nat)} \neq 0 \) using Ord_0_lt_csucc[OF Ord_nat] by auto

{ fix \( M \) assume \( M : M \leq (\text{Pow}(\text{csucc(nat)}) \cup \{\text{csucc(nat)}\} \cup S. S \in (\text{CoCountable csucc(nat)})-{0}) \)

let \( N \) = \( \{U \in M. U \in \text{Pow}(\text{csucc(nat)})\} \)

1237
let MN = {U ∈ M | U ∈ Pow(csucc(nat))}
have unM: M = (\bigcup MP) \cup (\bigcup MN) by auto
have csucc(nat) ∉ csucc(nat) using mem_not_refl by auto
with M have MN: MN = {U ∈ M | U ∈ {{csucc(nat)} \cup S | S ∈ (CoCountable csucc(nat)) \setminus \{0\}}} by auto
having
have unMP: \bigcup MP ∈ Pow(csucc(nat)) by auto
then have MN = 0 \rightarrow \bigcup M ∈ (Pow(csucc(nat)) \cup {\{csucc(nat)\} \cup S | S ∈ (CoCountable csucc(nat)) \setminus \{0\}})
using unM by auto
moreover
{ assume MN \neq 0
with MN have \{U ∈ M | U ∈ {{csucc(nat)} \cup S | S ∈ (CoCountable csucc(nat)) \setminus \{0\}}} \neq 0 by auto
then obtain U where U: U ∈ {{csucc(nat)} \cup S | S ∈ (CoCountable csucc(nat)) \setminus \{0\}}
by blast
then obtain S where S: U = {csucc(nat)} \cup S \in (CoCountable csucc(nat)) \setminus \{0\}
by auto
with U MN have csucc(nat) ∈ U \in MN by auto
then have a1: csucc(nat) ∈ \bigcup MN by auto
let SC = {S ∈ (CoCountable csucc(nat)) | {csucc(nat)} \cup S ∈ M}
have unSC: \bigcup SC ∈ (CoCountable csucc(nat)) using CoCar_is_topology[OF InfCard_csucc[OF InfCard_nat]] unfolding IsATopology_def unfolding Cocountable_def by blast
{ fix s assume s ∈ {csucc(nat)} \cup \bigcup SC
then have s = csucc(nat) \cup S \in \bigcup SC by auto
then have s \in \bigcup MN \vee \exists S ∈ SC. s \in S by auto
then have s \in \bigcup MN \vee \exists S ∈ (CoCountable csucc(nat)). {csucc(nat)} \cup S \in M
\wedge s ∈ S)
by auto
with MN have s \in \bigcup MN \vee \exists S ∈ (CoCountable csucc(nat)). {csucc(nat)} \cup S \in MN
\wedge s ∈ S)
by auto
then have s \in \bigcup MN by blast
}
then have {csucc(nat)} \cup \bigcup SC \subseteq \bigcup MN by blast
moreover
{ fix s assume s ∈ \bigcup MN
then obtain U where U: U ∈ {{csucc(nat)} \cup S | S ∈ (CoCountable csucc(nat))} by auto
with M have U ∈ {{csucc(nat)} \cup S | S ∈ (CoCountable csucc(nat))} by auto
then obtain S where S: U = {csucc(nat)} \cup S \in (CoCountable csucc(nat))
by auto
with U(1) have s = csucc(nat) \vee s ∈ S by auto
with S U(2) have s = csucc(nat) \vee s ∈ SC by auto
then have s ∈ {csucc(nat)} \cup SC by auto
}
then have \bigcup MN \subseteq {csucc(nat)} \cup \bigcup SC by blast
ultimately have unMN: \bigcup MN = {csucc(nat)} \cup \bigcup SC by auto
from unSC have b1: csucc(nat) \setminus \bigcup SC < csucc(nat) \vee \bigcup SC = 0 unfolding Cocountable_def
CoCardinal_def
  by auto
  {   
    assume 0∈SC
    then have {csucc(nat)}∈M by auto
    then have {csucc(nat)}∈{{csucc(nat)}∪S. S∈(CoCountable csucc(nat))-{0}}
    using mem_not_refl
    M by auto
    then obtain S where S:S∈(CoCountable csucc(nat))-{0} {csucc(nat)}={csucc(nat)}∪S
    by auto
    {   
      fix x assume x∈S
      then have x∈{csucc(nat)}∪S by auto
      with S(2) have x∈{csucc(nat)} by auto
      then have x=csucc(nat) by auto
    }
    then have S⊆{csucc(nat)} by auto
    with S(1) have S={csucc(nat)} by auto
    with S(1) have csucc(nat)-{csucc(nat)}≺csucc(nat) unfolding Cocountable_def
    CoCardinal_def
      by auto moreover
    then have csucc(nat)-{csucc(nat)}=csucc(nat) using mem_not_refl[of csucc(nat)] by force
    ultimately have False by auto
  }
  then have 0∉SC by auto moreover
  from S U(1) have S∈SC by auto
  ultimately have S∪SC S≠0 by auto
  then have noe:∪SC≠0 by auto
  with b1 have csucc(nat)-∪SC<csucc(nat) by auto
  moreover have csucc(nat)-{(∪SC U∪MP)⊆csucc(nat)-∪SC by auto
  then have csucc(nat)-{(∪SC U∪MP)⊆csucc(nat)-∪SC using subset_imp_lepoll
  by auto
  ultimately have csucc(nat)-{(∪SC U∪MP)<csucc(nat) using lesspoll_trans1
  by auto moreover
  have ∪SC⊆∪(CoCountable csucc(nat)) using unSC by auto
  then have ∪SC⊆csucc(nat) using union_cocardinal[of noe] unfolding Cocountable_def by auto
  ultimately have (∪SC U∪MP)∈(CoCountable csucc(nat))
  using unMP unfolding Cocountable_def CoCardinal_def by auto
  then have {csucc(nat)}∪(∪SC U∪MP)∈(Pow(csucc(nat)) U {{csucc(nat)}∪S. S∈(CoCountable csucc(nat))-{0}})
  using noe by auto moreover
  from unM unMN have ∪M={({csucc(nat)}∪∪SC) U∪MP by auto
  then have ∪M={csucc(nat)}∪(∪SC U∪MP) by auto
  ultimately have ∪M∈(Pow(csucc(nat)) U {{csucc(nat)}∪S. S∈(CoCountable csucc(nat))-{0}}) by auto
  }
  ultimately have ∪M∈(Pow(csucc(nat)) U {{csucc(nat)}∪S. S∈(CoCountable

1239
csucc(nat)−(0)) by auto

} then have ∀M∈Pow(Pow(csucc(nat)) ∪ {csucc(nat)∪S. S∈CoCountable
csucc(nat)−(0)})). \bigcup M∈(Pow(csucc(nat)) ∪ {csucc(nat)∪S. S∈CoCountable
csucc(nat)−(0)}) by auto

moreover

{ fix U V assume UV:U∈(Pow(csucc(nat)) ∪ {csucc(nat)∪S. S∈CoCountable
csucc(nat)−(0)}) V∈(Pow(csucc(nat)) ∪ {csucc(nat)∪S. S∈CoCountable
csucc(nat)−(0)})

  assume csucc(nat)\notin U \lor csucc(nat)\notin V

  with UV have U∈Pow(csucc(nat)) \lor V∈Pow(csucc(nat)) by auto

  then have U\cap V∈Pow(csucc(nat)) by auto

  then have U\cap V∈{csucc(nat)∪S. S∈CoCountable
csucc(nat)−(0)}) by auto

} moreover

{ assume csucc(nat)\notin U \land csucc(nat)\notin V

  then obtain SU SV where S:U={csucc(nat)}∪SU V={csucc(nat)}∪SV

SU∈(CoCountable csucc(nat))−{0}

  SV∈(CoCountable csucc(nat))−{0) using UV mem_not_refl by auto

  from S(1,2) have U\cap V∈(SU\cap SV) by auto moreover

  from S(3,4) have SU\cap SV∈(CoCountable csucc(nat)) using CoCar_is_topology[OF
InfCard_csucc[OF InfCard_nat]] unfolding IsATopology_def

  unfolding Ccocountable_def by blast moreover

  from S(3,4) have SU\cap SV\neq 0 using ccountable_in_csucc_nat_HConn un-

  folding IsHConnected_def

  by auto ultimately

  have U\cap V∈{csucc(nat)∪S. S∈CoCountable csucc(nat)−(0)}) by auto

  then have U\cap V∈(Pow(csucc(nat)) ∪ {csucc(nat)∪S. S∈CoCountable
csucc(nat)−(0)}) by auto

} ultimately have U\cap V∈(Pow(csucc(nat)) ∪ {csucc(nat)∪S. S∈CoCountable
csucc(nat)−(0)}) by auto

} then have ∀U∈(Pow(csucc(nat)) ∪ {csucc(nat)∪S. S∈CoCountable csucc(nat)−(0)}).

∀V∈(Pow(csucc(nat)) ∪ {csucc(nat)∪S. S∈CoCountable csucc(nat)−(0)}).

∀U\cap V∈(Pow(csucc(nat)) ∪ {csucc(nat)∪S. S∈CoCountable csucc(nat)−(0)})

by auto

ultimately show thesis unfolding IsATopology_def by auto

qed

This topology is defined over N+ ∪ {N+} or csucc(nat)∪{csucc(nat)}.

lemma extension_pow_union:

shows \bigcup (Pow(csucc(nat)) ∪ {csucc(nat)∪S. S∈CoCountable csucc(nat)−(0)})=csucc(nat)∪

proof

have noE:csucc(nat)\neq 0 using Ord_0_lt_csucc[OF Ord_nat] by auto
have \( \bigcup \{\text{Pow}(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable csucc}(\text{nat})) - \{0}\} = \bigcup \{\text{Pow}(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable csucc}(\text{nat})) - \{0\}\} \)

by blast

also have \( \ldots = \text{csucc}(\text{nat}) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable csucc}(\text{nat})) - \{0\} \)

by auto

ultimately have \( A: \bigcup \{\text{Pow}(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable csucc}(\text{nat})) - \{0\}\} = \text{csucc}(\text{nat}) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable csucc}(\text{nat})) - \{0\} \)

by auto

have \( \bigcup (\text{CoCountable csucc}(\text{nat})) \in (\text{CoCountable csucc}(\text{nat})) \) using CoCar_is_topology[of InfCard_csucc[of InfCard_nat]]

unfolding IsATopology_def Cocountable_def by auto

then have \( \text{csucc}(\text{nat}) \in (\text{CoCountable csucc}(\text{nat})) \) using union_cocardinal[of noE]

unfolding Cocountable_def by auto

with \( \text{noE} \)

have \( \text{csucc}(\text{nat}) \in (\text{CoCountable csucc}(\text{nat})) - \{0\} \) by auto

then have \( \{\text{csucc}(\text{nat})\} \cup \text{csucc}(\text{nat}) \subseteq \bigcup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable csucc}(\text{nat})) - \{0\} \)

by blast

with \( A \)

show \( \text{csucc}(\text{nat}) \cup \{\text{csucc}(\text{nat})\} \subseteq \bigcup \{\text{Pow}(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable csucc}(\text{nat})) - \{0\}\} \)

by auto

{-
  fix x assume x:x\in(\bigcup \{\text{Pow}(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable csucc}(\text{nat})) - \{0\}\})
  x\neq\text{csucc}(\text{nat})
  then obtain U where U:U\in(\{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable csucc}(\text{nat})) - \{0\})
  x\in U
  by blast
  then obtain S where S:S=x\in(\text{csucc}(\text{nat}) \cup S \in (\text{CoCountable csucc}(\text{nat})) - \{0\})
  by auto
  with U(2) x(2) have x\in S by auto
  with S(2) have x\in(\text{CoCountable csucc}(\text{nat})) by auto
  then have x\in(\text{csucc}(\text{nat})) using union_cocardinal[of noE] unfolding Cocountable_def by auto
}

then have \( (\bigcup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable csucc}(\text{nat})) - \{0\}) \subseteq \text{csucc}(\text{nat}) \cup \{\text{csucc}(\text{nat})\} \)

by blast

with \( A \)

show \( \bigcup \{\text{Pow}(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable csucc}(\text{nat})) - \{0\}\} \subseteq \text{csucc}(\text{nat}) \cup \{\text{csucc}(\text{nat})\} \)

by blast

qed

This topology has a discrete open subspace.

lemma extension_pow_subspace:
  shows \( (\text{Pow}(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable csucc}(\text{nat})) - \{0\}) \) (restricted to) \( \text{csucc}(\text{nat}) \) = \( \text{Pow}(\text{csucc}(\text{nat})) \)
  and \( \text{csucc}(\text{nat}) \in (\text{Pow}(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable csucc}(\text{nat})) - \{0\}) \)

proof
  show \( \text{csucc}(\text{nat}) \in (\text{Pow}(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable csucc}(\text{nat})) - \{0\}) \) by auto

1241
\begin{verbatim}
{  fix x assume x∈(Pow(csucc(nat)) ∪ \{csucc(nat)\}∪S. S∈(CoCountable
csucc(nat))-{0})){restricted to}csucc(nat)
th then obtain R where x=csucc(nat)\cap R R∈(Pow(csucc(nat)) ∪ \{csucc(nat)\}∪S.
S∈(CoCountable csucc(nat))-{0})){restricted to}csucc(nat)
unfolding RestrictedTo_def
by auto
then have x∈Pow(csucc(nat)) by auto
}
then show (Pow(csucc(nat)) ∪ \{csucc(nat)\}∪S. S∈(CoCountable csucc(nat))-{0})){restricted
to}csucc(nat)⊆Pow(csucc(nat)) by auto
qed

This topology is Hausdorff.

\textbf{theorem} extension_pow_T2:
  shows (Pow(csucc(nat)) ∪ \{csucc(nat)\}∪S. S∈(CoCountable csucc(nat))-{0})){is
T\textsubscript{2}}
\textbf{proof-}
  have noE:csucc(nat)≠0 using Ord_0_lt_csucc[OF Ord_nat] by auto
  {  fix A B assume A∈⋃(Pow(csucc(nat)) ∪ \{csucc(nat)\}∪S. S∈(CoCountable
csucc(nat))-{0})) B∈⋃(Pow(csucc(nat)) ∪ \{csucc(nat)\}∪S. S∈(CoCountable
csucc(nat))-{0}))
A≠B
  then have AB:A∈csucc(nat)∪\{csucc(nat)\} B∈csucc(nat)∪\{csucc(nat)\}
A≠B using extension_pow_union by auto
  {  assume A≠csucc(nat) B≠csucc(nat)
  then have A∈csucc(nat) B∈csucc(nat) using AB by auto
  then have sub:{A}∈Pow(csucc(nat)) \{B}∈Pow(csucc(nat)) by auto
  then have \{A\}∈(Pow(csucc(nat)) ∪ \{csucc(nat)\}∪S. S∈(CoCountable
csucc(nat))-{0})){restricted to}csucc(nat)
\{B\}∈(Pow(csucc(nat)) ∪ \{csucc(nat)\}∪S. S∈(CoCountable csucc(nat))-{0})){restricted
to}csucc(nat) using extension_pow_subspace(1)
  by auto
  then obtain RA RB where \{A\}=csucc(nat)\cap RA \{B\}=csucc(nat)\cap RB RA∈(Pow(csucc(nat))
∪ \{csucc(nat)\}∪S. S∈(CoCountable csucc(nat))-{0})) RB∈(Pow(csucc(nat)) ∪ \{csucc(nat)\}∪S. S∈(CoCountable
csucc(nat))-{0}))
unfolding RestrictedTo_def by auto
  then have (A)∈(Pow(csucc(nat)) ∪ \{csucc(nat)\}∪S. S∈(CoCountable
csucc(nat))-{0})){restricted to}csucc(nat)
\{B\}∈(Pow(csucc(nat)) ∪ \{csucc(nat)\}∪S. S∈(CoCountable csucc(nat))-{0})){restricted
to}csucc(nat) using extension_pow_subspace(1)
  by auto

1242
\end{verbatim}
using extension_pow_subspace(2) extension_pow_top unfolding IsATopology_def by auto

moreover from AB(3) have \( \{A\} \cap \{B\} = 0 \) by auto ultimately have \( \exists U \in (\text{Pow}(\text{csucc}(\text{nat}))) \cup \{\{\text{csucc}(\text{nat})\}\cup S. S \in (\text{CoCountable} \text{ csucc}(\text{nat}))-\{0\}\}. \exists V \in (\text{Pow}(\text{csucc}(\text{nat}))) \cup \{\{\text{csucc}(\text{nat})\}\cup S. S \in (\text{CoCountable} \text{ csucc}(\text{nat}))-\{0\}\}. \)

moreover |
\begin{align*}
&\text{assume } A = \text{csucc}(\text{nat}) \vee B = \text{csucc}(\text{nat}) \\
&\text{with } AB(3) \text{ have disj: } (A = \text{csucc}(\text{nat}) \wedge B \neq \text{csucc}(\text{nat})) \vee (B = \text{csucc}(\text{nat}) \wedge A \neq \text{csucc}(\text{nat}))
\end{align*}

by auto

\begin{align*}
&\text{assume } \text{ass}: A = \text{csucc}(\text{nat}) \vee B \neq \text{csucc}(\text{nat}) \\
&\text{then have } p: B \in \text{csucc}(\text{nat}) \text{ using } AB(2) \text{ by auto} \\
&\text{have } \{B\} \approx 1 \text{ using singleton_eqpoll_1 by auto} \\
&\text{then have } \{B\} \preceq \text{nat} \text{ using eq_lesspoll_trans n_lesspoll_nat by auto} \\
&\text{then have } \{B\} \preceq \text{csucc}(\text{nat}) \text{ using Card_less_csucc_eq_le[OF Card_nat]} \text{ by auto} \\
&\text{with } p \text{ have } \{B\} \{\text{is closed in}\} (\text{CoCountable} \text{ csucc}(\text{nat})) \text{ unfolding Ccoountable_def using closed_sets_cocodinal[OF noE]} \text{ by auto} \\
&\text{then have } \text{csucc}(\text{nat})-\{B\} \in (\text{CoCountable} \text{ csucc}(\text{nat})) \text{ unfolding IsClosed_def Ccoountable_def using union_cocodinal[OF noE]} \text{ by auto more-} \over \\
&\text{assume } \text{csucc}(\text{nat})-\{B\} = 0 \\
&\text{with } p \text{ have } \text{csucc}(\text{nat}) = \{B\} \text{ by auto} \\
&\text{then have } \text{csucc}(\text{nat}) \approx 1 \text{ using singleton_eqpoll_1 by auto} \\
&\text{then have } \text{csucc}(\text{nat}) \preceq \text{nat} \text{ using eq_lesspoll_trans n_lesspoll_nat lesspoll_imp_lepoll by auto} \\
&\text{then have } \text{csucc}(\text{nat}) \preceq \text{csucc}(\text{nat}) \text{ using Card_less_csucc_eq_le[OF Card_nat]} \\
&\text{by auto} \\
&\text{then have } \text{False by auto}
\end{align*}

ultimately have \( \{\text{csucc}(\text{nat})\} \cup (\text{csucc}(\text{nat})-\{B\}) \in (\{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable} \text{ csucc}(\text{nat}))-\{0\}\}. \)

by auto

then have \( U_1: (\text{csucc}(\text{nat}) \cup (\text{csucc}(\text{nat})-\{B\}) \in (\text{Pow}(\text{csucc}(\text{nat}))) \cup (\{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable} \text{ csucc}(\text{nat}))-\{0\}\). \)

by auto

then have \( B \in (\text{Pow}(\text{csucc}(\text{nat}))) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable} \text{ csucc}(\text{nat}))-\{0\}\} \{\text{restricted to}\} \text{csucc}(\text{nat}) \\
\text{using extension_pow_subspace(1) by auto} \\
\text{then obtain } R \text{ where } R \in \text{Pow}(\text{csucc}(\text{nat}))) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable} \text{ csucc}(\text{nat}))-\{0\}\} \{B\} = \text{csucc}(\text{nat}) \cap R
\)
unfolding RestrictedTo_def by auto
then have U2:{B}∈Pow(csucc(nat)) ∪ \{{\text{csucc(nat)}}\}∪S. S∈(CoCountable csucc(nat))¬\{0\} using extension_pow_subspace(2)
  extension_pow_top unfolding IsATopology_def by auto
have \{(\text{csucc(nat)})∪(\text{csucc(nat)}¬\{B\})\}¬\{B\}¬0 using p mem_not_refl[of csucc(nat)] by auto
have \{(\text{csucc(nat)})∪(\text{csucc(nat)}¬\{A\})\}¬\{B\}¬0 using p mem_not_refl[of csucc(nat)] by auto
moreover
  assume ¬(A=csucc(nat)∧B≠csucc(nat))
then have ass:B = csucc(nat) ∧ A ≠ csucc(nat) using disj by auto
then have p:A∈csucc(nat) using AB(1) by auto
have \{A\}=1 using singleton_eqpoll_1 by auto
then have \{A\}¬nat using eq_lesspoll_trans n_lesspoll_nat by auto
then have \{A\}≤nat using lesspoll_imp_lepoll by auto
then have \{A\}¬csucc(nat) using Card_less_csucc_eq_le[OF Card_nat] by auto
moreover
  assume csucc(nat)¬\{A\}=0
with p have \{A\}¬\{csucc(nat)¬A}\∈(CoCountable csucc(nat)) unfolding Cocountable_def
  using closed_sets_cocardinal[OF noE] by auto
then have csucc(nat)¬\{A\}∈(CoCountable csucc(nat)) unfolding IsClosed_def
  using union_cocardinal[OF noE] by auto
moreover
  assume csucc(nat)¬\{A\}=0
with p have csucc(nat)=\{A\} by auto
then have csucc(nat)≈1 using singleton_eqpoll_1 by auto
then have csucc(nat)≤nat using eq_lesspoll_trans n_lesspoll_nat
then have csucc(nat)<csucc(nat) using Card_less_csucc_eq_le[OF Card_nat] by auto
ultimately
  have \{csucc(nat)\}∪(csucc(nat)¬\{A\})∈\{csucc(nat)\}∪S. S∈(CoCountable csucc(nat))¬\{0\} by auto
then have U1:csucc(nat)∪(csucc(nat)¬\{A\})∈(Pow(csucc(nat)) ∪ \{{\text{csucc(nat)}}\}∪S. S∈(CoCountable csucc(nat))¬\{0\}) by auto
then have \{A\}∈Pow(csucc(nat)) using p by auto
then have \{A\}∈Pow(csucc(nat)) ∪ \{{\text{csucc(nat)}}\}∪S. S∈(CoCountable csucc(nat))¬\{0\}¬\{\text{csucc(nat)}\}¬(\text{csucc(nat)}¬\{A\})¬\{0\}¬\{A\}¬csucc(nat)¬R

1244
The topology we built is not discrete; i.e., not every set is open.

**Theorem extension_pow_notDiscrete:**

shows \{csucc(nat)\} \notin (Pow(csucc(nat)) \cup \{csucc(nat)\}\cup S. S \in (CoCountable csucc(nat))-\{0\})

**Proof**

assume \{csucc(nat)\} \in (Pow(csucc(nat)) \cup \{csucc(nat)\}\cup S. S \in (CoCountable csucc(nat))-\{0\})

then have \{csucc(nat)\}\in\{csucc(nat)\}\cup S. S \in (CoCountable csucc(nat))-\{0\})

using mem_not_refl by auto

then obtain S where S:S \in (CoCountable csucc(nat))-\{0\} \{csucc(nat)\}=\{csucc(nat)\}\cup S by auto

{ fix x assume x\in S
  then have x\in\{csucc(nat)\}\cup S by auto

qed
with \(S(2)\) have \(x \in \{\text{csucc}(\text{nat})\}\) by auto
then have \(x = \text{csucc}(\text{nat})\) by auto

then have \(S \subseteq \{\text{csucc}(\text{nat})\}\) by auto
with \(S(1)\) have \(S = \{\text{csucc}(\text{nat})\}\) by auto
with \(S(1)\) have \(\text{csucc}(\text{nat}) - \{\text{csucc}(\text{nat})\} < \text{csucc}(\text{nat})\) unfolding \(\text{Cocountable_def}\)
\(\text{CoCardinal_def}\)
by auto moreover
then have \(\text{csucc}(\text{nat}) - \{\text{csucc}(\text{nat})\} = \text{csucc}(\text{nat})\) using \(\text{mem_not_refl[of csucc(nat)]}\) by force
ultimately show \(\text{False}\) by auto
qed

The topology we built is anti-compact.

\begin{quote}
\textbf{Theorem} \(\text{extension_pow_antiCompact}\):
\textbf{shows} \((\text{Pow}(\text{csucc}(\text{nat})) \cup \{\{\text{csucc}(\text{nat})\}\cup S. S \in (\text{CoCountable }\text{csucc}(\text{nat})) - \{0\}\})\)\{\text{is anti-compact}\}
\textbf{proof}\-
\begin{itemize}
\item have \(\text{noE:csucc(nat)} \neq 0\) using \(\text{Ord}_0\_lt\_csucc[\text{OF Ord_nat}]\) by auto
\{ \begin{itemize}
\item fix \(K\) assume \(K:K \subseteq \bigcup (\text{Pow}(\text{csucc}(\text{nat})) \cup \{\{\text{csucc}(\text{nat})\}\cup S. S \in (\text{CoCountable }\text{csucc}(\text{nat})) - \{0\}\})\)\{\text{restricted to}K\}\){\text{is compact in}((\text{Pow}(\text{csucc}(\text{nat})) \cup \{\{\text{csucc}(\text{nat})\}\cup S. S \in (\text{CoCountable }\text{csucc}(\text{nat})) - \{0\}\})\)\{\text{restricted to}K\})}
from \(K(1)\) have \(\text{sub:K} \subseteq \text{csucc(nat)} \cup \{\text{csucc(nat)}\}\) using \(\text{extension_pow_union}\)
by auto
have \((\bigcup (\text{Pow}(\text{csucc}(\text{nat})) \cup \{\{\text{csucc}(\text{nat})\}\cup S. S \in (\text{CoCountable }\text{csucc}(\text{nat})) - \{0\}\})\)\{\text{restricted to}K\}) = (\text{csucc(nat)} \cup \{\text{csucc(nat)}\}) \cap K
using \(\text{extension_pow_union}\) unfolding \(\text{RestrictedTo_def}\) by auto moreover
from \(\text{sub have (csucc(nat)} \cup \{\text{csucc(nat)}\}) \cap K = K\) by auto
ultimately have \((\bigcup (\text{Pow}(\text{csucc}(\text{nat})) \cup \{\{\text{csucc}(\text{nat})\}\cup S. S \in (\text{CoCountable }\text{csucc}(\text{nat})) - \{0\}\})\)\{\text{restricted to}K\}) = K\) by auto
with \(K(2)\) have \(K\)\{\text{is compact in}((\text{Pow}(\text{csucc}(\text{nat})) \cup \{\{\text{csucc}(\text{nat})\}\cup S. S \in (\text{CoCountable }\text{csucc}(\text{nat})) - \{0\}\})\)\{\text{restricted to}K\}) \subseteq S(\text{CoCountable }\text{csucc}(\text{nat})) - \{0\}\} by auto
then have \(\text{comp:K}\)\{\text{is compact in}((\text{Pow}(\text{csucc}(\text{nat})) \cup \{\{\text{csucc}(\text{nat})\}\cup S. S \in (\text{CoCountable }\text{csucc}(\text{nat})) - \{0\}\})\)\{\text{restricted to}K\})\)
using \(\text{compact_subspace_imp_compact}\) by auto
\{ \begin{itemize}
\item assume \(ss:K \subseteq \text{csucc}(\text{nat})\)
then have \(K\)\{\text{is compact in}((\text{Pow}(\text{csucc}(\text{nat})) \cup \{\{\text{csucc}(\text{nat})\}\cup S. S \in (\text{CoCountable }\text{csucc}(\text{nat})) - \{0\}\})\)\{\text{restricted to}csucc(\text{nat})\})\)
using \(\text{compact_imp_compact_subspace}\) \(\text{comp Compact_is_card_nat}\) by auto
then have \(K\)\{\text{is compact in}((\text{Pow}(\text{csucc}(\text{nat})) \cup \{\{\text{csucc}(\text{nat})\}\cup S. S \in (\text{CoCountable }\text{csucc}(\text{nat})) - \{0\}\})\)\{\text{restricted to}K\})\)
using \(\text{compact_imp_compact_subspace}\) by auto
\end{itemize}
\end{itemize}
\end{itemize}
\end{quote}
Compact_is_card_nat by auto moreover
have \( \bigcup (\text{Pow}(csucc(nat)) \{\text{restricted to}\} K) = K \) using \( \text{ss} \) unfolding RestrictedTo_def
by auto
ultimately have \( (\bigcup (\text{Pow}(csucc(nat)) \{\text{restricted to}\} K)) \{\text{is compact in}\} (\text{Pow}(csucc(nat)) \{\text{restricted to}\} K) \) by auto
then have \( K \{\text{is in the spectrum of}\} (\lambda T. (\bigcup T) \{\text{is compact in}\} T) \)
using pow_anti_compact
  unfolding IsAntiComp_def antiProperty_def using \( \text{ss} \) by auto
}
moreover
{
  assume \( \neg (K \subseteq csucc(nat)) \)
  with sub have \( csucc(nat) \in K \) by auto
  with sub have \( \text{ss}: K - \{csucc(nat)\} \subseteq csucc(nat) \) by auto
  \{
    assume \( \text{prec}: K - \{csucc(nat)\} < csucc(nat) \)
    then have \( (K - \{csucc(nat)\}) \{\text{is closed in}\} (\text{CoCountable} \ csucc(nat)) \)
      using closed_sets_cocardinal[OF noE] \( \text{ss} \) unfolding Cocardinal_def
    by auto
    then have \( \text{csucc}(nat) - (K - \{csucc(nat)\}) \in (\text{CoCountable} \ csucc(nat)) \)
      unfolding IsClosed_def Cocardinal_def
    using union_cocardinal[OF noE] by auto
  \}
  ultimately have \( \{csucc(nat)\} \cup (csucc(nat) - (K - \{csucc(nat)\})) \in \{\text{csucc}(nat)\} \cup S \).
  \( S \in (\text{CoCountable} \ csucc(nat)) - \{0\} \)
  by auto
  moreover have \( \{\text{csucc}(nat)\} \cup (\text{csucc}(nat) - (K - \{csucc(nat)\})) = \{\text{csucc}(nat)\} \cup \text{csucc}(nat) - (K - \{csucc(nat)\}) \) by blast
  ultimately have \( \text{csucc}(nat) - (K - \{csucc(nat)\}) \in \{\text{csucc}(nat)\} \cup S \).
  \( S \in (\text{CoCountable} \ csucc(nat)) - \{0\} \)
  by auto
  then have \( \{\text{csucc}(nat)\} \cup \text{csucc}(nat) - (K - \{csucc(nat)\}) \in (\text{Pow}(\text{csucc}(nat)) \cup \{\text{csucc}(nat)\} \cup S) \).
  \( \text{S} \subseteq (\text{CoCountable} \ csucc(nat)) - \{0\} \)
  by auto
  then have \( (\bigcup (\text{Pow}(\text{csucc}(nat)) \cup \{\text{csucc}(nat)\} \cup S) \subseteq (\text{CoCountable} \ csucc(nat)) - \{0\}) \)
    using extension_pow_union by auto
  then have \( \{\text{csucc}(nat)\} \{\text{is closed in}\} (\text{Pow}(\text{csucc}(nat)) \cup \{\text{csucc}(nat)\} \cup S) \).
  \( S \subseteq (\text{CoCountable} \ csucc(nat)) - \{0\} \)
  unfolding IsClosed_def using \( \text{ss} \) by auto

1247
with \( \cap \) \( (K \cap (K - \{\text{csucc}(\text{nat})\})) \{\text{is compact in}\} (\text{Pow}(\text{csucc}(\text{nat}))) \)
\( \cup \{\{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable} \ \text{csucc}(\text{nat}))-\{0\}\} \) using compact_closed

moreover have \( K \cup (K - \{\text{csucc}(\text{nat})\}) = (K - \{\text{csucc}(\text{nat})\}) \) by auto
ultimately have \( K \cup (K - \{\text{csucc}(\text{nat})\}) \{\text{is compact in}\} (\text{Pow}(\text{csucc}(\text{nat}))) \)
\( \cup \{\{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable} \ \text{csucc}(\text{nat}))-\{0\}\} \) by auto
with \( \text{ss} \) have \( K \cup (K - \{\text{csucc}(\text{nat})\}) \{\text{is compact in}\} (\text{Pow}(\text{csucc}(\text{nat}))) \)
using compact_imp_compact_subspace comp Compact_is_card_nat by auto

ultimately have \( (K - \{\text{csucc}(\text{nat})\}) \{\text{is compact in}\} (\text{Pow}(\text{csucc}(\text{nat}))) \)
using \( \text{ss} \) unfolding RestrictedTo_def by auto

then have \( (K - \{\text{csucc}(\text{nat})\}) \{\text{is compact in}\} (\text{Pow}(\text{csucc}(\text{nat}))) \)
using extension_pow_subspace(1) by auto

then have \( (\bigcup (\text{Pow}(\text{csucc}(\text{nat}))) \{\text{restricted to}\} (K - \{\text{csucc}(\text{nat})\})) \{\text{is compact in}\} T \)
using \( \text{ss} \) unfolding IsAntiComp_def antiProperty_def using \( \text{ss} \) by auto

then have \( (K - \{\text{csucc}(\text{nat})\}) \approx \text{csucc}(\text{nat}) \) using lepoll_iff_leqpoll subset_imp_lepoll[of \( K - \{\text{csucc}(\text{nat})\} \) \( \text{csucc}(\text{nat}) \)] by auto
then have \( \text{csucc}(\text{nat}) \approx K - \{\text{csucc}(\text{nat})\} \) using eqpoll_sym by auto
then have \( \text{nat} < K - \{\text{csucc}(\text{nat})\} \) using lesspoll_eq_trans lt_csucc[OF Ord_nat] \( \text{lt} \) Card_imp_lesspoll[OF Card_csucc[OF Ord_nat]] by auto
then have \( \text{nat} < (K - \{\text{csucc}(\text{nat})\}) \) using lepoll_iff_leqpoll by auto
then obtain \( f \) where \( f \in \text{inj}(\text{nat}, K - \{\text{csucc}(\text{nat})\}) \) using fun_is_surj by auto
ultimately have \( f \in \text{bij}(\text{nat}, \text{range}(f)) \) unfolding bij_def inj_def surj_def by auto
then have \( \text{nat} \approx \text{range}(f) \) unfolding eqpoll_def by auto
then have \( e : \text{range}(f) \approx \text{nat} \) using eqpoll_sym by auto
then have \( \text{as} 2 : \text{range}(f) < \text{csucc}(\text{nat}) \) using \( \text{lt} \) Card_imp_lesspoll[OF
Card_csucc[OF Ord_nat]
  it_csucc[OF Ord_nat] eq_lesspoll_trans by auto
  then have range(f){is closed in}(CoCountable csucc(nat)) using
  closed_sets_cocardinal[of csucc(nat)] unfolding Ccountable_def using func1_1_L5B[OF
  fun] ss noE by auto
  then have csucc(nat)-(range(f))∈(CoCountable csucc(nat)) unfolding
  IsClosed_def Ccountable_def using union_cocardinal[OF noE] by auto
  ultimately have {csucc(nat)} ∪(csucc(nat)-(range(f)))∈{{csucc(nat)}∪S.
  S∈(CoCountable csucc(nat))-{0}}
  by auto
  moreover have {csucc(nat)} ∪(csucc(nat)-(range(f)))=(csucc(nat)}
  ∪ csucc(nat)-(range(f)) using func1_1_L5B[OF fun] by blast
  ultimately have {{csucc(nat)} ∪ csucc(nat)-(range(f))∈{{csucc(nat)}∪S.
  S∈(CoCountable csucc(nat))}-{0}} by auto
  then have {(csucc(nat)) ∪ csucc(nat)-(range(f))∈(Pow(csucc(nat))
  ∪ {(csucc(nat))∪S. S∈(CoCountable csucc(nat))}-{0})
  by auto
  moreover have csucc(nat) ∪ {csucc(nat)}=csucc(nat) ∪ csucc(nat) by auto
  ultimately have (csucc(nat)) ∪ {csucc(nat)}-(range(f))∈(Pow(csucc(nat))
  ∪ {(csucc(nat))∪S. S∈(CoCountable csucc(nat))}-{0})
  by auto
  then have (∪(Pow(csucc(nat)) ∪ {(csucc(nat))∪S. S∈(CoCountable
  csucc(nat))}-{0}))-(range(f))∈(Pow(csucc(nat)) ∪ {(csucc(nat))∪S. S∈(CoCountable
  csucc(nat))}-{0})
  using extension_pow_union by auto
  moreover have range(f)⊆∪(Pow(csucc(nat)) ∪ {(csucc(nat))∪S. S∈(CoCountable
  csucc(nat))}-{0}) using func1_1_L5B[OF fun] by auto
  ultimately have (range(f)) {is closed in}(Pow(csucc(nat)) ∪ {(csucc(nat))∪S.
  S∈(CoCountable csucc(nat))}-{0}) unfolding IsClosed_def by blast
  with comp have (∩(range(f))) {is compact in}(Pow(csucc(nat))
  ∪ {(csucc(nat))∪S. S∈(CoCountable csucc(nat))}-{0}) using compact_closed
  Compact_is_card_nat by auto
  moreover have K∩(range(f))=(range(f)) using func1_1_L5B[OF fun]
  by auto
  ultimately have (range(f)) {is compact in}(Pow(csucc(nat)) ∪ {(csucc(nat))∪S.
  S∈(CoCountable csucc(nat))}-{0}) by auto
  with ss func1_1_L5B[OF fun] have (range(f)) {is compact in}(Pow(csucc(nat))
  ∪ {(csucc(nat))∪S. S∈(CoCountable csucc(nat))}-{0}){restricted to}csucc(nat)
  using compact_imp_compact_subspace[of range(f) nat Pow(csucc(nat))
  ∪ {(csucc(nat))∪S. S∈(CoCountable csucc(nat))}-{0}] csucc(nat) comp Compact_is_card_nat

1249
by auto
  then have \((\text{range}(f))\)\{is compact in\}(\Pow(\text{csucc}(\text{nat})))\) using \text{extension_pow_subspace}\(1\)
by auto
  then have \((\text{range}(f))\)\{is compact in\}(\Pow(\text{csucc}(\text{nat}))\{\text{restricted to}\}(\text{range}(f)))\) using \text{compact_imp_compact_subspace}
Compact_is_card_nat by auto
moreover
  have \(\bigcup(\Pow(\text{csucc}(\text{nat}))\{\text{restricted to}\}(\text{range}(f)))=(\text{range}(f))\)
using \text{ss funcl_1_L5B}[OF fun] unfolding \text{RestrictedTo_def} by auto
ultimately have \((\bigcup(\Pow(\text{csucc}(\text{nat}))\{\text{restricted to}\}(\text{range}(f))))\)\{is compact in\}(\Pow(\text{csucc}(\text{nat})))\)
  unfolding \text{IsAntiComp_def antiProperty_def} using \text{ss funcl_1_L5B}[OF fun] by auto
then have Finite(\text{range}(f)) using \text{compact_spectrum} by auto moreover
  then have Finite(\text{nat}) using e eqpoll_imp_Finite_iff by auto ultimately
  have False using nat_not_Finite by auto
By auto
ultimately have \(K\)\{is in the spectrum of\}(\(\lambda T. (\bigcup T)\)\{is compact in\} \(T\))
by auto
then show thesis unfolding \text{IsAntiComp_def antiProperty_def} by auto qed

If a topological space is KC, then its one-point compactification is US.

\text{theorem (in topology0) KC_imp_OP_comp_is_US:}
  \text{assumes T\{is KC\}
  \text{shows (\{one-point compactification of\}T)\{is US\}}
proof-

- \text{fix \(N\ x\ y\ assume A:\text{N:nat}\rightarrow\bigcup(\{one-point compactification of\}T)\ (N,Le)\rightarrow_X
x\{in\}(\{one-point compactification of\}T)\ (N,Le)\rightarrow_N y\{in\}(\{one-point compactification of\}T)\ x\neq y
  have \text{dir:Le directs nat using \text{Le_directs_nat}}(2).
  from A(1) have \text{dom:domain}(N)=\text{nat using \text{func1_1_L1} by auto
  with dir A(1) have \text{NET:}(N,Le)\{is a net on\}\bigcup(\{one-point compactification of\}T)\ unfolding \text{IsNet_def by auto
  have \text{xy:x}\in\bigcup(\{one-point compactification of\}T)\ y\in\bigcup(\{one-point compactification of\}T)\}
  using A(2,3) \text{topology0.NetConverges_def}[OF _ NET] unfolding \text{topology0_def using \text{op_comp_is_top} dom by auto
  then have \text{pp:x}\in\bigcup T \{\bigcup T\} y\in\bigcup T \{\bigcup T\} using \text{op_compact_total by auto
  from A(2) have \text{comp:}V\in\Pow(\bigcup(\{one-point compactification of\}T)).
\[ x \in \text{Interior}(U, \{\text{one-point compactification of} T\}) \rightarrow \\
(\exists t \in \text{nat. } \forall m \in \text{nat. } (t, m) \in \text{Le} \rightarrow N \; m \in U) \text{ using topology0.NetConverges_def[OF _ NET, of } x] \\
\text{unfolding topology0_def using op_comp_is_top dom op_compact_total} \\
\text{by auto} \\
\text{from A(3) have op2: } \forall U \in \text{Pow}(\bigcup \{\text{one-point compactification of} T\}). \\
y \in \text{Interior}(U, \{\text{one-point compactification of} T\}) \rightarrow \\
(\exists t \in \text{nat. } \forall m \in \text{nat. } (t, m) \in \text{Le} \rightarrow N \; m \in U) \text{ using topology0.NetConverges_def[OF _ NET, of } y] \\
\text{unfolding topology0_def using op_comp_is_top dom op_compact_total} \\
\text{by auto} \\
\{ \\
\begin{align*}
\text{assume p:x} & \in \bigcup T \\
\text{assume B: } & \exists n \in \text{nat. } \forall m \in \text{nat. } (n, m) \in \text{Le} \rightarrow N m = \bigcup T \\
\text{have } & \bigcup T \in (\{\text{one-point compactification of} T\}) \text{ using open_subspace} \\
\text{by auto} \\
\text{then have } & \bigcup T = \text{Interior}(\bigcup T, \{\text{one-point compactification of} T\}) \text{ using topology0.Top_2_L3} \\
\text{unfolding topology0_def using op_comp_is_top by auto} \\
\text{then have } & x \in \text{Interior}(\bigcup T, \{\text{one-point compactification of} T\}) \text{ using p(1) by auto moreover} \\
\text{have } & \bigcup T \in \text{Pow}(\bigcup (\{\text{one-point compactification of} T\})) \text{ using open_subspace(1)} \\
\text{by auto} \\
\text{ultimately have } & \exists t \in \text{domain}(\text{fst}(N, \text{Le})). \; \forall m \in \text{domain}(\text{fst}(N, \text{Le})). \\
(t, m) & \in \text{snd}(N, \text{Le}) \rightarrow \text{fst}(N, \text{Le}) \; m \in \bigcup T \text{ using A(2)} \\
\text{using topology0.NetConverges_def[OF _ NET] op_comp_is_top unfolding topology0_def by blast} \\
\text{then have } & \exists t \in \text{nat. } \forall m \in \text{nat. } (t, m) \in \text{Le} \rightarrow N \; m \in \bigcup T \text{ using dom} \\
\text{by auto} \\
\text{then obtain } & t \text{ where } t : t \in \text{nat} \; \forall m \in \text{nat. } (t, m) \in \text{Le} \rightarrow N \; m \in \bigcup T \\
\text{by auto} \\
\text{from B obtain } & n \text{ where } n : n \in \text{nat} \; \forall m \in \text{nat. } (n, m) \in \text{Le} \rightarrow N m = \bigcup T \text{ by auto} \\
\text{from } & t(1) \; n(1) \text{ dir obtain } z \text{ where } z : z \in \text{nat} \; (n, z) \in \text{Le} \; (t, z) \in \text{Le unfolding IsDirectedSet_def} \\
\text{by auto} \\
\text{from } & t(2) \; z(1, 3) \text{ have } N z \in \bigcup T \text{ by auto moreover} \\
\text{from } & n(2) \; z(1, 2) \text{ have } N z = \bigcup T \text{ by auto ultimately} \\
\text{have False using mem_not_refl by auto} \\
\} \\
\text{then have reg: } \forall n \in \text{nat. } \exists m \in \text{nat. } N m \neq \bigcup T \land (n, m) \in \text{Le} \text{ by auto} \\
\text{let } & \text{NN} = \{n, N(n) \mid i. N i \neq \bigcup T \land (n, i) \in \text{Le}\}. n \in \text{nat}\} \\
\{ \\
\text{fix } x \; z \text{ assume A1: } (x, z) \in \text{NN} \\
\{ \\
\text{fix } y' \text{ assume A2: } (x, y') \in \text{NN} \\
\text{with A1 have } z = y' \text{ by auto} \\
\} \\
\} \\
\} \\
\} \\
\} \\
\}
then have $\forall y'. (x, y') \in \text{NN} \rightarrow z = y'$ by auto
}
then have $\forall x. z. (x, z) \in \text{NN} \rightarrow (\forall y'. (x, y') \in \text{NN} \rightarrow z = y')$ by auto moreover
{
  fix $n$ assume as:$\forall n \in \text{nat}$
  with reg obtain $m$ where $\text{Nm} \cap (n, m) \in \text{Le} \cap \text{nat}$ by auto
  then have $\text{Li} : \forall \mu. i. \text{Ni} \cap (n, i) \in \text{Le} \neq \text{UT} \cap (n, \mu) i. \text{Ni} \cap \text{UT} \neq (n, i) \in \text{Le}$ unfolding $\text{LeastI}$ by auto
  then have $\text{N}(\mu, n, m) \neq (n, i) \in \text{Le}$ by auto
  then have $\forall (\mu, i. \text{Ni} \neq \text{UT} \cap (n, i) \in \text{Le}) \in \text{UT} \cup \text{Le}$ (one-point compactification of $\text{UT}$) using $\text{apply_type}$ by auto
  with an have $(n, N(\mu, i. \text{Ni} \neq \text{UT} \cap (n, i) \in \text{Le})) \in \text{nat} \times \text{UT}$ (one-point compactification of $\text{UT}$) by auto
}
then have $\text{NN} \cap \text{Pow}(\text{nat} \times \text{UT})$ (one-point compactification of $\text{UT}$) unfolding $\text{Pi_def}$ $\text{function_def}$ $\text{domain_def}$ by auto
{
  fix $n$ assume as:$\forall n \in \text{nat}$
  with reg obtain $m$ where $\text{Nm} \cap (n, m) \in \text{Le} \cap \text{nat}$ by auto
  then have $\text{Li} : \forall \mu. i. \text{Ni} \cap (n, i) \in \text{Le} \neq \text{UT} \cap (n, \mu) i. \text{Ni} \cap \text{UT} \neq (n, i) \in \text{Le}$ unfolding $\text{LeastI}$ by auto
  then have $\forall (\mu, n, m) \neq (n, i) \in \text{Le}$ by auto
  then have $\forall (n, N(\mu, i. \text{Ni} \neq \text{UT} \cap (n, i) \in \text{Le})) \in \text{nat} \times \text{UT}$ using $\text{apply_equality}$ by auto
}
then have $\forall n. \forall \text{NN} \neq \text{UT} \rightarrow \text{UT}$ by auto
then have $\forall n. \forall \text{NN} \in \text{UT} \cap \text{UT}$ (one-point compactification of $\text{UT}$) unfolding $\text{IsNet_def}$ using $\text{R_def}$ by auto
{
  fix $U$ assume $U : \forall U \subseteq \text{UT} \cap \text{int}(U)$
  have $\text{int}T : \text{int}(U) \cap \text{UT}$ unfolding $\text{Top_2_L2}$ by auto
  then have $\text{int}(U) \in \text{UT}$ (one-point compactification of $\text{UT}$) unfolding $\text{OPCompaction_def}$ by auto
  then have $\text{Interior}(\text{int}(U), \text{UT}) = \text{int}(U)$ using $\text{topology0}$ $\text{Top_2_L3}$ unfolding $\text{topology0}$ $\text{def}$ using $\text{op_comp_is_top}$ by auto
  with $\text{UT}(2)$ have $x \in \text{Interior}(\text{int}(U), \text{UT})$ by auto
  then have $\forall x : \forall x \in \text{int}(U)$ (one-point compactification of $\text{UT}$) by auto
  with $\text{int}(U)$ have $\exists r \in \text{nat}. \forall s \in \text{nat}. (r, s) \in \text{Le} \rightarrow \text{N}(s) \in \text{int}(U)$ unfolding $\text{comp}$ by auto
  then obtain $r$ where $r : \forall r \in \text{nat}. \forall s \in \text{nat}. (r, s) \in \text{Le} \rightarrow \text{N}(s) \in \text{UT}$
}
ing Top_2_L1 by auto

\{ 
  fix s assume AA:\langle r,s \rangle \in \text{Le}
  with reg obtain m where \text{Nm} \neq \bigcup T \langle s,m \rangle \in \text{Le} by auto
  then have \langle s,\mu i. \text{Ni} \neq \bigcup T \langle s,i \rangle \in \text{Le} \rangle \in \text{Le} using \text{LeastI[of } \lambda m.

\text{Nm} \neq \bigcup T \land \langle s,m \rangle \in \text{Le} m
  nat_into_Ord by auto
  with AA have \langle r,\mu i. \text{Ni} \neq \bigcup T \land \langle s,i \rangle \in \text{Le} \rangle \in \text{Le} using \text{le_trans by auto}

  with r_def(2) have \text{N(}\mu i. \text{Ni} \neq \bigcup T \land \langle s,i \rangle \in \text{Le}) \in \text{U} by blast
  then have \text{NNs} \in \text{U} using \text{apply_equality[OF _ NFun] AA by auto}

  \}

then have \forall s \in \text{nat. } \langle r,s \rangle \in \text{Le} \rightarrow \text{NNs} \in \text{U} by auto

with r_def(1) have \exists r \in \text{nat. } \forall s \in \text{nat. } \langle r,s \rangle \in \text{Le} \rightarrow \text{NNs} \in \text{U} by auto

{ }

p(1) op_comp_is_top

unfolding topology0_def using xy(1) dom2 by auto

{ 
  fix U assume U:U \subseteq \bigcup T y \in \text{int(U)}
  have intT:int(U) \in T using Top_2_L2 by auto
  then have int(U) \in \text{(one-point compactification of} T) unfolding \text{OPCompactification_def}
  by auto
  then have \text{Interior(int(U),\text{(one-point compactification of} T))=int(U)}
  using topology0.Top_2_L3
    unfolding topology0_def using \text{op_comp_is_top by auto}
    with U(2) have y \in \text{Interior(int(U),\text{(one-point compactification of} T))}
    unfolding \text{op2 op_compact_total by auto}
    then obtain r where r_def:r \in \text{nat \forall s \in \text{nat. } \langle r,s \rangle \in \text{Le} \rightarrow \text{Ns} \in \text{U using Top_2_L1 by auto}

  { 
    fix s assume AA:\langle r,s \rangle \in \text{Le}
    with reg obtain m where \text{Nm} \neq \bigcup T \langle s,m \rangle \in \text{Le} by auto
    then have \langle s,\mu i. \text{Ni} \neq \bigcup T \langle s,i \rangle \in \text{Le} \rangle \in \text{Le} using \text{LeastI[of } \lambda m.

    \text{Nm} \neq \bigcup T \land \langle s,m \rangle \in \text{Le} m
      nat_into_Ord by auto
      with AA have \langle r,\mu i. \text{Ni} \neq \bigcup T \land \langle s,i \rangle \in \text{Le} \rangle \in \text{Le} using \text{le_trans by auto}

      with r_def(2) have \text{N(}\mu i. \text{Ni} \neq \bigcup T \land \langle s,i \rangle \in \text{Le}) \in \text{U} by blast
      then have \text{NNs} \in \text{U using \text{apply_equality[OF _ NFun] AA by auto}

      \}

    then have \forall s \in \text{nat. } \langle r,s \rangle \in \text{Le} \rightarrow \text{NNs} \in \text{U by auto
      with r_def(1) have \exists r \in \text{nat. } \forall s \in \text{nat. } \langle r,s \rangle \in \text{Le} \rightarrow \text{NNs} \in \text{U by auto

    \}

1253
then have $\forall U \in \text{Pow}(\bigcup T). \ y \in \text{int}(U) 
\rightarrow (\exists r \in \text{nat.} \ \forall s \in \text{nat.} \ \langle r, s \rangle \in \text{Le} \rightarrow s \in U)$ by auto
then have cony: $(NN, \text{Le}) \rightarrow (\forall y \in \bigcup T. y \in \text{int}(U))$ by auto

unfolding topology0_def using xy(2) dom2 by auto
with conx assms have $x=y$ using KC_imp_US unfolding IsUS_def by auto

with A(4) have False by auto

moreover

{ assume AAA: $x/\notin \bigcup T \lor y/\notin \bigcup T$

with pp have $x=\bigcup T \lor y=\bigcup T$ by auto

{ assume B: $\exists n \in \text{nat.} \ \forall m \in \text{nat.} \ \langle n, m \rangle \in \text{Le} \rightarrow N \in \bigcup T$

have $\bigcup T \in \text{pow}(\{\text{one-point compactification of} T\})$ using open_subspace by auto
then have $\bigcup T=\text{Interior}(\bigcup T, \{\text{one-point compactification of} T\})$

unfolding topology0.Top_2_L3 using y by auto

ultimately have $\exists t \in \text{domain}(\text{fst}(\langle N, \text{Le} \rangle)). \ \forall m \in \text{domain}(\text{fst}(\langle N, \text{Le} \rangle)). \ (t, m) \in \text{snd}(\langle N, \text{Le} \rangle) \rightarrow \text{fst}(\langle N, \text{Le} \rangle) \ m \in \bigcup T$ using A(3)

unfolding topology0.NetConverges_def[OF _ NET] op_comp_is_top using topology0.def by blast
then have $\exists t \in \text{nat.} \ \forall m \in \text{nat.} \ (t, m) \in \text{Le} \rightarrow N \in \bigcup T$

from B obtain n where $n:n \in \text{nat.} \ \forall m \in \text{nat.} \ (n, m) \in \text{Le} \rightarrow N \in \bigcup T$

from t(1) n(1) dir obtain z where $z:z \in \text{nat.} \ (n, z) \in \text{Le} \ (t, z) \in \text{Le}$

unfolding IsDirectedSet_def by auto

from t(2) z(1,3) have Nz$\in \bigcup T$ by auto ultimately
from n(2) z(1,2) have Nz$=\bigcup T$ by auto

ultimately have False using mem_not_refl by auto

} then have reg: $\forall n \in \text{nat.} \ \exists m \in \text{nat.} \ N \notin \bigcup T \land (n, m) \in \text{Le}$ by auto

let NN$=\{(n, N(\mu \ i. \ N_i \notin \bigcup T \land (n, i) \in \text{Le})). \ n \in \text{nat}\}$

{ fix $x \ z$ assume A1: $\langle x, z \rangle \in \text{NN}$

}
fix y' assume A2:⟨x,y'⟩∈NN
with A1 have z=y' by auto
} then have ∀y'. (x,y')∈NN → z=y' by auto
} then have ∀x z. ⟨x, z⟩ ∈ NN → (∀y'. ⟨x,y'⟩∈NN → z=y') by auto
moreover

{ fix n assume as:n∈nat
with reg obtain m where Nm≠∪T ∧ ⟨n,m⟩∈Le m∈nat by auto
then have LI:N(μ i. Ni≠∪T ∧ ⟨n,i⟩∈Le)≠∪T (n,μ i. Ni≠∪T ∧ ⟨n,i⟩∈Le) using LeastI[of λm. Nm≠∪T ∧ ⟨n,m⟩∈Le m]
∧ ⟨n,i⟩∈Le by auto
then have (μ i. Ni≠∪T ∧ ⟨n,i⟩∈Le)∈nat by auto
then have N(μ i. Ni≠∪T ∧ ⟨n,i⟩∈Le)∈∪({one-point compactification of}T) using apply_type[of A(1)] op_compact_total by auto
with as have [n,N(μ i. Ni≠∪T ∧ ⟨n,i⟩∈Le)∈nat×∪({one-point compactification of}T)] by auto
} then have NN∈Pow(nat×∪({one-point compactification of}T)) by auto
ultimately have NFun:NN:nat→∪({one-point compactification of}T) using apply_equality[of _ NFun] by auto

then have noy:∀n∈nat. NNn≠∪T by auto
then have ∀n∈nat. NNn∈∪T using apply_type[of NFun] op_compact_total by auto
then have R:NN:nat→∪T using func1_1_L1A[of NFun] by auto
have dom2:domain(NN)=nat by auto
then have net2:(NN,Le) is a net on ∪T unfolding IsNet_def using R dir by auto

{ fix U assume U⊆∪T y∈int(U)
have intT:int(U)∈T using Top_2_L2 by auto
then have int(U)∈({one-point compactification of}T) unfolding OPCompactification_def by auto
then have Interior(int(U),{one-point compactification of}T)=int(U) using topology0.Top_2_L3 unfolding topology0_def using op_comp_is_top by auto
with U(2) have y∈Interior(int(U),{one-point compactification of}T)
of} T by auto
  with int T have (∃ r ∈ nat. ∀ s ∈ nat. ⟨r, s⟩ ∈ Le → Ns ∈ int(U)) using op2 op_comp_total by auto
  then obtain r where r_def : r ∈ nat. ⟨r, s⟩ ∈ Le → Ns ∈ U using Top_2_L1 by auto
  obtain r where r_def : r ∈ nat. ∀ s ∈ nat. ⟨r, s⟩ ∈ Le → Ns ∈ U using op2 op_comp_total by auto
  then
  { fix s assume AA : ⟨r, s⟩ ∈ Le with reg
    obtain m where Nm ≠ ∪ T ∧ ⟨s, m⟩ ∈ Le by auto
    then have ⟨s, µ i. Ni ≠ ∪ T ∧ ⟨s, i⟩ ∈ Le⟩ ∈ Le using LeastI[of λ m. Nm ≠ ∪ T ∧ ⟨s, m⟩ ∈ Le]
    by auto
  }
  then have ∀ U ∈ Pow(∪ T). y ∈ int(U) → (∃ r ∈ nat. ∀ s ∈ nat. ⟨r, s⟩ ∈ Le → Ns ∈ U) by auto
  then have y op_comp_is_top unfolding topology0_def using xy(2) dom2 by auto
  let A = {y} ∪ NNNat
  { fix M assume Acov : A ⊆ ∪ M ⊆ T
    then have y ∈ ∪ M by auto
    then obtain V where V : y ∈ V V ∈ M by auto
    with Acov(2) have VT : V ∈ int T by auto
    then have V = int V using Top_2_L3 by auto
    with V(1) have y ∈ int V by auto
    with cony obtain r where rr : r ∈ nat. ∀ s ∈ nat. ⟨r, s⟩ ∈ Le → Ns ∈ V
    unfolding NetConverges_def[of net2, of y] using dom2 VT y by auto
  }
  have NresFun : restrict(NN, {n ∈ nat. ⟨n, r⟩ ∈ Le}) : {n ∈ nat. ⟨n, r⟩ ∈ Le} → ∪ T
  using restrict_fun
  [OF R, of {n ∈ nat. ⟨n, r⟩ ∈ Le}] by auto
  then have restric(NN, {n ∈ nat. ⟨n, r⟩ ∈ Le}) ∈ surj({n ∈ nat. ⟨n, r⟩ ∈ Le}, range(restrict(NN, {n ∈ nat. ⟨n, r⟩ ∈ Le}))
  unfolding fun_is_surj by auto moreover
  have {n ∈ nat. ⟨n, r⟩ ∈ Le} ⊆ nat by auto
  then have {n ∈ nat. ⟨n, r⟩ ∈ Le} ⊆ nat using subset_imp_lepoll by auto
  ultimately have range(restrict(NN, {n ∈ nat. ⟨n, r⟩ ∈ Le})) ⊆ {n ∈ nat. ⟨n, r⟩ ∈ Le}
  using surj_fun_inv_2 by auto
  moreover
  have {n ∈ nat. ⟨n, 0⟩ ∈ Le} = {0} by auto
  then have Finite({n ∈ nat. ⟨n, 0⟩ ∈ Le}) by auto moreover

1256
\begin{verbatim}
{ fix j assume as: j \in \mathbb{N} \exists \{ n \in \mathbb{N} . \langle n, j \rangle \in \text{Le} \} 
{ fix t assume t \in \{ n \in \mathbb{N} . \langle n, \text{succ}(j) \rangle \in \text{Le} \} 
  then have t \in \mathbb{N} \langle t, \text{succ}(j) \rangle \in \text{Le} \text{ by auto} 
  then have t \leq \text{succ}(j) \text{ by auto} 
  then have t \subseteq \text{succ}(j) \text{ using le_imp_subset by auto} 
  then have j \in t \cup \{ j \} \text{ using succ_explained by auto} 
  then have j \in t \subseteq \text{succ}(j) \text{ by auto} 
  then have j \in t \subseteq j \text{ using subset_imp_subset by auto} 
  then have \langle t, \text{succ}(j) \rangle \in \text{Le} \text{ by auto} 
  then have \{ n \in \mathbb{N} . \langle n, j \rangle \in \text{Le} \} \cup \{ \text{succ}(j) \} \subseteq \{ n \in \mathbb{N} . \langle n, \text{succ}(j) \rangle \in \text{Le} \} \cup \{ \text{succ}(j) \} \text{ by auto} 
  moreover have Finite(\{ n \in \mathbb{N} . \langle n, \text{succ}(j) \rangle \in \text{Le} \} \cup \{ \text{succ}(j) \}) \text{ by auto} 
  ultimately have Finite(\{ n \in \mathbb{N} . \langle n, j \rangle \in \text{Le} \}) \text{ by auto} 
  \} 
  then have \forall j \in \mathbb{N} \exists \{ n \in \mathbb{N} . \langle n, j \rangle \in \text{Le} \} \to Finite(\{ n \in \mathbb{N} . \langle n, \text{succ}(j) \rangle \in \text{Le} \}) \text{ by auto} 
  ultimately have Finite(\{ n \in \mathbb{N} . \langle n, \text{succ}(j) \rangle \in \text{Le} \}) \text{ by auto} 
  \} 
  then have (\{ n \in \mathbb{N} . \langle n, j \rangle \in \text{Le} \}) \{\text{is in the spectrum of}\} \{ \lambda T. (T \text{ restricted to}) \mathbb{N} \{ n \in \mathbb{N} . \langle n, r \rangle \in \text{Le} \} \} \{\text{is in the spectrum of}\} \{ \lambda T. (T \text{ restricted to}) \mathbb{N} \{ n \in \mathbb{N} . \langle n, r \rangle \in \text{Le} \} \} \text{ by auto} 
  \} 
  un
\end{verbatim}
a topology\{using Top_1_L4 unfolding topology0_def by auto
ultimately have (NN\{n ∈ nat . (n, r) ∈ Le\})\{is compact in\}(T\{restricted
to\}NN\{n ∈ nat . (n, r) ∈ Le\})
  unfolding Spec_def by force
then have (NN\{n ∈ nat . (n, r) ∈ Le\})\{is compact in\}(T) using compact_subspace_imp_compact by auto
moreover from Acov(1) have (NN\{n ∈ nat . (n, r) ∈ Le\})⊆(∪M)
  unfolding IsCompact_def by blast
ultimately have \textcircled{1}
now assume s:A)
s∈V by auto
then have s∈\{NNn. n∈nat\} using func_imagedef[of NFun] by auto
then obtain n where n:n∈nat s=NNn by auto
{ assume ⟨r,n⟩∈Le
  with rr have NNn∈V by auto
  with n(2) s(2) have False by auto
} then have ⟨r,n⟩∉Le by auto
with rr(1) n(1) have ¬(r≤n) by auto
then have n≤r using Ord_linear_le[where thesis=⟨n,r⟩∈Le]
ultimately have s∈NN\{n∈nat. (n,r)∈Le\} by auto
moreover have \{n∈nat. (n,r)∈Le\}⊆nat by auto
ultimately have s∈NN\{n∈nat. (n,r)∈Le\} by auto
moreover have ss:A⊆∪\mathcal{V} by auto
then have A⊆∪\mathcal{V} by auto ultimately have ∃\mathcal{N}∈FinPow(M). A⊆∪\mathcal{N} by auto
moreover have ss:A⊆∪\mathcal{V} \{is compact in\}(T) using func1_1_L6(2)[OF R] \{by blast ultimately have A\{is compact in\}(T) unfolding IsCompact_def by auto more-
over with assms have A\{is closed in\}(T) unfolding IsKC_def IsCompact_def

1258
by auto ultimately
  have A ∈ {B ∈ Pow(∪ T). B is compact in}(T) ∧ B is closed in}(T) using ss by auto
  then have {∪ T} ∪ (∪ T - A) ∈ (one-point compactification of)T unfolding OPCompactification_def
  by auto
  then have {∪ T} ∪ (∪ T - A) = Interior({∪ T} ∪ (∪ T - A), {one-point compactification of)T) using topology0.Top_2_L3 op_comp_is_top unfolding topology0_def by auto
  moreover have \{ assume x ∈ A
  with A(4) have x ∈ NNat by auto
  then have x ∈ {NNn. n ∈ nat} using func_imagedef[OF NFun] by auto
  with noy x have False by auto
  \}
  with y have x ∈ {∪ T} ∪ (∪ T - A) using x by force ultimately
  have x ∈ Interior({∪ T} ∪ (∪ T - A), {one-point compactification of)T) using op_compact_total by auto moreover
  have (∀ U ∈ Pow({\{one-point compactification of)T}). x ∈ Interior(U, {one-point compactification of)T) → (∃ t ∈ nat. ∀ m ∈ nat. ⟨ t, m ⟩ ∈ Le → N m ∈ U)) using A(2) dom topology0.NetConverges_def[OF _ NET] op_comp_is_top unfolding topology0_def by auto
  ultimately have ∃ t ∈ nat. ∀ m ∈ nat. ⟨ t, m ⟩ ∈ Le → N m ∈ {∪ T} ∪ (∪ T - A) by blast
  then obtain r where r_def: r ∈ nat ∀ s ∈ nat. ⟨ r, s ⟩ ∈ Le → N s ∈ {∪ T} ∪ (∪ T - A) by auto
  \{
  fix s assume AA: (r, s) ∈ Le
  with reg obtain m where Nm \neq ∪ T \langle s, m \rangle ∈ Le by auto
  then have \langle s, \mu i. Ni \neq ∪ T \langle s, i \rangle ∈ Le \rangle using LeastI[of \lambda m. Nm \neq ∪ T \langle s, m \rangle ∈ Le]
  Nm \neq ∪ T \langle s, m \rangle ∈ Le by auto
  with AA have \langle r, \mu i. Ni \neq ∪ T \langle s, i \rangle ∈ Le \rangle ∈ Le using le_trans by auto
  with r_def(2) have N(\mu i. Ni \neq ∪ T \langle s, i \rangle ∈ Le) ∈ (∪ T) ∪ (∪ T - A)
  by auto
  then have NNs ∈ (∪ T) ∪ (∪ T - A) using apply_equality[OF _ NFun]
  AA by auto
  with noy have NNs ∈ (∪ T - A) using AA by auto
  moreover have NNs ∈ {NNt. t ∈ nat} using AA by auto
  then have NNs ∈ NNNat using func_imagedef[OF NFun] by auto
  then have NNs ∈ A by auto
  ultimately have False by auto
  \}
  moreover have r ∈ succ(r) using succ_explained by auto
  then have r ∈ succ(r) using subset_imp_le nat_into_Ord \langle r ∈ nat \rangle
  nat_succI
by auto
then have \((r, \text{succ}(r)) \in \mathcal{L}_e\) using \(<r \in \text{nat}>\) nat_succI by auto
ultimately have False by auto
}\)
then have \(x \notin \bigcup T\) by auto
with \(xy(1)\) AAA have \(y \notin \bigcup T\) \(x \in \bigcup T\) using op_compact_total by auto
with \(xy(2)\) have \(y \in \bigcup T\) and \(x \in \bigcup T\) using op_compact_total by auto
\{
assume \(B : \exists n \in \text{nat}. \ \forall m \in \text{nat}. \ (n, m) \in \mathcal{L}_e \rightarrow Nm = \bigcup T\)
have \(\bigcup T \in \{\text{one-point compactification of } T\}\) using open_subspace by auto
then have \(\bigcup T = \text{Interior}(\bigcup T, \{\text{one-point compactification of } T\})\) using topology0.Top_2_L3
unfolding topology0_def using op_comp_is_top by auto
then have \(x \in \text{Interior}(\bigcup T, \{\text{one-point compactification of } T\})\) using x by auto moreover
have \(\bigcup T \in \text{Pow}(\bigcup \{\text{one-point compactification of } T\})\) using open_subspace(1) by auto
ultimately have \(\exists t \in \text{domain}(\text{fst}(\langle N, \mathcal{L}_e \rangle)). \ \forall m \in \text{domain}(\text{fst}(\langle N, \mathcal{L}_e \rangle)). \ (t, m) \in \bigcup T\) using dom by auto
then obtain \(t\) where \(t : t \in \text{nat} \ \forall m \in \text{nat}. \ (t, m) \in \mathcal{L}_e \rightarrow Nm \in \bigcup T\) by auto
from \(B\) obtain \(n\) where \(n : n \in \text{nat} \ \forall m \in \text{nat}. \ (n, m) \in \mathcal{L}_e \rightarrow Nm \in \bigcup T\) by auto
from \(t(1)\) \(n(1)\) dir obtain \(z\) where \(z : z \in \text{nat} \ \forall n \in \text{nat} \ \forall z \in \text{nat}. \ \langle n, z \rangle \in \mathcal{L}_e \langle t, z \rangle \in \mathcal{L}_e\) unfolding IsDirectedSet_def by auto
from \(t(2)\) \(z(1, 3)\) have \(Nz \in \bigcup T\) by auto moreover
from \(n(2)\) \(z(1, 2)\) have \(Nz = \bigcup T\) by auto ultimately have False using mem_not_refl by auto
\}
then have \(\reg : \forall n \in \text{nat}. \exists m \in \text{nat}. \ Nm \neq \bigcup T \land (n, m) \in \mathcal{L}_e\) by auto
let \(\text{NN} = \{\langle n, N(\mu i. Ni \neq \bigcup T \land (n, i) \in \mathcal{L}_e) \rangle. \ n \in \text{nat}\}\)
\{
fix \(x z\) assume \(A1 : \langle x, z \rangle \in \text{NN}\)
\{
fix \(y'\) assume \(A2 : \langle x, y' \rangle \in \text{NN}\)
with \(A1\) have \(z = y'\) by auto
\}
then have \(\forall y'. \langle x, y' \rangle \in \text{NN} \rightarrow z = y'\) by auto
\}
then have \(\forall x z. \ (x, z) \in \text{NN} \rightarrow (\forall y'. \langle x, y' \rangle \in \text{NN} \rightarrow z = y')\) by auto moreover
\{
fix \(n\) assume \(\text{as : } n \in \text{nat}\)

1260
with reg obtain m where $N \neq U \land \langle n, m \rangle \in \text{Le} \cap \text{nat}$ by auto
then have LI: $N(\mu \ i. \ N \neq U \land \langle n, i \rangle \in \text{Le}) \neq U \land \langle n, \mu \ i. \ N \neq U \land \langle n, i \rangle \in \text{Le} \rangle \in \text{Le}$ using LeastI[of $\lambda m. \ N \neq U \land \langle n, m \rangle \in \text{Le} \land \text{nat} \cap \text{Ord} \cap \text{of} \ m$] by auto
then have $N(\mu \ i. \ N \neq U \land \langle n, i \rangle \in \text{Le}) \neq U \land \langle n, \mu \ i. \ N \neq U \land \langle n, i \rangle \in \text{Le} \rangle \in \text{Le}$ using apply_type[OF A(i)] op_compact_total by auto
with as have $\langle n, N(\mu \ i. \ N \neq U \land \langle n, i \rangle \in \text{Le} \rangle \in \text{Le} \rangle \in \text{Le} \in \text{nat} \times U \in \langle \text{one-point compactification of} \rangle T \in \text{by auto}$
then have $\text{NN} \in \text{Pow}(\text{nat} \times \cup \langle \text{one-point compactification of} \rangle T)$ by auto
ultimately have $N\text{Fun}: N\text{nat} \to \cup \langle \text{one-point compactification of} \rangle T$ unfolding $\Pi \text{def}$ function_def domain_def by auto
{ fix n assume as:n\in\text{nat}
with reg obtain m where $N \neq U \land \langle n, m \rangle \in \text{Le} \in \text{nat} \in \text{by auto}
then have LI: $N(\mu \ i. \ N \neq U \land \langle n, i \rangle \in \text{Le}) \neq \cup \langle n, \mu \ i. \ N \neq U \land \langle n, i \rangle \in \text{Le} \rangle \neq \cup \langle n, i, \mu \ i. \ N \neq U \land \langle n, i \rangle \in \text{Le} \rangle \in \text{Le}$ using LeastI[of $\lambda m. \ N \neq U \land \langle n, m \rangle \in \text{Le} \land \text{nat} \cap \text{Ord} \cap \text{of} \ m$] by auto
then have $\text{NN} \neq U \in \text{by auto}
then have \text{NN} \in \text{nat} \in \cup \langle \text{one-point compactification of} \rangle T \text{using apply}\_\text{equality}[OF _\text{NPun}] \text{by auto}
}
then have $\text{noy}: \forall n \in \text{nat}. \ 0 \neq U \in \text{by auto}
then have $\forall n \in \text{nat}. \ \text{NN} \in \text{nat} \in \cup \langle \text{one-point compactification of} \rangle T \text{using apply}\_\text{type}[OF \text{NPun}] \text{op}\_\text{compact}\_\text{total} \text{by auto}
then have $R: \text{NN}: \text{nat} \in \cup \langle \text{one-point compactification of} \rangle T \text{using func1\_L1A}[OF \text{NPun}] \text{by auto}
then have $\text{dom2}: \text{domain}(\text{NN}) = \text{nat} \in \text{by auto}
then have $\text{net2}: \langle \text{NN} \in \text{nat} \cap \text{nat} \cap \cup \langle \text{one-point compactification of} \rangle T \text{unfolding \text{IsNet}\_\text{def}} \text{using R \text{dir}} \text{by auto}$
{
fix U assume $U: U \subseteq U \in \text{int}(U) \in \text{by auto}
have $\text{int}T: \text{int}(U) \in \text{U} \in \text{by auto}
then have $\text{int}(U) \in \langle \text{one-point compactification of} \rangle T \text{unfolding \text{OPCompactification}\_\text{def}} \text{by auto}
then have $\text{Interior}(\text{int}(U), \langle \text{one-point compactification of} \rangle T) = \text{int}(U) \text{using topology0.Top2\_L3} \text{unfolding topology0\_def using op}\_\text{comp}\_\text{is}\_\text{top} \text{by auto}
with U(2) have $x \in \text{Interior}(\text{int}(U), \langle \text{one-point compactification of} \rangle T) \text{by auto}
with \text{int}T \text{have} (\exists r \in \text{nat}. \ \forall s \in \text{nat}. \ (r, s) \in \text{Le} \to Ns \in \text{int}(U)) \text{using \text{comp op}\_\text{compact}\_\text{total} \text{by auto}
then obtain r where $r \text{def}: r \in \text{nat} \in \forall s \in \text{nat}. \ (r, s) \in \text{Le} \to Ns \in U \text{using Top2\_L1 by auto}$
{ fix s assume AA: $(r, s) \in \text{Le}
with reg obtain m where $Nm \neq U \land \langle s, m \rangle \in \text{Le} \in \text{by auto}
then have $\langle s, \mu \ i. \ Nm \neq U \land \langle s, i \rangle \in \text{Le} \text{using LeastI[of} \lambda m. \ Nm \neq U \land \langle s, m \rangle \in \text{Le} \rangle \in \text{by auto}$

1261
\text{nat_into_Ord} \text{ by auto}
\text{with } \AA \text{ have } \{r, \mu \text{ i. } \Ni \neq \bigcup T \land \langle s, i \rangle \in \Le \} \in \Le \text{ using } \text{le_trans by auto}
\text{with } \text{r_def(2)} \text{ have } N(\mu \text{ i. } \Ni \neq \bigcup T \land \langle s, i \rangle \in \Le) \in U \text{ by blast}
\text{then have } \NNs \subseteq U \text{ using } \text{apply_equality[OF _ NFun] } \AA \text{ by auto}
\text{then have } \forall s \in \text{nrat}. \langle r, s \rangle \in \Le \rightarrow \NNs \subseteq U \text{ by auto}
\text{with } \text{r_def(1)} \text{ have } \exists r \in \text{nrat}. \forall s \in \text{nrat}. \langle r, s \rangle \in \Le \rightarrow \NNs \subseteq U \text{ by auto}
\text{then have } \forall U \in \text{Pow}(\bigcup T). \ x \in \text{int}(U)
\rightarrow \langle \exists r \in \text{nrat}. \forall s \in \text{nrat}. \langle r, s \rangle \in \Le \rightarrow \NNs \subseteq U \text{ by auto}
\text{then have } \text{cony: } (\langle N, N \rangle \rightarrow N) x(\text{in}) T \text{ using } \text{NetConverges_def[OF net2]}
x \text{ op_comp_is_top}
\text{unfolding } \text{topology0_def using } x y(2) \text{ dom2 by auto}
\text{let } A = \{x \} \subseteq \text{nNNnat}
\{\text{fix } M \text{ assume } \text{Acov}: A \subseteq \bigcup M \subseteq T
\text{then have } x \in \bigcup M \text{ by auto}
\text{then obtain } V \text{ where } V : x \in V \subseteq M \text{ by auto}
\text{with } \text{Acov(2)} \text{ have } VT : V \subseteq T \text{ by auto}
\text{then have } V = \text{int}(V) \text{ using } \text{Top_2_L3 by auto}
\text{with } V(1) \text{ have } x \in \text{int}(V) \text{ by auto}
\text{with } \text{cony VT obtain } r \text{ where } \text{rr : } r \in \text{nrat} \forall s \in \text{nrat}. \langle r, s \rangle \in \Le \rightarrow \NNs \subseteq V
\text{unfolding } \text{NetConverges_def[OF net2, of x] using } \text{dom2 y by auto}
\text{have } \text{NresFun: } \text{restrict}(NN, \{n \in \text{nrat}. \langle n, r \rangle \in \Le \} : \{n \in \text{nrat}. \langle n, r \rangle \in \Le \} \rightarrow \bigcup T
\text{using } \text{restrict_fun}
\[\text{OF R, of } \{n \in \text{nrat}. \langle n, r \rangle \in \Le \} \} \text{ by auto}
\text{then have } \text{restrict}(NN, \{n \in \text{nrat}. \langle n, r \rangle \in \Le \}) \subseteq \text{surj}(\{n \in \text{nrat}. \langle n, r \rangle \in \Le \}, \text{range( } \text{restrict}(NN, \{n \in \text{nrat}. \langle n, r \rangle \in \Le \} ))
\text{using } \text{fun_is_surj by auto moreover}
\text{have } \{n \in \text{nrat}. \langle n, r \rangle \in \Le \} \subseteq \text{nrat by auto}
\text{then have } \text{nnat(} \{n \in \text{nrat}. \langle n, r \rangle \in \Le \} \subseteq \text{nrat using } \text{subset_imp_lepoll by auto}
\text{ultimately have } \text{range( } \text{restrict}(NN, \{n \in \text{nrat}. \langle n, r \rangle \in \Le \} )) \subseteq \{n \in \text{nrat}. \langle n, r \rangle \in \Le \} \text{ using } \text{surj_fun_inv_2 by auto}
\text{moreover}
\text{have } \{n \in \text{nrat}. \langle n, 0 \rangle \in \Le \} = \{0\} \text{ by auto}
\text{then have } \text{Finite}(\{n \in \text{nrat}. \langle n, 0 \rangle \in \Le \}) \text{ by auto moreover}
\{\text{fix } j \text{ assume as: } j \in \text{nrat } \text{Finite}(\{n \in \text{nrat}. \langle n, j \rangle \in \Le \})
\{\text{fix } t \text{ assume } t \in \{n \in \text{nrat}. \langle n, \text{succ}(j) \rangle \in \Le \}
\text{then have } t \in \text{nrat } \langle t, \text{succ}(j) \rangle \in \Le \text{ by auto}
\text{then have } t \leq \text{succ}(j) \text{ by auto}
\text{then have } t \leq \text{succ}(j) \text{ using } \text{le_imp_subset by auto}
\text{then have } t \subseteq \text{U} \{j\} \text{ using } \text{succ_explained by auto}
\text{then have } j \in \text{tVt} \subseteq j \text{ by auto}
\text{then have } j \in \text{tVt} \subseteq j \text{ using } \text{subset_imp_le } t \in \text{nrat} \langle j \in \text{nrat} \rangle \text{ nat_into_Ord by auto}
\text{then have } j \cup \{j\} \subseteq tVt \leq j \text{ using } t \in \text{nrat} \langle j \in \text{nrat} \rangle \text{ nat_into_Ord}
\}
\}
\}
1262
unfolding Ord_def by auto
  then have succ(j) ≤ t ∧ t ≤ j using succ_explained by auto
  with t ≤ succ(j) have t = succ(j) ∨ t ≤ j by auto
  with t ∈ nat ∧ j ∈ nat have t ∈ {n ∈ nat. ⟨n, j⟩ ∈ Le} ∪ {succ(j)}
  by auto
\}
  then have {n ∈ nat. ⟨n, succ(j)⟩ ∈ Le} ⊆ {n ∈ nat. ⟨n, j⟩ ∈ Le} ∪ {succ(j)}
  by auto
moreover have Finite({n ∈ nat. ⟨n, j⟩ ∈ Le} ∪ {succ(j)}) using as(2)
Finite_cons
  by auto
ultimately have Finite({n ∈ nat. ⟨n, succ(j)⟩ ∈ Le})
  by auto
moreover have ∃j ∈ nat. Finite({n ∈ nat. ⟨n, j⟩ ∈ Le}) → Finite({n ∈ nat. ⟨n, j⟩ ∈ Le})
  by auto
ultimately have Finite(range(restrict(NN, {n ∈ nat . ⟨n, r⟩ ∈ Le})))
  using lepoll_Finite[of range(restrict(NN, {n ∈ nat . ⟨n, r⟩ ∈ Le}))]
ind_on_nat[OF r ∈ nat, where P = λt. Finite({n ∈ nat. ⟨n, t⟩ ∈ Le})]
  by auto
then have Finite(range(restrict(NN, {n ∈ nat . ⟨n, r⟩ ∈ Le})))
  using range_image_domain[of NresFun]
by auto
then have Finite(range(restrict(NN, {n ∈ nat . ⟨n, r⟩ ∈ Le})))
  using restrict_image
by auto
then have (NN{n ∈ nat . ⟨n, r⟩ ∈ Le}){is in the spectrum of} (∪ T){is compact in} T
  using compact_spectrum by auto
moreover have ∪ (T{restricted to}NN{n ∈ nat . ⟨n, r⟩ ∈ Le}) = ∪ T \ NN{n ∈ nat . ⟨n, r⟩ ∈ Le}
  unfolding RestrictedTo_def by auto moreover
have ∪ T \ NN{n ∈ nat . ⟨n, r⟩ ∈ Le} = NN{n ∈ nat . ⟨n, r⟩ ∈ Le}
  using func1_1_L6(2)[OF R] by blast
moreover have (T{restricted to}NN{n ∈ nat . ⟨n, r⟩ ∈ Le}){is a topology}
  using Top_1_L4 unfolding topology0_def by auto
ultimately have (NN{n ∈ nat . ⟨n, r⟩ ∈ Le}){is compact in} (T{restricted to}NN{n ∈ nat . ⟨n, r⟩ ∈ Le})
  unfolding Spec_def by force
then have (NN{n ∈ nat . ⟨n, r⟩ ∈ Le}){is compact in} (T) using compact_subspace_imp_compact by auto
moreover from Acov(1) have (NN{n ∈ nat . ⟨n, r⟩ ∈ Le}) ⊆ ∪ M by auto
moreover note Acov(2) ultimately
obtain N where N : N ∈ FinPow(M) (NN{n ∈ nat . ⟨n, r⟩ ∈ Le}) ⊆ ∪ N
  unfolding IsCompact_def by blast
1263
from Ω(1) have Ω ∪ {V} ∈ FinPow(M) using V(2) unfolding FinPow_def
by auto moreover

{ fix s assume s:s ∈ A s ∉ V
  with V(1) have s:∈ N N nat by auto
  then have s:∈ {NNn. n ∈ nat} using func_imagedef[OF NFun] by auto
  then obtain n where n:n ∈ nat by auto
  { assume ⟨r, n⟩ ∈ Le
    with rr have N N n ∈ V by auto
    with n(2) s(2) have False by auto
  }
  then have ⟨r, n⟩ /∈ Le by auto
  with rr(1) n(1) have ¬ (r ≤ n) by auto
  then have n ≤ r using Ord_linear_le[where thesis = ⟨n, r⟩ ∈ Le] nat_into_Ord[OF rr(1)]
  nat_into_Ord[OF n(1)] by auto
  with rr(1) n(1) have ⟨n, r⟩ ∈ Le by auto
  with n(2) have s:∈ {NNt. t ∈ {n∈ nat. ⟨n, r⟩ ∈ Le}} by auto more-
  over
  have {n∈ nat. ⟨n, r⟩ ∈ Le} ⊆ nat by auto
  ultimately have s:∈ N N n ∈ nat. ⟨n, r⟩ ∈ Le} using func_imagedef[OF NFun]
  by auto
}

then have A ⊆ ∪ Ω ∪ V by auto
then have A ⊆ (Γ ∪ {V}) by auto ultimately
have ∃ N ∈ FinPow(M). A ⊆ Ω by auto
moreover
have ss: A ⊆ ∪ (T) using func1_1_L6(2)[OF R] x by blast ultimately
have A(is compact in)(T) unfolding IsCompact_def by auto more-
over
with assms have A(is closed in)(T) unfolding IsKC_def IsCompact_def
by auto ultimately
have A∈ B:Pow(∪ T). B(is compact in)(T) ∧ B(is closed in)(T) us-
ing ss by auto
then have {∪ T} ∪ (∪ T − A) ∈ {{one-point compactification of}T} un-
folding QPCompactification_def by auto
then have {∪ T} ∪ (∪ T − A) = Interior({∪ T} ∪ (∪ T − A), {one-point compactification
of}T) using topology0.Top_2_L3 op_comp_is_top
  unfolding topology0_def by auto moreover

{ assume y∈ A
  with A(4) have y:∈ N N nat by auto
  then have y:∈ {NNn. n ∈ nat} using func_imagedef[OF NFun] by auto

1264
then obtain \( n \) where \( n \in \mathbb{N} \) by auto
with noy have False by auto
}
with y have \( y \in \bigcup T \cup (\bigcup T-A) \) by force ultimately
have \( y \in \text{Interior}(\bigcup T \cup (\bigcup T-A), \{\text{one-point compactification of} \ T\}) \)
\( \{\bigcup T \cup (\bigcup T-A)\} \in \text{Pow}(\bigcup \{\text{one-point compactification of} \ T\}) \)
using opCompact_total by auto moreover
have \( (\forall U \in \text{Pow}(\bigcup \{\text{one-point compactification of} \ T\}). \ y \in \text{Interior}(U, \{\text{one-point compactification of}\ T\}) \) \)
using A(3) dom topology0.NetConverges_def[of _ NET] op_comp_is_top
unfolding topology0_def by auto
ultimately have \( \exists t \in \mathbb{N}. \forall m \in \mathbb{N}. \langle t, m \rangle \in \text{Le} \rightarrow N m \in \bigcup T \cup (\bigcup T-A) \)
by blast
then obtain \( r \) where \( r \in \mathbb{N} \) by auto

In the one-point compactification of an anti-compact space, ever subspace that contains the infinite point is compact.
theorem (in topology0) anti_comp_imp_OP_inf_comp:
  assumes T:is anti-compact A⊆∪((one-point compactification of)T) U∈A
  shows A:is compact in((one-point compactification of)T)
proof-
  {  
    fix M assume M:M⊆((one-point compactification of)T) A⊆∪M
    with assms(3) obtain U where U:∪T⊆U U⊆M by auto
    with M(1) obtain K where K:K:is compact in)T K:is closed in)T U=(∪T)∩(∪T-K)
    unfolding OPCompactification_def using mem_not_refl[of ∪T] by auto
    from K(1) have K:is compact in)(T{restricted to}K) using compact_imp_compact_subspace
    Compact_is_card_nat
    by auto
    moreover have ∪(T{restricted to}K)=∪T∩K unfolding RestrictedTo_def
    by auto
    with K(1) have ∪(T{restricted to}K)=K unfolding IsCompact_def by auto
    ultimately have (∪(T{restricted to}K)){is compact in}(T{restricted to}K) by auto
    with assms(1) have K:is in the spectrum of}{(λT. (∪T){is compact in)T) using auto
    unfolding IsAntiComp_def
    antiProperty_def using K(1) unfolding IsCompact_def by auto
    then have finK:Finite(K) using compact_spectrum by auto
    from assms(2) have A-U⊆(∪T∪(∪T))-U using op_compact_total by auto
    with K(3) have A-U⊆K by auto
    with finK have Finite(A-U) using subset_Finite by auto
    then have (A-U){is in the spectrum of}(λT. (∪T){is compact in)T) using auto
    unfolding RestrictedTo_def using assms(2) K(3) op_compact_total by auto
    moreover have (((one-point compactification of)T){restricted to}(A-U)){is a topology) using topology0.Top_1_L4
    op_comp_is_top unfolding topology0_def by auto
    ultimately have (A-U){is compact in}(((one-point compactification of)T){restricted to};(A-U))
    unfolding Spec_def by auto
    then have (A-U){is compact in}(((one-point compactification of)T)
    using compact_subspace_imp_compact by auto
    moreover have A-U⊆∪M using M(2) by auto
    moreover note M(1) ultimately obtain N where N:N∈FinPow(M) A-U⊆∪N unfolding IsCompact_def by blast
    from N(1) U(2) have N ∪{U}∈FinPow(M) unfolding FinPow_def by auto
    moreover from N(2) have A⊆∪(N ∪{U}) by auto
    ultimately have ∃R∈FinPow(M). A⊆∪R by auto
  }
  then show thesis using op_compact_total assms(2) unfolding IsCompact_def by auto

1266
As a last result in this section, the one-point compactification of our topology is not a KC space.

**Theorem** extension_pow_OP_not_KC:

\[ \neg ((\text{one-point compactification of } (\text{Pow}(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable csucc}(\text{nat})) - \{0\})) \{\text{is KC}\}) \]

**Proof**

- **Assumption** have noE:csucc(nat) ≠ 0 using Ord_0_lt_csucc[OF Ord_nat] by auto
- **Let** T = (\text{Pow}(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable csucc}(\text{nat})) - \{0\})
- **Assumption** assume ass: (\text{one-point compactification of} T) \{\text{is KC}\}
- **From** extension_pow_notDiscrete have \{\text{csucc}(\text{nat})\} /∈ (\text{Pow}(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable csucc}(\text{nat})) - \{0\})
- **By** auto

\[
\{\\text{assume csucc(nat)} = \text{csucc(nat)} \cup \{\text{csucc(nat)}\} \therefore \text{csucc(nat)} \in (\text{Pow}(\text{csucc}(\text{nat}) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable csucc}(\text{nat})) - \{0\}))
\]

**Then** have \text{dist:csucc(nat)} ≠ \text{csucc(nat)} \cup \{\text{csucc(nat)}\} by blast

- **Assume** \text{csucc(nat)} ∈ (\text{Pow}(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable csucc}(\text{nat})) - \{0\})
- **Then** have \text{dist:csucc(nat)} = \text{csucc(nat)} \cup \{\text{csucc(nat)}\} by auto
- **Ultimately** have \text{csucc(nat)} ∈ csucc(nat) by auto
- **Then** have False using mem_not_refl by auto

**Then** have \text{dist:csucc(nat)} ≠ \text{csucc(nat)} \cup \{\text{csucc(nat)}\} by blast

- **Assume** \text{csucc(nat)} ∈ (\text{Pow}(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable csucc}(\text{nat})) - \{0\})
- **Then** have \text{dist:csucc(nat)} = \text{csucc(nat)} \cup \{\text{csucc(nat)}\} by auto
- **Ultimately** have \text{dist:csucc(nat)} = \text{csucc(nat)} by auto
- **Then** have False using mem_not_refl by auto

**Then** have \text{dist:csucc(nat)} ≠ \text{csucc(nat)} \cup \{\text{csucc(nat)}\} by blast

- **Assume** \text{csucc(nat)} ∈ (\text{Pow}(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable csucc}(\text{nat})) - \{0\})
- **Then** have \text{dist:csucc(nat)} = \text{csucc(nat)} \cup \{\text{csucc(nat)}\} by auto
- **Ultimately** have \text{dist:csucc(nat)} = \text{csucc(nat)} by auto
- **Then** have False using mem_not_refl by auto
then have n:\neg((\bigcup\{\text{one-point compactification of }T\}\setminus\{\text{csucc(nat)}\})\text{is closed in}\{\text{one-point compactification of }T\})\text{ by auto moreover from dist have }\bigcup T\subseteq(\bigcup\{\text{one-point compactification of }T\}\setminus\{\text{csucc(nat)}\})\text{using topology0.op_compact_total unfolding topology0_def using extension_pow_top extension_pow_union by auto then have }((\bigcup\{\text{one-point compactification of }T\}\setminus\{\text{csucc(nat)}\})\text{is compact in}\{\text{one-point compactification of }T\})\text{ using topology0.anti_comp_imp_OP_inf_comp[of T}}

(\bigcup\{\text{one-point compactification of }T\}\setminus\{\text{csucc(nat)}\})\text{ unfolding topology0_def using extension_pow_antiCompact extension_pow_top by auto with ass have }((\bigcup\{\text{one-point compactification of }T\}\setminus\{\text{csucc(nat)}\})\text{is closed in}\{\text{one-point compactification of }T\})\text{ unfolding IsKC_def by auto with n show False by auto qed}

In conclusion, US \not\Rightarrow KC.

### 84.8 Other types of properties

In this section we will define new properties that aren’t defined as anti-properties and that are not separation axioms. In some cases we will consider their anti-properties.

### 84.9 Definitions

A space is called perfect if it has no isolated points. This definition may vary in the literature to similar, but not equivalent definitions.

**definition**

IsPerf (_ {is perfect}) where

\(T\{\text{is perfect}\} \equiv \forall x \in \bigcup T. \{x\} \not\subseteq T\)

An anti-perfect space is called scattered.

**definition**

IsScatt (_ {is scattered}) where

\(T\{\text{is scattered}\} \equiv T\{\text{is anti-}\}IsPerf\)

A topological space with two disjoint dense subspaces is called resolvable.

**definition**

IsRes (_ {is resolvable}) where

\(T\{\text{is resolvable}\} \equiv \exists U \in \text{Pow}(\bigcup T). \exists V \in \text{Pow}(\bigcup T). \text{Closure}(U,T)=\bigcup T \land \text{Closure}(V,T)=\bigcup T \land U \cap V = 0\)

A topological space where every dense subset is open is called submaximal.

**definition**

IsSubMax (_ {is submaximal}) where

\(T\{\text{is submaximal}\} \equiv \forall U \in \text{Pow}(\bigcup T). \text{Closure}(U,T)=\bigcup T \rightarrow U \subseteq T\)
A subset of a topological space is nowhere-dense if the interior of its closure is empty.

definition IsNowhereDense (_ {is nowhere dense in} _) where
A {is nowhere dense in} T ≡ A ⊆ T ∧ Interior(Closure(A,T),T)=0

A topological space is then a Luzin space if every nowhere-dense subset is countable.

definition IsLuzin (_ {is luzin}) where
T {is luzin} ≡ ∀ A ∈ Pow(∪ T). (A {is nowhere dense in} T) → A ≲ nat

An also useful property is local-connexion.

definition IsLocConn (_ {is locally-connected}) where
T {is locally-connected} ≡ T {is locally}(λ T. λ B. ((T {restricted to} B) {is connected}))

An SI-space is an anti-resolvable perfect space.

definition IsAntiRes (_ {is anti-resolvable}) where
T {is anti-resolvable} ≡ T {is anti-} IsRes

definition IsSI (_ {is Strongly Irresolvable}) where
T {is Strongly Irresolvable} ≡ (T {is anti-resolvable}) ∧ (T {is perfect})

84.10 First examples

Firstly, we need to compute the spectrum of the being perfect.

lemma spectrum_perfect:
shows (A {is in the spectrum of} IsPerf) ←→ A=0
proof
assume A {is in the spectrum of} IsPerf
then have Pow(A) {is perfect} unfolding Spec_def using Pow_is_top by auto
then have ∀ b ∈ A. {b} ∉ Pow(A) unfolding IsPerf_def by auto
then show A=0 by auto
next
assume A: A=0
{ fix T assume T: T {is a topology} ∪ T=A
with T(2) A have ∪ T=0 by auto
then have ∪ T=0 using eqpoll_0_is_0 by auto
then have T {is perfect} unfolding IsPerf_def by auto
}
then show A {is in the spectrum of} IsPerf unfolding Spec_def by auto
The discrete space is clearly scattered:

lemma pow_is_scattered:
shows $\text{Pow}(A)$ (is scattered)
proof-
{  
  fix $B$ assume $B:B \subseteq \bigcup \text{Pow}(A)$ (Pow(A){restricted to}B){is perfect}  
  from $B(1)$ have $\text{Pow}(A)$ (restricted to)$B=\text{Pow}(B)$ unfolding RestrictedTo_def  
  by blast  
  with $B(2)$ have $\text{Pow}(B)$ (is perfect) by auto  
  then have $\forall b \in B. \ (b) \not\in \text{Pow}(B)$ unfolding IsPerf_def by auto  
  then have $B=\emptyset$ by auto  
}
then show thesis using spectrum_perfect unfolding IsScatt_def antiProperty_def by auto
qed

The trivial topology is perfect, if it is defined over a set with more than one point.

lemma trivial_is_perfect:  
assumes $\exists x \ y. \ x \in X \wedge y \in X \wedge x \neq y$  
shows $\{0,X\}$ (is perfect)
proof-
{  
  fix $r$ assume $\{r\} \in \{0,X\}$  
  then have $X=\{r\}$ by auto  
  with asms have False by auto  
}
then show thesis unfolding IsPerf_def by auto
qed

The trivial topology is resolvable, if it is defined over a set with more than one point.

lemma trivial_is_resolvable:  
assumes $\exists x \ y. \ x \in X \wedge y \in X \wedge x \neq y$  
shows $\{0,X\}$ (is resolvable)
proof-
from asms obtain $x \ y$ where $xy:x \in X \ y \in X \ x \neq y$ by auto 
{  
  fix $A$ assume $A:A$(is closed in)$\{0,X\} \ A \subseteq X$  
  then have $X-A \in \{0,X\}$ unfolding IsClosed_def by auto  
  then have $X-A=0 \vee X-A=X$ by auto  
  with $A(2)$ have $A=X \vee X-A=X$ by auto moreover  
  {  
    assume $X-A=X$  
    then have $X-(X-A)=0$ by auto  
    with $A(2)$ have $A=0$ by auto  
  }  
}

1270
ultimately have \( A=\emptyset \vee A=X \) by auto
then have \( A=\emptyset \vee A=X \) by auto

then have \( cl: \forall A \in \text{Pow}(X). A \{\text{is closed in}\} \{0,X\} \rightarrow A=\emptyset \vee A=X \) by auto
from xy(3) have \( \{x\} \cap \{y\} = \emptyset \) by auto moreover

\{ have \( \{X\}\{\text{is a partition of}\}X \) using indiscrete_partition xy(1) by auto
then have \( \text{top:topology0(PTopology X \{X\})} \) using topology0_ptopology by auto
have \( X \neq 0 \) using xy(1) by auto
then have \( \langle \text{PTopology X \{X\}}=\{0,X\} \rangle \) using indiscrete_ptopology[of X] by auto

with top have \( \text{top0:topology0(\{0,X\})} \) by auto
then have \( x \in \text{Closure}(\{x\},\{0,X\}) \) using topology0.clContainsSet xy(1) by auto

by auto moreover
have \( \text{Closure}(\{x\},\{0,X\}) \{\text{is a partition of}\} \{0,X\} \) using topology0.clIsClosed top0 xy(1) by auto
moreover note cl
moreover have \( \text{Closure}(\{x\},\{0,X\}) \subseteq X \) using topology0.Top3L11(1)
top0 xy(1) by auto
ultimately have \( \text{Closure}(\{x\},\{0,X\})=X \) by auto

moreover
\{ have \( \{X\}\{\text{is a partition of}\}X \) using indiscrete_partition xy(1) by auto
then have \( \text{top:topology0(PTopology X \{X\})} \) using topology0_ptopology by auto
have \( X \neq 0 \) using xy(1) by auto
then have \( \langle \text{PTopology X \{X\}}=\{0,X\} \rangle \) using indiscrete_ptopology[of X] by auto

with top have \( \text{top0:topology0(\{0,X\})} \) by auto
then have \( y \in \text{Closure}(\{y\},\{0,X\}) \) using topology0.clContainsSet xy(2) by auto

by auto moreover
have \( \text{Closure}(\{y\},\{0,X\}) \{\text{is a partition of}\} \{0,X\} \) using topology0.clIsClosed top0 xy(2) by auto
moreover note cl
moreover have \( \text{Closure}(\{y\},\{0,X\}) \subseteq X \) using topology0.Top3L11(1)
top0 xy(2) by auto
ultimately have \( \text{Closure}(\{y\},\{0,X\})=X \) by auto

\}
ultimately show \( \text{thesis} \) using xy(1,2) unfolding IsRes_def by auto
qed

The spectrum of Luzin spaces is the class of countable sets, so there are lots of examples of Luzin spaces.

lemma spectrum_Luzin:
shows \((A \in \text{the spectrum of} \text{IsLuzin}) \iff A \subseteq \mathbb{N}\)

proof

assume \(A : A \in \text{the spectrum of} \text{IsLuzin}\)

\{ assume \(A = 0\)
then have \(A \subseteq \mathbb{N}\) using empty_lepollI by auto \}

moreover \{ assume \(A \neq 0\)
then obtain \(x\) where \(x \in A\) by auto
\}

moreover \{ fix \(M\) assume \(M \subseteq \{0, \{x\}, A\}\)
then have \(\bigcup M \in \{0, \{x\}, A\}\) using \(x\) by blast
\}

moreover \{ fix \(U, V\) assume \(U \subseteq \{0, \{x\}, A\}\) \(V \subseteq \{0, \{x\}, A\}\)
then have \(U \cap V \in \{0, \{x\}, A\}\) by auto
\}

ultimately have \(\text{top} : \{0, \{x\}, A\} \in \text{a topology}\) unfolding IsATopology_def by auto

moreover have \(\text{tot} : \bigcup \{0, \{x\}, A\} = A\) using \(x\) by auto

moreover note \(A\) ultimately have \(\text{luz} : \{0, \{x\}, A\} \in \text{luzin}\) unfolding Spec_def by auto

moreover have \(\{x\} \in \{0, \{x\}, A\}\) by auto
then have \(\bigcup \{0, \{x\}, A\} \in \text{closed in} \{0, \{x\}, A\}\) using topology0.Top_3_L9 unfolding topology0_def using \(x\) by blast
then have \((A - \{x\}) \in \text{closed in} \{0, \{x\}, A\}\) using \(x\) by auto
then have \(\text{Closure}(A - \{x\}, \{0, \{x\}, A\}) = A - \{x\}\) using \(x\) tot top topology0.Top_3_L8[of \(\{0, \{x\}, A\}\)] unfolding topology0_def by auto
then have \(B : \text{Interior}(\text{Closure}(A - \{x\}, \{0, \{x\}, A\}), \{0, \{x\}, A\}) = \text{Interior}(A - \{x\}, \{0, \{x\}, A\})\) by auto

then have \(C : \text{Interior}(\text{Closure}(A - \{x\}, \{0, \{x\}, A\}), \{0, \{x\}, A\}) \subseteq A - \{x\}\) using topology0.Top_2_L1 unfolding topology0_def by auto
then have \(D : \text{Interior}(\text{Closure}(A - \{x\}, \{0, \{x\}, A\}), \{0, \{x\}, A\}) \subseteq \{0, \{x\}, A\}\) using topology0.Top_2_L2 unfolding topology0_def using \(x\) by auto

from \(x\) have \(\neg (A \subseteq A - \{x\})\) by auto
with \(C, D\) have \(\text{Interior}(\text{Closure}(A - \{x\}, \{0, \{x\}, A\}), \{0, \{x\}, A\}) = 0\) by auto
then have \((A - \{x\}) \in \text{nowhere dense in} \{0, \{x\}, A\}\) unfolding IsNowhereDense_def using \(x\) by auto

with \(\text{luz}\) have \(A - \{x\} \subseteq \mathbb{N}\) unfolding IsLuzin_def using \(x\) by auto
then have \(U_1 : A - \{x\} < \text{csucc}(\mathbb{N})\) using Card_less_csucc_eq_le[OF Card_nat] by auto

have \(\{x\} \approx 1\) using singleton_eqpoll_1 by auto
then have \( \{x\} \triangleq \text{nat} \) using \( \text{n_lesspoll_nat eq_lesspoll_trans} \) by auto
then have \( \{x\} \preceq \text{nat} \) using \( \text{lesspoll_imp_lepoll} \) by auto
then have \( U_2 : \{x\} \triangleq \text{csucc(nat)} \) using \( \text{Card_less_csucc_eq_le[OF Card_nat]} \) by auto
with \( U_1 \) have \( U : (A - \{x\}) \cup \{x\} \triangleq \text{csucc(nat)} \) using \( \text{less_less_imp_un_less[OF _ _ InfCard_csucc[OF InfCard_nat]]} \) by auto
ultimately show \( A \preceq \text{nat} \) by auto

next
assume \( A : A \preceq \text{nat} \)
\{ fix \( T \) assume \( T : T \{\text{is a topology}\} \cup T \approx A \)
\{ fix \( B \) assume \( B \subseteq T \{\text{is nowhere dense in} T\} \)
then have \( B \subseteq T \{\text{using subset_imp_lepoll by auto}\} \)
with \( T(2) \) have \( B \preceq A \{\text{using lepoll_trans by blast}\} \)
\}
then have \( \forall B \in \text{Pow}(\cup T). (B \{\text{is nowhere dense in} T\} \rightarrow B \preceq \text{nat} \) by auto
then show \( A \{\text{is in the spectrum of}\} \text{IsLuzin unfolding Spec_def by auto}\)
qed

84.11 Structural results

Every resolvable space is also perfect.

theorem (in topology0) resolvable_imp_perfect:
  assumes \( T \{\text{is resolvable}\} \)
  shows \( T \{\text{is perfect}\} \)
proof-
\{ assume \( \neg (T \{\text{is perfect}\}) \)
then obtain \( x \) where \( x : x \in \cup T \{x\} \in T \) unfolding \( \text{IsPerf_def}\) by auto
then have \( c_1 : (\cup T - \{x\}) \{\text{is closed in} T\} \) using \( \text{Top_3_L9} \) by auto
from asms obtain \( U V \) where \( UV : U \subseteq \cup T \subseteq \cup T \ c_1(U) = \cup T \ c_1(V) = \cup T \ U \cap V = 0 \)
unfolding \( \text{IsRes_def}\) by auto
\{ fix \( W \) assume \( x \not\in W \ W \subseteq \cup T \)
then have \( W \subseteq \cup T - \{x\} \) by auto
then have \( c_1(W) \subseteq \cup T - \{x\} \) using \( c_1 \text{ Top_3_L13} \) by auto
with \( x(1) \) have \( \neg (\cup T \subseteq c_1(W)) \) by auto
then have \( \neg (c_1(W) = \cup T) \) by auto
\}

1273
with UV have False by auto
}
then show thesis by auto
qed

The spectrum of being resolvable follows:

\[ \text{corollary spectrum_resolvable:}
\]
\[ \text{shows } (A \text{is in the spectrum of} \text{IsRes}) \iff A=0
\]

\[ \text{proof}
\]
\[ \text{assume } A:A \text{is in the spectrum of} \text{IsRes}
\]
\[ \text{have } \forall T. T \text{is a topology} \implies \text{IsRes}(T) \implies \text{IsPerf}(T) \text{ using topology0.resolvable_imp_perfect}
\]
\[ \text{unfolding topology0_def by auto}
\]
\[ \text{with } A \text{ have } A \text{is in the spectrum of} \text{IsPerf} \text{ using P_imp_Q_spec_inv[of IsRes IsPerf] by auto}
\]
\[ \text{then show } A=0 \text{ using spectrum_perfect by auto}
\]
next
\[ \text{assume } A:A=0
\]
\[ \{\]
\[ \text{fix } T \text{ assume } T:T \text{is a topology} \cup T \approx A
\]
\[ \text{with } T(2) \text{ A have } \cup T \approx 0 \text{ by auto}
\]
\[ \text{then have } \cup T=0 \text{ using eqpoll_0_is_0 by auto}
\]
\[ \text{then have Closure}(0,T)=\cup T \text{ using topology0.Top_3_L2 T(1)
}\]
\[ \text{topology0.Top_3_L8 unfolding topology0_def by auto}
\]
\[ \text{then have } T \text{is resolvable} \text{ unfolding IsRes_def by auto}
\]
\[ \text{then show } A \text{is in the spectrum of} \text{IsRes unfolding Spec_def by auto}
\]
qed

The cofinite space over \( \mathbb{N} \) is a \( T_1 \), perfect and luzin space.

\[ \text{theorem cofinite_nat_perfect:}
\]
\[ \text{shows } (\text{CoFinite } \mathbb{N}) \text{is perfect}
\]
\[ \text{proof-}
\]
\[ \{\]
\[ \text{fix } x \text{ assume } x:x \in \cup (\text{CoFinite } \mathbb{N}) \{x\} \in (\text{CoFinite } \mathbb{N})
\]
\[ \text{then have } xn:x \in \mathbb{N} \text{ using union_cocardinal unfolding Cofinite_def}
\]
by auto
\[ \text{with } x(2) \text{ have } \text{nat} \prec \text{nat unfolding Cofinite_def CoCardinal_def}
\]
by auto
\[ \text{moreover have } \text{Finite}(\{x\}) \text{ by auto}
\]
\[ \text{then have } \{x\} \prec \text{nat unfolding Finite_def using n_lesspoll_nat eq_lesspoll_trans}
\]
by auto
\[ \text{ultimately have } (\text{nat} \prec \{x\}) \cup \{x\} \prec \text{nat using less_less_imp_un_less[0F
}\]
- - \text{InfCard_natu} \text{ by auto}
\[ \text{moreover have } (\text{nat} \prec \{x\}) \cup \{x\} = \text{nat using xn by auto
}\]
\[ \text{ultimately have } \text{False by auto}
\]
\[ \}
\[ \text{then show thesis unfolding IsPerf_def by auto}
\]
qed
Theorem cofinite_nat_luzin:
  Shows (CoFinite nat) {is luzin}
Proof:
  Have nat {is in the spectrum of} IsLuzin using spectrum_Luzin by auto
  Moreover have ∪ (CoFinite nat) = nat using union_cocardinal unfolding Cofinite_def
  By auto
  Ultimately show thesis unfolding Spec_def by auto
Qed

The cocountable topology on \( \mathbb{N}^+ \) or csucc(nat) is also T_1, perfect and luzin;
but defined on a set not in the spectrum.

Theorem cocountable_csucc_nat_perfect:
  Shows (CoCountable csucc(nat)) {is perfect}
Proof:
  Have noE: csucc(nat) ≠ 0 using lt_csucc[OF Ord_nat] by auto
  \{ Fix x assume x: x ∈ ∪ (CoCountable csucc(nat)) \{x\} ∈ (CoCountable csucc(nat))
  Then have x(x) ∈ csucc(nat) using union_cocardinal noE unfolding Cocountable_def
  By auto
  With x(2) have csucc(nat) - {x} ⊂ csucc(nat) unfolding Cocountable_def
  CoCardinal_def by auto
  Moreover have Finite({x}) by auto
  Then have \{x\} ⊂ nat unfolding Finite_def using n_lesspoll_nat eq_lesspoll_trans
  By auto
  Then have \{x\} ⊂ nat using lesspoll_imp_lepoll by auto
  Then have \{x\} ⊂ csucc(nat) using Card_less_csucc_eq_le[OF Card_nat]
  By auto
  Ultimately have (csucc(nat) - \{x\}) ∪ \{x\} ⊂ csucc(nat) using less_less_imp_un_less[OF
  _ _ InfCard_csucc[OF InfCard_nat]] by auto
  Moreover have (csucc(nat) - \{x\}) ∪ \{x\} = csucc(nat) using xn by auto
  Ultimately have False by auto
\}
Then show thesis unfolding IsPerf_def by auto
Qed

Theorem cocountable_csucc_nat_luzin:
  Shows (CoCountable csucc(nat)) {is luzin}
Proof:
  Have noE: csucc(nat) ≠ 0 using lt_csucc[OF Ord_nat] by auto
  \{ Fix B assume B: B Pow (∪ (CoCountable csucc(nat))) B {is nowhere dense
  in}(CoCountable csucc(nat)) \(¬ (B ⊆ \text{nat})\)
  From B(1) have B ⊂ csucc(nat) using union_cocardinal noE unfolding
  Cocountable_def by auto moreover
The existence of $T_2$, uncountable, perfect and Luzin spaces is unprovable in ZFC. It is related to the CH and Martin’s axiom.

85 Real valued metric spaces

theory MetricSpace_ZF_1 imports Real_ZF_2 begin

The development of metric spaces in IsarMathLib is different from the usual treatment of the subject because the notion of a metric (or a pseudometric) is defined in the MetricSpace_ZF theory a more generally as a function valued in an ordered loop. This theory file brings the subject closer to the standard way by specializing that general definition to the usual special case where the value of the metric are nonnegative real numbers.

85.1 Context and notation

The reals context (locale) defined in the Real_ZF_2 theory fixes a model of reals (i.e. a complete ordered field) and defines notation for things like zero, one, the set of positive numbers, absolute value etc. For metric spaces we reuse the notation defined there.

The pmetric_space1 locale extends the reals locale, adding the carrier $X$ of the metric space and the metric $d$ to the context, together with the assumption that $d : X \times X \rightarrow \mathbb{R}$ is a pseudo metric. An alternative would be to define the pmetric_space1 as an extension of the pmetric_space1 context,
but that is in turn an extension of the \texttt{loop1} locale that defines notation for left and right division which which do not want in the context of real numbers.

```
locale pmetric_space1 = reals +
  fixes X and d
  assumes pmetricAssum: IsApseudoMetric(d,X,R,Add,R0Ord)
  fixes ball
  defines ball_def [simp]: ball(c,r) ≡ Disk(X,d,R0Ord,c,r)
```

The propositions proven in the \texttt{pmetric_space} context defined in \texttt{Metric_Space_ZF} theory are valid in the \texttt{pmetric_space1} context.

```
lemma (in pmetric_space1) pmetric_space_pmetric_space1_valid:
  shows pmetric_space(\langle d,X,R,Add,R0Ord,d,X \rangle)
  unfolding pmetric_space_def pmetric_space_axioms_def loop1_def
  using pmetricAssum reals_loop by simp
```

It is convenient to have the collection of all open balls in given (p)metrics defined as a separate notion.

```
definition (in pmetric_space1) Open_Balls
  where Open_Balls ≡ \bigcup_{c \in X} \{ball(c,r) . r \in R_+}\}
```

```
Topology on a metric space is defined as the collection of sets that are unions of open balls of the (p)metric.
```

```
definition (in pmetric_space1) Metric_Topology
  where Metric_Topology ≡ \{\bigcup A . A \in Pow(Open_Balls)\}
```

The \texttt{metric_space1} locale (context) specializes the \texttt{pmetric_space1} context by adding the assumption of identity of indiscernibles.

```
locale metric_space1 = pmetric_space1 +
  assumes ident_indisc: \forall x \in X. \forall y \in Y. d(\langle x, y \rangle) = 0 \rightarrow x=y
```

The propositions proven in the \texttt{metric_space} context defined in \texttt{Metric_Space_ZF} theory are valid in the \texttt{metric_space1} context.

```
lemma (in metric_space1) metric_space_metric_space1_valid:
  shows metric_space(\langle R,Add,R0Ord,d,X \rangle)
  unfolding metric_space_def metric_space_axioms_def
  using pmetric_space_pmetric_space1_valid ident_indisc
  by simp
```

85.2 Metric spaces are Hausdorff as topological spaces

The usual (real-valued) metric spaces are a special case of ordered loop valued metric spaces defined in the \texttt{MetricSpace_ZF} theory, hence they are \textit{T}_2 as topological spaces.

Since in the \texttt{pmetric_space1} context \(\mathcal{D}\) is a pseudometrics the (p)metric topology as defined above is indeed a topology, the set of open balls is the base
of that topology and the carrier of the topology is the underlying (p)metric space carrier \( X \).

**Theorem** \((\text{in } \text{pmetric_space1})\) \( \text{rpmetric_is_top} \):

\[
\text{shows} \quad \text{Metric\_Topology \{is a topology\}} \\
\text{Open\_Balls \{is a base for\} Metric\_Topology} \\
\bigcup \text{Metric\_Topology} = X
\]

*unfolding* \( \text{Open\_Balls\_def Metric\_Topology\_def} \)

*using* \( \text{rord\_down\_directs pmetric\_space_pmetric\_space1\_valid} \)

\( \text{pmetric\_space.pmetric_is_top by simp} \)

The topology generated by a metric is Hausdorff (i.e. \( T_2 \)).

**Theorem** \((\text{in } \text{metric\_space1})\) \( \text{rmetric\_space_T2} \):

\[
\text{shows} \quad \text{Metric\_Topology \{is } T_2 \text{\}}
\]

*unfolding* \( \text{Open\_Balls\_def Metric\_Topology\_def} \)

*using* \( \text{rord\_down\_directs metric\_space_metric\_space1\_valid} \)

\( \text{metric\_space.metric\_space_T2 by simp} \)

end

86 Uniform spaces

**Theory** UniformSpace_ZF \( \text{imports Topology\_ZF\_2 Topology\_ZF\_4a} \)

begin

This theory defines uniform spaces and proves their basic properties.

86.1 Definition and motivation

Just like a topological space constitutes the minimal setting in which one can speak of continuous functions, the notion of uniform spaces (commonly attributed to André Weil) captures the minimal setting in which one can speak of uniformly continuous functions. In some sense this is a generalization of the notion of metric (or metrizable) spaces and topological groups.

There are several definitions of uniform spaces. The fact that these definitions are equivalent is far from obvious (some people call such phenomenon cryptomorphism). We will use the definition of the uniform structure (or "uniformity") based on entourages. This was the original definition by Weil and it seems to be the most commonly used. A uniformity consists of entourages that are binary relations between points of space \( X \) that satisfy a certain collection of conditions, specified below.

**Definition**

\( \text{IsUniformity \{is a uniformity on\} \_ 90} \)

\[
\Phi \{\text{is a uniformity on} \ X \equiv (\Phi \{\text{is a filter on} \ (X\times X)) \\
\land (\forall U \in \Phi. \ id(X) \subseteq U \land (\exists V \in \Phi. \ V \circ V \subseteq U) \land \text{converse}(U) \in \Phi}\)
\]
If $\Phi$ is a uniformity on $X$, then the every element $V$ of $\Phi$ is a certain relation on $X$ (a subset of $X \times X$) and is called an "entourage". For an $x \in X$ we call $V\{x\}$ a neighborhood of $x$. The first useful fact we will show is that neighborhoods are non-empty.

lemma neigh_not_empty:
  assumes $\Phi$ {is a uniformity on} $X$ $W \in \Phi$ and $x \in X$
  shows $W\{x\} \neq 0$ and $x \in W\{x\}$

proof -
  from assms(1,2) have $id(X) \subseteq W$
    unfolding IsUniformity_def IsFilter_def by auto 
    with $x \in X$ show $x \in W\{x\}$ and $W\{x\} \neq 0$ by auto
  qed

The filter part of the definition of uniformity for easier reference:

lemma unif_filter: assumes $\Phi$ {is a uniformity on} $X$
  shows $\Phi$ {is a filter on} $(X \times X)$
  using assms unfolding IsUniformity_def by simp

The second part of the definition of uniformity for easy reference:

lemma entourage_props:
  assumes $\Phi$ {is a uniformity on} $X$ and $A \in \Phi$
  shows
    $A \subseteq X \times X$
    $id(X) \subseteq A$
    $\exists V \in \Phi. \ V \circ V \subseteq A$
    $converse(A) \in \Phi$

proof -
  from assms show $id(X) \subseteq A$ $\exists V \in \Phi. \ V \circ V \subseteq A$ converse(A) $\in \Phi$
    unfolding IsUniformity_def by auto 
  from assms show $A \subseteq X \times X$
    using unif_filter unfolding IsFilter_def by blast
  qed

The definition of uniformity states (among other things) that for every member $U$ of uniformity $\Phi$ there is another one, say $V$ such that $V \circ V \subseteq U$. Sometimes such $V$ is said to be half the size of $U$. The next lemma states that $V$ can be taken to be symmetric.

lemma half_size_symm: assumes $\Phi$ {is a uniformity on} $X$ $W \in \Phi$
  shows $\exists V \in \Phi. \ V \circ V \subseteq W$ $\land$ $V = converse(V)$

proof -
  from assms obtain $U$ where $U \in \Phi$ and $U \circ U \subseteq W$
  unfolding IsUniformity_def by auto
  let $V = U \cap converse(U)$
  from assms(1) $\langle U \in \Phi \rangle$ have $V \in \Phi$ and $V = converse(V)$
    unfolding IsUniformity_def IsFilter_def by auto 
  moreover from $\langle U \circ U \subseteq W \rangle$ have $V \circ V \subseteq W$ by auto
  ultimately show thesis by blast
Inside every member $W$ of the uniformity $\Phi$ we can find one that is symmetric and smaller than a third of size $W$. Compare with the Metamath’s theorem with the same name.

**Lemma ustex3sym:** assumes $\Phi \{\text{is a uniformity on}\} X A \in \Phi$

shows $\exists B \in \Phi. B \circ (B \circ B) \subseteq A \land B=\text{converse}(B)$

**Proof** -
- from assm obtain $C$ where $C \subseteq \Phi$ and $C \circ C \subseteq A$
  - unfolding $\text{IsUniformity_def}$ by auto
- from assm(1) $C \subseteq \Phi$ obtain $B$ where $B \in \Phi. B \circ B \subseteq C \land B=\text{converse}(B)$
  - using $\text{half_size_symm}$ by blast
- with assm(1) $C \subseteq \Phi$ have $(B \circ B) \circ (B \circ B) \subseteq A$ by blast
- with assm(1) $B \subseteq \Phi$ have $B \circ (B \circ B) \subseteq A$
  - using $\text{entourage_props(1,2)}$ by blast
- with $B \subseteq \Phi$ $\text{B=converse(B)}$ show thesis by blast

qed

If $\Phi$ is a uniformity on $X$ then every element of $\Phi$ is a subset of $X \times X$ whose domain is $X$.

**Lemma uni_domain:**
- assumes $\Phi \{\text{is a uniformity on}\} X W \in \Phi$
- shows $W \subseteq X \times X \land \text{domain}(W) = X$

**Proof** -
- from assm show $W \subseteq X \times X$ unfolding $\text{IsUniformity_def}$ $\text{IsFilter_def}$ by blast
- show $\text{domain}(W) = X$
  - proof
    - from assm show $\text{domain}(W) \subseteq X$ unfolding $\text{IsUniformity_def}$ $\text{IsFilter_def}$
      - by auto
    - from assm show $X \subseteq \text{domain}(W)$ unfolding $\text{IsUniformity_def}$ by blast
  - qed

qed

If $\Phi$ is a uniformity on $X$ and $W \in \Phi$ the for every $x \in X$ the image of the singleton $\{x\}$ by $W$ is contained in $X$. Compare the Metamath’s theorem with the same name.

**Lemma ustimasn:**
- assumes $\Phi \{\text{is a uniformity on}\} X W \in \Phi \land x \in X$
- shows $W\{x\} \subseteq X$
  - using assms uni_domain(1) by auto

Uniformity $\Phi$ defines a natural topology on its space $X$ via the neighborhood system that assigns the collection $\{V(\{x\}) : V \in \Phi\}$ to every point $x \in X$. In the next lemma we show that if we define a function this way the values of that function are what they should be. This is only a technical fact
which is useful to shorten the remaining proofs, usually treated as obvious
in standard mathematics.

**lemma neigh_filt_fun:**

assumes \( \Phi \) {is a uniformity on} \( X \)
defines \( M \equiv \{\langle x, \{V{x}.V \in \Phi\}\rangle. x \in X\}\)
shows \( M : X \to \text{Pow}(\text{Pow}(X)) \) and \( \forall x \in X. \ M(x) = \{V{x}.V \in \Phi\}\)

**proof**

- from asms have \( \forall x \in X. \ \{V{x}.V \in \Phi\} \in \text{Pow}(\text{Pow}(X)) \)
  using IsUniformity_def IsFilter_def image_subset by auto
  with asms show \( M : X \to \text{Pow}(\text{Pow}(X)) \) using ZF_fun_from_total by simp

qed

In the next lemma we show that the collection defined in lemma neigh_filt_fun
is a filter on \( X \). The proof is kind of long, but it just checks that all filter
conditions hold.

**lemma filter_from_uniformity:**

assumes \( \Phi \) {is a uniformity on} \( X \) and \( x \in X \)
defines \( M \equiv \{\langle x, \{V{x}.V \in \Phi\}\rangle. x \in X\}\)
shows \( M(x) \) {is a filter on} \( X \)

**proof**

- from asms have PhiFilter: \( \Phi \) {is a filter on} \( (X \times X) \) and
  \( M : X \to \text{Pow}(\text{Pow}(X)) \) and \( M(x) = \{V{x}.V \in \Phi\} \)
  using IsUniformity_def neigh_filt_fun by auto
  have \( 0 \not\in M(x) \)
  proof
  - from asms \( \langle x \in X \rangle \) have \( 0 \not\in \{V{x}.V \in \Phi\} \) using neigh_not_empty by blast
    with \( \langle M(x) = \{V{x}.V \in \Phi\} \rangle \) show \( 0 \not\in M(x) \) by simp
    qed
  moreover have \( X \in M(x) \)
  proof
    note \( \langle M(x) = \{V{x}.V \in \Phi\} \rangle \)
    moreover from asms have \( X \times X \in \Phi \) unfolding IsUniformity_def IsFilter_def
    by blast
    hence \( (X \times X)\{x\} \in \{V{x}.V \in \Phi\} \) by auto
    moreover from \( \langle x \in X \rangle \) have \( (X \times X)\{x\} = X \) by auto
    ultimately show \( X \in M(x) \) by simp
    qed
  moreover from \( \langle M : X \to \text{Pow}(\text{Pow}(X)) \rangle \) \( \langle x \in X \rangle \) have \( M(x) \subseteq \text{Pow}(X) \) using
  apply_funtype
  by blast
  moreover have LargerIn: \( \forall B \in M(x). \ \forall C \in \text{Pow}(X). \ B \subseteq C \rightarrow C \in M(x) \)
  proof
  { fix B assume \( B \in M(x) \)
    fix \( C \) assume \( C \in \text{Pow}(X) \) and \( B \subseteq C \)
    from \( \langle M(x) = \{V{x}.V \in \Phi\} \rangle \) \( \langle B \in M(x) \rangle \) obtain \( U \) where
  }
\[ U \subseteq \Phi \text{ and } B = U\{x\} \text{ by auto} \]

let \( V = U \cup C \times C \)

from \( \text{assms } U \in \Phi \) have \( V \in \text{Pow}(X \times X) \) and \( U \subseteq V \)

using \text{IsUniformity_def IsFilter_def} by auto

with \( U \in \Phi \) \text{PhiFilter} have \( \text{V} \in \Phi \) using \text{IsFilter_def} by simp

moreover from \( \text{assms } U \in \Phi \) \( x \in X \) \( B = U\{x\} \) \( B \subseteq C \) have \( C = V\{x\} \)

using \text{neigh_not_empty image_greater_rel} by simp

ultimately have \( C \in \{V\{x\}.V \in \Phi\} \) by auto

with \( M(x) = \{V\{x\}.V \in \Phi\} \) have \( C \in M(x) \) by simp

thus thesis by blast

qed

moreover have \( \forall A \in M(x). \forall B \in M(x). A \cap B \in M(x) \)

proof -

\{ fix \( A \) \( B \) assume \( A \in M(x) \) and \( B \in M(x) \)

with \( M(x) = \{V\{x\}.V \in \Phi\} \) obtain \( V_A \) \( V_B \) where

\( A = V_A\{x\} \) \( B = V_B\{x\} \) and \( V_A \in \Phi \) \( V_B \in \Phi \)

by auto

let \( C = V_A\{x\} \cap V_B\{x\} \)

from \( \text{assms } V_A \in \Phi \) \( V_B \in \Phi \) have \( V_A \cap V_B \in \Phi \) using \text{IsUniformity_def IsFilter_def} by simp

moreover have \( V_A \cap V_B \in \Phi \) \text{PhiFilter} have \( \text{V} \in \Phi \) using \text{IsFilter_def} by simp

moreover note \( \text{LargerIn} \)

ultimately have \( C \in M(x) \) by simp

with \( A = V_A\{x\} \) \( B = V_B\{x\} \) have \( A \cap B \in M(x) \) by blast

} thus thesis by simp

qed

A rephrasing of \text{filter_from_uniformity}: if \( \Phi \) is a uniformity on \( X \), then

\( \{V\{x\}.V \in \Phi\} \) is a filter on \( X \) for every \( x \in X \).

lemma \text{unif_filter_at_point}:

assumes \( \Phi \) \text{ is a uniformity on } X \text{ and } x \in X \)

shows \( \{V\{x\}.V \in \Phi\} \) \text{ is a filter on } X

using \( \text{assms filter_from_uniformity ZF_fun_from_tot_val1} \) by simp

A frequently used property of filters is that they are "upward closed" i.e. supersets of a filter element are also in the filter. The next lemma makes this explicit for easy reference as applied to the natural filter created from a uniformity.

corollary \text{unif_filter_up_closed}:

assumes \( \Phi \) \text{ is a uniformity on } X \text{ and } x \in X \text{ U } \in \{V\{x\}.V \in \Phi\} \text{ W } \subseteq X \text{ U } \subseteq W \)

shows \( W \in \{V\{x\}.V \in \Phi\} \)

1282
The function defined in the premises of lemma \texttt{neigh_filt_fun} (or \texttt{filter_from_uniformity}) is a neighborhood system. The proof uses the existence of the "half-the-size" neighborhood condition $(\exists V \in \Phi. V \circ V \subseteq U)$ of the uniformity definition, but not the converse $(U) \in \Phi$ part.

\textbf{theorem} \texttt{neigh_from_uniformity}:
\begin{itemize}
\item assumes $\Phi \{\text{is a uniformity on} \ X\}$
\item shows $\{\langle x, (V(x). V \in \Phi) \rangle.x \in X\} \{\text{is a neighborhood system on} \ X\}$
\end{itemize}
\textbf{proof} -
\begin{itemize}
\item from \texttt{assms \ M} \texttt{: X} $\rightarrow \texttt{Pow(Pow(X))}$ and \texttt{Mval: } $\forall x \in X. \ M(x) = \{V(x). V \in \Phi\}$
\item using \texttt{IsUniformity_def} \texttt{neigh_filt_fun} by \texttt{auto}
\item moreover from \texttt{assms} have $\forall x \in X. (M(x) \{\text{is a filter on} \ X\})$ using \texttt{filter_from_uniformity}
\item moreover
\item \{ fix $x$ assume $x \in X$
\item have $\forall N \in M(x). x \in N \land (\exists U \in M(x). \forall y \in U.(N \in M(y)))$
\item proof -
\item \{ fix $N$ assume $N \in M(x)$
\item have $x \in N \land \exists U \in M(x). \forall y \in U.(N \in M(y))$
\item proof -
\item from $\langle M: X \rightarrow \text{Pow(Pow(X))} \rangle. \text{Mval: } \langle x \in X. \langle N \in M(x) \rangle \rangle$
\item obtain $U$ where $U \in \Phi$ and $N = U \{x\}$ by \texttt{auto}
\item with \texttt{assms \ Mval} show $x \in N$ using \texttt{neigh_not_empty} by \texttt{simp}
\item from \texttt{assms \ Mval} obtain $V$ where $V \in \Phi$ and $V \circ V \subseteq U$
\item unfolding \texttt{IsUniformity_def} by \texttt{auto}
\item let $W = V \{x\}$
\item from $\langle V \in \Phi \rangle. \text{Mval: } \langle x \in X. \text{have } W \in M(x) \rangle$ by \texttt{auto}
\item moreover have $\forall y \in W. N \in M(y)$
\item proof -
\item \{ fix $y$ assume $y \in W$
\item with $\langle M: X \rightarrow \text{Pow(Pow(X))} \rangle. \langle x \in X. \langle V \in \Phi \rangle. \langle V \circ V \subseteq U \rangle. \langle x \in X. \text{have } N \in M(x) \rangle \rangle$
\item using \texttt{apply_functype} by \texttt{blast}
\item with \texttt{assms \ Mval} have $M(y) \{\text{is a filter on} \ X\}$ using \texttt{filter_from_uniformity}
\item by \texttt{simp}
\item moreover from \texttt{assms \ Mval} have $V \{y\} \in M(y)$
\item using \texttt{neigh_filt_fun} by \texttt{auto}
\item moreover from $\langle M: X \rightarrow \text{Pow(Pow(X))} \rangle. \langle x \in X. \langle N \in M(x) \rangle \rangle$
\item using \texttt{apply_functype} by \texttt{blast}
\item moreover from $\langle V \circ V \subseteq U \rangle. \langle y \in W \rangle$
\item have $V \{y\} \subseteq (V \circ V) \{x\}$ and $(V \circ V) \{x\} \subseteq U \{x\}$
\item by \texttt{auto}
\item with $\langle N = U \{x\} \rangle$ have $V \{y\} \subseteq N$ by \texttt{blast}
\item ultimately have $N \in M(y)$ using \texttt{IsFilter_def} by \texttt{simp}
\item \}
\item thus thesis by \texttt{simp}
\item qed
\end{itemize}
ultimately show $\exists U \in \mathcal{M}(x). \forall y \in U. (N \in \mathcal{M}(y))$ by auto
qed
}

thus thesis by simp

ultimately show thesis unfolding IsNeighSystem_def by simp

When we have a uniformity $\Phi$ on $X$ we can define a topology on $X$ in a (relatively) natural way. We will call that topology the UniformTopology($\Phi$). We could probably reformulate the definition to skip the $X$ parameter because if $\Phi$ is a uniformity on $X$ then $X$ can be recovered from (is determined by) $\Phi$.

definition
UniformTopology($\Phi$,X) $\equiv \{U \in \text{Pow}(X). \forall x \in U. \exists \{V\{x\}. V \in \Phi\} \}$

An identity showing how the definition of uniform topology is constructed. Here, the $M = \{\langle t, \{V\{t\}. V \in \Phi\} \rangle : t \in X\}$ is the neighborhood system (a function on $X$) created from uniformity $\Phi$. Then for each $x \in X$, $M(x) = \{V\{t\}. V \in \Phi\}$ is the set of neighborhoods of $x$.

lemma uniftop_def_alt:
shows UniformTopology($\Phi$,X) = \{U $\in \text{Pow}(X). \forall x \in U. \exists \{\langle t, \{V\{t\}. V \in \Phi\} \rangle : t \in X\}\(x)\}

proof -
let $M = \{\langle x, \{V\{x\}. V \in \Phi\} \rangle. x \in X\}$

have $\forall U \in \text{Pow}(X). \forall x \in U. M(x) = \{\langle t, \{V\{t\}. V \in \Phi\} \rangle. t \in X\}(x)$
using ZF_fun_from_tot_val1 by auto

then show thesis unfolding UniformTopology_def by auto

The collection of sets constructed in the UniformTopology definition is indeed a topology on $X$.

theorem uniform_top_is_top:
assumes $\Phi$ {is a uniformity on} $X$
shows UniformTopology($\Phi$,X) {is a topology} and $\bigcup$ UniformTopology($\Phi$,X) = $X$
using assms neigh_from_uniformity uniftop_def_alt topology_from_neighs by auto

If we have a uniformity $\Phi$ we can create a neighborhood system from it in two ways. We can create a neighborhood system directly from $\Phi$ using the formula $X \ni x \mapsto \{V\{x\}. x \in X\}$ (see theorem neigh_from_uniformity). Alternatively we can construct a topology from $\Phi$ as in theorem uniform_top_is_top and then create a neighborhood system from this topology as in theorem neigh_from_topology. The next theorem states that these two ways give the same result.

theorem neigh_unif_same: assumes $\Phi$ {is a uniformity on} $X$
Another form of the definition of topology generated from a uniformity.

**Lemma uniftop_def-alt1:**

Assumes \( \Phi \) is a uniformity on \( X \).

**shows**

\[ \text{UniformTopology}(\Phi, X) = \{ U \subseteq X : \forall x \in U. \exists W \in \Phi. W(x) \subseteq U \} \]

**Proof**

Let \( T = \text{UniformTopology}(\Phi, X) \).

**shows**

\[ T \subseteq \{ U \subseteq X : \forall x \in U. \exists W \in \Phi. W(x) \subseteq U \} \]

Unfolding \text{UniformTopology_def} by auto.

Thus \( \{ U \subseteq X : \forall x \in U. \exists W \in \Phi. W(x) \subseteq U \} \subseteq T \) by blast.

**Images of singletons by entourages are neighborhoods of those singletons.**

**Lemma image_singleton_ent_nei:**

Assumes \( \Phi \) is a uniformity on \( X \).

**shows**

\[ V(x) \in \{ \text{set neighborhood system of} \ \text{UniformTopology}(\Phi, X) \} \]

**Proof**

From \text{assms}(1,4) have \( \mathcal{M} = \{ \langle x, \{ V(x) : V \in \Phi \} \rangle : x \in X \} \) using neigh_unif_same by simp.

Hence \( \forall x \in U. U \subseteq \{ V(x) : V \in \Phi \} \) using \text{ZF_fun_from_tot_val1} by auto.

**The set neighborhoods of a singleton \( \{ x \} \) where \( x \in X \) consist of images of the singleton by the entourages \( W \in \Phi \). See also the Metamath's theorem with the same name.**

**Lemma utopsnneip:**

Assumes \( \Phi \) is a uniformity on \( X \).

**shows**

\[ \{ W(x) : x \in X, W \in \Phi \} \]
proof - 
let T = UniformTopology(Φ,X)
let M = {neighborhood system of} T
from assms(1,2) have x ∈ ∪ T
    using uniform_top_is_top(2) by simp
with assms(3) have M(x) = S(x)
    using neigh_from_nei by simp
moreover from assms(1) have M(x) = {W{x}.W ∈ Φ}
    using neigh_unif_same by simp
with assms(2) have M(x) = {W{x}.W ∈ Φ}
    using ZF_fun_from_tot_val1 by simp
ultimately show thesis by simp
qed

Images of singletons by entourages are set neighborhoods of those singletons.
See also the Metamath theorem with the same name.

**corollary utopsnnei:** assumes Φ is a uniformity on X W ∈ Φ x ∈ X
defines S ≡ {set neighborhood system of} UniformTopology(Φ,X)
shows W{x} ∈ S{x} using assms utopsnneip by auto

If Φ is a uniformity on X that generates a topology T, R is any relation on X (i.e. R ⊆ X × X), W is a symmetric entourage (i.e. W ∈ Φ, and W is symmetric (i.e. equal to its converse)), then the closure of R in the product topology is contained the the composition V ◦ (M ◦ V).

**lemma utop3cls:**
assumes Φ is a uniformity on X R ⊆ X × X W ∈ Φ W=converse(W)
defines J ≡ UniformTopology(Φ,X)
shows Closure(R,J × t J) ⊆ W O (R O W)
proof
let M = {set neighborhood system of} (J × t J)
fix z assume zMem: z ∈ Closure(R,J × t J)
from assms(1,5) have Jtop: J {is a topology} and ∪ J = X
    using uniform_top_is_top by auto
then have JJtop: (J × t J) {is a topology} and JxJ: ∪ (J × t J) = X × X
    unfolding topology0_def by auto
with assms(2) have topology0(J × t J) and R ⊆ ∪ (J × t J)
    using Top_1_4_T1(1,3) by auto
then have Closure(R,J × t J) ⊆ ∪ (J × t J)
    using topology0.Top_3_L11(1) by simp
with ‹ z ∈ Closure(R,J × t J) › JxJ have z ∈ X × X by auto
let x = fst(z)
let y = snd(z)
from ‹ z ∈ X × X › have x ∈ X y ∈ X z = (x,y) by auto
with assms(1,3,5) Jtop have (W{x}) × (W{y}) ∈ M({x} × {y})
    using utopsnnei neitx by simp
moreover from ‹ z = (x,y) › have {x} × {y} = {z}
uniform spaces are regular \((T_3)\).

\[
\text{theorem utopreg:} \quad \text{assumes } \Phi \{\text{is a uniformity on } X\} \\
\text{shows UniformTopology}(\Phi, X) \{\text{is regular}\}
\]

\[
\text{proof -} \\
\quad \text{let } J = \text{UniformTopology}(\Phi, X) \\
\quad \text{let } S = \{\text{set neighborhood system of } J\} \\
\quad \text{from assms have } \bigcup J = X \\
\quad \text{and } J_{top}: J \{\text{is a topology}\} \text{ and cntx: topology0(J)} \\
\quad \text{using uniform_top_is_top unfolding topology0_def by auto} \\
\quad \text{have } \forall U \in J. \forall x \in U. \exists V \in J. x \in V \land \text{Closure}(V, J) \subseteq U
\]

\[
\text{proof -} \\
\quad \{ \text{ fix } U \text{ x assume } U \in J \text{ x } U \\
\quad \text{then have } U \in S\{x\} \text{ using open_nei_singl by simp} \\
\quad \text{from } \langle U \in J \rangle \text{ have } U \subseteq \bigcup J \text{ by auto} \\
\quad \text{with } \langle x \in U \rangle \langle \bigcup J = X \rangle \text{ have } x \in X \text{ by auto} \\
\quad \text{from assms(1) } \langle x \in X \rangle \langle U \in S\{x\} \rangle \text{ obtain } A \\
\quad \text{where } U=A\{x\} \text{ and } A \in \Phi \\
\quad \text{using utopsneip by auto} \\
\quad \text{from assms(1) } \langle A \in \Phi \rangle \text{ obtain } W \text{ where} \\
\quad \text{W} \in \Phi \text{ W } 0 \text{ (W } 0 \text{ W) } \subseteq A \text{ and Wsymm: W=} \text{converse(W)} \\
\quad \text{using ustex3sym by blast} \\
\quad \text{with assms(1) } \langle x \in X \rangle \text{ have } W\{x\} \in S\{x\} \text{ and } W\{x\} \subseteq X \\
\quad \text{using utopsnei ustimasn by auto} \\
\quad \text{from } \langle W\{x\} \in S\{x\} \rangle \text{ have } \exists V \in J. \{x\} \subseteq V \land V \subseteq W\{x\} \\
\quad \text{by (rule nei2)} \\
\quad \text{then obtain } V \text{ where } V \in J \text{ x } V \subseteq W\{x\} \\
\quad \text{by blast} \\
\quad \text{have } \text{Closure}(V, J) \subseteq U \\
\quad \text{proof -} \\
\quad \quad \text{from assms(1) } \langle W \in \Phi \rangle \langle \bigcup J = X \rangle \text{ have } W \subseteq X \times X \\
\quad \quad \text{using entourage_props(1) by simp} \\
\quad \quad \text{from cntx } \langle W\{x\} \subseteq X \rangle \langle \bigcup J = X \rangle \langle V \subseteq W\{x\} \rangle \\
\quad \quad \text{have } \text{Closure}(V, J) \subseteq \text{Closure}(W\{x\}, J) \\
\quad \quad \text{using topology0.top_closure_mono by simp} \\
\quad \quad \text{also have } \text{Closure}(W\{x\}, J) \subseteq \text{Closure}(W, J \times J)\{x\} \\
\quad \quad \text{proof -} \\
\quad \quad \quad \text{from } \langle W \subseteq X \times X \rangle \langle x \in X \rangle \langle \bigcup J = X \rangle \\
\quad \quad \quad \text{have } W \subseteq \bigcup J \times \bigcup J \text{ by auto}
with \(<J \{\text{is a topology}\}> \text{ show thesis}
using imasncls by simp
qed
also from \text{assms(1)} <W \subseteq X \times X> \text{ Wsymm } <W \circ (W \circ W) \subseteq A>
have Closure(W,J \times J \{x\}) \subseteq A\{x\}
using utop3cls by blast
finally have Closure(V,J) \subseteq A\{x\}
by simp
with \(<U=A\{x\}> \text{ show thesis by auto}
qed
with \(<V \in J> <x \in V> \text{ have } \exists V \in J. x \in V \land Closure(V,J) \subseteq U>
by blast
} thus thesis by simp
qed
with Jtop show J \{\text{is regular}\} using is_regular_def_alt
by simp
qed

end

87 More on uniform spaces

theory UniformSpace_ZF_1 imports func_ZF_1 UniformSpace_ZF Topology_ZF_2 begin

This theory defines the maps to study in uniform spaces and proves their basic properties.

87.1 Uniformly continuous functions

Just as the the most general setting for continuity of functions is that of topological spaces, uniform spaces are the most general setting for the study of uniform continuity.

A map between 2 uniformities is uniformly continuous if it preserves the entourages:

definition
IsUniformlyCont (_ \{is uniformly continuous between\}_{and} \_ 90) where
f:X \rightarrow Y \rightarrow \Phi \{\text{is a uniformity on} \ X \rightarrow \Gamma \{\text{is a uniformity on} \ Y \rightarrow

\{\text{is uniformly continuous between}\} \Phi \{\text{and}\} \Gamma \equiv \forall V \in \Gamma. (\text{ProdFunction}(f,f)-V)
\in \Phi

Any uniformly continuous function is continuous when considering the topologies on the uniformities.

lemma uniformly_cont_is_cont:
assumes f:X \rightarrow Y \Phi \{\text{is a uniformity on} \ X \rightarrow \Gamma \{\text{is a uniformity on} \ Y \rightarrow
f {is uniformly continuous between} Φ {and} Γ
shows \text{IsContinuous}(\text{UniformTopology}(\Phi,X),\text{UniformTopology}(\Gamma,Y),f)

\text{proof} -
\{ fix U assume \text{op: } U \in \text{UniformTopology}(\Gamma,Y)
\text{have } f-(U) \in \text{UniformTopology}(\Phi,X)
\text{proof} -
\{ from \text{assms(1)} have f-(U) \subseteq X using func1_1_L3 by simp
moreover
\{ fix x xa assume \text{as: } \langle x,xa \rangle \in f xa \in U
\text{with } \text{assms(1)} have x:x \in X unfolding Pi_def by auto
\text{from as(2) } \text{op have } U:U \in \{ \langle t,\{V(t).V \in \Gamma\}\rangle.t \in Y\}(xa) using uniftop_def_alt by auto
\text{have } \{ \langle t,\{V(t).V \in \Gamma\}\rangle.t \in X\} \in \Pi(X,%t. \{\{V(t).V \in \Gamma\}\}) unfolding Pi_def function_def by simp
\text{have } \forall t. t \in (\text{ProdFunction}(f,f)-V){x} \leftrightarrow \langle x,t \rangle \in \text{ProdFunction}(f,f)-(V)
\text{using image_def by auto}
\text{with } \text{assms(1)} x \text{ have } \forall t. t: (\text{ProdFunction}(f,f)-V){x} \leftrightarrow (t \in X \wedge (f(t),\langle x,xa \rangle) \in V)
\text{using prodFunVimage by auto}
\text{with } \text{assms(1)} as(1) have \forall t. t \in (\text{ProdFunction}(f,f)-V){x} \leftrightarrow (t \in X \wedge f(t) \in U) by auto
\text{with } \text{assms(1)} U \text{ have } \forall t. t \in (\text{ProdFunction}(f,f)-V){x} \leftrightarrow t \in f-U using func1_1_L15 by simp
\text{hence } f-U = (\text{ProdFunction}(f,f)-V){x} by blast
\text{with } \text{ent have } f-(U) \subseteq \{V(x) . V \in \Phi\} by auto
moreover
\text{have } \{\langle t,\{V(t).V \in \Phi\}\rangle.t \in X\} \in \Pi(X,%t. \{\{V(t).V \in \Phi\}\}) unfolding Pi_def function_def by auto
\text{ultimately have } f-(U) \in \{\langle t, \{V \{t \} . V \in \Phi\}\rangle . t \in X\}(x) using x apply_equality by auto
\text{ultimately show } f-(U) \in \text{UniformTopology}(\Phi,X) using uniftop_def_alt by auto
qed \}
then show thesis unfolding IsContinuous_def by simp qed

1289
88 Alternative definitions of uniformity

theory UniformSpace_ZF_2 imports UniformSpace_ZF
begin

The UniformSpace_ZF theory defines uniform spaces based on entourages (also called surroundings sometimes). In this theory we consider an alternative definition based on the notion of uniform covers.

88.1 Uniform covers

Given a set $X$ we can consider collections of subsets of $X$ whose unions are equal to $X$. Any such collection is called a cover of $X$. We can define a relation on the set of covers of $X$, called "star refinement" (definition below). A collection of covers is a "family of uniform covers" if it is a filter with respect to the start refinement ordering. A member of such family is called a "uniform cover", but one has to remember that this notion has meaning only in the context of the whole family of uniform covers. Looking at a specific cover in isolation we cannot say whether it is a uniform cover or not.

The set of all covers of $X$ is called $\text{Covers}(X)$.

\begin{definition}
$\text{Covers}(X) \equiv \{ P \in \text{Pow}(\text{Pow}(X)). \bigcup P = X\}$
\end{definition}

A cover of a nonempty set must have a nonempty member.

\begin{lemma}
$\text{cover_nonempty}$: assumes $X \neq 0 P \in \text{Covers}(X)$
shows $\exists U \in P. U \neq 0$
using assms unfolding Covers_def by blast
\end{lemma}

A "star" of $R$ with respect to $\mathcal{R}$ is the union of all $S \in \mathcal{R}$ that intersect $R$.

\begin{definition}
$\text{Star}(R, \mathcal{R}) \equiv \bigcup \{ S \in \mathcal{R}. S \cap R \neq 0 \}$
\end{definition}

An element of $\mathcal{R}$ is a subset of its star with respect to $\mathcal{R}$.

\begin{lemma}
$\text{element_subset_star}$: assumes $U \in P$ shows $U \subseteq \text{Star}(U, P)$
using assms unfolding Star_def by auto
\end{lemma}

An alternative formula for star of a singleton.

\begin{lemma}
$\text{star_singleton}$: shows $(\bigcup \{ V \times V. V \in P\})\{x\} = \text{Star}\{x\}, P$
unfolding Star_def by blast
\end{lemma}

Star of a larger set is larger.
lemma star_mono: assumes U ⊆ V shows Star(U,P) ⊆ Star(V,P)
  using assms unfolding Star_def by blast

In particular, star of a set is larger than star of any singleton in that set.

corollary star_single_mono: assumes x∈U shows Star({x},P) ⊆ Star(U,P)
  using assms star_mono by auto

A cover \( R \) (of \( X \)) is said to be a "barycentric refinement" of a cover \( C \) iff for every \( x \in X \) the star of \( \{ x \} \) in \( R \) is contained in some \( C \in C \).

definition
  IsBarycentricRefinement (_ <^B _ 90)
  where
  P <^B Q ≡ ∀x∈P. ∃U∈Q. Star({x},P) ⊆ U

A cover is a barycentric refinement of the collection of stars of the singletons \( \{ x \} \) as \( x \) ranges over \( X \).

lemma singl_star_bary:
  assumes P ∈ Covers(X) shows P <^B \{Star({x},P). x∈X}\)
  using assms unfolding Covers_def IsBarycentricRefinement_def by blast

A cover \( R \) is a "star refinement" of a cover \( C \) iff for each \( R \in R \) there is a \( C \in C \) such that the star of \( R \) with respect to \( R \) is contained in \( C \).

definition
  IsStarRefinement (_ <^* _ 90)
  where
  P <^* Q ≡ ∀U∈P. ∃V∈Q. Star(U,P) ⊆ V

Every cover star-refines the trivial cover \( \{ X \} \).

lemma cover_stref_triv: assumes P ∈ Covers(X) shows P <^* \{X\}
  using assms unfolding Star_def IsStarRefinement_def Covers_def by auto

Star refinement implies barycentric refinement.

lemma star_is_bary: assumes Q∈Covers(X) and Q <^* P shows Q <^B P
  proof -
  from assms(1) have ∪Q = X unfolding Covers_def by simp
  { fix x assume x∈X
    with ∪Q = X obtain R where R∈Q and x∈R by auto
    with assms(2) obtain U where U∈P and Star(R,Q) ⊆ U
    unfolding IsStarRefinement_def by auto
    from x∈R R∈Q ⊆ U have Star({x},Q) ⊆ U
    using star_single_mono by blast
    with R∈P have ∃U∈P. Star({x},Q) ⊆ U by auto
  } with ∪Q = X show thesis unfolding IsBarycentricRefinement_def by simp
  qed

Barycentric refinement of a barycentric refinement is a star refinement.

lemma bary_bary_star:
assumes \( P \in \text{Covers}(X) \) \( Q \in \text{Covers}(X) \) \( R \in \text{Covers}(X) \) \( P \subseteq^\ast Q \) \( Q \subseteq^\ast R \) \( X \neq 0 \)
shows \( P \subseteq^\ast R \)
proof
{ fix \( U \) assume \( U \in P \)
  { assume \( U = 0 \)
    then have \( \text{Star}(U,P) = 0 \) unfolding \text{Star_def} by simp
    from assms(6,3) obtain \( V \) where \( V \in R \) using \text{cover_nonempty} by auto
    with \( \langle \text{Star}(U,P) = 0 \rangle \) have \( \exists V \in R. \text{Star}(U,P) \subseteq V \) by auto
  }
moreover
{ assume \( U \neq 0 \)
  then obtain \( x_0 \) where \( x_0 \in U \) by auto
  with assms(1,2,5) \( \langle U \in P \rangle \) obtain \( V \) where \( V \in R \) and \( \text{Star}\{x_0\},Q \) \( \subseteq V \) unfolding \text{Covers_def} \text{IsBarycentricRefinement_def} by blast
  have \( \text{Star}(U,P) \subseteq V \) proof
    { fix \( W \) assume \( W \in P \) and \( W \cap U \neq 0 \)
      from \( \langle W \cap U \neq 0 \rangle \) obtain \( x \) where \( x \in W \cap U \) by auto
      with assms(2) \( \langle U \in P \rangle \) have \( x \in \bigcup P \) by auto
      with assms(4) obtain \( C \) where \( C \in Q \) and \( \text{Star}\{x\},P \) \( \subseteq C \)
      unfolding \text{IsBarycentricRefinement_def} by blast
      with \( \langle U \in P \rangle \) \( \langle W \in P \rangle \) \( \langle x \in W \cap U \rangle \) \( \langle x_0 \in U \rangle \) \( \langle \text{Star}(x_0),Q \) \( \subseteq V \rangle \) have \( W \subseteq V \)
      unfolding \text{Star_def} by blast
    }
  then show \( \text{Star}(U,P) \subseteq V \) unfolding \text{Star_def} by auto
  qed
  with \( \langle V \in R \rangle \) have \( \exists V \in R. \text{Star}(U,P) \subseteq V \) by auto
}
ultimately have \( \exists V \in R. \text{Star}(U,P) \subseteq V \) by auto
} then show \( P \subseteq^\ast R \) unfolding \text{IsStarRefinement_def} by simp
qed

The notion of a filter defined in \texttt{Topology_ZF} is not sufficiently general to use it to define uniform covers, so we write the conditions directly. A nonempty collection \( \Theta \) of covers of \( X \) is a family of uniform covers if

a) if \( R \in \Theta \) and \( C \) is any cover of \( X \) such that \( R \) is a star refinement of \( C \), then \( C \in \Theta \).

b) For any \( C,D \in \Theta \) there is some \( R \in \Theta \) such that \( R \) is a star refinement of both \( C \) and \( R \).

This departs slightly from the definition in Wikipedia that requires that \( \Theta \) contains the trivial cover \( \{X\} \). As we show in lemma \texttt{unicov\_contains\_trivial} below we don’t loose anything by weakening the definition this way.

definition
\begin{align*}
\text{AreUniformCovers} & \equiv \{ \text{are uniform covers of} \} \text{ are uniform covers of } X \equiv \Theta \subseteq \text{Covers}(X) \land \Theta \neq 0 \land \\
( \forall R \in \Theta. \forall C \in \text{Covers}(X). ( ((R \subseteq^\ast C) \longrightarrow C \in \Theta)) \land \\
\end{align*}

1292
\( \forall C \in \Theta. \forall D \in \Theta. \exists R \in \Theta. (R \prec C) \land (R \prec D) \)

A family of uniform covers contain the trivial cover \( \{X\} \).

**Lemma unicov_contains_triv:** assumes \( \Theta \) {are uniform covers of} \( X \) shows \( \{X\} \in \Theta \)

**Proof** -
from assms obtain \( R \) where \( R \in \Theta \) unfolding AreUniformCovers_def by blast
with assms show thesis using cover_stref_triv
  unfolding AreUniformCovers_def Covers_def by auto
qed

If \( \Theta \) are uniform covers of \( X \) then we can recover \( X \) from \( \Theta \) by taking \( \bigcup \bigcup \Theta \).

**Lemma space_from_unicov:** assumes \( \Theta \) {are uniform covers of} \( X \) shows \( X = \bigcup \bigcup \Theta \)

**Proof**
from assms show \( X \subseteq \bigcup \bigcup \Theta \) using unicov_contains_triv
  unfolding AreUniformCovers_def by auto
from assms show \( \bigcup \bigcup \Theta \subseteq X \) unfolding AreUniformCovers_def Covers_def
  by auto
qed

Every uniform cover has a star refinement.

**Lemma unicov_has_star_ref:**
assumes \( \Theta \) {are uniform covers of} \( X \) and \( P \in \Theta \)
shows \( \exists Q \in \Theta. (Q \prec^* P) \)
using assms unfolding AreUniformCovers_def by blast

In particular, every uniform cover has a barycentric refinement.

**Corollary unicov_has_bar_ref:**
assumes \( \Theta \) {are uniform covers of} \( X \) and \( P \in \Theta \)
shows \( \exists Q \in \Theta. (Q \prec^B P) \)

**Proof** -
from assms obtain \( Q \) where \( Q \in \Theta \) and \( Q \prec^* P \)
  using unicov_has_star_ref by blast
with assms show thesis
  unfolding AreUniformCovers_def using star_is_bary by blast
qed

From the definition of uniform covers we know that if a uniform cover \( P \) is a star-refinement of a cover \( Q \) then \( Q \) is in a uniform cover. The next lemma shows that in order for \( Q \) to be a uniform cover it is sufficient that \( P \) is a barycentric refinement of \( Q \).

**Lemma unicov_bary_cov:**
assumes \( \Theta \) {are uniform covers of} \( X \) \( P \in \Theta \) \( Q \in \text{Covers}(X) \) \( P \prec^B Q \) and \( X \neq \emptyset \)
shows \( Q \in \Theta \)

**Proof** -
from \texttt{assms(1,2)} obtain \( R \) where \( R \in \Theta \) and \( R \leq B \) P
  using \texttt{unicov_has_bar_ref} by blast
from \texttt{assms(1,2,3)} \( \langle R \in \Theta \rangle \) have
  \( P \in \text{Covers}(X) \) Q \( \in \text{Covers}(X) \) R \( \in \text{Covers}(X) \)
  unfolding \texttt{AreUniformCovers_def} by auto
with \texttt{assms(1,3,4,5)} \( \langle R \leq B \rangle \) \( \langle R \in \Theta \rangle \) show thesis
  using \texttt{bary_bary_star} unfolding \texttt{AreUniformCovers_def} by auto
qed

A technical lemma to simplify proof of the \texttt{uniformity_from_unicov} theorem.

\textbf{lemma} \texttt{star_ref_mem}: \texttt{assumes} \( U \in P \) \( P<^* Q \) \texttt{and} \( \bigcup \{ W \times W. W \in Q \} \subseteq A \)
\texttt{shows} \( U \times U \subseteq A \)
\texttt{proof} -
  from \texttt{assms(1,2)} obtain \( W \) where \( W \in Q \) and \( \bigcup \{ S \in P. S \cap U \neq 0 \} \subseteq W \)
  unfolding \texttt{IsStarRefinement_def} \texttt{Star_def} by \texttt{auto}
with \texttt{assms(1,3)} show \( U \times U \subseteq A \) by \texttt{blast}
qed

An identity related to square (in the sense of composition) of a relation of the form \( \bigcup \{ U \times U. U \in P \} \). I am amazed that Isabelle can see that this is true without an explicit proof, I can’t.

\textbf{lemma} \texttt{rel_square_starr}: shows
  \( (\bigcup \{ U \times U. U \in P \}) O (\bigcup \{ U \times U. U \in P \}) = \bigcup \{ U \times \text{Star}(U,P). U \in P \} \)
unfolding \texttt{Star_def} by \texttt{blast}

An identity similar to \texttt{rel_square_starr} but with \texttt{Star} on the left side of the Cartesian product:

\textbf{lemma} \texttt{rel_square_starl}: shows
  \( (\bigcup \{ U \times U. U \in P \}) O (\bigcup \{ U \times U. U \in P \}) = \bigcup \{ \text{Star}(U,P) \times U. U \in P \} \)
unfolding \texttt{Star_def} by \texttt{blast}

A somewhat technical identity about the square of a symmetric relation:

\textbf{lemma} \texttt{rel_sq_image}:
  \texttt{assumes} \( W = \text{converse}(W) \) \texttt{domain}(W) \subseteq X
\texttt{shows} \( \text{Star} \{ x \}, \{ W(t). t \in X \} \) = \( (W O W \{ x \}) \)
\texttt{proof}
  have I: \( \text{Star} \{ x \}, \{ W(t). t \in X \} \) = \( \bigcup \{ S \in \{ W(t). t \in X \}. x \in S \} \)
    unfolding \texttt{Star_def} by \texttt{auto}
  \{ \texttt{fix y assume y \in Star} \{ x \}, \{ W(t). t \in X \} \}
    with I obtain \( S \) where \( y \in S \) \( x \in S \) \( S \in \{ W(t). t \in X \} \) by \texttt{auto}
  from \( \langle S \in \{ W(t). t \in X \} \rangle \) obtain \( t \) where \( t \in X \) and \( S = W(t) \)
    by \texttt{auto}
  with \( \langle x \in S \rangle \) \( \langle y \in S \rangle \) have \( \langle t, x \rangle \in W \) and \( \langle t, y \rangle \in W \)
    by \texttt{auto}
  from \( \langle t, x \rangle \in W \) have \( \langle x, t \rangle \in \text{converse}(W) \) by \texttt{auto}
  with \texttt{assms(1)} \( \langle t, y \rangle \in W \) have \( y \in (W O W \{ x \}) \)
    using \texttt{rel_compdef} by \texttt{auto}
  \} then show \( \text{Star} \{ x \}, \{ W(t). t \in X \} \) \( \subseteq (W O W \{ x \}) \)
1294
by blast
{ fix y assume y ∈ (W ∩ W) \{x\}
then obtain t where \(\langle x, t \rangle \in W\) \(\land\) \(\langle t, y \rangle \in W\)
using rel_compdef by auto
from assms(2) \(\langle t, y \rangle \in W\) have \(t \in X\) by auto
from \(\langle x, t \rangle \in W\) have \(\langle t, x \rangle \in converse(W)\) by auto
with assms(1) I \(\langle t, y \rangle \in W\) \(\land\) \(t \in X\)
have \(t \in X\) by auto
with assms I \(\langle t, y \rangle \in W\) have \(y \in Star(\{x\}, \{W\}. t \in X]\)
by auto
} then show \((W ∩ W)\{x\} \subseteq Star(\{x\}, \{W\}. t \in X))\)
by blast
qed

Given a family of uniform covers of \(X\) we can create a uniformity on \(X\) by taking the supersets of \(\bigcup\{A \times A : A \in P\}\) as \(P\) ranges over the uniform covers. The next definition specifies the operation creating entourages from uniform covers.

definition UniformityFromUniCov(X, Θ) ≡ Supersets(X × X, \{\bigcup\{U × U. U ∈ P\}. P ∈ Θ\})

For any member \(P\) of a cover \(Θ\) the set \(\bigcup\{U × U : U ∈ P\}\) is a member of UniformityFromUniCov(X, Θ).

lemma basic_unif: assumes \(Θ \subseteq Covers(X)\) \(P ∈ Θ\)
shows \(\bigcup\{U × U. U ∈ P\} ∈ UniformityFromUniCov(X, Θ)\)
using assms unfolding UniformityFromUniCov_def Supersets_def Covers_def
by blast

If \(Θ\) is a family of uniform covers of \(X\) then UniformityFromUniCov(X, Θ) is a uniformity on \(X\).

theorem uniformity_from_unicov:
assumes \(Θ \{are uniform covers of\} X \ X \neq 0\)
shows UniformityFromUniCov(X, Θ) \{is a uniformity on\} \(X\)
proof -
let \(Φ = UniformityFromUniCov(X, Θ)\)
have \(Φ \{is a filter on\} (X × X)\)
proof -
have \(0 \notin Φ\)
proof -
{ assume \(0 \in Φ\)
then obtain P where \(P ∈ Θ\) and \(0 = \bigcup\{U × U. U ∈ P\}\)
unfolding UniformityFromUniCov_def Supersets_def by auto
hence \(\bigcup P = 0\) by auto
with assms \(P ∈ Θ\) have False unfolding AreUniformCovers_def Covers_def
by auto
} thus thesis by auto
qed
moreover have \(X × X ∈ Φ\)
proof -
from assms have \(X × X ∈ \{\bigcup\{U × U. U ∈ P\}. P ∈ Θ\}\)

using univcov_contains_triv unfolding AreUniformCovers_def
by auto
then show thesis unfolding Supersets_def UniformityFromUniCov_def
by blast
qed
moreover have \( \Phi \subseteq \text{Pow}(X \times X) \)
unfolding UniformityFromUniCov_def Supersets_def by auto
moreover have \( \forall A \in \Phi. \forall B \in \Phi. A \cap B \in \Phi \)
proof -
{ fix A B assume \( A \in \Phi \) \( B \in \Phi \)
  then have \( A \cap B \subseteq X \times X \)
  unfolding UniformityFromUniCov_def Supersets_def by auto
  from \( A \in \Phi \) \( B \in \Phi \) obtain \( P \)
  where \( \bigcup \{ U \times U. U \in P \} \subseteq A \cap B \)
  unfolding AreUniformCovers_def by blast
  qed
  with \( A \cap B \subseteq X \times X \) \( P \in \Theta \) have \( A \cap B \in \Phi \)
  unfolding Supersets_def UniformityFromUniCov_def by auto
  thus thesis by auto
qed
moreover have \( \forall B \in \Phi. \forall C \in \text{Pow}(X \times X). B \subseteq C \rightarrow C \in \Phi \)
proof -
{ fix B C assume \( B \in \Phi \) \( C \in \text{Pow}(X \times X) \) \( B \subseteq C \)
  from \( B \in \Phi \) obtain \( P_B \) where \( \bigcup \{ U \times U. U \in P_B \} \subseteq B \)
  unfolding UniformityFromUniCov_def Supersets_def by auto
  with \( \forall C \in \text{Pow}(X \times X). C \subseteq C \in \Phi \) have \( C \in \Phi \)
  unfolding UniformityFromUniCov_def by blast
  thus thesis by auto
  qed
ultimately show thesis unfolding IsFilter_def by simp
qed
moreover have \( \forall A \in \Phi. \text{id}(X) \subseteq A \land (\exists B \in \Phi. B \circ B \subseteq A) \land \text{converse}(A) \in \Phi \)
proof -
{ fix A assume \( A \in \Phi \)
  then obtain \( P \) where \( \bigcup \{ U \times U. U \in P \} \subseteq A \)
  unfolding UniformityFromUniCov_def Supersets_def by auto
  have \( \text{id}(X) \subseteq A \)
  qed
1296
proof -
  from assms(1) \langle P \in \Theta \rangle have \bigcup P = X unfolding AreUniformCovers_def
Covers_def
  by auto
with \langle \bigcup \{ U \times U . U \in P \} \subseteq A \rangle show thesis by auto
qed
moreover have \exists B \in \Phi . B \circ B \subseteq A
proof -
  from assms(1) \langle P \in \Theta \rangle have \bigcup \{ U \times U . U \in P \} \subseteq A
  unfolding AreUniformCovers_def Covers_def UniformityFromUniCov_def
Supersets_def
  by auto
from assms(1) \langle P \in \Theta \rangle obtain Q where Q \in \Theta and Q \textless\textasciitilde P using unicov_has_star_ref
by blast
let B = \bigcup \{ U \times U . U \in Q \}
from assms(1) \langle Q \in \Theta \rangle have B \in \Phi
  unfolding AreUniformCovers_def Covers_def UniformityFromUniCov_def
Supersets_def
  by auto
moreover have B \circ B \subseteq A
proof -
  have II: B \circ B = \bigcup \{ U \times \text{Star}(U,Q) . U \in Q \} using rel_square_starr
    by simp
have \forall U \in Q . \exists V \in P . U \times \text{Star}(U,Q) \subseteq V \times V
proof
  fix U assume U \in Q
  with \langle Q \textless\textasciitilde P \rangle obtain V where V \in P and Star(U,Q) \subseteq V
    unfolding IsStarRefinement_def by blast
  with \langle U \in Q \rangle have V \in P and U \times \text{Star}(U,Q) \subseteq V \times V using element_subset_star
    by auto
  thus \exists V \in P . U \times \text{Star}(U,Q) \subseteq V \times V by auto
  qed
hence \bigcup \{ U \times \text{Star}(U,Q) . U \in Q \} \subseteq \bigcup \{ V \times V . V \in P \} by blast
with \langle \bigcup \{ V \times V . V \in P \} \subseteq A \rangle have \bigcup \{ U \times \text{Star}(U,Q) . U \in Q \} \subseteq A by blast
with II show thesis by simp
  qed
ultimately show thesis by auto
  qed
moreover from \langle A \in \Phi \rangle \langle P \in \Theta \rangle \langle \bigcup \{ U \times U . U \in P \} \subseteq A \rangle have converse(A)
  \in \Phi
  unfolding AreUniformCovers_def UniformityFromUniCov_def Supersets_def
    by auto
ultimately show id(X) \subseteq A \land (\exists B \in \Phi . B \circ B \subseteq A) \land converse(A) \in \Phi
    by simp
  qed
ultimately show \Phi \{ is a uniformity on \} X unfolding IsUniformity_def

1297
Given a uniformity $\Phi$ on $X$ we can create a family of uniform covers by taking the collection of covers $P$ for which there exist an entourage $U \in \Phi$ such that for each $x \in X$, there is an $A \in P$ such that $U(\{x\}) \subseteq A$. The next definition specifies the operation of creating a family of uniform covers from a uniformity.

**definition**

$\text{UniCovFromUniformity}(X, \Phi) \equiv \{ P \in \text{Covers}(X). \exists U \in \Phi. \forall x \in X. \exists A \in P. U(\{x\}) \subseteq A \}$

When we convert the quantifiers into unions and intersections in the definition of $\text{UniCovFromUniformity}$ we get an alternative definition of the operation that creates a family of uniform covers from a uniformity. Just a curiosity, not used anywhere.

**lemma** $\text{UniCovFromUniformityDef}$: assumes $X \neq 0$

shows $\text{UniCovFromUniformity}(X, \Phi) = (\bigcup U \in \Phi. \forall x \in X. \exists A \in P. U(\{x\}) \subseteq A)$

proof

{ fix $P$ assume $P \in \{ P \in \text{Covers}(X). \exists U \in \Phi. \forall x \in X. \exists A \in P. U(\{x\}) \subseteq A \}$

then have $P \in \text{Covers}(X)$ and $\forall x \in X. \exists A \in P. U(\{x\}) \subseteq A$ by auto

then obtain $U$ where $U \in \Phi$ and $\forall x \in X. \exists A \in P. U(\{x\}) \subseteq A$ by auto

with assms $P \in \text{Covers}(X)$ have $P \in (\bigcap x \in X. \{ P \in \text{Covers}(X). \exists A \in P. U(\{x\}) \subseteq A \})$

by auto

with $\langle U \in \Phi \rangle$ have $P \in (\bigcap x \in X. \{ P \in \text{Covers}(X). \exists A \in P. U(\{x\}) \subseteq A \})$

by blast

} then show

$\{ P \in \text{Covers}(X). \exists U \in \Phi. \forall x \in X. \exists A \in P. U(\{x\}) \subseteq A \} \subseteq$

$(\bigcup U \in \Phi. \forall x \in X. \{ P \in \text{Covers}(X). \exists A \in P. U(\{x\}) \subseteq A \})$

using subset_iff by simp

{ fix $P$ assume $P \in (\bigcup U \in \Phi. \forall x \in X. \{ P \in \text{Covers}(X). \exists A \in P. U(\{x\}) \subseteq A \})$

then obtain $U$ where $U \in \Phi$ and $\forall x \in X. \exists A \in P. U(\{x\}) \subseteq A$ by auto

with $\langle U \in \Phi \rangle$ have $P \in \text{Covers}(X). \exists A \in P. U(\{x\}) \subseteq A$ by auto

} then show $(\bigcup U \in \Phi. \forall x \in X. \{ P \in \text{Covers}(X). \exists A \in P. U(\{x\}) \subseteq A \}) \subseteq$

$\{ P \in \text{Covers}(X). \exists U \in \Phi. \forall x \in X. \exists A \in P. U(\{x\}) \subseteq A \}$ by auto

qed

then show thesis unfolding $\text{UniCovFromUniformity_def}$ by simp
If $\Phi$ is a (diagonal) uniformity on $X$, then covers of the form $\{W\{x\} : x \in X\}$ are members of $\text{UniCovFromUniformity}(X, \Phi)$.

**Lemma cover_image:**
- assumes $\Phi$ {is a uniformity on} $X$ $W \in \Phi$
- shows $\{W\{x\} : x \in X\} \in \text{UniCovFromUniformity}(X, \Phi)$

**Proof** -
- let $P = \{W\{x\} : x \in X\}$
- have $P \in \text{Covers}(X)$
  - from asssms have $W \subseteq X \times X$ and $P \in \text{Pow(Pow}(X))$
  - using entourage<Props>(1) by auto
  - moreover have $\bigcup P = X$
    - from $W \subseteq X \times X$ show $\bigcup P \subseteq X$ by auto
    - from asssms show $X \subseteq \bigcup P$ using neigh_not_empty(2) by auto
  - ultimately show thesis unfolding $\text{Covers}_\text{def}$ by simp

- moreover from asssms(2) have $\exists W \in \Phi. \forall x \in X. \exists A \in P. W\{x\} \subseteq A$
  - by auto
  - ultimately show thesis unfolding $\text{UniCovFromUniformity}_\text{def}$ by simp

**Qed**

If $\Phi$ is a (diagonal) uniformity on $X$, then every two elements of $\text{UniCovFromUniformity}(X, \Phi)$ have a common barycentric refinement.

**Lemma common_bar_refinemnt:**
- assumes $\Phi$ {is a uniformity on} $X$
  - $\Theta = \text{UniCovFromUniformity}(X, \Phi)$
  - $C \in \Theta$
  - $D \in \Theta$
- shows $\exists R \in \Theta. (R <^B C) \land (R <^B D)$

**Proof** -
- from asssms(2,3) obtain $U$ where $U \in \Phi$ and $I$: $\forall x \in X. \exists C \in \Theta. U\{x\} \subseteq C$
  - unfolding $\text{UniCovFromUniformity}_\text{def}$ by auto
- from asssms(2,4) obtain $V$ where $V \in \Phi$ and $II$: $\forall x \in X. \exists D \in \Theta. V\{x\} \subseteq D$
  - unfolding $\text{UniCovFromUniformity}_\text{def}$ by auto
- from asssms(1) $\langle U \in \Phi \rangle$ $\langle V \in \Phi \rangle$ have $U \cap V \in \Phi$
  - unfolding $\text{IsUniformity}_\text{def}$ $\text{IsFilter}_\text{def}$ by auto
  - with asssms(1) obtain $W$ where $W \in \Phi$ and $W \circ W \subseteq U \cap V$ and $W = \text{converse}(W)$
    - using half_size_symm by blast
- from asssms(1) $\langle W \in \Phi \rangle$ have domain($W$) $\subseteq X$
  - unfolding $\text{IsUniformity}_\text{def}$ $\text{IsFilter}_\text{def}$ by auto
- let $P = \{W\{t\} : t \in X\}$
- have $P \in \Theta$ $P <^B C$ $P <^B D$

**Proof** -
- from asssms(1,2) $\langle W \in \Phi \rangle$ show $P \in \Theta$ using cover_image by simp
with \texttt{assms(2)} have $\bigcup P = X$ unfolding \texttt{UniCovFromUniformity_def Covers_def}
by simps
{ fix $x$ assume $x \in X$
  from $\langle W = \text{converse}(W), \langle \text{domain}(W) \subseteq X \rangle \langle W \cap W \subseteq U \cap V \rangle$ have $\text{Star}\{\{x\}, P\} \subseteq U\{x\}$ and $\text{Star}\{\{x\}, P\} \subseteq V\{x\}$
  using rel_sq_image by auto
  from $\langle x \in X \rangle$ I obtain $C$ where $C \in C$ and $U\{x\} \subseteq C$
  by auto
  with $\langle \text{Star}\{\{x\}, P\} \subseteq U\{x\} \rangle \langle C \in C \rangle$
  have $\exists C \in C. \text{Star}\{\{x\}, P\} \subseteq C$
  by auto
  moreover
  from $\langle x \in X \rangle$ II obtain $D$ where $D \in D$ and $V\{x\} \subseteq D$
  by auto
  with $\langle \text{Star}\{\{x\}, P\} \subseteq V\{x\} \rangle \langle D \in D \rangle$
  have $\exists D \in D. \text{Star}\{\{x\}, P\} \subseteq D$
  by auto
  ultimately have $\exists C \in C. \text{Star}\{\{x\}, P\} \subseteq C$ and $\exists D \in D. \text{Star}\{\{x\}, P\} \subseteq D$
  by auto
  } hence $\forall x \in X. \exists C \in C. \text{Star}\{\{x\}, P\} \subseteq C$ and $\forall x \in X. \exists D \in D. \text{Star}\{\{x\}, P\} \subseteq D$
  by auto
  with $\langle \bigcup P = X \rangle$ show $P <^B C$ and $P <^B D$
  unfolding \texttt{IsBarycentricRefinement_def} by auto
  qed
  thus thesis by auto
  qed

If $\Phi$ is a (diagonal) uniformity on $X$, then every element of $\text{UniCovFromUniformity}(X, \Phi)$ has a barycentric refinement there.

corollary bar_refinement_ex:
  assumes $\Phi$ {is a uniformity on} $X$ $\Theta = \text{UniCovFromUniformity}(X, \Phi)$ $C \in \Theta$
  shows $\exists R \in \Theta$. ($R <^B C$)
  using \texttt{assms common_bar_refinement} by blast

If $\Phi$ is a (diagonal) uniformity on $X$, then $\text{UniCovFromUniformity}(X, \Phi)$ is a family of uniform covers.

theorem unicov_from_uniformity: assumes $\Phi$ {is a uniformity on} $X$ and $X \neq \emptyset$
  shows $\text{UniCovFromUniformity}(X, \Phi)$ {are uniform covers of} $X$
proof -
  let $\Theta = \text{UniCovFromUniformity}(X, \Phi)$
  from \texttt{assms(1)} have $\Theta \subseteq \text{Covers}(X)$ unfolding \texttt{UniCovFromUniformity_def}
  by auto
  moreover
  from \texttt{assms(1)} have $\{x\} \in \Theta$
  unfolding \texttt{Covers_def IsUniformity_def IsFilter_def UniCovFromUniformity_def}
  by auto
hence $\Theta \neq 0$ by auto
moreover have $\forall R \in \Theta. \forall C \in \text{Covers}(X). ((R \prec^* C) \rightarrow C \in \Theta)$
proof -
\{ fix $R \ C$ assume $R \in \Theta \ C \in \text{Covers}(X) \ R \prec^* C$
  have $C \in \Theta$
proof -
  from $\langle R \in \Theta \rangle$ obtain $U$ where $U \in \Phi$ and I: $\forall x \in X. \exists R \in R. \ U(\{x\}) \subseteq$
  unfolding $\text{UniCovFromUniformity\_def}$ by auto
  \{ fix $x$ assume $x \in X$
    with I obtain $R$ where $R \in R$ and $U(\{x\}) \subseteq R$ by auto
  from $\langle R \in R \rangle$ $\langle R \prec^* C \rangle$ obtain $C$ where $C \in C$ and $\text{Star}(R,R) \subseteq$
  unfolding $\text{IsStarRefinement\_def}$ by auto
    with $\langle U \in \Phi \rangle$ $\langle C \in \text{Covers}(X) \rangle$ show thesis unfolding $\text{UniCovFromUniformity\_def}$ by auto
\} hence $\forall x \in X. \exists C \in C. \ U(\{x\}) \subseteq C$
proof -
\{ fix $C \ D$ assume $C \in \Theta \ D \in \Theta$
  with assms(1) obtain $P$ where $P \in \Theta$ and $P \prec^B C \ P \prec^B D$
    using $\text{common\_bar\_refinement}$ by blast
  from assms(1) $\langle P \in \Theta \rangle$ obtain $R$ where $R \in \Theta$ and $R \prec^B P$
    using $\text{bar\_refinement\_ex}$ by blast
  from $\langle R \in \Theta \rangle$ $\langle P \in \Theta \rangle$ $\langle C \in \Theta \rangle$ $\langle D \in \Theta \rangle$ have
    $P \in \text{Covers}(X) \ R \in \text{Covers}(X) \ C \in \text{Covers}(X) \ D \in \text{Covers}(X)$
    unfolding $\text{UniCovFromUniformity\_def}$ by auto
  with assms(2) $\langle R \prec^B P \rangle$ $\langle P \prec^B C \rangle$ $\langle P \prec^B D \rangle$ have $R \prec^* C$ and $R \prec^*$
\} thus thesis by simp
qed
ultimately show thesis unfolding $\text{AreUniformCovers\_def}$ by simp
qed
The $\text{UniCovFromUniformity}$ operation is the inverse of $\text{UniformityFromUniCov}$.

**Theorem unicov_from_unif_inv:** assumes $\Theta \{\text{are uniform covers of} \ X \ X \neq 0$
shows $\text{UniCovFromUniformity}(X,\text{UniformityFromUniCov}(X,\Theta)) = \Theta$
proof
let $\Phi = \text{UniformityFromUniCov}(X,\Theta)$
let $L = \text{UniCovFromUniformity}(X,\Phi)$
from assms have I: $\Phi \{\text{is a uniformity on} \ X$

1301
using uniformity_from_unicov by simp
with assms(2) have II: L {are uniform covers of} X
using unicov_from_uniformity by simp
{ fix P assume P∈L
with I obtain Q where Q∈L and Q <^B P
using bar_refinement_ex by blast
from \langle Q∈L \rangle obtain U where U∈Φ and III:∀x∈X.∃A∈Q. U(x) ⊆ A
unfolding UnicovFromUniformity_def by auto
from \langle U∈Φ \rangle have U ∈ Supersets(X×X,\{ U×U. P∈P \})
unfolding UniformityFromUniCov_def by simp
then obtain B where B ⊆ X×X B ⊆ U and ∃C∈{ U×U. P∈P }. C ⊆ B
unfolding Supersets_def by auto
then obtain C where C∈{ U×U. P∈P } and C ⊆ B by auto
then obtain R where R∈Θ and C = \bigcup\{ V×V. V∈R \} by auto
with \langle C∈B \rangle \langle B∈U \rangle have \bigcup\{ U×U. V∈R \} ⊆ U by auto
from assms(1) II \langle P∈L \rangle \langle Q∈L \rangle \langle R∈Θ \rangle have
IV: P∈Covers(X) Q∈Covers(X) R∈Covers(X)
unfolding AreUniformCovers_def by auto
have R <^B Q
proof -
{ fix x assume x∈X
with III obtain A where A∈Q and U(x) ⊆ A by auto
with \langle \bigcup\{ V×V. V∈R \} ⊆ U \rangle have \bigcup\{ U×U. V∈R \} \{x\} ⊆ A
by auto
with \langle A∈Q \rangle have ∃A∈Q. Star({x},R) ⊆ A using star_singleton by auto
} then have ∀x∈X. ∃A∈Q. Star({x},R) ⊆ A by simp
moreover from \langle R∈Covers(X) \rangle have \bigcup R = X unfolding Covers_def
by simp
ultimately show thesis unfolding IsBarycentricRefinement_def
by simp
qed
with assms(2) \langle Q <^B P \rangle IV have R <^* P using bary_bary_star by simp
with assms(1) \langle R∈Θ \rangle \langle P∈Covers(X) \rangle have P∈Θ
unfolding AreUniformCovers_def by simp
} thus L⊆Θ by auto
{ fix P assume P∈Θ
with assms(1) have P ∈ Covers(X)
unfolding AreUniformCovers_def by auto
from assms(1) \langle P∈Θ \rangle obtain Q where Q ∈ Θ and Q <^B P
using unicov_has_bar_ref by blast
let A = \bigcup\{ V×V. V∈Q \}
have A ∈ Φ
proof -
from assms(1) \langle Q∈Θ \rangle have A ⊆ X×X and A ∈ { \bigcup\{ V×V. V∈Q \}. Q∈Θ }
unfolding AreUniformCovers_def Covers_def by auto
then show thesis
using superset_gen unfolding UniformityFromUniCov_def
by auto
qed
with I obtain B where B ∈ Φ B O B ⊆ A and B = converse(B)
using half_size_symm by blast
let R = {B{x}. x ∈ X}
from I II 〈B ∈ Φ〉 have R ∈ L and \( \bigcup R = X \)
using cover_image unfolding UniCovFromUniformity_def Covers_def
by auto
have R <^B P
proof -
{ fix x assume x ∈ X
  from assms(1) 〈Q ∈ Θ〉 have \( \bigcup Q = X \)
  unfolding AreUniformCovers_def Covers_def by auto
  with 〈Q <^B P〉 x ∈ X obtain C where C ∈ P and Star({x},Q) ⊆ C
  unfolding IsBarycentricRefinement_def by auto
  from 〈B = converse(B)〉 I 〈B ∈ Φ〉 have Star({x},R) = (B O B){x}
  using uni_domain rel_sq_image by auto
  moreover from 〈(B O B) ⊆ A〉 have (B O B){x} ⊆ A{x} by blast
  moreover have A{x} = Star({x},Q) using star_singleton by simp
  ultimately have Star({x},R) ⊆ Star({x},Q) by auto
  with 〈Star({x},Q) ⊆ C> 〈C ∈ P〉 have ∃C ∈ P. Star({x},R) ⊆ C
  by auto
} with 〈\( \bigcup R = X \)〉 show thesis unfolding IsBarycentricRefinement_def
by auto
qed
with assms(2) II 〈P ∈ Covers(X)〉 〈R ∈ L〉 〈R <^B P〉 have P ∈ L
using unicov_bary_cov by simp
} thus Θ ⊆ L by auto
qed

The UniformityFromUniCov operation is the inverse of UniCovFromUniformity.

theorem unif_from_unicov_inv: assumes Φ {is a uniformity on} X X ≠\0
shows UniformityFromUniCov(X,UniCovFromUniformity(X,Φ)) = Φ
proof
let Θ = UniCovFromUniformity(X,Φ)
let L = UniformityFromUniCov(X,Θ)
from assms have I: Θ {are uniform covers of} X
  using unicov_from_uniformity by simp
with assms have II: L {is a uniformity on} X
  using uniformity_from_unicov by simp
{ fix A assume A ∈ Φ
  with assms(1) obtain B where B ∈ Φ B O B ⊆ A and B = converse(B)
  using half_size_symm by blast
  from assms(1) 〈A ∈ Φ〉 have A ⊆ X×X using uni_domain(1)
  by simp
  let P = {B{x}. x ∈ X}
  from assms(1) 〈B ∈ Φ〉 have P ∈ Θ using cover_image
  by simp
  let C = \( \bigcup \{U × U. U ∈ P\} \)
}
from I $P \in \Theta$ have $C \in L$
unfolding AreUniformCovers_def using basic_unif by blast
from assms(1) $B \in \Phi$ $B = \text{converse}(B)$ $B \circ B \subseteq A$ have $C \subseteq A$
unfolding basic_unif by blast
with II $A \subseteq X \times X$ $C \subseteq L$ have $A \in L$
unfolding IsUniformity_def IsFilter_def by simp
thus $\Phi \subseteq L$ by auto
{ fix $A$ assume $A \in L$
  with II have $A \subseteq X \times X$ using entourage_props(1) by simp
  from $A \in L$ obtain $P$ where $P \in \Theta$ and $\bigcup \{ U \times U. U \in P \} \subseteq A$
  unfolding UniformityFromUniCov_def Supersets_def by blast
  from $P \in \Theta$ obtain $B$ where $B \in \Phi$ and $\forall x \in X. \exists V \in P. B(x) \subseteq V$
  unfolding UniCovFromUniformity_def by auto
  have $B \subseteq A$
  proof -
    from assms(1) $B \in \Phi$ have $B \subseteq \bigcup \{ B(x) \times B(x). x \in X \}$
    using entourage_props(1,2) refl_union_singl_image by simp
    moreover have $\bigcup \{ B(x) \times B(x). x \in X \} \subseteq A$
    proof -
      { fix $x$ assume $x \in X$
        with III obtain $V$ where $V \in P$ and $B(x) \subseteq V$ by auto
        hence $B(x) \times B(x) \subseteq \bigcup \{ U \times U. U \in P \}$ by auto
      } hence $\bigcup \{ B(x) \times B(x). x \in X \} \subseteq \bigcup \{ U \times U. U \in P \}$ by blast
      with $\bigcup \{ U \times U. U \in P \} \subseteq A$ show thesis by blast
    qed
  qed
  ultimately show thesis by auto
  qed
  with assms(1) $B \in \Phi$ $A \subseteq X \times X$ have $A \in \Phi$
  unfolding IsUniformity_def IsFilter_def by simp
} thus $L \subseteq \Phi$ by auto
qed

end

89  Topological groups - introduction

theory TopologicalGroup_ZF imports Topology_ZF_3 Group_ZF_1 Semigroup_ZF
begin

This theory is about the first subject of algebraic topology: topological groups.

89.1  Topological group: definition and notation

Topological group is a group that is a topological space at the same time. This means that a topological group is a triple of sets, say $(G,f,T)$ such that $T$ is a topology on $G$, $f$ is a group operation on $G$ and both $f$ and the
operation of taking inverse in $G$ are continuous. Since IsarMathLib defines topology without using the carrier, (see Topology_ZF), in our setup we just use $\bigcup T$ instead of $G$ and say that the pair of sets $(\bigcup T, f)$ is a group. This way our definition of being a topological group is a statement about two sets: the topology $T$ and the group operation $f$ on $G = \bigcup T$. Since the domain of the group operation is $G \times G$, the pair of topologies in which $f$ is supposed to be continuous is $T$ and the product topology on $G \times G$ (which we will call $\tau$ below).

This way we arrive at the following definition of a predicate that states that pair of sets is a topological group.

**definition**

\[
\text{IsAtopologicalGroup}(T, f) \equiv (T \{\text{is a topology}\}) \land \text{IsAgroup}(\bigcup T, f) \land 
\text{IsContinuous}(\text{ProductTopology}(T, T), T, f) \land 
\text{IsContinuous}(T, T, \text{GroupInv}(\bigcup T, f))
\]

We will inherit notation from the topology0 locale. That locale assumes that $T$ is a topology. For convenience we will denote $G = \bigcup T$ and $\tau$ to be the product topology on $G \times G$. To that we add some notation specific to groups. We will use additive notation for the group operation, even though we don’t assume that the group is abelian. The notation $g + A$ will mean the left translation of the set $A$ by element $g$, i.e. $g + A = \{g + a | a \in A\}$. The group operation $G$ induces a natural operation on the subsets of $G$ defined as $(A, B) \mapsto \{x + y | x \in A, y \in B\}$. Such operation has been considered in func_ZF and called $f$ ”lifted to subsets of” $G$. We will denote the value of such operation on sets $A, B$ as $A + B$. The set of neigboorhoods of zero (denoted $\mathcal{N}_0$) is the collection of (not necessarily open) sets whose interior contains the neutral element of the group.

**locale** topgroup = topology0 +

**fixes** G
**defines** G_def [simp]: G $\equiv$ $\bigcup T$

**fixes** prodtop ($\tau$)
**defines** prodtop_def [simp]: $\tau$ $\equiv$ ProductTopology(T,T)

**fixes** f

**assumes** Ggroup: IsAgroup(G,f)

**assumes** fcon: IsContinuous(\tau,T,f)

**assumes** inv_cont: IsContinuous(T,T,GroupInv(G,f))

**fixes** groip (infixl 90)
**defines** groip_def [simp]: $x+y$ $\equiv$ $f(x,y)$

1305
The first lemma states that we indeed talk about topological group in the context of topgroup locale.

lemma (in topgroup) topGroup: shows IsAtopologicalGroup(T,f)
  using topSpaceAssum Ggroup fcon inv_cont IsAtopologicalGroup_def by simp

If a pair of sets \((T, f)\) forms a topological group, then all theorems proven in the topgroup context are valid as applied to \((T, f)\).

lemma topGroupLocale: assumes IsAtopologicalGroup(T,f)
  shows topgroup(T,f)
  using assms IsAtopologicalGroup_def topgroup_def
topgroup_axioms.intro topology0_def by simp

We can use the group0 locale in the context of topgroup.

lemma (in topgroup) group0_valid_in_tgroup: shows group0(G,f)
  using Ggroup group0_def by simp

We can use the group0 locale in the context of topgroup.

sublocale topgroup < group0 G f gzero grop grinv
  unfolding group0_def gzero_def grop_def grinv_def using Ggroup by auto
We can use semigr0 locale in the context of topgroup.

**lemma** (in topgroup) semigr0_valid_in_tgroup: shows semigr0(G,f)
  using Ggroup IsAgroup_def IsAmonoid_def semigr0_def by simp

We can use the prod_top_spaces0 locale in the context of topgroup.

**lemma** (in topgroup) prod_top_spaces0_valid: shows prod_top_spaces0(T,T,T)
  using topSpaceAssum prod_top_spaces0_def by simp

Negative of a group element is in group.

**lemma** (in topgroup) neg_in_tgroup: assumes g ∈ G shows (-g) ∈ G
  using assms inverse_in_group by simp

Sum of two group elements is in the group.

**lemma** (in topgroup) group_op_closed_add: assumes x1 ∈ G x2 ∈ G
  shows x1+x2 ∈ G
  using assms group_op_closed by simp

Zero is in the group.

**lemma** (in topgroup) zero_in_tgroup: shows 0 ∈ G
  using group0_2_L2 by simp

Another lemma about canceling with two group elements written in additive notation

**lemma** (in topgroup) inv_cancel_two_add: assumes x1 ∈ G x2 ∈ G
  shows x1+(-x2)+x2 = x1
  x1+x2+(-x2) = x1
  (-x1)+(x1+x2) = x2
  x1+((-x1)+x2) = x2
  using assms inv_cancel_two by auto

Useful identities proven in the Group_ZF theory, rewritten here in additive notation. Note since the group operation notation is left associative we don’t really need the first set of parentheses in some cases.

**lemma** (in topgroup) cancel_middle_add: assumes x1 ∈ G x2 ∈ G x3 ∈ G
  shows (x1+(-x2))+(x2+(-x3)) = x1+ (-x3)
  (((-x1)+x2)+((-x2)+x3) = (-x1)+ x3
  (- (x1+x2)) + (x1+x3) = (-x2)+x3
  (x1+x2) + (- (x1+x2)) =x1+ (-x3)
  (-x1) + (x1+x2+x3) + (-x3) = x2
  proof -
  from assms have x1+ (-x3) = (x1+(-x2))+(x2+(-x3))
    using group0_2_L14A(1) by blast
  thus (x1+(-x2))+(x2+(-x3)) = x1+ (-x3) by simp
from assms have \((-x_1)+x_3=((-x_1)+x_2)+((-x_2)+x_3)\)
  using group0_2_L14A(2) by blast

thus \((-x_1)+x_2)+((-x_2)+x_3)=(-x_1)+x_3\) by simp
from assms show \((- (x_1+x_2)) + (x_1+x_3) = (-x_2)+x_3\)
  using cancel_middle(1) by simp

from assms show \((x_1+x_2) + ((-x_1)+x_2)) =x_1 + (-x_3)\)
  using cancel_middle(2) by simp
from assms show \((-x_1) + (x_1+x_2)+x_3) + (-x_3) = x_2\)
  using cancel_middle(3) by simp

qed

We can cancel an element on the right from both sides of an equation.

lemma (in topgroup) cancel_right_add:
  assumes \(x_1 \in G\ \ x_2 \in G\ \ x_3 \in G\ \ x_1+x_2 = x_3+x_2\)
  shows \(x_1 = x_3\)
  using assms cancel_right by simp

We can cancel an element on the left from both sides of an equation.

lemma (in topgroup) cancel_left_add:
  assumes \(x_1 \in G\ \ x_2 \in G\ \ x_3 \in G\ \ x_1+x_2 = x_1+x_3\)
  shows \(x_2 = x_3\)
  using assms cancel_left by simp

We can put an element on the other side of an equation.

lemma (in topgroup) put_on_the_other_side:
  assumes \(x_1 \in G\ \ x_2 \in G\ \ x_3 \in G\ \ x_1+x_2 = x_3\)
  shows \(x_3+(-x_2) = x_1\) \(\\text{and}\ (-x_1)+x_3 = x_2\)
  using assms group0_2_L18 by auto

A simple equation from lemma simple_equation0 in Group_ZF in additive notation

lemma (in topgroup) simple_equation0_add:
  assumes \(x_1 \in G\ \ x_2 \in G\ \ x_3 \in G\ \ x_1+(-x_2) = (-x_3)\)
  shows \(x_3 = x_2 + (-x_1)\)
  using assms simple_equation0 by blast

A simple equation from lemma simple_equation1 in Group_ZF in additive notation

lemma (in topgroup) simple_equation1_add:
  assumes \(x_1 \in G\ \ x_2 \in G\ \ x_3 \in G\ \ (-x_1)+x_2 = (-x_3)\)
  shows \(x_3 = (-x_2) + x_1\)
  using assms simple_equation1 by blast

The set comprehension form of negative of a set. The proof uses the ginv_image lemma from Group_ZF theory which states the same thing in multiplicative notation.

lemma (in topgroup) ginv_image_add: assumes \(V \subseteq G\)
shows \((-V) \subseteq G\) and \((-V) = \{-x. x \in V\} \)
using assms ginv_image by auto

The additive notation version of ginv_image_el lemma from Group_ZF theory

\textbf{lemma (in topgroup) ginv_image_el_add:} assumes \(V \subseteq G\) \(x \in (-V)\)
shows \((-x) \in V\)
using assms ginv_image_el by simp

Of course the product topology is a topology (on \(G \times G\)).

\textbf{lemma (in topgroup) prod_top_on_G:}
\quad shows \(\tau \{\text{is a topology}\} \text{ and } \bigcup \tau = G \times G\)
using topSpaceAssum Top_1_4_T1 by auto

Let’s recall that \(f\) is a binary operation on \(G\) in this context.

\textbf{lemma (in topgroup) topgroup_f_binop:}
\quad shows \(f : G \times G \rightarrow G\)
using Ggroup group0_def group0.group_oper_fun by simp

A subgroup of a topological group is a topological group with relative topology and restricted operation. Relative topology is the same as \(T \{\text{restricted to}\} H\) which is defined to be \(\{V \cap H : V \in T\}\) in ZF1 theory.

\textbf{lemma (in topgroup) top_subgroup:}
\quad assumes \(A1: \text{IsAsubgroup}(H,f)\)
\shows IsAtopologicalGroup\((\bigcup \tau_0, f_H)\)
proof -
\quad let \(\tau_0 = T \{\text{restricted to}\} H\)
\quad let \(f_H = \text{restrict}(f, H \times H)\)
\quad have \(\bigcup \tau_0 = G \cap H\) using union_restrict by simp
\quad also from \(A1\) have \(... = H\)
\quad using group0_3_L2 by blast
\quad finally have \(\bigcup \tau_0 = H\) by simp
\quad have \(\tau_0 \{\text{is a topology}\}\) using Top_1_L4 by simp
\quad moreover from \(A1\) \(\bigcup \tau_0 = H\) have IsAgroup\((\bigcup \tau_0, f_H)\)
\quad using IsAsubgroup_def by simp
\quad moreover have IsContinuous\((\text{ProductTopology}(\tau_0, \tau_0), \tau_0, f_H)\)
proof -
\quad have two_top_spaces0\((\tau, T, f)\)
\quad using topSpaceAssum prod_top_on_G topgroup_f_binop prod_top_on_G
\quad two_top_spaces0_def by simp
\quad moreover
\quad from \(A1\) have \(H \subseteq G\) using group0_3_L2 by simp
\quad then have \(H \times H \subseteq \bigcup \tau\) using prod_top_on_G by auto
\quad moreover have IsContinuous\((\tau, T, f)\) using fcon by simp
\quad ultimately have
\quad IsContinuous\((\tau \{\text{restricted to}\} H \times H, T \{\text{restricted to}\} f_H(H \times H), f_H)\)
\quad using two_top_spaces0.restr_restr_image_cont
\quad by simp
\quad moreover have
\quad ProductTopology\((\tau_0, \tau_0) = \tau \{\text{restricted to}\} H \times H\) using topSpaceAssum
\quad prod_top_restr_comm

1309
by simp 
moreover from A1 have $f_H(H \times H) = H$ using image_subgr_op 
by simp 
ultimately show thesis by simp 
qed

moreover have IsContinuous($\tau_0, \tau_0, \text{GroupInv}(\bigcup \tau_0, f_H)$) 

proof - 
let $g = \text{restrict} (\text{GroupInv}(G,f), H)$ 
have $\text{GroupInv}(G,f) : G \to G$ 
  using Ggroup group0_2_T2 by simp 
then have two_top_spaces0(T,T,GroupInv(G,f)) 
  using topSpaceAssum two_top_spaces0_def by simp 
moreover from A1 have $H \subseteq \bigcup T$ using group0_3_L2 by simp 
ultimately have 
  IsContinuous($\tau_0, T \{\text{restricted to}\} g(H), g$) 
  using inv_cont two_top_spaces0.restr_restr_image_cont 
  by simp 
moreover from A1 have $g(H) = H$ using restr_inv_onto by simp 
moreover 
from A1 have $\text{GroupInv}(H, f_H) = g$ using group0_3_T1 by simp 
with $\bigcup \tau_0 = H$ have $g = \text{GroupInv}(\bigcup \tau_0, f_H)$ by simp 
ultimately show thesis by simp 
qed 
ultimately show thesis unfolding IsAtopologicalGroup_def by simp 
qed 

89.2 Interval arithmetic, translations and inverse of set

In this section we list some properties of operations of translating a set and reflecting it around the neutral element of the group. Many of the results are proven in other theories, here we just collect them and rewrite in notation specific to the topgroup context.

Different ways of looking at adding sets.

lemma (in topgroup) interval_add: assumes $A \subseteq G$ $B \subseteq G$ shows 
$A + B \subseteq G$ 
$A + B = f(A \times B)$ 
$A + B = \{ \bigcup x \in A. x + B \}$ 
$A + B = \{ x+y. \langle x, y \rangle \in A \times B \}$ 

proof - 
from assms show $A + B \subseteq G$ and $A + B = f(A \times B)$ and $A + B = \{ x+y. \langle x, y \rangle \in A \times B \}$ 
  using topgroup_f_binop lift_subsets_explained by auto 
from assms show $A + B = \{ \bigcup x \in A. x + B \}$ using image_ltrans_union by simp 
qed 

If the neutral element is in a set, then it is in the sum of the sets.

lemma (in topgroup) interval_add_zero: assumes $A \subseteq G$ $0 \in A$
shows $0 \in A+A$

proof -
from assms have $0+0 \in A+A$ using interval_add(4) by auto
then show $0 \in A+A$ using group0_2_L2 by auto
qed

Some lemmas from Group_ZF_1 about images of set by translations written in additive notation

lemma (in topgroup) lrtrans_image: assumes $V \subseteq G \ x \in G$
shows $x+V = \{x+v. v \in V\}$
$V+x = \{v+x. v \in V\}$
using assms ltrans_image rtrans_image by auto

Right and left translations of a set are subsets of the group. This is of course typically applied to the subsets of the group, but formally we don’t need to assume that.

lemma (in topgroup) lrtrans_in_group_add: assumes $x \in G$
shows $x+V \subseteq G$ and $V+x \subseteq G$
using assms lrtrans_in_group by auto

A corollary from interval_add

corollary (in topgroup) elements_in_set_sum: assumes $A \subseteq G \ B \subseteq G$
t $\in A+B$ shows $\exists s \in A. \exists q \in B. t=s+q$
using assms interval_add(4) by auto

A corollary from lrtrans_image

corollary (in topgroup) elements_in_ltrans: assumes $B \subseteq G \ g \in G \ t \in g+B$
shows $\exists q \in B. t=g+q$
using assms lrtrans_image(1) by simp

Another corollary of lrtrans_image

corollary (in topgroup) elements_in_rtrans: assumes $B \subseteq G \ g \in G \ t \in B+g$ shows $\exists q \in B. t=q+g$
using assms lrtrans_image(2) by simp

Another corollary from interval_add

corollary (in topgroup) elements_in_set_sum_inv: assumes $A \subseteq G \ B \subseteq G \ t=s+q \ s \in A \ q \in B$
shows $t \in A+B$
using assms interval_add by auto

Another corollary of lrtrans_image

corollary (in topgroup) elements_in_ltrans_inv: assumes $B \subseteq G \ g \in G \ q \in B \ t=g+q$
shows $t \in g+B$
using assms lrtrans_image(1) by auto

1311
Another corollary of \texttt{rtrans_image_add}

\textbf{lemma (in topgroup) elements_in_rtrans_inv:} \par
\texttt{assumes } B \subseteq G \texttt{ g} \in G \texttt{ q} \in B \texttt{ t}=q+g \par
\texttt{shows } t \in B+g \par
\texttt{using assms lrtrans_image(2) by auto}

Right and left translations are continuous.

\textbf{lemma (in topgroup) trans_cont:} \texttt{assumes } g \in G \texttt{ shows}
\texttt{IsContinuous(T,T,RightTranslation(G,f,g)) and IsContinuous(T,T,LeftTranslation(G,f,g))}
\texttt{using assms trans_eq_section topgroup_f_binop fcon prod_top_spaces0_valid prod_top_spaces0.fix_1st_var_cont prod_top_spaces0.fix_2nd_var_cont by auto}

Left and right translations of an open set are open.

\textbf{lemma (in topgroup) open_tr_open:} \texttt{assumes } g \in G \texttt{ and } V \in T \\
\texttt{shows } g+V \in T \texttt{ and } V+g \in T \\
\texttt{using assms neg_in_tgroup trans_cont IsContinuous_def trans_image_vimage by auto}

Right and left translations are homeomorphisms.

\textbf{lemma (in topgroup) tr_homeo:} \texttt{assumes } g \in G \texttt{ shows}
\texttt{IsAhomeomorphism(T,T,RightTranslation(G,f,g)) and IsAhomeomorphism(T,T,LeftTranslation(G,f,g))}
\texttt{using assms trans_bij trans_cont open_tr_open bij_cont_open_homeo by auto}

Left translations preserve interior.

\textbf{lemma (in topgroup) ltrans_interior:} \texttt{assumes } A1: g \in G \texttt{ and A2: A} \subseteq G \\
\texttt{shows } g + \text{int}(A) = \text{int}(g+A) \\
\texttt{proof -} \\
\texttt{from assms have } A \subseteq \bigcup T \texttt{ and IsAhomeomorphism(T,T,LeftTranslation(G,f,g))}
\texttt{using tr_homeo by auto} \\
\texttt{then show thesis using int_top_invariant by simp qed}

Right translations preserve interior.

\textbf{lemma (in topgroup) rtrans_interior:} \texttt{assumes } A1: g \in G \texttt{ and A2: A} \subseteq G \\
\texttt{shows } \text{int}(A) + g = \text{int}(A+g) \\
\texttt{proof -} \\
\texttt{from assms have } A \subseteq \bigcup T \texttt{ and IsAhomeomorphism(T,T,RightTranslation(G,f,g))}
\texttt{using tr_homeo by auto} \\
\texttt{then show thesis using int_top_invariant by simp qed}

1312
Translating by an inverse and then by an element cancels out.

**Lemma (in topgroup)** trans_inverse_elem: assumes \( g \in G \) and \( A \subseteq G \)
shows \( g + ((-g) + A) = A \)
using assms neg_in_tgroup trans_comp_image group0_2_L6 trans_neutral
image_id_same
by simp

Inverse of an open set is open.

**Lemma (in topgroup)** open_inv_open: assumes \( V \in T \) shows \((-V) \in T \)
using assms inv_image_vimage inv_cont IsContinuous_def
by simp

Inverse is a homeomorphism.

**Lemma (in topgroup)** inv_homeo: shows IsAhomeomorphism(T,T,GroupInv(G,f))
using group_inv_bij inv_cont open_inv_open bij_cont_open_homeo by simp

Taking negative preserves interior.

**Lemma (in topgroup)** int_inv_inv_int: assumes \( A \subseteq G \)
shows \( \text{int}(-A) = -(\text{int}(A)) \)
using assms inv_homeo int_top_invariant
by simp

### 89.3 Neighborhoods of zero

Zero neighborhoods are (not necessarily open) sets whose interior contains the neutral element of the group. In the topgroup locale the collection of neighborhoods of zero is denoted \( N_0 \).

The whole space is a neighborhood of zero.

**Lemma (in topgroup)** zneigh_not_empty: shows \( G \in N_0 \)
using topSpaceAssum IsATopology_def Top_2_L3 zero_in_tgroup
by simp

Any element that belongs to a subset of the group belongs to that subset with the interior of a neighborhood of zero added.

**Lemma (in topgroup)** elem_in_int_sad: assumes \( A \subseteq G \) \( g \in A \) \( H \in N_0 \)
shows \( g \in A + \text{int}(H) \)
proof -
from assms(3) have \( 0 \in \text{int}(H) \) and \( \text{int}(H) \subseteq G \) using Top_2_L2 by auto
with assms(1,2) have \( g+0 \in A + \text{int}(H) \) using elements_in_set_sum_inv
by simp
with assms(1,2) show thesis using group0_2_L2 by auto
qed

Any element belongs to the interior of any neighborhood of zero left translated by that element.

**Lemma (in topgroup)** elem_in_int_ltrans:
assumes \( g \in G \) and \( H \in N_0 \)
shows \( g \in \text{int}(g+H) \) and \( g \in \text{int}(g+H) + \text{int}(H) \)
proof -
from assms(2) have \(0 \in \text{int}(H)\) and \(\text{int}(H) \subseteq G\) using Top_2_L2 by auto
with assms(1) have \(g \in g + \text{int}(H)\) using neut_trans_elem by simp
with assms show \(g \in \text{int}(g+H)\) using ltrans_interior by simp
from assms(1) have \(\text{int}(g+H) \subseteq G\) using lrtrans_in_group_add(1) Top_2_L1
by blast
with \(g \in \text{int}(g+H)\) assms(2) show \(g \in \text{int}(g+H) + \text{int}(H)\)
using elem_in_int_sad by simp
qed

Any element belongs to the interior of any neighborhood of zero right
translated by that element.

lemma (in topgroup) elem_in_int_rtrans:
assumes A1: \(g \in G\) and A2: \(H \in N_0\)
shows \(g \in \text{int}(H+g)\) and \(g \in \text{int}(H+g) + \text{int}(H)\)
proof -
from A2 have \(0 \in \text{int}(H)\) and \(\text{int}(H) \subseteq G\) using Top_2_L2 by auto
with A1 have \(g \in \text{int}(H+g) + \text{int}(H)\) using neut_trans_elem by simp
with assms show \(g \in \text{int}(H+g)\) using rtrans_interior by simp
from assms(1) have \(\text{int}(H+g) \subseteq G\) using lrtrans_in_group_add(2) Top_2_L1
by blast
with \(g \in \text{int}(H+g)\) assms(2) show \(g \in \text{int}(H+g) + \text{int}(H)\)
using elem_in_int_sad by simp
qed

Negative of a neighborhood of zero is a neighborhood of zero.

lemma (in topgroup) neg_neigh_neigh: assumes \(H \in N_0\)
shows \((-H) \in N_0\)
proof -
from assms have \(\text{int}(H) \subseteq G\) and \(0 \in \text{int}(H)\) using Top_2_L1 by auto
with assms have \(0 \in \text{int}(-H)\) using neg_in_tgroup by simp
moreover have \(\text{GroupInv}(G,f):G \rightarrow G\) using Ggroup group0_2_T2 by simp
then have \((-H) \subseteq G\) using func1_1_L6 by simp
ultimately show thesis by simp
qed

Left translating an open set by a negative of a point that belongs to it makes
it a neighborhood of zero.

lemma (in topgroup) open_trans_neigh: assumes A1: \(U \in T\) and \(g \in U\)
shows \((-g)+U \in N_0\)
proof -
let \(H = (-g)+U\)
from assms have \(g \in G\) by auto
then have \((-g) \in G\) using neg_in_tgroup by simp
with A1 have \(H \in T\) using open_tr_open by simp
hence \(H \subseteq G\) by auto

1314
moreover have $0 \in \text{int}(H)$
proof -
  from assms have $U \subseteq G$ and $g \in U$ by auto
  with $\langle H \in T \rangle$ show $0 \in \text{int}(H)$ using elem_trans_neut Top_2_L3 by auto
  qed
ultimately show thesis by simp
qed

Right translating an open set by a negative of a point that belongs to it makes it a neighborhood of zero.

lemma (in topgroup) open_trans_neigh_2: assumes $A1: U \in T$ and $g \in U$ shows $U+(\neg g) \in N_0$
proof -
  let $H = U+(\neg g)$
  from assms have $g \in G$ by auto
  then have $\neg g \in G$ using neg_in_tgroup by simp
  hence $H \subseteq G$ by auto
  moreover have $0 \in \text{int}(H)$
  proof -
    from assms have $U \subseteq G$ and $g \in U$ by auto
    with $\langle H \in T \rangle$ show $0 \in \text{int}(H)$ using elem_trans_neut Top_2_L3 by auto
    qed
ultimately show thesis by simp
qed

Right and left translating an neighborhood of zero by a point and its negative makes it back a neighborhood of zero.

lemma (in topgroup) lrtrans_neigh: assumes $W \in N_0$ and $x \in G$ shows $x+(W+(\neg x)) \in N_0$ and $(x+W)+(-x) \in N_0$
proof -
  from assms(2) have $x+(\neg x) \subseteq G$ using lrtrans_in_group_add(1) by simp
  moreover have $0 \in \text{int}(x+(\neg x)))$
  proof -
    from assms(2) have $\text{int}(\neg x) \subseteq G$
    using neg_in_tgroup lrtrans_in_group_add(2) Top_2_L1 by blast
    with assms(2) have $(x+\text{int}((W+(\neg x)))) = \{x+y. y \in \text{int}(W+(\neg x))\}$
    using lrtrans_image(1) by simp
    moreover from assms have $(-x) \in \text{int}(W+(\neg x))$
    using neg_in_tgroup elem_in_int_lrtrans(1) by simp
    ultimately have $x+(\neg x) \in x+\text{int}(\neg x))$ by auto
    with assms show thesis using group0_2_L6 neg_in_tgroup lrtrans_in_group_add(2)
    lrtrans_interior
    by simp
  qed
ultimately show $x+(W+(\neg x)) \in N_0$ by simp
from assms(2) have $(x+W)+(-x) \subseteq G$ using lrtrans_in_group_add(2) neg_in_tgroup

1315
by simp
moreover have 0 ∈ int((x+W)+(-x))
proof -
  from assms(2) have int((x+W)) ⊆ G using lrtrans_in_group_add(1) Top_2_L1
  by blast
  with assms(2) have int(x+W) + (-x) = {y+(-x). y ∈ int(x+W)}
  using neg_in_tgroup lrtrans_image(2) by simp
moreover from assms have x ∈ int(x+W) using elem_in_int_ltrans(1)
  by simp
  ultimately have x+(-x) ∈ int(x+W) + (-x) by auto
  with assms(2) have 0 ∈ int(x+W) + (-x) using group0_2_L6 by simp
  with assms show thesis using group0_2_L6 neg_in_tgroup lrtrans_in_group_add(1)
  by auto
  qed
ultimately show (x+W)+(-x) ∈ N_0 by simp
qed

If A is a subset of B translated by −x then its translation by x is a subset of B.

lemma (in topgroup) trans_subset:
  assumes A ⊆ ((-x)+B) x ∈ G B ⊆ G
  shows x+A ⊆ B
proof-
  from assms(1) have x+A ⊆ (x+ ((-x)+B)) by auto
  with assms(2,3) show x+A ⊆ B
  using neg_in_tgroup trans_comp_image group0_2_L6 trans_neutral image_id_same
  by simp
  qed

Every neighborhood of zero has a symmetric subset that is a neighborhood of zero.

theorem (in topgroup) exists_sym_zerohood:
  assumes U ∈ N_0
  shows ∃ V ∈ N_0. (V ⊆ U ∧ (-V)=V)
proof
  let V = U ∩ (-U)
  have U ⊆ G using assms unfolding zerohoods_def by auto
  then have V ⊆ G by auto
  have invg: GroupInv(G, f) ∈ G → G using group0_2_T2 Ggroup by auto
  have invb: GroupInv(G, f) ∈ bij(G, G) using group_inv_bij(2) by auto
  have (-V)=GroupInv(G, f)-V unfolding setninv_def using inv_image_vimage
    by auto
  also have ...=(GroupInv(G, f)-U)∩(GroupInv(G, f)-(-U)) using invim_inter_inter_invim
  invg
    by auto
  also have ...=(-U)∩(GroupInv(G, f)-(GroupInv(G, f)∪U))
    unfolding setninv_def using inv_image_vimage by auto
also from \( \langle U \subseteq G \rangle \) have \( \ldots = (-U) \cap U \) using \texttt{inj_vimage_image invb unfolding bij_def} by auto

finally have \((-V) = V\) by auto

then show \( V \subseteq U \land (-V) = V \) by auto

from asssms have \((-U) \in N_0\) using \texttt{neg_neigh_neigh} by auto

with asssms have \( 0 \in \text{int}(U) \cap \text{int}(-U) \) unfolding \texttt{zerhoods_def} by auto

ultimately have \( 0 \in \text{int}(V) \) by \( \text{(rule set_mem_eq)} \)

with \( \langle V \subseteq G \rangle \) show \( V \in N_0 \) using \texttt{zerhoods_def} by auto

qed

We can say even more than in \texttt{exists_sym_zerohood}: every neighborhood of zero \( U \) has a symmetric subset that is a neighborhood of zero and its set double is contained in \( U \).

\begin{verbatim}
theorem (in topgroup) \texttt{exists_procls_zerohood}: 
asssms \( U \in N_0 \) shows \( \exists V \in N_0. (V \subseteq U \land (V+V) \subseteq U \land (-V)=V) \)
proof
have \( \text{int}(U) \in T \) using \texttt{Top_2_L2} by auto
then have \( \mathcal{f}^{-1}(\text{int}(U)) \in \tau \) using \texttt{fcon IsContinuous_def} by auto
moreover have \( \mathcal{f}\langle 0, 0 \rangle = 0 \) using \texttt{group0_2_L2} by auto
moreover have \( 0 \in \text{int}(U) \) using asssms unfolding \texttt{zerhoods_def} by auto
then have \( \mathcal{f}^{-1}\{0\} \subseteq \mathcal{f}^{-1}(\text{int}(U)) \) using \texttt{func1_1_L8 vimage_def} by auto
then have \( \text{GroupInv}(G, \mathcal{f}) \subseteq \mathcal{f}^{-1}(\text{int}(U)) \) using \texttt{group0_2_T3} by auto
then have \( \langle 0, 0 \rangle \in \mathcal{f}^{-1}(\text{int}(U)) \) using \texttt{zero_in_tgroup unfolding GroupInv_def} by auto
ultimately obtain \( W \in T \) and \( V \in T \) and \( \text{cartsub:} W \times V \subseteq \mathcal{f}^{-1}(\text{int}(U)) \) and \( \text{zerhood:} \langle 0, 0 \rangle \in W \times V \)
using \texttt{prod_top_point_neighb topSpaceAssum unfolding prodtop_def by force}
then have \( 0 \in W \) and \( 0 \in V \) by auto
then have \( 0 \in W \cap V \) by auto
have sub:\( W \cap V \subseteq G \) using \( W \cap V \subseteq \mathcal{f}^{-1}(\text{int}(U)) \) and \( \text{zerhood:} \langle 0, 0 \rangle \in W \times V \)
have assoc:\( f \in G \times G \rightarrow G \) using \texttt{group_oper_fun} by auto
{ fix \( t \) assume \( t \in W \cap V \) and \( s \in W \cap V \)
then have \( t \in W \) and \( s \in W \) by auto
then have \( (t,s) \in W \times V \) by auto
then have \( (t,s) \in f^{-1}(\text{int}(U)) \) using \texttt{cartsub by auto}
then have \( f(t,s) \in \text{int}(U) \) using \texttt{func1_1_L15 assoc by auto}
} hence \( \{f(t,s). \langle t,s \rangle \in (W \cap V) \times (W \cap V) \subseteq \text{int}(U) \} \) by auto
then have \( (W \cap V) \times (W \cap V) \subseteq \text{int}(U) \)
unfolding \texttt{setadd_def using lift_subsets_explained(4) assoc sub by auto}
then have \( (W \cap V) \times (W \cap V) \subseteq \text{int}(U) \)
using \texttt{Top_2_L1} by auto
from \texttt{topSpaceAssum} have \( W \cap V \subseteq T \) using \( W \cap V \subseteq T \) unfolding \texttt{IsATopology_def}
}\end{verbatim}
89.4 Closure in topological groups

This section is devoted to a characterization of closure in topological groups.

Closure of a set is contained in the sum of the set and any neighborhood of zero.

```isar
lemma (in topgroup) cl_contains_zneigh:
  assumes A1: A ⊆ G and A2: H ∈ \mathcal{N}_0
  shows cl(A) ⊆ A+H
proof
  fix x assume x ∈ cl(A)
  from A1 have cl(A) ⊆ G using Top_3_L11 by simp
  with \langle x ∈ cl(A) \rangle have x∈G by auto
  have V = int(x + (-H)) using Top_2_L2 by auto
  have V = x + (-int(H))
  proof
    from A2 \langle x∈G \rangle have V = x + int(-H)
    using neg_neigh_neigh ltrans_interior by simp
    with A2 show thesis using int_inv_inv_int by simp
  qed
  have A\cap V ≠ 0
  proof
    from A2 \langle x∈G \rangle have V∈\mathcal{T} and x ∈ cl(A) ∩ V
    using neg_neigh_neigh elem_in_int_ltrans(1) Top_2_L2 by auto
    with A1 show A\cap V ≠ 0 using cl_inter_neigh by simp
  qed
  then obtain y where y∈A and y∈V by auto
  with \langle V = x + (-int(H)) \rangle \langle int(H) ⊆ G \rangle \langle x∈G \rangle have x ∈ y+int(H)
  using ltrans_inv_in by simp
```

1318
with \( y \in A \) have \( x \in \bigcup \{ y \in A \mid y + H \} \) using Top_2_L1 func1_1_L8 by auto
with assms show \( x \in A + H \) using interval_add(3) by simp
qed

The next theorem provides a characterization of closure in topological groups in terms of neighborhoods of zero.

**Theorem (in topgroup) cl_topgroup:**

assumes \( A \subseteq G \)

shows \( \text{cl}(A) = \bigcap_{H \in \mathcal{N}_0} (A + H) \)

proof

from assms show \( \text{cl}(A) \subseteq \bigcap_{H \in \mathcal{N}_0} (A + H) \)

using zneigh_not_empty cl_contains_zneigh by auto

next

\{ fix \( x \) assume \( x \in \bigcap_{H \in \mathcal{N}_0} (A + H) \)

then have \( x \in A + G \)

using zneigh_not_empty by auto

with assms have \( x \in G \)

by simp

have \( \forall U \in \mathcal{T}. \ x \in U \rightarrow U \cap A \neq \emptyset \)

proof -

\{ fix \( U \) assume \( U \in \mathcal{T} \) and \( x \in U \)

let \( H = -((-x)+U) \)

from \( U \in \mathcal{T} \) and \( x \in U \) have \( (-x)+U \subseteq G \) and \( H \in \mathcal{N}_0 \)

using open_trans_neigh neg_neigh_neigh by auto

with \( x \in \bigcap_{H \in \mathcal{N}_0} (A + H) \) have \( x \in A + H \) by auto

with assms \( H \in \mathcal{N}_0 \) obtain \( y \) where \( y \in A \) and \( x \in y + H \)

using interval_add(3) by auto

have \( y \in U \)

proof -

from assms \( y \in A \) have \( y \in G \) by auto

with \( (-x)+U \subseteq G \) and \( x \in y + H \) have \( y \in x + ((-x)+U) \)

using ltrans_inv_in by simp

with \( U \in \mathcal{T} \) \( x \in G \) show \( y \in U \)

using neg_in_tgroup trans_comp_image group0_2_L6 trans_neutral

image_id_same

by auto

qed

with assms \( y \in A \) have \( U \cap A \neq \emptyset \) by auto

\} thus thesis by simp

with assms \( x \in A \) have \( x \in \text{cl}(A) \) using inter_neigh_cl by simp

\} thus \( \bigcap_{H \in \mathcal{N}_0} (A + H) \subseteq \text{cl}(A) \) by auto

qed

### 89.5 Sums of sequences of elements and subsets

In this section we consider properties of the function \( G^n \rightarrow G, x = (x_0, x_1, \ldots, x_{n-1}) \mapsto \sum_{i=0}^{n-1} x_i. \) We will model the cartesian product \( G^n \) by the space of sequences \( n \rightarrow G, \) where \( n = \{0,1,\ldots,n-1\} \) is a natural number. This space is equipped with a natural product topology defined in Topology_ZF_3.

Let’s recall first that the sum of elements of a group is an element of the
lemma (in topgroup) sum_list_in_group:
  assumes n ∈ nat and x: succ(n)→G
  shows (∑ x) ∈ G
proof -
  from assms have semigr0(G,f) and n ∈ nat x: succ(n)→G
  using semigr0_valid_in_tgroup by auto
  then have Fold1(f,x) ∈ G by (rule semigr0.prod_type)
  thus (∑ x) ∈ G by simp
qed

In this context x+y is the same as the value of the group operation on the elements x and y. Normally we shouldn’t need to state this as separate lemma.

lemma (in topgroup) grop_def1: shows f⟨x,y⟩ = x+y by simp

Another theorem from Semigroup_ZF theory that is useful to have in the additive notation.

lemma (in topgroup) shorter_set_add:
  assumes n ∈ nat and x: succ(succ(n))→G
  shows (∑ x) = (∑ Init(x)) + (x(succ(n)))
proof -
  from assms have semigr0(G,f) and n ∈ nat x: succ(succ(n))→G
  using semigr0_valid_in_tgroup by auto
  then have Fold1(f,x) = f⟨Fold1(f,Init(x)),x(succ(n))⟩ by (rule semigr0.shorter_seq)
  thus thesis by simp
qed

Sum is a continuous function in the product topology.

theorem (in topgroup) sum_continuous: assumes n ∈ nat
  shows IsContinuous(SeqProductTopology(succ(n),T),T,{⟨x, ∑ x⟩.x ∈ succ(n)→G})
proof -
  note "n ∈ nat"
  moreover have IsContinuous(SeqProductTopology(succ(0),T),T,{⟨x, x(0)⟩. x ∈ 1→G})
  using semigr0_valid_in_tgroup semigr0.prod_of_1elem by simp
  moreover have IsAhomeomorphism(SeqProductTopology(1,T),T,{⟨x, x(0)⟩. x ∈ 1→⋃T})
  using topSpaceAssum singleton_prod_top1 by simp
  ultimately show thesis using IsAhomeomorphism_def by simp
qed

moreover have ∀k∈nat.
  IsContinuous(SeqProductTopology(succ(k),T),T,{⟨x, ∑ x⟩.x ∈ succ(k)→G})
  \implies
  IsContinuous(SeqProductTopology(succ(succ(k)),T),T,{⟨x, ∑ x⟩.x ∈ succ(succ(k))→G})
proof -

{ fix } k assume k ∈ nat
let s = { ⟨x, ∑ x⟩. x ∈ succ(k) → G }
let g = { ⟨p, ⟨s(fst(p)), snd(p)⟩⟩. p ∈ (succ(k) → G) × G }
let h = { ⟨x, ⟨Init(x), x(succ(k))⟩⟩. x ∈ succ(succ(k)) → G }

let ϕ = SeqProductTopology(succ(k), T)
let ψ = SeqProductTopology(succ(succ(k)), T)
assume IsContinuous(ϕ, T, s)
from ‹ k ∈ nat † have s : (succ(k) → G) → G
  using sum_list_in_group ZF_fun_from_total by simp
have h : (succ(succ(k)) → G) → (succ(k) → G) × G
proof -
{ fix x assume x ∈ succ(succ(k)) → G
  with ‹ k ∈ nat † have Init(x) ∈ (succ(k) → G)
    using init_props by simp
  with ‹ k ∈ nat † ‹ x : succ(succ(k)) → G †
    have ⟨ Init(x), x(succ(k)) ⟩ ∈ (succ(k) → G) × G
    using apply_funtype by blast
} then show thesis using ZF_fun_from_total by simp
qed

moreover have g : ((succ(k) → G) × G) → (G × G)
proof -
{ fix p assume p ∈ (succ(k) → G) × G
  hence fst(p) : succ(k) → G and snd(p) ∈ G by auto
  with ‹ s : (succ(k) → G) → G † have ⟨ s(fst(p)), snd(p) ⟩ ∈ G × G
    using apply_funtype by blast
} then show g : ((succ(k) → G) × G) → (G × G) using ZF_fun_from_total
  by simp
qed

moreover have f : G × G → G using topgroup_f_binop by simp
ultimately have f ∘ g ∘ h : (succ(succ(k)) → G) → G using comp_fun
  by blast
from ‹ k ∈ nat † have IsContinuous(ψ, ProductTopology(ϕ, T), h)
  using topSpaceAssum finite_top_prod_homeo IsAhomeomorphism_def
  by simp
moreover have IsContinuous(ProductTopology(ϕ, T), τ, g)
proof -
from topSpaceAssum have
  T { is a topology } ϕ { is a topology } ∪ ϕ = succ(k) → G
  using seq_prod_top_is_top by auto
moreover from ‹ ∪ ϕ = succ(k) → G † ‹ s : (succ(k) → G) → G †
  have s : ∪ ϕ → T by simp
moreover note ‹ IsContinuous(ϕ, T, s) †
moreover from ‹ ∪ ϕ = succ(k) → G †
  have g = { ⟨ p, ⟨ s(fst(p)), snd(p) ⟩ ⟩. p ∈ ∪ ϕ × T } by simp
ultimately have IsContinuous(ProductTopology(ϕ, T), ProductTopology(T, T), g)
  using cart_prod_contl by blast
thus thesis by simp

1321
moreover have IsContinuous(τ,T,f) using fcon by simp
moreover have \{(x,\sum x).x\in succ(succ(k))\rightarrow G\} = f \circ g \circ h
proof
- let d = \{(x,\sum x).x\in succ(succ(k))\rightarrow G\}
  from \langle k\in \text{nat} \rangle have \forall x\in succ(succ(k))\rightarrow G. (\sum x) \in G
    using sum_list_in_group by blast
  then have d:(succ(succ(k))\rightarrow G)\rightarrow G
    using sum_list_in_group ZF_fun_from_total by simp
moreover note \langle f \circ g \circ h :(succ(succ(k))\rightarrow G)\rightarrow G \rangle
moreover have \forall x\in succ(succ(k))\rightarrow G. d(x) = (f \circ g \circ h)(x)
proof
  fix x assume x \in succ(succ(k))\rightarrow G
  then have I: h(x) = \langle \text{Init}(x),x(succ(k)) \rangle
    using ZF_fun_from_tot_val1 by simp
  moreover from \langle k\in \text{nat} \rangle \langle x: succ(succ(k))\rightarrow G \rangle
    have \text{Init}(x): succ(k)\rightarrow G
      using init_props by simp
  moreover from \langle k\in \text{nat} \rangle \langle x: succ(succ(k))\rightarrow G \rangle
    have II: x(succ(k)) \in G
      using apply_funtype by blast
  ultimately have h(x) \in (succ(k)\rightarrow G)\times G by simp
  then have g(h(x)) = \langle \text{fst}(h(x)),\text{snd}(h(x)) \rangle
    using ZF_fun_from_tot_val1 by simp
  with I have g(h(x)) = \langle \text{Init}(x),x(succ(k)) \rangle
    by simp
  with \langle \text{Init}(x): succ(k)\rightarrow G \rangle have g(h(x)) = (\sum x)
    using ZF_fun_from_tot_val1 by simp
  with \langle x: succ(succ(k))\rightarrow G \rangle have f(g(h(x))) = d(x)
    using ZF_fun_from_tot_val1 by simp
moreover from
  \langle h: (succ(succ(k))\rightarrow G)\rightarrow (succ(k)\rightarrow G)\times G \rangle
  \langle g: (succ(k)\rightarrow G)\rightarrow (G\times G) \rangle
  \langle f: (G\times G)\rightarrow \times succ(succ(k))\rightarrow G \rangle
  have (f \circ g \circ h)(x) = f(g(h(x))) by (rule func1_1_L18)
  ultimately show d(x) = (f \circ g \circ h)(x) by simp
qed
ultimately show \{(x,\sum x).x\in succ(succ(k))\rightarrow G\} = f \circ g \circ h
using func_eq by simp
qed
moreover note \langle IsContinuous(\psi,T,f) \rangle
ultimately have IsContinuous(\psi,T,\{(x,\sum x).x\in succ(succ(k))\rightarrow G\})
using comp_cont3 by simp
} thus thesis by simp
qed
ultimately show thesis by (rule ind_on_nat)
90 Topological groups 1

theory TopologicalGroup_ZF_1 imports TopologicalGroup_ZF Topology_ZF_properties_2 begin

This theory deals with some topological properties of topological groups.

90.1 Separation properties of topological groups

The topological groups have very specific properties. For instance, $G$ is $T_0$ iff it is $T_3$.

theorem (in topgroup) cl_point:
  assumes $x \in G$
  shows $\text{cl} \{x\} = (\bigcap H \in \mathcal{N}_0. x+H)$
proof-
  have $c : \text{cl} \{x\} = (\bigcap H \in \mathcal{N}_0. \{x\}+H)$ using cl_topgroup assms by auto
  fix $H$
  assume $H \in \mathcal{N}_0$
  then have $(x)+H=x+H$ using interval_add(3) assms by auto
  with $H \in \mathcal{N}_0$ have $\{x\}+H=x+\{x\}+H \in \mathcal{N}_0$ by auto
  moreover
  fix $H$
  assume $H \in \mathcal{N}_0$
  then have $(x)+H=x+H$ using interval_add(3) assms by auto
  with $H \in \mathcal{N}_0$ have $x+H \in \mathcal{N}_0$ by auto
  ultimately have $\{x\}+H \subseteq x+H \subseteq \mathcal{N}_0$ by auto
  then have $\bigcap H \in \mathcal{N}_0. \{x\}+H) \subseteq \bigcap H \in \mathcal{N}_0. x+H) \subseteq \mathcal{N}_0$ by auto
  with $c$ show $\text{cl} \{x\} = (\bigcap H \in \mathcal{N}_0. x+H)$ by auto
qed

We prove the equivalence between $T_0$ and $T_1$ first.

theorem (in topgroup) neu_closed_imp_T1:
  assumes $\{0\}$ is closed in $T$
  shows $T$ is $T_1$
proof-
\{ 
fix x z assume xG : x ∈ G and zG : z ∈ G and dis : x ≠ z 
then have clx : cl(\{x\}) = (∩H∈N₀. x + H) using cl_point by auto 
\{ 
fix y 
assume y ∈ cl(\{x\}) 
with clx have y ∈ (∩H∈N₀. x + H) by auto 
then have t : ∀H∈N₀. y ∈ x + H by auto 
from y ∈ cl(\{x\}) > xG have yG : y ∈ G using Top_3_L11(1) G_def by auto 
\{ 
fix H 
assume HNeig : H ∈ N₀ 
with t have y ∈ x + H by auto 
then obtain n where y = x + n and n ∈ H unfolding ltrans_def grop_def 
LeftTranslation_def by auto 
with HNeig have nG : n ∈ G unfolding zerohoods_def by auto 
from y = x + n and n ∈ H have (-x) + y ∈ N₀ using group0.group0_2_L18(2) 
group0_valid_in_tgroup xG nG yG unfolding grinv_def grop_def 
by auto 
\} 
then have el : (-x) + y ∈ (∩N₀) using zneigh_not_empty by auto 
have cl(\{0\}) = (∩H∈N₀. 0 + H) using cl_point zero_in_tgroup by auto 
moreover 
\{ 
fix H assume H ∈ N₀ 
then have H ⊆ G unfolding zerohoods_def by auto 
then have 0 + H = H using image_id_same group0.trans_neutral(2) 
group0_valid_in_tgroup unfolding gzero_def ltrans_def 
by auto 
with H ∈ N₀ have 0 + H ∈ {0 + H. H ∈ N₀} by auto 
\} 
then have \{0 + H. H ∈ N₀\} = N₀ by blast 
ultimately have cl(\{0\}) = (∩N₀) by auto 
with el have (-x) + y ∈ cl(\{0\}) by auto 
then have (-x) + y ∈ \{0\} using assms Top_3_L8 G_def zero_in_tgroup 
by auto 
then have (-x) + y = 0 by auto 
then have y = (-x) using group0.group0_2_L9(2) group0_valid_in_tgroup 
eg_in_tgroup xG yG unfolding group_def grinv_def by auto 
then have y = x using group0.group_inv_of_inv group0_valid_in_tgroup 
xG unfolding grinv_def by auto 
\} 
then have cl(\{x\}) ⊆ (x) by auto 
then have cl(\{x\}) = (x) using xG cl_contains_set G_def by blast 
then have \{x\} (is closed in) T using Top_3_L8 xG G_def by auto 
then have (∪T) - (x) ∈ T using IsClosed_def by auto 
moreover from dis zG G_def have z ∈ ((∪T) - (x)) ∧ x ∈ ((∪T) - (x)) by auto 
ultimately have ∃V ∈ T. z ∈ V \ x ∉ V by (safe, auto) 
\} 
}
then show T{is T1} using isT1_def by auto
qed

theorem (in topgroup) T0_imp_neu_closed:
  assumes T{is T0}
  shows {0}{is closed in}T
proof-  
  { fix x assume x∈cl({0}) and x≠0  
    have cl({0})=⋂H∈N0. 0+H by cl_point_zero_in_tgroup by auto  
    moreover  
    { fix H assume H∈N0  
      then have H⊆G unfolding zerohoods_def by auto  
      then have 0+H=H using image_id_same group0_trans_neutral(2) group0_valid_in_tgroup unfolding gzero_def ltrans_def by auto  
    }  
    ultimately have cl({0})=⋂H∈N0. H by auto  
  } 
  then have {0+H. H∈N0}∈N0 by blast  
  ultimately have cl({0})=⋂N0 by auto  
  from x≠0 and x∈cl({0}) obtain U where U∈T and (x∉U∧0∈U)∨(0∉U∧x∈U) using assms Top_3_L11(1) zero_in_tgroup unfolding isT0_def G_def by blast  
  moreover  
  { assume 0∈U  
    with U∈T have U∈N0 using zerohoods_def G_def Top_2_L3 by auto  
  }  
  finally have x∈U by auto  
  ultimately have 0∉U and x∈U by auto  
  with x∈cl({0}) have False using cl_inter_neigh zero_in_tgroup G_def by blast  
  then have cl({0})⊆{0} by auto  
  then have cl({0})={0} using zero_in_tgroup cl_contains_set G_def by blast  
  then show thesis using Top_3_L8 zero_in_tgroup unfolding G_def by auto  
qed

90.2 Existence of nice neighbourhoods.

lemma (in topgroup) exist_basehoods_closed:
  assumes U∈N0
  shows ∃V∈N0. cl(V)⊆U
proof- 
  from assms obtain V where V∈N0 V⊆U (V+V)⊆U (-V)=V using exists_procls_zerohood by blast  
  have inv_fun: GroupInv(G,f)∈G→G using group0_2_T2 Ggroup by auto  
  have f_fun:f∈G×G→G using group0_group_oper_fun group0_valid_in_tgroup
proof

\{ 
\begin{align*}
&\text{fix } x \text{ assume } x \in \mathbb{C}(V) \\
&\quad \text{with } \langle V \in \mathbb{N}_0 \rangle \text{ have } x \in T \subseteq U T \text{ using Top_3_L11(1) unfolding zerohoods_def G_def by blast+} \\
&\quad \text{with } \langle V \in \mathbb{N}_0 \rangle \text{ have } x \in \mathbb{C}(x+V) \text{ using elem_in_int_ltrans G_def by auto} \\
&\quad \text{with } \langle V \subseteq U T, x \in \mathbb{C}(V) \rangle \text{ have } \mathbb{C}(x+V) \cap V \neq 0 \text{ using cl_inter_neigh Top_2_L2 by blast} \\
&\quad \text{then have } (x+V) \cap V \neq 0 \text{ using Top_2_L1 by blast} \\
&\quad \text{then obtain } q \text{ where } q \in (x+V) \text{ and } q \in V \text{ by blast} \\
&\quad \text{with } \langle V \subseteq U T, x \in U T \rangle \text{ obtain } v \text{ where } q = x+V \text{ v } \in V \text{ unfolding ltrans_def group_def using group0.ltrans_image} \\
&\quad \text{group0_valid_in_tgroup unfolding G_def by auto} \\
&\quad \text{from } \langle V \subseteq U T, q \in V \rangle \text{ have } v \in U T q \in U T \text{ by auto} \\
&\quad \text{with } \langle q = x+v, x \in U T \rangle \text{ have } q-v = x \text{ using group0.group0_2_L18(1) group0_valid_in_tgroup unfolding G_def} \\
&\quad \text{unfolding grsub_def grinv_def group_def by auto} \\
&\quad \text{moreover from } \langle V \in V \rangle \text{ have } (-V) \subseteq (-V) \text{ unfolding setinv_def grinv_def using func_image_def} \\
&\quad \text{inv_fun } \langle V \subseteq U T \rangle \text{ G_def by auto} \\
&\quad \text{then have } (-V) = V \text{ using } (-V) = V \text{ by auto} \\
&\quad \text{with } \langle q \in V \rangle \text{ have } (q, -v) \subseteq V \times V \text{ by auto} \\
&\quad \text{then have } f(q, -v) \in V + V \text{ using lift_subset_suff f_fun } \langle V \subseteq U T \rangle \text{ unfolding setadd_def by auto} \\
&\quad \text{with } \langle V + V \subseteq U \rangle \text{ have } q - v \in U \text{ unfolding grsub_def group_def by auto} \\
&\quad \text{with } \langle q - v = x \rangle \text{ have } x \in U \text{ by auto} \\
&\quad \text{then have } \mathbb{C}(V) \subseteq U \text{ by auto} \\
&\quad \text{with } \langle V \in \mathbb{N}_0 \rangle \text{ show thesis by auto} \\
\end{align*}
\}

qed

90.3 Rest of separation axioms

theorem(in topgroup) T1_imp_T2:
passes T\{is T_1\}.
shows T\{is T_2\}.
proof
\{ 
\begin{align*}
&\text{fix } x \text{ y assume } \text{as: } x \in U T \text{ y } \in U T \text{ x } \neq y \\
&\quad \text{assume } (-y) + x = 0 \\
&\quad \text{with } \text{as}(1, 2) \text{ have } y = x \text{ using group0.group0_2_L11[where a=y and b=x] group0_valid_in_tgroup by auto} \\
&\quad \text{with } \text{as}(3) \text{ have } \text{False by auto} \\
&\quad \text{then have } (-y) + x \neq 0 \text{ by auto} \\
&\quad \text{then have } 0 \neq (-y) + x \text{ by auto} \\
&\quad \text{from } \langle y \in U T \rangle \text{ have } (-y) \in U T \text{ using neg_in_tgroup G_def by auto} \\
&\quad \text{with } \langle x \in U T \rangle \text{ have } (-y) + x \in U T \text{ using group0.group_op_closed[where a=y and b=x] group0_valid_in_tgroup unfolding} \\
\end{align*}
\}
\[ G \text{ by auto} \]

with 

assms \(\langle 0 \neq -(y)+x \rangle \) obtain \( U \) where \( U \subseteq T \) and \((y)+x \notin U \) and \( 0 \in U \) unfolding \isT\_1 \_def using \zero\_in\_tgroup

by auto

then have \( U \subseteq N_0 \) unfolding \zerohoods\_def \G\_def using \Top\_2\_L3 by auto

then obtain \( U \) where \( Q \subseteq N_0 \) \( (Q+Q) \subseteq U \) \( (-Q)=Q \) using \exists\text{-procl}_\text{-zerohood}

by blast

with \( \langle -(y)+x \rangle \in U \rangle \) have \( -(y)+x \notin Q \) by auto

from \( \langle Q \subseteq N_0 \rangle \) have \( Q \subseteq G \) unfolding \zerohoods\_def by auto

{ 
assume \( x \in y+Q \)

with \( \langle Q \subseteq G \rangle \langle y \in \bigcup T \rangle \) obtain \( u \) where \( u \in Q \) and \( x=y+u \) unfolding \ltrans\_def

grop\_def using group0.ltrans\_image group0.valid\_in\_tgroup

unfolding \G\_def by auto

with \( \langle Q \subseteq G \rangle \) have \( u \in \bigcup T \) unfolding \G\_def by auto

with \( \langle x=y+u \rangle \langle y \in \bigcup T \rangle \langle x \in \bigcup T \rangle \langle Q \subseteq G \rangle \) have \( -(y)+x=u \) using group0.group0\_2\_L18(2)

group0_valid\_in\_tgroup unfolding \G\_def

unfolding grsub\_def \grinv\_def grop\_def by auto

with \( \langle u \in Q \rangle \) have \( -(y)+x \in Q \) by auto

then have False using \( \langle -(y)+x \notin Q \rangle \) by auto

} then have \( x \notin y+Q \) by auto moreover

{ 
assume \( y \in x+Q \)

with \( \langle Q \subseteq G \rangle \langle x \in \bigcup T \rangle \) obtain \( u \) where \( u \in Q \) and \( y=x+u \) unfolding \ltrans\_def

grop\_def using group0.ltrans\_image group0.valid\_in\_tgroup

unfolding \G\_def by auto

with \( \langle Q \subseteq G \rangle \) have \( u \in \bigcup T \) unfolding \G\_def by auto

with \( \langle x=y+u \rangle \langle y \in \bigcup T \rangle \langle x \in \bigcup T \rangle \langle Q \subseteq G \rangle \) have \( -(y)+x=u \) using group0.group0\_2\_L18(2)

group0_valid\_in\_tgroup unfolding \G\_def

unfolding grsub\_def \grinv\_def grop\_def by auto

with \( \langle u \in Q \rangle \) have \( -(y)+x \in Q \) by auto

then have \( x \in y+Q \) using \( \langle x \in \bigcup T \rangle \langle y \in \bigcup T \rangle \langle u \in Q \rangle \) by auto

moreover from \( \langle u \in Q \rangle \) have \( (-u) \in (-Q) \) unfolding \setn\_inv\_def \grinv\_def

using func\_image\_def [OF group0.2_T2 [OF Ggroup] \( \langle Q \subseteq G \rangle \)] by auto

ultimately have \( -(y)+x \in Q \) using \( \langle -(y)+x \notin Q \rangle \langle -Q)=Q \) unfolding \setn\_inv\_def

ginv\_def by auto

then have False using \( \langle -(y)+x \notin Q \rangle \) by auto

} then have \( y \notin x+Q \) by auto moreover

{ 
fix \( t \)

assume \( t \in (x+Q) \cap (y+Q) \)

then have \( t \in (x+Q) \) \( t \in (y+Q) \) by auto

with \( \langle Q \subseteq G \rangle \langle x \in \bigcup T \rangle \langle y \in \bigcup T \rangle \) obtain \( u \) \( v \) where \( u \in Q \) \( v \in Q \) and \( t=x+u \)

\( t=y+v \) unfolding \ltrans\_def grop\_def using group0.ltrans\_image [OF group0.valid\_in\_tgroup]

unfolding \G\_def by auto

1327
then have \( x+u = y+v \) by \textit{auto}.

moreover from \( \langle u \in Q \rangle \langle v \in Q \rangle \langle Q \subseteq G \rangle \) have \( u \in \bigcup T \ v \in \bigcup T \) unfolding \textit{G_def}

by \textit{auto}

moreover note \( \langle x \in \bigcup T \rangle \langle y \in \bigcup T \rangle \)

ultimately have \( (-y)+(x+u) = v \) using \textit{group0.group0_2_L18(2)[OF group0_valid_in_tgroup, of y v x+u] group0.group_op_closed[OF group0_valid_in_tgroup, of x u]}

unfolding \textit{G_def}

unfolding \textit{grsub_def grinv_def grop_def} by \textit{auto}

moreover note \( \langle x \in \bigcup T \rangle \langle y \in \bigcup T \rangle \langle u \in \bigcup T \rangle \)

ultimately have \( (-y)+x+u = v \) using \textit{group0.group_oper_assoc[OF group0_valid_in_tgroup]}

unfolding \textit{grop_def}

unfolding \textit{grsub_def grinv_def grop_def} by \textit{force}

moreover from \( \langle u \in Q \rangle \) have \( (-u) \in (-Q) \) unfolding \textit{setninv_def grinv_def}

using \textit{func_imagedef[OF group0_2_T2[OF Ggroup] \langle Q \subseteq G \rangle]} by \textit{auto}

then have \( (-u) \in (-Q) \) using \textit{setadd_def} by \textit{auto}

ultimately have \( (-y)+x \in U \) using \textit{isT2_def} by \textit{auto}

qed

Here follow some auxiliary lemmas.

\textbf{lemma (in topgroup) trans_closure:}

assumes \( x \in G \ A \subseteq G \)

shows \( cl(x+A) = x + cl(A) \)

\textbf{proof-}

have \( \bigcup T - (\bigcup T - (x+A)) = (x+A) \) unfolding \textit{ltrans_def} using \textit{group0.group0_5_L1(2)[OF group0_valid_in_tgroup assms(1)]]}

unfolding \textit{image_def range_def domain_def converse_def Pi_def} by \textit{auto}

1328
then have cl(x+A)= ∪ T-int(∪ T-(x+A)) using Top_3_L11(2) [of ∪ T-(x+A)]
by auto
moreover have x+G=G using surj_image_eq group0.trans_bij(2) [OF group0_valid_in_tgroup
assms(1)] bij_def by auto
then have ∪ T-(x+A)=x+(∪ T-A) using inj_image_dif [of LeftTranslation(G, f, x)GG, OF _ assms(2)]
unfolding ltrans_def G_def using group0.trans_bij(2) [OF group0_valid_in_tgroup
assms(1)] bij_def by auto
then have ∪ T-(x+G)=x+∪ T-A using ltrans_interior [OF assms(1), of ∪ T-A] unfolding ltrans_def
by auto
have ∪ T-int(∪ T-A)=cl(∪ T-(∪ T-A)) using Top_3_L11(2) [of ∪ T-A] by force
have ∪ T-(∪ T-A)=A using assms(2) G_def
by auto
with ‹cl(x+A)= ∪ T-int(∪ T-(x+A))› show thesis by auto
qed

lemma (in topgroup) trans_interior2: assumes A1: g ∈ G and A2: A ⊆ G
shows int(A)+g = int(A+g)
proof -
  from assms have A ⊆ ∪ T and IsAhomeomorphism(T,T,RightTranslation(G,f,g))
  using tr_homeo by auto
  then show thesis using int_top_invariant by simp
qed

lemma (in topgroup) trans_closure2: assumes x∈G A ⊆ G
shows cl(A+x)=cl(A)+x
proof-
  have ∪ T-(∪ T-(A+x))=(A+x) unfolding ltrans_def using group0.group0_5_L1(1) [OF
  group0_valid_in_tgroup assms(1)] unfolding image_def range_def domain_def converse_def Pi_def by auto
then have cl(A+x)= ∪ T-int(∪ T-(A+x)) using Top_3_L11(2) [of ∪ T-(A+x)]
by auto
moreover have G+x=G using surj_image_eq group0.trans_bij(1) [OF group0_valid_in_tgroup
\textbf{assms(1)} bij_def by auto
then have \( \bigcup T-(A+x) = \bigcup T-A + x \)
using inj_image_dif[of RightTranslation(G, f, x)GG, OF _ assms(2)] unfolding rtrans_def G_def by auto
then have \( \mathit{int(\bigcup T-(A+x)}) = \mathit{int(\bigcup T-A + x)} \)
using trans_interior2[OF assms(1),of \( \bigcup T-A \)] unfolding G_def by force
have \( \bigcup T-\mathit{int(\bigcup T-A)} = \mathit{cl(\bigcup T-(\bigcup T-A))} \)
using Top_3_L11(2)[of \( \bigcup T-A \)] by force
have \( \bigcup T-\mathit{int(\bigcup T-A)} = A \)
using assms(2) G_def by auto
with \( \mathit{cl(A+x)} = \bigcup T-\mathit{int(\bigcup T-A)} \)
show thesis by auto
qed

\textbf{lemma (in topgroup) trans_subset:}
assumes \( A \subseteq (-x)+B \) \( x \in G \subseteq GB \subseteq G \)
sshows \( x+A \subseteq B \)
proof-
\{ 
fix t assume t \in x+A
with \( x \in G \) \( A \subseteq GB \) obtain u where u \in A t=x+u unfolding ltrans_def grop_def using group0.ltrans_image[OF group0_valid_in_tgroup]
unfolding G_def by auto
with \( x \in G \) \( A \subseteq GB \) \( u \in A \) have \( (-x)+t = u \)
using group0.group0_2_L18(2)[OF group0_valid_in_tgroup, of x u t]
group0.group_op_closed[OF group0_valid_in_tgroup, of \( x+u \)] unfolding grop_def grunt_def by auto
with \( u \in A \) have \( (-x)+t \in A \)
by auto
with \( A \subseteq (-x)+B \) have \( (-x)+t \in (-x)+B \)
by auto
with \( B \subseteq G \) obtain v where \( (-x)+t \in (-x)+v \) \( v \in B \)
unfolding ltrans_def grop_def using neg_in_tgroup[OF \( x \in G \)] group0.ltrans_image[OF group0_valid_in_tgroup]
unfolding G_def by auto
have LeftTranslation(G,f,-x) \in inj(G,G) using group0.trans_bij(2)[OF group0_valid_in_tgroup neg_in_tgroup[OF \( x \in G \)]] bij_def by auto
then have eq:\( \forall A \subseteq G. \forall B \subseteq G. \) LeftTranslation(G,f,-x)A=LeftTranslation(G,f,-
x)B $\rightarrow$ A=B unfolding inj_def by auto

{  
  fix A B assume A\in G B\in G  
  assume f(-x,A)=f(-x,B)  
  then have LeftTranslation(G,f,-x)A=LeftTranslation(G,f,-x)B using group0.group0_5_L2(2) [OF group0_valid_in_tgroup neg_in_tgroup [OF \langle x\in G \rangle]]  
  \langle A\in G \rangle \langle B\in G \rangle by auto  
  with eq \langle A\in G \rangle \langle B\in G \rangle have A=B by auto  
}
then have eq1: $\forall A \in G. \forall B \in G. f(-x,A) = f(-x,B) \rightarrow A=B$ by auto

from \langle A \subseteq G \rangle \langle u \in A \rangle have u \in G by auto  
with \langle v \in B \rangle \langle B \subseteq G \rangle \langle t=x+u \rangle have t \in G v \in G using group0.group_op_closed [OF group0_valid_in_tgroup \langle x \in G \rangle, of u] unfolding G_def by auto  
with eq1 \langle (-x)+t=(-x)+v \rangle have t=v unfolding G_def by auto  
with \langle v \in B \rangle have t \in B by auto
	hen then show thesis by auto
qed

Every topological group is regular, and hence $T_3$. The proof is in the next section, since it uses local properties.

90.4 Local properties

In a topological group, all local properties depend only on the neighbourhoods of the neutral element; when considering topological properties. The next result of regularity, will use this idea, since translations preserve closed sets.

lemma (in topgroup) local_iff_neutral:  
  assumes $\forall U \in T \cap N_0. \exists N \in N_0. N \subseteq U \land P(N,T) \land \forall x \in G. P(N,T) \rightarrow P(x+N,T)$  
  shows $T\{is locally}\ P$
proof-  
{  
  fix x U assume x\in U \cup T U \subseteq T x \in U  
  then have \langle-x\rangle+U \subseteq T \setminus N_0 using open_tr_open(1) open_trans_neigh neg_in_tgroup unfolding G_def by auto  
  with assms(1) obtain N where N\subseteq((-x)+U)P(N,T)N\in N_0 by auto  
  note \langle x\in U \rangle \langle N\subseteq((-x)+U) \rangle moreover  
  from \langle U\in T \rangle have U\subseteq U by auto moreover  
  from \langle N\in N_0 \rangle have N\subseteq G unfolding zerohoods_def by auto  
  ultimately have \langle x+N \subseteq U \rangle using trans_subset unfolding G_def by auto moreover  
  from \langle N\subseteq G \rangle \langle x \in U \rangle assms(2) \langle P(N,T) \rangle have P((x+N),T) using P(N,T) unfolding G_def by auto moreover

1331
from \( \langle N \in \mathbb{N}_0 \rangle \langle x \in \bigcup T \rangle \) have \( x \in \text{int}(x+N) \) using \text{elem_in_int_ltrans} unfolding \( \text{G_def} \) by auto
ultimately have \( \exists N \in \text{Pow}(U) \). \( x \in \text{int}(N) \land P(N,T) \) by auto
} then show thesis unfolding \( \text{IsLocally_def[OF topSpaceAssum]} \) by auto qed

lemma (in topgroup) trans_closed:
assumes \( A \) (is closed in) \( Tx \in G \)
shows \( (x+A) \) (is closed in) \( T \)
proof-
from assms(1) have \( \text{cl}(A) = A \) using \text{Top_3_L8} unfolding \( \text{IsClosed_def} \) by auto
then have \( x + \text{cl}(A) = x + A \) by auto
then have \( \text{cl}(x+A) = x + A \) using \text{trans_closure assms} unfolding \( \text{IsClosed_def} \) by auto
moreover have \( x + A \subseteq G \) unfolding \( \text{ltrans_def} \) using \text{group0.group0_5_L1(2)[OF group0_valid_in_tgroup \( x \in G \):]}
unfolding \( \text{image_def range_def domain_def converse_def Pi_def} \) by auto
ultimately show thesis using \text{Top_3_L8} unfolding \( \text{G_def} \) by auto
qed

As it is written in the previous section, every topological group is regular.

theorem (in topgroup) topgroup_reg:
shows \( T \) (is regular)
proof-
{ fix \( U \) assume \( U \in T \setminus \mathcal{N}_0 \)
then obtain \( V \) where \( \text{cl}(V) \subseteq UV \subseteq \mathcal{N}_0 \) using \text{exist_basehoods_closed} by blast
then have \( V \subseteq \text{cl}(V) \) using \text{cl_contains_set} unfolding \( \text{zerohoods_def G_def} \) by auto
then have \( \text{int}(V) \subseteq \text{int(cl(V))} \) using \text{interior_mono} by auto
with \( \langle V \in \mathcal{N}_0 \rangle \) have \( \text{cl}(V) \in \mathcal{N}_0 \) unfolding \( \text{zeroshoods_def G_def} \) using \text{Top_3_L11(1)} by auto
from \( \langle V \in \mathcal{N}_0 \rangle \) have \( \text{cl}(V) \) (is closed in) \( T \) using \text{cl_is_closed unfolding zeroshoods_def G_def} by auto
with \( \langle cl(V) \in \mathcal{N}_0 \rangle \langle cl(V) \subseteq U \rangle \) have \( \exists N \in \mathcal{N}_0 \). \( N \subseteq \text{Ann}(\text{is closed in})T \) by auto
} then have \( \forall U \in T \setminus \mathcal{N}_0 \). \( \exists N \in \mathcal{N}_0 \). \( N \subseteq \text{Ann}(\text{is closed in})T \) by auto moreover have \( \forall N \in \text{Pow}(G). ( \forall x \in G. \ (N \text{ is closed in})T \rightarrow (x+N) \text{ is closed in} T) \)
using \text{trans_closed by auto}
ultimately have \( T \) (is locally-closed) using \text{local_iff_neutral unfolding IsLocallyClosed_def by auto}
then show \( T \) (is regular) using \text{regular_locally_closed by auto}
qed

The promised corollary follows:
**91.1 Natural uniformities in topological groups: definitions and notation**

There are two basic uniformities that can be defined on a topological group.

**Definition of left uniformity**

\[
\text{definition (in topgroup) leftUniformity} \\
\text{where leftUniformity} \equiv \{V \in \text{Pow}(G \times G). \exists U \in N_0. \{(s,t) \in G \times G. (-s)+t \in U\} \subseteq V\}
\]

**Definition of right uniformity**

\[
\text{definition (in topgroup) rightUniformity} \\
\text{where rightUniformity} \equiv \{V \in \text{Pow}(G \times G). \exists U \in N_0. \{(s,t) \in G \times G. s+(-t) \in U\} \subseteq V\}
\]

Right and left uniformities are indeed uniformities.

**Lemma (in topgroup) side_uniformities:**

shows leftUniformity \(\text{is a uniformity on}\) \(G\) and rightUniformity \(\text{is a uniformity on}\) \(G\)

**Proof:**

\[
\begin{align*}
\text{assume } 0 & \in \text{leftUniformity} \\
\text{then obtain } U & \text{ where } U:U \in N_0 \{(s,t) \in G \times G. (-s)+t \in U\} \subseteq 0 \text{ unfolding leftUniformity_def} \\
\text{by auto} \\
\text{have } (0,0):G \times G & \text{ using zero_in_tgroup by auto} \\
\text{moreover have } (-0)+0 & = 0 \\
& \text{ using group0_valid_in_tgroup group0.group_inv_of_one group0.group0_2_L2} \\
\text{zero_in_tgroup} & \text{ by auto} \\
\text{moreover have } 0 & \in \text{int}(U) \text{ using U(1) by auto} \\
\text{then have } 0 & \in U \text{ using Top_2_L1 by auto}
\end{align*}
\]
ultimately have \( \{0,0\} \subseteq \{(s,t) \in G \times G. \ (-s)+t \in U\} \) by auto

with \( U(2) \) have \( \{0,0\} \subseteq 0 \) by blast

hence False by auto

\}

hence \( 0 \notin \text{leftUniformity} \) by auto

moreover have \( \text{leftUniformity} \subseteq \text{Pow}(G \times G) \) unfolding \text{leftUniformity_def} by auto

moreover

\{

have \( G \times G \subseteq \text{Pow}(G \times G) \) by auto

moreover have \( \{(s,t) : G \times G. \ (-s)+t : G\} \subseteq G \times G \) by auto

note \text{zneigh_not_empty}

ultimately have \( G \times G \subseteq \text{leftUniformity} \) unfolding \text{leftUniformity_def} by auto

moreover

\{

fix A B assume as: \( A \in \text{leftUniformity} \) \( B \in \text{leftUniformity} \)

from as(1) obtain \( AU \) where \( AU: AU \subseteq N_0 \{(s,t) \in G \times G. \ (-s)+t \in AU\} \subseteq A \)

\( A \in \text{Pow}(G \times G) \)

unfolding \text{leftUniformity_def} by auto

from as(2) obtain \( BU \) where \( BU: BU \subseteq N_0 \{(s,t) \in G \times G. \ (-s)+t \in BU\} \subseteq B \)

\( B \in \text{Pow}(G \times G) \)

unfolding \text{leftUniformity_def} by auto

from \( AU(1) \) \( BU(1) \) have \( 0 \in \text{int}(AU) \cap \text{int}(BU) \) by auto

moreover from \( AU \) \( BU \) have \( \text{op}: \text{int}(AU) \cap \text{int}(BU) \subseteq T \) using \text{Top_2_L2 topSpaceAssum IsATopology_def}

by auto

moreover

have \( \text{int}(AU) \cap \text{int}(BU) \subseteq \text{AU} \cap \text{BU} \) using \text{Top_2_L1 by auto}

with \( \text{op} \) have \( \text{int}(AU) \cap \text{int}(BU) \subseteq \text{int}(AU \cap BU) \) using \text{Top_2_L5 by auto}

moreover note \( AU(1) \) \( BU(1) \)

ultimately have \( AU \cap BU: N_0 \) unfolding \text{zerohoods_def} by auto

moreover have \( \{(s,t) \in G \times G. \ (-s)+t \in AU \cap BU\} \subseteq \{(s,t) \in G \times G. \ (-s)+t \in AU\} \)

by auto

with \( AU(2) \) \( BU(2) \) have \( \{(s,t) \in G \times G. \ (-s)+t \in AU \cap BU\} \subseteq A \cap B \) by auto

ultimately have \( A \cap B \in \{V \in \text{Pow}(G \times G). \exists U \in N_0. \{(s,t) \in G \times G. \ (-s)+t \in U\} \subseteq V\} \)

using \( AU(3) \) \( BU(3) \) by blast

then have \( A \cap B \in \text{leftUniformity} \) unfolding \text{leftUniformity_def} by simp

\}

hence \( \forall A \in \text{leftUniformity}. \ \forall B \in \text{leftUniformity}. \ A \cap B \in \text{leftUniformity} \) by auto

moreover

\{

fix B C assume as: \( B \in \text{leftUniformity} \) \( C \in \text{Pow}(G \times G) \) \( B \subseteq C \)

from as(1) obtain \( BU \) where \( BU: BU \subseteq N_0 \{(s,t) \in G \times G. \ (-s)+t \in BU\} \subseteq B \)

1334
unfolding leftUniformity_def by blast 
from as(3) BU(2) have \{s,t\}\in G\times G. (-s)+t \in BU \subseteq C by auto 
with as(2) BU(1) have C \subseteq \{V\in Pow(G\times G). \exists U\in N_0. \{s,t\}\in G\times G. (-s)+t 
\in U\} by auto 
then have C \in leftUniformity unfolding leftUniformity_def by auto  
} 
then have \forall B\in leftUniformity. \forall C\in Pow(G\times G). B \subseteq C \rightarrow C \in leftUniformity 
by auto 
ultimately have leftFilter: leftUniformity \{is a filter on\} (G\times G) unfolding IsFilter_def 
by auto 
{ 
assume 0\in rightUniformity 
then obtain U where U:U \subseteq 0 unfolding rightUniformity_def 
by auto 
have (0,0):G\times G using zero_in_tgroup by auto 
moreover have 0+(-0) = 0 
using group0_valid_in_tgroup group0.group_inv_of_one group0.group0_2_L2 
zero_in_tgroup 
by auto 
moreover 
have 0\in int(U) using U(1) by auto 
then have 0\in U using Top_2_L1 by auto 
ultimately have (0,0)\in\{s,t\}\in G\times G. s+(-t) \in U\} by auto 
with U(2) have (0,0)\in 0 by blast 
hence False by auto 
} 
then have 0\notin rightUniformity by auto 
moreover have rightUniformity \subseteq Pow(G\times G) unfolding rightUniformity_def 
by auto 
moreover 
{ 
have G\times G\in Pow(G\times G) by auto 
moreover have \{(-s)+t:G\} \subseteq G\times G by auto 
moreover note zneigh_not_empty 
ultimately have G\times G \in rightUniformity unfolding rightUniformity_def 
by auto 
} 
moreover 
{ 
fix A B assume as:A\in rightUniformity B\in rightUniformity 
from as(1) obtain AU where AU:AU \subseteq N_0 \{s,t\}\in G\times G. s+(-t) \in AU\subseteq A 
A\in Pow(G\times G) 
unfolding rightUniformity_def by auto 
from as(2) obtain BU where BU:BU \subseteq N_0 \{s,t\}\in G\times G. s+(-t) \in BU\subseteq B 
B\in Pow(G\times G) 
unfolding rightUniformity_def by auto
from AU(1) BU(1) have 0 \in \text{int}(AU) \cap \text{int}(BU) \text{ by auto} 
moreover from AU BU have \text{op: int}(AU) \cap \text{int}(BU) \subseteq T 
using Top_2_L2 topSpaceAssum IsATopology_def by auto 
moreover have \text{int}(AU) \cap \text{int}(BU) \subseteq AU \cap BU \text{ using Top_2_L1 by auto} 
moreover note AU(1) BU(1) 
ultimately have AU \cap BU: N_0 \text{ unfolding zerohoods_def by auto} 
moreover have \{ (s,t) \in G \times G. s+(-t) \in AU \cap BU \} \subseteq V \text{ by auto} 
with AU(2) BU(2) have \{ (s,t) \in G \times G. s+(-t) \in AU \cap BU \} \subseteq A \cap B \text{ by auto} 
ultimately have A \cap B \in \{ V \in \text{Pow}(G \times G). \exists U \in N_0. \{ (s,t) \in G \times G. s+(-t) \in U \} \subseteq V \} 
using AU(3) BU(3) by blast 
then have A \cap B \in \text{rightUniformity unfolding rightUniformity_def by simp} 
hence \forall A \in \text{rightUniformity}. \forall B \in \text{rightUniformity}. A \cap B \in \text{rightUniformity} 
by auto 
moreover 
\{ 
\text{fix B C assume as:B \in \text{rightUniformity} C \in \text{Pow}(G \times G) B \subseteq C} 
from as(1) obtain BU where BU:BU \in N_0 \{ (s,t) \in G \times G. s+(-t) \in BU \} \subseteq B 
\text{ unfolding rightUniformity_def by blast} 
from as(3) BU(2) have \{ (s,t) \in G \times G. s+(-t) \in BU \} \subseteq C \text{ by auto} 
then have C \in \text{rightUniformity using as(2) BU(1) unfolding rightUniformity_def by auto} 
\text{ then have } \forall B \in \text{rightUniformity. } \forall C \in \text{Pow}(G \times G). B \subseteq C \rightarrow C \in \text{rightUniformity} 
by auto 
ultimately have \text{rightFilter: rightUniformity \{ is a filter on \} } (G \times G) 
unfolding IsFilter_def 
by auto 
\{ 
\text{fix U assume as:U \in \text{leftUniformity}} 
from as obtain V where V:V \in N_0 \{ (s,t) \in G \times G. (-s)+t \in V \} \subseteq U 
\text{ unfolding leftUniformity_def by auto} 
then have V \subseteq G \text{ by auto} 
\{ 
\text{fix x assume as: x \in id(G)} 
from as obtain V where V:V \in N_0 \{ (s,t) \in G \times G. (-s)+t \in V \} \subseteq U 
\text{ unfolding leftUniformity_def by auto} 
from V(1) have 0 \in \text{int}(V) \text{ by auto} 
then have V0:0 \in V \text{ using Top_2_L1 by auto} 
from as2 obtain t where t:V=x=(t,t) t:G \text{ by auto} 
from t(2) have \(-t)+t =0 \text{ using group0_valid_in_tgroup group0.group0_2_L6 by auto} 
with V0 t V(2) have x \in U \text{ by auto}
then have \( \text{id}(G) \subseteq U \) by auto
moreover
\{
\begin{align*}
\text{fix } x & \text{ assume } \text{ass}: x \in \{ (s,t) \in G \times G. (\neg s)+t \in -V \} \\
\text{then obtain } s \ t & \text{ where } \\text{as}: s \in G, t \in G, (\neg s)+t \in -V \text{ by force from as(3) } \langle \forall G \rangle \text{ have } (\neg s)+t \in \{ -q. q \in V \} \text{ using ginv_image_add by simp} \\
\text{then obtain } q & \text{ where } q \in V \ (\neg s)+t = -q \text{ by auto with } \langle V \subseteq G \rangle \text{ have } q \in G \text{ by auto with } \langle s \in G \rangle \langle t \in G \rangle \langle -s)+t = -q \rangle \text{ have } q = (-t)+s \text{ using simple_equation1_add by blast} \\
\text{ultimately have } & \langle s,t \rangle \in \text{converse}(U) \text{ by auto}
\end{align*}
\}
moreover have \( (-V) : N_0 \text{ using neg_neigh_neigh V(1)} \) by auto
moreover note as ultimately have \( \text{converse}(U) \in \text{leftUniformity} \) unfolding \text{leftUniformity_def} by auto
moreover
\{
\begin{align*}
\text{from } V(1) & \text{ obtain } W \text{ where } W : W : N_0, W + W \subseteq V \text{ using exists_procls_zerohood by blast} \\
\text{fix } x & \text{ assume as: } x \in \{ (s,t) \in G \times G. (\neg s)+t \in W \} \ 0 \ (s,t) \in G \times G. (\neg s)+t \in W \\
\text{then obtain } x_1 \ x_2 \ x_3 & \text{ where } \\
x : x_1 \in G, x_2 \in G, x_3 \in G, (\neg x_1)+x_2 \in W, (\neg x_2)+x_3 \in W, x = \langle x_1, x_3 \rangle \text{ unfolding comp_def by auto} \\
\text{from } W(1) & \text{ have } W+W = f(W \times W) \text{ using interval_add(2) by auto moreover from } W(1) \text{ have } W : W : W \subseteq G \times G \text{ by auto moreover from } x(4,5) \text{ have } ((\neg x_1)+x_2, (\neg x_2)+x_3) : W \times W \text{ by auto with } W \text{ have } f((\neg x_1)+x_2, (\neg x_2)+x_3)) : f(W \times W) \\
\text{using func_imagedef topgroup_f_binop by auto ultimately have } ((\neg x_1)+x_2) \ + ((\neg x_2)+x_3) : W+W \text{ by auto moreover from } x(1,2,3) \text{ have } ((\neg x_1)+x_2) + ((\neg x_2)+x_3) = (-x_1)+x_3 \text{ using cancel_middle_add(2) by simp ultimately have } (-x_1)+x_3 \in W+W \text{ by auto with } W(2) \text{ have } (-x_1)+x_3 \in V \text{ by auto with } x(1,3,6) \text{ have } x : (s,t) \in G \times G. (\neg s)+t \in V \text{ by auto} \\
\end{align*}
\}
then have \( \{ (s,t) \in G \times G. (\neg s)+t \in W \} \ 0 \ (s,t) \in G \times G. (\neg s)+t \in W \text{ } \subseteq U \)
using $V(2)$ by auto moreover
have $\{(s, t) \in G \times G. (-s) + t \in W\} \subseteq \text{leftUniformity}

unfolding \text{leftUniformity_def} \text{ using } W(1) \text{ by auto}
ultimately have $\exists Z \in \text{leftUniformity}. \ Z \circ Z \subseteq U \text{ by auto}
}
ultimately have $\forall U \in \text{leftUniformity}. \ id(G) \subseteq U \land \exists Z \in \text{leftUniformity}. \ Z \circ Z \subseteq U \text{ by auto}
}
ultimately have $\forall U \in \text{leftUniformity}. \ id(G) \subseteq U \land \exists Z \in \text{leftUniformity}. \ Z \circ Z \subseteq U \land \text{converse}(U) \in \text{leftUniformity}

by blast
}
then have $\forall U \in \text{leftUniformity}. \ id(G) \subseteq U \land \exists Z \in \text{leftUniformity}. \ Z \circ Z \subseteq U \land \text{converse}(U) \in \text{leftUniformity}

by auto
with \text{leftFilter} \text{ show } \text{leftUniformity} \{\text{is a uniformity on} \} \ G \text{ unfolding \text{IsUniformity_def} by auto}

\{ fix U assume as:U \in \text{rightUniformity}
from as obtain V where V:V \in N_0 \{ (s, t) \in G \times G. \ s + (-t) \in V\} \subseteq U

unfolding \text{rightUniformity_def} \text{ by auto}
\{ fix x assume as2:x \in id(G)
from as obtain V where V:V \in N_0 \{ (s, t) \in G \times G. \ s + (-t) \in V\} \subseteq U

from V(1) have 0 \in \text{int}(V) \text{ by auto}
then have V0:0 \in V \text{ using Top_2_L1} \text{ by auto}
from as2 obtain t where t:x=(t, t) \in G \text{ by auto}
from t(2) have t+(-t) = 0 \text{ using group0_valid_in_tgroup group0.group0_2_L6}

by auto
with V0 t V(2) have x \in U \text{ by auto}
\}
then have $id(G) \subseteq U \text{ by auto}$
moreover
\{
\{
fix x assume as: x \in (s, t) \in G \times G. \ s + (-t) \in -V\}
then obtain s t where as:s \in G \land t \in G \land s + (-t) \in -V \land x = (s, t)

by force
from as(3) V(1) have s + (-t) \in \{ -q. q \in V\}

using \text{ginv_image_add} \text{ by simp}
then obtain q where q:q \in V \land s + (-t) = -q \text{ by auto}
with $\langle V \in N_0 \rangle$ have q \in G \text{ by auto}
with as(1, 2) q \langle 1, 2 \rangle \text{ have t+(-s) \in V using simple_equation0_add}

by blast
with as(1, 2, 4) V(2) have x \in \text{converse}(U) \text{ by auto}
\}
then have $\{(s, t) \in G \times G. \ s + (-t) \in -V\} \subseteq \text{converse}(U) \text{ by auto}$
moreover from V(1) have $(-V) \in N_0 \text{ using neg_neigh_neigh} \text{ by auto}$

1338
ultimately have converse(U) ∈ rightUniformity using as rightUniformity_def

by auto
}
moreover
{
from V(1) obtain W where W:W:N_0 W + W ⊆ V using exists_procls_zerohood
by blast
{
fix x assume as:x:{⟨s,t⟩∈G×G. s+(-t) ∈ W} ∩ {⟨s,t⟩∈G×G. s+(-t) ∈ W}
then obtains x_1 x_2 x_3 where
x:x_1:G x_2:G x_3:G x_1+(-x_2) ∈ W x_2+(-x_3) ∈ W x=(x_1,x_3)
unfolding comp_def by auto
from W(1) have W+W = f(W×W) using interval_add(2) by auto
moreover from W(1) have W×W×G×G by auto
moreover from x(4,5) have (x_1+(-x_2),x_3+(-x_3)) ∈ W×W by auto
with W have f(x_1+(-x_2),x_3+(-x_3)) ∈ f(W×W)
using func_imagedef topgroup_f_binop by auto
ultimately have (x_1+(-x_2))+(x_2+(-x_3)) ∈ W+W by auto
moreover from x(1,2,3) have (x_1+(-x_2))+(x_2+(-x_3)) = x_1+(-x_3)
using cancel_middle_add(1) by simp
ultimately have x_1+(-x_3) ∈ W+W by auto
with W(2) have x_1+(-x_3) ∈ V by auto
then have x ∈ {⟨s,t⟩∈G×G. s+(-t) ∈ V} using x(1,3,6) by auto
}
with V(2) have {⟨s,t⟩∈G×G. s+(-t) ∈ W} ∩ {⟨s,t⟩∈G×G. s+(-t) ∈ W} ⊆ U
by auto
moreover from W(1) have {⟨s,t⟩∈G×G. s+(-t) ∈ W} ∈ rightUniformity
unfolding rightUniformity_def by auto
ultimately have ∃Z∈rightUniformity. Z O Z ⊆ U by auto
}
ultimately have id(G) ⊆ U ∧ (∃Z∈rightUniformity. Z O Z ⊆ U) ∧ converse(U)∈rightUniformity

by blast
}
then have
∀U∈rightUniformity. id(G) ⊆ U ∧ (∃Z∈rightUniformity. Z O Z ⊆ U) ∧ converse(U)∈rightUniformity

by auto
with rightFilter show rightUniformity {is a uniformity on} G unfolding IsUniformity_def
by auto
qed

The topologies generated by the right and left uniformities are the original
group topology.

lemma (in topgroup) top_generated_side_uniformities:
  shows UniformTopology(leftUniformity,G) = T and UniformTopology(rightUniformity,G) = T
proof -
  let M = {⟨t, {V {t} . V ∈ leftUniformity}⟩ . t ∈ G}
  have fun:M:G→Pow(Pow(G)) using neigh_from_uniformity side_uniformities(1)
    by auto
  let N = {⟨t, {V {t} . V ∈ rightUniformity}⟩ . t ∈ G}
  have funN:N:G→Pow(Pow(G)) using neigh_from_uniformity side_uniformities(2)
    by auto
  { fix U assume op:U∈T
    hence U⊆G by auto
      fix x assume x:x∈U
        with op have xg:x∈G and (-x) ∈ G using neg_in_tgroup by auto
        then have ⟨x, {V{x}. V ∈ leftUniformity}⟩ ∈ {⟨t, {V{t}. V ∈ leftUniformity}⟩ . t ∈ G}
          by auto
        with fun have app:M(x) = {V{x}. V ∈ leftUniformity} using ZF_fun_from_tot_val
          by auto
        have (-x)+U : N \ using open_trans_neigh op x by auto
          then have V:{(s,t)∈G×G. (-s)+t∈((-x)+U)} ∈ leftUniformity
            unfolding leftUniformity_def by auto
          with xg have N:∀ t∈G. t:{(s,t)∈G×G. (-s)+t∈((-x)+U)}{x} ↔ (-x)+t∈((-x)+U)
            using image_iff by auto
            fix t assume t:t∈G
              assume as:(-x)+t∈((-x)+U)
              then have (-x)+t∈LeftTranslation(G,f,-x)U by auto
              then obtain q where q:q∈U (q,(-x)+t)∈LeftTranslation(G,f,-x)
                using image_iff by auto
                with op have q∈G by auto
                from q(2) have (-x)+q = (-x)+t unfolding LeftTranslation_def
                  by auto
                with (¬x) ∈ G} q∈G} t∈G} have q = t using neg_in_tgroup
                  cancel_left_add
                    by blast
                    with q(1) have t∈U by auto
    moreover
\{ 
  assume \( t : t \in U \)
  with \( \langle U \subseteq G \rangle \ \langle \langle -x \rangle \in G \rangle \) have \( (-x) + t \in ((-x) + U) \)
  using lrtrans_image(1) by auto
\}
ultimately have \( (-x) + t \in ((-x) + U) \) \( \iff \ t : U \)
by blast

with \( N \) have \( \forall t : t \in G. t : \{ (s, t) : \in G \times G. \ (-s) + t \in ((-x) + U) \} \iff t \in U \)
by auto

moreover from \( \langle x \in G \rangle \) fun\( N \) have app\( : N(x) = \{ V \cdot V \in \text{rightUniformity} \} \)
using ZF_fun_from_tot_val by simp
moreover
from \( x \) op have open\( Trans : U + (\langle -x \rangle) : \mathcal{N}_0 \) using open_trans_neigh_2 by auto
then have \( V : \langle s, t \rangle \in G \times G. s + t \in (U + (\langle -x \rangle)) \) \( \in \text{rightUniformity} \)
unfolding rightUniformity_def by auto
with \( x \) \( g \)
\( N : \forall t : t \in G. t : \{ (s, t) : \in G \times G. s + t \in (U + (\langle -x \rangle)) \} \iff t + (\langle -x \rangle) \in (U + (\langle -x \rangle)) \)
using vimage_iff by auto
moreover
\{ 
  fix \( t \) assume \( t : t \in G \)
  \{ 
    assume \( as : t + (\langle -x \rangle) \in (U + (\langle -x \rangle)) \)
    hence \( t + (\langle -x \rangle) \in \text{RightTranslation}(G, f, \langle -x \rangle) U \) by auto
    then obtain \( q \) where \( q : q \in U \ \langle q, t + (\langle -x \rangle) \rangle \in \text{RightTranslation}(G, f, \langle -x \rangle) \)
  \}
using image_iff by auto
with op have \( q : q \in G \) by auto
from \( q(2) \) have \( q + (\langle -x \rangle) = t + (\langle -x \rangle) \) unfolding RightTranslation_def by auto
with \( q \in G \) \( \langle -x \rangle \in G \) \( \langle t \in G \rangle \) have \( q = t \) using cancel_right_add by simp
with \( q(1) \) have \( t : t \in U \) by auto
\}
moreover
\{ 
  assume \( t : t \in U \)
  with \( \langle (\langle -x \rangle) \in G \rangle \ \langle U \subseteq G \rangle \) have \( t + (\langle -x \rangle) \in (U + (\langle -x \rangle)) \) using lrtrans_image(2) by auto
\}

1341
ultimately have $t \vdash (-x) \in (U+(-x)) \leftrightarrow t : U$ by blast

with $N$ have $\forall t \in G. \; t : \{(s,t) \in G \times G. \; s+(-t) \in (U+(-x))\} - \{x\} \leftrightarrow t : U$ by blast

with $op$ have $\forall t. \; t : \{(s,t) \in G \times G. \; s+(-t) \in (U+(-x))\} - \{x\} \leftrightarrow t : U$ by auto

hence $\{(s,t) \in G \times G. \; s+(-t) \in (U+(-x))\} - \{x\} = U$ by auto

then have $U = converse(\{(s,t) \in G \times G. \; s+(-t) \in (U+(-x))\}) \{x\}$

unfolding $vimage_def$ by simp

with $V$ app have $U \in \{(t, \{V \{t\} . V \in rightUniformity\}) \cdot t \in G\}(x)$

using $side_uniformities(2)$ $IsUniformity_def$ by auto

ultimately have $U \in \{(t, \{V \{t\} . V \in leftUniformity\}) \cdot t \in G\}(x)$ and

$U \in \{(t, \{V \{t\} . V \in rightUniformity\}) \cdot t \in G\}(x)$ by auto

hence $\forall x \in U. \; U \in \{(t, \{V \{t\} . V \in leftUniformity\}) \cdot t \in G\} \cdot x$ and

$\forall x \in U. \; U \in \{(t, \{V \{t\} . V \in rightUniformity\}) \cdot t \in G\} \cdot x$ by auto

moreover

{ fix $U$ assume as: $U \in Pow(G) \forall x \in U. \; U \in \{(t, \{V \{t\} . V \in leftUniformity\}) \cdot t \in G\}(x)$

{ fix $x$ assume $x : x \in U$

with as(1) have $xg : x \in G$ by auto

from $x$ as(2) have $U \in \{(t, \{V \{t\} . V \in leftUniformity\}) \cdot t \in G\}(x)$ by auto

with $xg$ fun have $U \in \{V \{x\} . V \in leftUniformity\}$ using $apply_equation$

by auto

then obtain $V$ where $V : U = V \{x\}$ $V \in leftUniformity$ by auto

from $V(2)$ obtain $W$ where $W : W \in N_0 \{(s,t) : G \times G. \; (-s)+t : W\} \subseteq V$

unfolding $leftUniformity_def$ by auto

from $W(2)$ have $A : \{(s,t) : G \times G. \; (-s)+t : W\} \{x\} \subseteq V \{x\}$ by auto

from $xg$ have $\forall q \in G. \; q \in (\{(s,t) : G \times G. \; (-s)+t : W\}) \leftrightarrow (-x)+q : W$

using $image_iff$ by auto

hence $B : \{(s,t) : G \times G. \; (-s)+t : W\} \{x\} = \{t \in G. \; (-x)+t : W\}$ by auto

from $W(1)$ have $WG : W \subseteq G$ by auto

{ fix $t$ assume $t : t \in x + W$
then have $t \in \text{LeftTranslation}(G, f, x)W$ by auto
then obtain $s$ where $s : s \in W \langle s, t \rangle \in \text{LeftTranslation}(G, f, x)$ using image_iff by auto
with $\langle W \subseteq G \rangle$ have $s \in G$ by auto
from $s(2)$ have $t = x + s$ $t \in G$ unfolding LeftTranslation_def by auto
with $\langle x \in G \rangle$ $\langle s \in G \rangle$ have $(-x) + t = s$ using put_on_the_other_side(2)
by simp
with $s(1)$ have $(-x) + t \in W$ by auto
with $\langle t \in G \rangle$ have $(-x) + t \in \{ s \in G. (-x) + s : W \}$ by auto
ultimately have $\exists \ Y \in T. x \in Y \land Y \subseteq U$ by auto
then have $U \in T$ using open_neigh_open by auto
moreover {fix $U$ assume as: $U \in \text{Pow}(G) \forall x \in U. U \in \{ \langle t, \{ V \{ t \} . V \in \text{leftUniformity}\} \rangle . t \in G \} \ x$}
by auto
moreover have $\int(x + W) \subseteq T$ using Top_2_L1 by auto
ultimately have $\exists Y \in T. x \in Y \land Y \subseteq U$ by auto
then have $U \in T$ using open_neigh_open by auto
} hence $\{ U \in \text{Pow}(G) \forall x \in U. U \in \{ \langle t, \{ V \{ t \} . V \in \text{leftUniformity}\} \rangle . t \in G \} \ x \}$ by auto
moreover
{fix $x$ assume x:xG
with as(1) have xG:xG by auto
from x as(2) have Ue:{t, \{ V \{ t \} . V \in \text{leftUniformity}\}} . t \in G) x
by auto
with xG funN have Ue:{V \{ x \} . V \in \text{rightUniformity}} using apply_equality
by auto
then obtain $V$ where $V : U = V(x)$ $V \in \text{rightUniformity}$ by auto
then have converse($V$) $\in \text{rightUniformity}$ using side_uniformities(2)
IsUniformity_def
by auto
then obtain $W$ where $W : W \in N_0 \{ \langle s, t \rangle : G \times G. s + (-t) : W \} \subseteq \text{converse}(V)$
unfolding rightUniformity_def by auto
from $W(2)$ have $A : \langle \langle s, t \rangle : G \times G. s + (-t) : W \rangle - \{ x \} \subseteq V(x)$ by auto
from xG have $\forall q \in G. q \in \{ \langle s, t \rangle : G \times G. s + (-t) : W \} - \{ x \}$ $\longleftrightarrow q + (-x) : W$
using image_iff by auto
hence $B : \langle \langle s, t \rangle : G \times G. s + (-t) : W \rangle - \{ x \} = \{ t \in G. t + (-x) : W \}$ by auto
from $W(1)$ have $W : W \subseteq G$ by auto
}
fix t assume t ∈ W+x
with ⟨x∈G⟩ ⟨W⊆G⟩ obtain s where s∈W and t=s+x using lrtrans_image(2)

by auto
with ⟨W⊆G⟩ have s∈G by auto
with ⟨x∈G⟩ ⟨t=s+x⟩ have t∈G using group_op_closed_add by simp
from ⟨x∈G⟩ ⟨s∈G⟩ ⟨t=s+x⟩ have t+(−x) = s using put_on_the_other_side

by simp
with ⟨s∈W⟩ ⟨t∈G⟩ have t ∈ {s∈G. s+(−x) ∈ W} by auto

then have W+x ⊆ {t:G. t+(−x):W} by auto
with B have W + x ⊆ {(s,t) ∈ G × G. s + (− t) ∈ W}−{x} by auto
with A have W + x ⊆ V {x} by blast
with V(1) have W + x ⊆ U by auto
then have int(W + x) ⊆ U using Top_2_L1 by blast
moreover
from xg W(1) have x∈int(W + x) using elem_in_int_rtrans(1) by auto
moreover have int(W + x)∈T using Top_2_L2 by auto
ultimately have ∃Y∈T. x∈Y ∧ Y⊆U by auto

then have U∈T using open_neigh_open by auto

ultimately have
{U ∈ Pow(G). ∀x∈U. U ∈ {{t, V{t} . V ∈ leftUniformity)}. t ∈ G}(x)}
= T
{U ∈ Pow(G). ∀x∈U. U ∈ {{t, V{t} . V ∈ rightUniformity)}. t ∈ G}(x)}
= T
by auto
then show UniformTopology(leftUniformity,G) = T and UniformTopology(rightUniformity,G)
= T
using uniftop_def_alt by auto
qed

The side uniformities are called this way because of how they affect left and
right translations. In the next lemma we show that left translations are
uniformly continuous with respect to the left uniformity.

lemma (in topgroup) left_mult_uniformity: assumes x∈G
shows
LeftTranslation(G,f,x) {is uniformly continuous between} leftUniformity
{and} leftUniformity
proof -
let P = ProdFunction(LeftTranslation(G, f, x), LeftTranslation(G, f, x))
from asms have L: LeftTranslation(G,f,x):G→G and leftUniformity {is
a uniformity on} G
using group0_5_L1 side_uniformities(1) by auto
moreover have ∀V ∈ leftUniformity. P−(V) ∈ leftUniformity
proof -

{ fix V assume V ∈ leftUniformity
  then obtain U where U ∈ N₀ and {⟨s,t⟩ ∈ G × G . (- s) + t ∈ U} ⊆ V
}

unfolding leftUniformity_def by auto

with as: V ⊆ G × G U ∈ N₀ {⟨s,t⟩ ∈ G × G . (- s) + t ∈ U} ⊆ V
unfolding leftUniformity_def by auto

{ fix z assume z ∈ {⟨s,t⟩ ∈ G × G . (- s) + t ∈ U}
  then obtain s t where st: z = ⟨s,t⟩ s ∈ G t ∈ G by auto
  from assms st have
    P(z) = ⟨LeftTranslation(G, f, x)(s), LeftTranslation(G, f, x)(t)⟩
    using prodFunctionApp group0_5_L1(2) by blast
  with assms st(2,3) have P(z) = ⟨x+s,x+t⟩ using group0_5_L2(2)
    by auto
  moreover from x ∈ G s ∈ G t ∈ G have (- (x+s)) + (x+t) = (-s)+t
    using cancel_middle_add(3) by simp
  with st2 have (- (x+s)) + (x+t) ∈ U by auto
  ultimately have P(z) ∈ {⟨s,t⟩ ∈ G × G . (- s) + t ∈ U} by auto
  using assms st(2,3) group_op_closed by auto
  with as(3) have P(z) ∈ V by force
  with L z have z ∈ P-(V) using prodFunction func1_1_L5A vimage_iff
    by blast
}

with as(2) have ∃U ∈ N₀. {⟨s,t⟩ ∈ G × G . (- s) + t ∈ U} ⊆ P-(V)
  by blast
with as(3) have P-(V) ∈ leftUniformity unfolding leftUniformity_def using prodFunction func1_1_L6A by blast

thus thesis by simp
qed

ultimately show thesis using IsUniformlyCont_def by auto
qed

Right translations are uniformly continuous with respect to the right uniformity.

lemma (in topgroup) right_mult_uniformity: assumes x ∈ G
  shows RightTranslation(G,f,x) {is uniformly continuous between} rightUniformity
  {and} rightUniformity
proof -
  let P = ProdFunction(RightTranslation(G, f, x), RightTranslation(G, f, x))
  from assms have R: RightTranslation(G,f,x):G→G and rightUniformity
    {is a uniformity on} G
    using group0_5_L1 side_uniformities(2) by auto
moreover have \( \forall V \in \text{rightUniformity}. \ P(V) \in \text{rightUniformity} \)

proof -
{ fix \( V \) assume \( V \in \text{rightUniformity} \)
then obtain \( U \) where \( U \in \mathcal{N}_0 \) and \( \{(s,t) \in G \times G . \ s + (-t) \in U\} \subseteq V \)

unfolding \( \text{rightUniformity_def} \) by auto
with \( \langle V \in \text{rightUniformity}\rangle \) have 
{ fix \( z \) assume \( z \in \{(s,t) \in G \times G . \ s + (-t) \in U\} \)
then obtain \( s \ t \) where \( st: z = \langle s, t \rangle \) \( s \in G \) \( t \in G \) by auto
from \( st(1) \) \( z \)
have \( asms (2,3) \) \( \text{group_op_closed} \) by auto
{ with \( as(3) \) have \( \exists U \in \mathcal{N}_0 . \ (s,t) \in G \times G \). \ s + (-t) \in U \)
by blast }
with \( as(2) \) have \( \exists U \in \mathcal{N}_0 . \ \{(s,t) \in G \times G . \ s + (-t) \in U\} \subseteq P-(V) \)
by blast
with \( \langle \text{RightTranslation}(G,f,x): G \to G\rangle \) \( \langle V \subseteq G \times G\rangle \) have \( P(V) \in \text{rightUniformity} \)
unfolding \( \text{rightUniformity_def} \) using prodFunction func1_1_L6A vimage_iff by auto
\}
thus thesis by simp
qed
ultimately show thesis using \( \text{IsUniformlyCont_def} \) by auto
qed

The third uniformity important on topological groups is called the uniformity of Roelcke.

definition \( \text{in topgroup} \) \( \text{roelckeUniformity} \)
where \( \text{roelckeUniformity} \equiv \{V \in \text{Pow}(G \times G). \ \exists U \in \mathcal{N}_0 . \ \{(s,t) \in G \times G . \ t \in (U+s)+U\} \subseteq V\} \)

The Roelcke uniformity is indeed a uniformity on the group.

lemma \( \text{in topgroup} \) \( \text{roelcke_uniformity} \):
shows \( \text{roelckeUniformity} \ \{\text{is a uniformity on} \ G \) 
proof -
let \( \Phi = \text{roelckeUniformity} \)

1346
∀ U ∈ Φ. id(G) ⊆ U ∧ \( \exists V ∈ Φ. \ V \cup V \subseteq U \) ∧ converse(U) ∈ Φ

proof
fix U assume U:U ∈ roelckeUniformity
then obtain V where V:{(s,t)∈G×G. t ∈((V+s)+V)} ⊆ U V ∈ ℕ₀ U:Pow(G×G)

unfolding roelckeUniformity_def by auto
from V(2) have VG:V ⊆ G by auto
have id(G) ⊆ U proof
from V(2) have 0∈int(V) by auto
then have V0:0∈V using Top_2.L1 by auto
thus id(G) ⊆ U by auto
qed
moreover have converse(U) ∈ Φ
proof
{ fix l assume l ∈ \{ (s,t)∈G×G. t ∈((-V)+s)+(-V) \}
then obtain s t where st:s∈G t∈((-V)+s)+(-V) l=(s,t)
by force
from \( \langle V \subseteq G \rangle \) have smG:(-V) ⊆ G using ginv_image_add(1) by simp
with \( \langle s \in G \rangle \) have VxG:(-V)+s ⊆ G using lrtrans_in_group_add(2)
by simp
from \( \langle V \subseteq G \rangle \) \( \langle t \in G \rangle \) have VsG:V+t ⊆ G using lrtrans_in_group_add(2)
by simp
from st(3) VxG smG obtain x y where xy:t = x+y x ∈ (-V)+s y∈(-V)
using elements_in_set_sum by blast
from xy(2) smG st(1) obtain z where z:x = z+s z∈(-V) using elements_in_rtrans
by blast
with \( \langle y∈(-V) \rangle \) \( \langle s∈G \rangle \) \( \langle t = x+y \rangle \)
have ts:(-z)+t+(−y) = s using cancel_middle_add(5) by blast
{ fix u assume u∈(-V)
with \( \langle V \subseteq G \rangle \) have (-u) ∈ V using ginv_image_el_add by simp
} hence R:∀u∈(-V).(-u) ∈ V by simp
with z(2) xy(3) have zy:(-z)∈V (-y)∈V by auto
from zy(1) VG st(2) have (-z)+t : V+t using lrtrans_image(2)
by auto
with zy(2) VG VsG have (-z)+t+(-y) : (V+t)+V
using interval_add(4) by auto
with ts have s:(V+t)+V by auto

1347
with \( \text{st}(1,2) \) have \( \{s,t\} \subseteq \text{converse}(\{(s,t) \in G \times G. \ t \in (V+s)+V\}) \)
using converse_iff by auto
with \( V(1) \) have \( \{s,t\} \subseteq \text{converse}(U) \) by auto
with \( \text{st}(4) \) have \( l \in \text{converse}(U) \) by auto
} then have \( \{s,t\} \subseteq G \times G. \ t \in ((-V)+s)+(-V) \) \( \subseteq \text{converse}(U) \) by auto
moreover from \( V(2) \) have \((-V): N' \subseteq \text{neg_neigh_neigh} \) by auto
ultimately have \( \exists V \in N'. \{s,t\} \subseteq (V+s)+V \subseteq \text{converse}(U) \) by auto
moreover
from \( V(3) \) have \( \text{converse}(U) \subseteq G \times G \) unfolding converse_def by auto
ultimately show \( \text{converse}(U) \in \text{roelckeUniformity} \) unfolding roelckeUniformity_def by auto
qed
moreover have \( \exists Z \in \Phi. \ Z : Z \subseteq U \)
proof -
from \( V(2) \) obtain \( \bar{W} \) where \( \bar{W}: \bar{W} \in N' \subseteq \bar{W} \subseteq V \) using exists_procls_zerohood by blast
moreover
\{ fix \( k \) assume \( \text{as:} k : \{\{s,t\} \in G \times G. \ t \in (W+s)+W\} \) \}
then obtain \( x_1, x_2, x_3 \) where
\( x, x_1, x_2, x_3 \in (W+x_1)+W \) \( x_3 \in (W+x_2)+W \) \( k = (x_1, x_3) \)
unfolding comp_def by auto
from \( x_1 \in G \) have \( \forall s, t \in G \) and \( V \cap (V+s) \subseteq G \)
using lrtrans_in_group_add(2) by auto
from \( x(4) \) \( \forall s, t \in G \) obtain \( x \ y \) where \( xy : x_2 = x+y \) \( x \in W+x_1 \ y \in W \)
using elements_in_set_sum by blast
from \( xy(2) \) \( \forall s, t \in G \) obtain \( x(1) \) obtain \( z \) where \( z : z = x+x_1 \ z \in \bar{W} \) using elements_in_rtrans by blast
from \( z(2) \) \( \forall s, t \in G \) obtain \( yzG : y \in G \) \( z \in G \) by auto
from \( x(2) \) \( \forall s, t \in G \) obtain \( \bar{W} : x_2 \subseteq G \) using lrtrans_in_group_add by simp
from \( x(5) \) \( \forall s, t \in G \) obtain \( x \ y' \) where \( xy : x_3 = x'+y' \) \( x' \in \bar{W}+x_2 \ y' \in \bar{W} \)
using elements_in_set_sum by blast
from \( xy2(2) \) \( \forall s, t \in G \) obtain \( z' \) where \( z : z' = x'_2 \ z' \in \bar{W} \) using elements_in_rtrans by blast
from \( z2(2) \) \( \forall s, t \in G \) obtain \( yzG : y' \in G \) \( z' \in G \) by auto
from \( xy(1) \) \( \forall s, t \in G \) obtain \( z' = (z' + (z+x_1)+y')) + y' \) by auto
with \( yzG \) \( yzG : x(1) \) have \( x_3 : x_3 = ((z'+z) + x_1) + (y+y') \)
using group_op_assoc group_op_closed by simp
from \( xy(3) \) \( z(2) \) \( xy2(3) \) \( z2(2) \) \( \forall s, t \in G \) have \( z' + z \in W+x+y' \) \( y+y' \in W+W \)
using interval_add(4) by auto
with \( W(2) \) have \( yzG : z' + z \in V \) \( y+y' \in V \) by auto
from \( yzG(1) \) \( \forall s, t \in G \) obtain \( (z'+z) + x_1 \in W+x_1 \) using lrtrans_image(2) by auto
with \( yzG(2) \) \( \forall s, t \in G \) have \( (z'+z) + x_1 + (y+y') \in (V+x_1)+V \)
using interval_add(4) by auto
with $x \in (V+x_1)+V$ by auto
with $x(1,3,6)$ have $k:\{(s,t)\in G\times G. t \in (V+s)+V\}$ by auto

by auto
moreover from $W(1)$ have $\{(s,t)\in G\times G. t \in (W+s)+W\}\in \text{roelckeUniformity}$

unfolding $\text{roelckeUniformity}_\text{def}$ by auto
ultimately show $\exists Z\in \text{roelckeUniformity}. Z \subseteq U$ by auto
qed
ultimately show $\text{id}(G) \subseteq U \land (\exists V \in \Phi. V \subseteq U) \land \text{converse}(U) \in \Phi$
by simp
qed
moreover
have $\text{roelckeUniformity}$ \{is a filter on\} $(G \times G)$
proof -
{
assume $0 \in \text{roelckeUniformity}$
then obtain $W$ where $U:W\in N_0 \\{(s,t)\in G\times G. t \in (W+s)+W\}\subseteq 0$
unfolding $\text{roelckeUniformity}_\text{def}$ by auto
have $(0,0):G\times G$ using $\text{zero_in_tgroup}$ by auto
moreover have $0 = 0+0+0$ using $\text{group0_2_L2 zero_in_tgroup}$ by auto
moreover
from $U(1)$ have $0\in \text{int}(W)$ by auto
then have $0\in W$ using $\text{Top_2_L1}$ by auto
with $\langle W\in N_0 \rangle$ have $0+0+0 \in (W+0)+W$
using $\text{group0_2_L2 group_op_closed trans_neutral_image interval_add_zero}$ by auto
ultimately have $(0,0)\in \{(s,t)\in G\times G. t \in (W+s)+W\}$ by auto
with $U(2)$ have False by blast
}
moreover
{
fix $x$ $xa$ assume as:$x \in \text{roelckeUniformity} xa\in x$
have $\text{roelckeUniformity} \subseteq \text{Pow}(G\times G)$ unfolding $\text{roelckeUniformity}_\text{def}$
by auto
with as have $xa \in G\times G$ by auto
}
moreover
{
have $G\times G\in \text{Pow}(G\times G)$ by auto
moreover
have $\{(s,t):G\times G. t \in (G+s)+G\} \subseteq G\times G$ by auto
moreover note $\text{zneigh_not_empty}$
ultimately have $G\times G\in \text{roelckeUniformity}$ unfolding $\text{roelckeUniformity}_\text{def}$
by auto
}
moreover

1349
\{ 
  fix A B assume as:A∈roelckeUniformity B∈roelckeUniformity
  from as(1) obtain AU where
  AU:AU∈ N₀ \{⟨s,t⟩∈G×G. t ∈(AU+s)+AU\} ⊆ A∈Pow(G×G) 
  unfolding roelckeUniformity_def by auto
  from as(2) obtain BU where
  BU:BU∈ N₀ \{⟨s,t⟩∈G×G. t ∈(BU+s)+BU\} ⊆ B∈Pow(G×G) 
  unfolding roelckeUniformity_def by auto
  moreover have op:int(AU)∩int(BU)∈T using Top_2_L2 topSpaceAssum
  unfolding IsATopology_def by auto
  moreover have int(AU)∩int(BU)⊆AU∩BU using Top_2_L5 
  by auto
  moreover note AU(1) BU(1)
  ultimately have interNeigh:AU∩BU∈ N₀ unfolding zerohoods_def by auto
  morevoer 
  \{ 
    fix z assume z ∈ \{⟨s,t⟩∈G×G. t ∈((AU∩BU)+s)+(AU∩BU)\} 
    then obtain s t where
    z=z⟨s,t⟩ s∈G t∈((AU∩BU)+s)+(AU∩BU) 
    by force
    from \langle AU∩BU \rangle <s∈G> have AU∩BU ⊆ G and (AU∩BU)+s ⊆ G 
    using lrtrans_in_group_add(2) by auto
    with \text{z(4)} obtain x y where t:t=x+y x∈(AU∩BU)+s y∈AU∩BU 
    using elements_in_set_sum by blast
    from \text{t(2)} \text{z(2)} interNeigh obtain q where x:x=q+s q ∈ AU∩BU using 
    lrtrans_image(2) 
    by auto
    with AU(1) BU(1) z(2) have x ∈ AU+s x ∈ BU+s using lrtrans_image(2) 
    by auto
    with \text{z(1,2,3)} have 
    z ∈ \{⟨s,t⟩∈G×G. t ∈(AU+s)+AU\} and z ∈\{⟨s,t⟩∈G×G. t ∈(BU+s)+BU\} 
    by auto
  \} 
  then have 
  \{⟨s,t⟩∈G×G. t ∈((AU∩BU)+s)+(AU∩BU)\} ⊆ 
  \{⟨s,t⟩∈G×G. t ∈(AU+s)+AU\}∩\{⟨s,t⟩∈G×G. t ∈(BU+s)+BU\} 
  by auto
  with AU(2) BU(2) have \{⟨s,t⟩∈G×G. t ∈((AU∩BU)+s)+(AU∩BU)\} ⊆ A∩B 
  by blast
\}
ultimately have \( A \cap B \in \text{roelckeUniformity} \) using \( \text{AU}(3) \) \( \text{BU}(3) \) unfolding \text{roelckeUniformity_def} by blast

moreover

\{ fix B C assume \( \text{as:B}\in\text{roelckeUniformity} \) \( C \subseteq (G \times G) \) \( B \subseteq C \)
\}
\{ from \( \text{as}(1) \) obtain \( \text{BU} \) where \( \text{BU}\in\mathcal{N}_0 \) \( \{ (s,t) \in G \times G. t \in (\text{BU}+s) + \text{BU} \} \subseteq \text{B} \)
\}

unfolding \text{roelckeUniformity_def} by blast
\{ from \( \text{as}(3) \) \( \text{BU}(2) \) have \( \{ (s,t) \in G \times G. t \in (\text{BU}+s) + \text{BU} \} \subseteq \text{C} \) by auto
\}
then have \( C \in \text{roelckeUniformity} \) using \( \text{as}(2) \) \( \text{BU}(1) \)

unfolding \text{roelckeUniformity_def} by auto
\}
ultimately show thesis unfolding \text{IsFilter_def} by auto
qed
ultimately show thesis using \text{IsUniformity_def} by auto
qed

The topology given by the roelcke uniformity is the original topology

**lemma** (in \text{topgroup}) \text{top_generated_roelcke_uniformity}:

shows \( \text{UniformTopology}(\text{roelckeUniformity}, G) = T \)
proof -
\{ let \( M = \{ \langle t, \{ V \{ t \} . V \in \text{roelckeUniformity} \} \rangle . t \in G \} \)
\{ have \( \text{fun:M:G} \rightarrow \text{Pow}(\text{Pow}(G)) \) using \text{IsNeighSystem_def neigh_from_uniformity roelcke_uniformity} by auto
\}
\{ fix \( U \) assume \( \text{as:U} \in \{ U \in \text{Pow}(G). \forall x \in U. U \in M x \} \)
\{ fix \( x \) assume \( x:x \in U \)
with \( \text{as} \) have \( xg:x \in G \) by auto
\}
\{ from \( x \) \( \text{as} \) have \( U \in \{ \langle t, \{ V \{ t \} . V \in \text{roelckeUniformity} \} \rangle . t \in G \} \) \( (x) \) by auto
with \( \text{fun} \langle x \in G \rangle \) have \( U \in \{ V \{ x \} . V \in \text{roelckeUniformity} \} \) using \text{ZF_fun_from_tot_val} by simp
\}
then obtain \( V \) where \( \text{V:U=V}\{x\} \) \( V \in \text{roelckeUniformity} \) by auto
\}
\{ from \( \text{V}(2) \) obtain \( W \) where \( \text{W:W} \in \mathcal{N}_0 \) \( \{ (s,t) \in G \times G. t \in (W+s)+W \} \subseteq V \) unfolding \text{roelckeUniformity_def} by auto
\}
\{ from \( \text{W}(1) \) have \( \text{WG:W} \subseteq G \) by auto
\}
\{ from \( \text{W}(2) \) have \( A: \{ (s,t) : G \times G. \ t: (W+s)+W \} \subseteq V \) \( \{ x \} \) by auto
have \( \{ (s,t) \in G \times G. t \in (W+s)+W \} \{ x \} = (W+x)+W \)
proof -
\{ let \( A = \{ (s,t) : G \times G. \ t \in (W+s)+W \} \)
\}
\{ from \( \text{WG} \) \( \langle x \in G \rangle \) have \( I: (W+x)+W \subseteq G \)
using \text{ltrans_in_group_add interval_add(1)} by auto
have \( A(x) = \{ t \in G. \langle x, t \rangle \in A \} \) by blast
\}
\}
moreover have \{t \in G. \langle x, t \rangle \in A\} \subseteq (W+x)+W by auto
moreover from \langle W \subseteq G \rangle \langle x \in G \rangle I have \langle (W+x)+W \subseteq \{t \in G. \langle x, t \rangle \in A\} \rangle
by auto ultimately show thesis by auto qed
with A V(1) have WU : (W+x)+W \subseteq U by auto
have int(W)+x \subseteq W+x using image_mono Top_2_L1 by simp
then have \((\text{int}(W)+x) \times (\text{int}(W)) \subseteq (W+x) \times W\) using Top_2_L1 by auto
then have \(f((\text{int}(W)+x) \times (\text{int}(W))) \subseteq f((W+x) \times W)\) using image_mono
by auto
moreover from xg WG have \langle \text{int}(W)+x, \text{int}(W) \rangle \in \text{Pow}(G) \times \text{Pow}(G) and \langle (W+x), W \rangle \in \text{Pow}(G) \times \text{Pow}(G)
using Top_2_L2 lrtrans_in_group_add(2) by auto
then have \((\text{int}(W)+x)+(\text{int}(W)) = f((\text{int}(W)+x) \times (\text{int}(W)))\) and
\((W+x)+W = f((W+x) \times W)\)
using interval_add(2) by auto
ultimately have \((\text{int}(W)+x)+(\text{int}(W)) \subseteq (W+x)+W\) by auto
with xg WG have \text{int}(W+x)+(\text{int}(W)) \subseteq (W+x)+W using rtrans_interior
by auto
moreover
\{ have \text{int}(W+x)+(\text{int}(W)) = (\bigcup t \in \text{int}(W+x). t+(\text{int}(W)))
using interval_add(3) Top_2_L2 by auto
moreover have \forall t \in \text{int}(W+x). t+(\text{int}(W)) = \text{int}(t+W)
proof -
\{ fix t assume t \in \text{int}(W+x)
from \langle x \in G \rangle have \langle W+x \rangle \subseteq G using lrtrans_in_group_add(2)
by simp with \(t \in \text{int}(W+x)\) have t \in G using Top_2_L2 by auto
with \langle W \subseteq G \rangle have t + \text{int}(W) = \text{int}(t+W) using ltrans_interior
by simp \}
thus thesis by simp qed
ultimately have \text{int}(W+x)+(\text{int}(W)) = (\bigcup t \in \text{int}(W+x). \text{int}(t+W))
by auto
with topSpaceAssum have \text{int}(W+x)+(\text{int}(W)) \in T using Top_2_L2
union_open by auto
\}
moreover from \langle x \in G \rangle \langle W \in N_0 \rangle have x \in \text{int}(W+x)+(\text{int}(W))
using elem_in_int_rtrans(2) by simp
moreover note WU
ultimately have \exists Y \in T. x \in Y \land Y \subseteq U by auto
\}
then have U \in T using open_neigh_open by auto
then have \( \{ U \in \text{Pow}(G) \mid \forall x \in U. U \in \{ \{ t, \{ V \mid V \in \text{roelckeUniformity} \} \} \} \subseteq T \)
by auto
moreover
\[
\text{fix } U \text{ assume } \text{op} : U \subseteq T
\]
\[
\text{fix } x \text{ assume } x : x \in U
\]
with \text{op} have \( xg : x \in G \) by auto
have \( (-x) + U \in N_0 \) using \text{open_trans_neigh} \( \text{op} x \) by auto
then obtain \( W \) where \( W : W \subseteq (-x) + U \) using \text{exists_procls_zerohood}
by blast
let \( V = x + (W + (-x)) \cap W \)
from \( \langle W \in N_0 \rangle \langle x \in G \rangle \) have \( xWx : x + (W + (-x)) : N_0 \) using \text{lrtrans_neigh}
by simp
from \( W(1) \) have \( GW : W \subseteq G \) by auto
from \( xWx \ W(1) \) have \( 0 \in \text{int}(x + (W + (-x))) \cap \text{int}(W) \) by auto
using \text{Top_2_L2} topSpaceAssum unfolding \text{IsATopology_def} by auto
have \( \text{int}(x + (W + (-x))) \cap \text{int}(W) \subseteq (x + (W + (-x))) \cap W \) using \text{Top_2_L1}
by auto
with \text{int} have \( \text{int}(x + (W + (-x))) \cap \text{int}(W) \subseteq ((x + (W + (-x))) \cap W) \)
using \text{Top_2_L5} by auto
moreover note \( xWx \ W(1) \)
ultimately have \( V \_ \text{NEIG} : V \in N_0 \) unfolding \text{zerohoods_def} by auto
\[
\text{fix } z \text{ assume } z : z \in (V + x) \cap W
\]
from \( W(1) \) have \( VG : V \subseteq G \) by auto
with \( \langle x \in G \rangle \) have \( VxG : V + x \subseteq G \) using \text{lrtrans_in_group_add}(2) by simp
from \( z VG VxG W(1) \) obtain \( a_1 \ b_1 \) where \( ab : z = a_1 + b_1 \ a_1 \in V + x \ b_1 \in V \)
using \text{elements_in_set_sum} by blast
from \( ab(2) \) \( xg \ VG \) obtain \( c_1 \) where \( c : a_1 = c_1 + x \ c_1 \in V \) using \text{elements_in_rtrans}
by blast
from \( ab(3) \) \( c(2) \) have \( bc : b_1 \in W \ c_1 \in x + (W + (-x)) \) by auto
from \( \langle x \in G \rangle \) have \( x + (W + (-x)) = \{ x + y. y \in (W + (-x)) \} \)
using \text{neg_in_tgroup} \( \text{lrtrans_in_group_add} \) \( \text{lrtrans_image} \) by auto
with \( c_1 \in x + (W + (-x)) \) obtain \( d \) where \( d : c_1 = x + d \ d \in W + (-x) \)
by auto
from \( \langle x \in G \rangle \langle W \in N_0 \rangle \langle d \in W + (-x) \rangle \) obtain \( e \) where \( e : d = e + (-x) \ e \in W \)
using \text{neg_in_tgroup} \( \text{lrtrans_in_group} \) \( \text{lrtrans_image}(2) \) by auto
from \( e(2) \) \( WG \) have \( eG : e \in G \) by auto
from \( \langle e \in W \rangle \langle W \subseteq G \rangle \langle b_1 \in W \rangle \) have \( eG b_1 \in G \) by auto
from \( \langle z = a_1 + b_1 \rangle \langle a_1 = c_1 + x \rangle \langle c_1 = x + d \rangle \langle d = e + (-x) \rangle \)
have \( z = x + (e + (-x)) + x + b_1 \) by simp
with \( \langle x \in G \rangle \langle e \in G \rangle \) have \( z = (x + e) + b_1 \) using cancel_middle(4) by simp
with \( \langle x \in G \rangle \langle e \in G \rangle \langle b_1 \in G \rangle \) have \( z = x + (e + b_1) \) using group_opr_assoc
by simp
moreover from \( e \in (2) \) ab \( (3) \) WG have \( e + b_1 \in \bar{W} + \bar{W} \) using elements_in_set_sum_inv
by auto
moreover note \( x \in G \)
moreover from \( \langle W \subseteq G \rangle \langle U \in T \rangle \) have \( W + W \subseteq G \) and \( U \subseteq G \) using interval_add(1)
by simp
ultimately have \( z \in x + (W + W) \) using elements_in_ltrans_inv interval_add(1)
by auto
moreover from \( \langle W \subseteq G \rangle \langle U \in T \rangle \) have \( W + W \subseteq (x + U) \)
by simp
ultimately have \( z \in (x + U) \) by auto
}
then have \( \text{sub} : (V + x) + V \subseteq U \) by auto
moreover from \( V \_\text{NEIG} \) have \( \text{unif} : \{ (s, t) \in G \times G. \ t : (V + s) + V \} \in \text{roelckeUniformity} \)
unfolding roelckeUniformity_def by auto
moreover from \( xg \) have
\( \forall q. q \in \{ (s, t) \in G \times G. \ t : (V + s) + V \} \iff q \in ((V + x) + V) \cap G \)
by auto
then have \( \{ (s, t) \in G \times G. \ t \in (V + s) + V \} = ((V + x) + V) \cap G \)
by auto
ultimately have \( \text{basic} : \{ (s, t) \in G \times G. \ t : (V + s) + V \} \subseteq U \) using op
by auto
have \( \text{add} : \{ (x) \times U \} \{ x \} = U \) by auto
from basic add have \( \{ (s, t) \in G \times G. \ t \in (V + s) + V \} \cup \{ (x) \times U \} \{ x \} = U \)
by auto
moreover have \( \forall B \in \text{roelckeUniformity} . \forall C \subseteq Pow(G \times G). \ B \subseteq C \)
using roelcke_uniformity unfolding IsUniformity_def IsFilter_def
by auto
moreover from \( op \ xg \) have \( G G : \{ (s, t) \in G \times G. \ t \in (V + s) + V \} \cup \{ (x) \times U \} : Pow(G \times G) \)
by auto
moreover have
\( \{ (s, t) \in G \times G. \ t \in (V + s) + V \} \subseteq \{ (s, t) \in G \times G. \ t \in (V + s) + V \} \cup \{ (x) \times U \} \)
by auto
moreover from \( R \) unif \( G G \) have
\( \{ (s, t) \in G \times G. \ t \in (V + s) + V \} \cup \{ (x) \times U \} \in \text{roelckeUniformity} \)
by auto
ultimately have \( \exists V \in \text{roelckeUniformity}. \ V \{ x \} = U \) by auto

1354
then have $U \in \{V \{x\} . V \in \text{roelckeUniformity}\}$ by auto
with $xg$ fun have $U \in \{\{t, \{V \{t\} . V \in \text{roelckeUniformity}\}\} . t \in G\}$
using $\text{apply\_equality}$ by auto

hence $\forall x \in U. U \in \{\{t, \{V \{t\} . V \in \text{roelckeUniformity}\}\} . t \in G\}$
by auto

ultimately have $\{U \in \text{Pow}(G). \forall x \in U. U \in \{\{t, \{V \{t\} . V \in \text{roelckeUniformity}\}\} . t \in G\}(x)\} = T$
by auto

then show thesis using $\text{uniftop\_def\_alt}$ by simp
qed

The inverse map is uniformly continuous in the Roelcke uniformity

**Theorem (in topgroup) inv_uniform_roelcke:**
shows
GroupInv($G,f$) {is uniformly continuous between} roelckeUniformity
{and} roelckeUniformity

proof -
let $P = \text{ProdFunction}(\text{GroupInv}(G,f), \text{GroupInv}(G,f))$
have $L: \text{GroupInv}(G,f):G \to G$ and $R:\text{roelckeUniformity}$ {is a uniformity on} $G$
using $\text{groupAssum group0\_2\_T2 roelcke\_uniformity}$ by auto
have $\forall V \in \text{roelckeUniformity}. P^{-1}(V) \in \text{roelckeUniformity}$
proof
fix $V$ assume $v:V \in \text{roelckeUniformity}$
then obtain $U$ where $U \in N_0 \subseteq V$ unfolding $\text{roelckeUniformity\_def}$ by auto
with $\langle V \in \text{roelckeUniformity}\rangle$ have
as:$V \subseteq G \times G U \in N_0 \{\{s,t\} \in G \times G . t \in U + s + U\} \subseteq V$
unfolding $\text{roelckeUniformity\_def}$ by auto
from as(2) obtain $W$ where $w:W \in N_0 W \subseteq U (-W) = W$ using $\text{exists\_sym\_zerohood}$
by blast
from $w(1)$ have $wg:W \subseteq G$ by auto
{
fix $z$ assume $z:z \in \{\{s,t\} \in G \times G . t \in W + s + W\}$
then obtain $s t$ where $st:z=\{s,t\} \in G \times G t \in G$ by auto
from $st(1)$ $z$ have $st2: t \in W + s + W$ by auto
with $\langle W \in N_0 \rangle$ $st(2)$ obtain $u v$ where $uv:t+u+v u \in W + s v \in W$
using $\text{interval\_add}(4) \text{lrtrans\_in\_group\_add}(2)$ by auto
from $\langle W \subseteq G \rangle \langle s \in G \rangle \langle u \in W + s \rangle$ obtain $q$ where $q:q \in W u=q+s$ using $\text{elements\_in\_rtrans}$
by blast

1355
from \( w(2) \) as \( (2) \) \( q \) \( st(2) \) have \( u \in U + s \) using \( \text{lrtrans\_image}(2) \) by auto
with \( w(2) \) \( uv(1,3) \) as \( (2) \) \( st(2) \) have \( t \in U + s + U \) using \( \text{interval\_add}(4) \)
\( \text{lrtrans\_in\_group\_add}(2) \) by auto
with \( st \) have \( z \in \{ \langle s, t \rangle \in G \times G \cdot t \in U + s + U \} \) by auto

then have
\[
\text{sub}: \{ \langle s, t \rangle \in G \times G \cdot t \in W + s + W \} \subseteq \{ \langle s, t \rangle \in G \times G \cdot t \in U + s + U \}
\]
by auto

\[
\{ \text{fix } z \text{ assume } z: z \in \{ \langle s, t \rangle \in G \times G \cdot t \in W + s + W \} \}
\]
then obtain \( s \ t \) where \( \text{st}: z = \langle s, t \rangle \)
\( s \in G \) \( t \in G \) by auto
from \( \text{st}(1) \) \( z \) have \( \text{st2}: t \in W + s + W \) by auto
with \( \langle W \in N_0 \rangle \) obtain \( u \ v \) where \( \text{uv}: t = u + v \)
\( u \in W + s \) \( v \in W \)
using \( \text{interval\_add}(4) \) \( \text{lrtrans\_in\_group\_add}(2) \) \( st(2) \) by auto

from \( \langle W \subseteq G \rangle \) \( \langle s \in G \rangle \) \( \langle q \in W \rangle \) have \( q \in G \)
by auto

moreover
from \( \text{st \ groupAssum} \) have \( \text{P}(z) = \langle \text{GroupInv}(G, f)(s), \text{GroupInv}(G, f)(t) \rangle \)
using \( \text{prodFunctionApp} \) \( \text{group0\_2\_T2} \) by blast
with \( \text{st}(2,3) \) have \( \text{P}(z) = \langle -s, -t \rangle \) by auto
ultimately have \( \text{P}(z) \in \{ \langle s, t \rangle \in G \times G \cdot t \in W + s + W \} \)
using \( \text{st}(2,3) \) \( \text{inverse\_in\_group} \) by auto
with \( \text{sub \ have} \) \( \text{P}(z) \in \{ \langle s, t \rangle \in G \times G \cdot t \in U + s + U \} \) by force
with \( \text{as}(3) \) have \( \text{P}(z) \in V \) by force
with \( z \) \( L \) have \( z \in \text{P}\(-V\) \) using \( \text{prodFunction} \) \( \text{func1\_1\_L5A} \) \( \text{vimage\_iff} \)
by blast

}\)
with \( w(1) \) have \( \exists U \subseteq N_0. \{ s, t \} \in G \times G \cdot t \in U + s + U \subseteq \text{P}\(-V\) \)
by blast
with \( L \) show \( \text{P}\(-V \) \) \( \in \text{roelckeUniformity} \)
unfolding \( \text{roelckeUniformity\_def} \) using \( \text{prodFunction} \) \( \text{func1\_1\_L6A} \) by blast
qed
with L R show thesis using IsUniformlyCont_def by auto
qed
end

92  Topological groups 2

theory TopologicalGroup_ZF_2 imports Topology_ZF_8 TopologicalGroup_ZF
Group_ZF_2
begin

This theory deals with quotient topological groups.

92.1  Quotients of topological groups

The quotient topology given by the quotient group equivalent relation, has
an open quotient map.

theorem (in topgroup) quotient_map_topgroup_open:
  assumes IsAsubgroup(H,f) A ∈ T
defines r ≡ QuotientGroupRel(G,f,H)
shows {⟨b,r{b}⟩. b ∈ ∪ T}A ∈ (T{quotient by}r)
proof
  have eqT:equiv(∪ T,r) and eqG:equiv(G,r)
    using group0.Group_ZF_2_4_L3 assms(1)
    unfolding r_def IsAnormalSubgroup_def
    using group0_valid_in_tgroup by auto
  have subA:A ⊆ G using assms(2) by auto
  have subH:H ⊆ G using group0.group0_3_L2[OF group0_valid_in_tgroup assms(1)].
  have A1:{⟨b,r{b}⟩. b ∈ ∪ T}-(∪ T}A)=H+A
    proof
      { fix t assume t∈{⟨b,r{b}⟩. b ∈ ∪ T}-(∪ T}A)
      then have ∃ m∈(∪ T}A). ⟨t,m⟩∈{⟨b,r{b}⟩. b ∈ ∪ T}A
        unfolding image_iff by auto
      then obtain m where m∈(∪ T}A). ⟨t,m⟩∈{⟨b,r{b}⟩. b ∈ ∪ T}
        by auto
      then obtain b where b∈A{⟨b,m⟩∈{⟨b,r{b}⟩. b ∈ ∪ T}A} and rel:r{t}=m
        using image_iff by auto
      then have r{t}=m by auto
      then have r{t}=r{b} using rel by auto
      with ⟨b∈A:subA have ⟨t,b⟩∈r using eq_equiv_class[of _ walls] by auto
      then have f{t,GroupInv(G,f)b}∈H unfolding r_def QuotientGroupRel_def
        by auto
      then obtain h where h∈H and prd:f{t,GroupInv(G,f)b}=h by auto
      then have h∈G using subH by auto
      have b∈G using ⟨b∈A:subA by auto
      then have (-b)∈G using neg_in_tgroup by auto
      from prd have h=t+(-b) by simp
\[
\text{with } t \in \mathbb{G} \text{ have } t = h + b \text{ using inv_cancel_two_add(1) by simp}
\]
then have \((b, r(b), t) \in f\) using apply_Pair[OF topgroup_f_binop] \((b, r(b)) \in \mathbb{G}\) by auto
moreover from \((h, b) \in \mathbb{H} \times A\) have \((h, b) \in H \times A\) by auto
ultimately have \(t \in f(H \times A)\) using image_iff by auto
with \(A \subseteq H + A\) have \(t \in H + A\) using interval_add(2) by auto
\}
then show \(\{(b, r(b)), b \in \bigcup T\} - \{(b, r(b)), b \in (\bigcup T)A\} \subseteq H + A\) by force
\{ fix \(t\) assume \(t \in H + A\)
with \(A \subseteq H + A\) have \(t \in f(H \times A)\) using interval_add(2) by auto
then obtain \(h\) where \(h \in H \times A\) \(\langle h, t \rangle \in f\) using image_iff by auto
then obtain \(h, a\) where \(h = (h, a)\) \(h \in H + A\) by auto
then have \(h \in G \times A\) using \(subH \subseteq A\) by auto
from \(\langle h, t \rangle \in f\) have \(t \in G\) using topgroup_f_binop unfolding Pi_def by auto
from \(h = (h, a)\) \(\langle h, t \rangle \in f\) have \(t = h + a\) using applyEquality topgroup_f_binop unfolding Pi_def by auto
with \(b \in \mathbb{G}\) \(\langle a, a \in G\rangle \), have \(t + (-a) = h \) using inv_cancel_two_add(2)
by simp
with \(h \in \mathbb{H}\) \(\langle t, a \in \mathbb{G}\rangle \) have \(\langle t, a \rangle \in f\) unfolding \(r\) def QuotientGroupRel_def by auto
then have \(r(t) = r(a)\) using eqT equiv_class_eq by auto
with \(\langle a, a \in G\rangle \) have \(\langle a, r(t) \rangle \in \{(b, r(b)), b \in \bigcup T\}\) by auto
with \(\langle a, a \in A\rangle \) have \(A_1 : \mathbb{r} \cap \mathbb{t} \in \{(b, r(b)), b \in (\bigcup T)A\}\) using image_iff by auto
from \(\langle t, r(t) \rangle \in \{(b, r(b)), b \in \bigcup T\}\) by auto
with \(A_1 : \mathbb{r} \cap \mathbb{t} \in \{(b, r(b)), b \in (\bigcup T)A\}\) using vimage_iff by auto
\}
then show \(H + A \subseteq \{(b, r(b)), b \in \bigcup T\} - \{(b, r(b)), b \in (\bigcup T)A\}\) by auto
\quad have \(H + A = \bigcup_{x \in H} (x + A)\) using interval_add(3) subA subH by auto moreover
\quad have \(\forall x \in H, x + A \subseteq T\) using open_tr_open(1) assms(2) subH by blast
\quad then have \(\{x + A, x \in H\} \subseteq T\) by auto
\quad then have \(\bigcup_{x \in H} (x + A) \subseteq T\) using topSpaceAssum unfolding IsATopology_def by auto
ultimately have \(H + A \subseteq T\) by auto
\quad with \(A_1 : \mathbb{r} \cap \mathbb{t} \in \{(b, r(b)), b \in (\bigcup T)A\}\) by auto
\quad then have \(\{(b, r(b)), b \in (\bigcup T)A\} \subseteq \text{topSpaceAssum}(\text{quotient topology in}(\langle \bigcup T \rangle / \mathbb{r})\{(b, r(b)), b \in (\bigcup T)\})\) by auto
\quad using QuotientTop_def topSpaceAssum quotient_proj_surj using
\quad funcl_1_L6(2) [OF quotient_proj_fun] by auto
\quad then show \(\{(b, r(b)), b \in (\bigcup T)A\} \subseteq (\text{quotient by} \mathbb{r})\) using EquivQuo_def [OF eqT] by auto
qed
A quotient of a topological group is just a quotient group with an appropriate topology that makes product and inverse continuous.

**Theorem (in topgroup)** quotient_top_group_F_cont:

assumes IsAnormalSubgroup(G,f,H)

defines r ≡ QuotientGroupRel(G,f,H)

defines F ≡ QuotientGroupOp(G,f,H)

shows IsContinuous(ProductTopology(T{quotient by}r),T{quotient by}r),T{quotient by}r,F)

**Proof**

have eqT:equiv(⋃T,r) and eqG:equiv(G,r) using group0.Group_ZF_2_4_L3

assms(1) unfolding r_def IsAnormalSubgroup_def

using group0_valid_in_tgroup by auto

have have:⟨⟨b,c⟩⟩∈(⋃T×⋃T):G×G→(G//r)×(G//r) using productequivrel_fun unfolding G_def by auto

have C:Congruent2(r,f) using Group_ZF_2_4_L5A[OF Ggroup assms(1)] unfolding r_def.

with eqT have IsContinuous(ProductTopology(T,T),ProductTopology(T{quotient by}r),T{quotient by}r),T{quotient by}r,F)

using product_quo_fun unfolding G_def by auto

have tprod:topology0(ProductTopology(T,T))

using Top_1_4_T1(1)[OF topSpaceAssum topSpaceAssum].

have Hfun:{⟨⟨b,c⟩⟩. ⟨⟨b,c⟩⟩∈(⋃T×⋃T)}∈surj(⋃ProductTopology(T,T),⋃ProductTopology(T,T)) using prod_equiv_rel_surj total_quo_equi[OF eqT] topology0.total_quo_func[OF tprod prod_equiv_rel_surj] unfolding F_def QuotientGroupOp_def r_def by auto

have cc:(F O {⟨⟨b,c⟩⟩. ⟨⟨b,c⟩⟩∈(⋃T×⋃T)}):G→G//r using comp_fun[OF fun EquivClass_1_T1[OF eqG C]] unfolding F_def QuotientGroupOp_def r_def by auto

then have have:⟨⟨b,c⟩⟩∈(⋃T×⋃T):G×G→(G//r)×(G//r) using quotient_func_cont[OF quotient_proj_surj] unfolding G_def by auto

moreover have IsContinuous(ProductTopology(T,T),T{quotient by}r),T{quotient by}r,F)

ultimately have cont:IsContinuous(ProductTopology(T,T),T{quotient by}r),⟨⟨b,r(b)⟩⟩. b∈⋃T) 0 f)
fix $A$ assume $A : A \in G \times G$
then obtain $g_1$ $g_2$ where $A_{\text{def}} : A = \langle g_1, g_2 \rangle$ $g_1 \in G$ $g_2 \in G$ by auto
then have $fA = g_1 + g_2$ and $p : g_1 + g_2 \in \bigcup T$ unfolding $grop$ def using
apply_type[OF topgroup_f_binop] by auto
then have $\{ \langle b, r\{b\} \rangle. b \in \bigcup T \} (fA) = \{ \langle b, r\{b\} \rangle. b \in \bigcup T \} (g_1 + g_2)$ by auto
with $p$ have $\{ \langle b, r\{b\} \rangle. b \in \bigcup T \} (fA) = r\{g_1 + g_2\}$ using apply_equality[OF _ quotient_proj_fun]
by auto
then have $Pr_1 : (\{ \langle b, r\{b\} \rangle. b \in \bigcup T \} O f)A = r\{g_1 + g_2\}$ using comp_fun_apply[OF _ A]
by auto
moreover
note fun ultimately have $(F O \{ \langle b, r\{b\}, r\{c\} \rangle. \langle b, c \rangle \in \bigcup T \times \bigcup T \}) A = r\{g_1 + g_2\}$
using comp_fun_apply[OF _ A] by auto
then have $(F O \{ \langle b, c \rangle, \langle r\{b\}, r\{c\} \rangle \} A = \{ \langle b, r\{b\} \rangle. b \in \bigcup T \}$
0 $f) A$ using $Pr_1$ by auto
} then have $(F O \{ \langle b, c \rangle, \langle r\{b\}, r\{c\} \} A = \{ \langle b, r\{b\} \rangle. b \in \bigcup T \}$
0 $f)$ using fun_extension[OF cc comp_fun[OF topgroup_f_binop quotient_proj_fun]]
unfolding $F$ def QuotientGroupOp_def r_def by auto
then have $A : \text{IsContinuous}(\text{ProductTopology}(T, T), \text{T(quotient by)} r, F \ O \{ \langle b, c \rangle, \langle r\{b\}, r\{c\} \}$.
$\langle b, c \rangle \in \bigcup T \times \bigcup T \} )$ using cont by auto
have $\text{IsAsubgroup}(H, f)$ using assms(1) unfolding $\text{IsAnormalSubgroup}$ def by auto
then have $\forall A \in \text{T}. \{ \langle b, r \{b\} \rangle. b \in \bigcup T \} A \in \{ \text{quotient by} r \}$ using quotient_map_topgroup_open unfolding $r$ def by auto
with $\text{eqT}$ have $\text{ProductTopology}\{\{ \text{quotient by} r, \{ \text{quotient by} r \} \} = \{ \text{quotient}$
topology in}$(\bigcup T) / r \times (\bigcup T) / r ) \{ \text{by} \}{ \langle b, c \rangle, \{ r\{b\}, r\{c\} \} }$. $\langle b, c \rangle \in \bigcup T \times \bigcup T \} \{ \text{from} \} \{ \text{ProductTopology}\{\langle b, c \rangle, \{ r\{b\}, r\{c\} \} \}$
using prod_quotient
by auto
with $A$ show $\text{IsContinuous}(\text{ProductTopology}(T \{ \text{quotient by} r, T \{ \text{quotient by} r \} )$, $\{ \text{quotient by} r \}, F \}$
using two_top_spaces0.cont_quotient_top[OF two Hfun Ffun] topology0.total_quo_func[OF tprod prod_equiv_rel_surj] unfolding $F$ def QuotientGroupOp_def r_def
by auto
qed
lemma (in group0) Group_ZF_2_4_L8:
assumes IsAnormalSubgroup(G,P,H)
defines r ≡ QuotientGroupRel(G,P,H)
and F ≡ QuotientGroupOp(G,P,H)
shows GroupInv(G//r,F):G//r → G//r
using group0_2_T2[OF Group_ZF_2_4_T1[OF assms(1)]]
groupAssum using assms(2,3)
by auto

theorem (in topgroup) quotient_top_group_INV_cont:
assumes IsAnormalSubgroup(G,f,H)
defines r ≡ QuotientGroupRel(G,f,H)
defines F ≡ QuotientGroupOp(G,f,H)
shows IsContinuous(T{quotient by}r,T{quotient by}r,GroupInv(G//r,F))
proof-
  have eqT:equiv(⋃T,r) and eqG:equiv(G,r) using group0.Group_ZF_2_4_L3
  unfolding r_def IsAnormalSubgroup_def
  using group0_valid_in_tgroup by auto
  have two:two_top_spaces0(T,T{quotient by}r,{⟨b,r{b}⟩. b ∈ G}) unfolding two_top_spaces0_def
  using topSpaceAssum equiv_quo_is_top[OF eqT] quotient_proj_fun total_quo_equi[OF eqT] by auto
  have IsContinuous(T,T,GroupInv(G,f)) using inv_cont.
  moreover
  { fix g assume G:g∈G
    then have GroupInv(G,f)g=-g using grinv_def by auto
    then have r({GroupInv(G,f)g})=GroupInv(G//r,F)(r{g}) using group0.Group_ZF_2_4_L7
      [OF group0_valid_in_tgroup assms(1) G] unfolding r_def F_def by auto
    then have {⟨b,r{b}⟩. b∈G}O GroupInv(G,f)=GroupInv(G//r,F)O {⟨b,r{b}⟩. b∈G} using comp_fun_apply[OF quotient_proj_fun G] comp_fun_apply[OF group0_2_T2[OF Ggroup] G] by auto
  } then have A1:{⟨b,r{b}⟩. b∈G}O GroupInv(G,f)=GroupInv(G//r,F)O {⟨b,r{b}⟩. b∈G} using fun_extension[OF comp_fun[OF quotient_proj_fun_group0.Group_ZF_2_4_L8[OF group0_valid_in_tgroup assms(1)]]]
    unfolding r_def F_def by auto
  have IsContinuous(T,T{quotient by}r,{⟨b,r{b}⟩. b∈⋃T}) using quotient_func_cont[OF quotient_proj_surj]
    unfolding EquivQuo_def[OF eqT] by auto
  ultimately have IsContinuous(T,T{quotient by}r,{⟨b,r{b}⟩. b∈⋃T})O GroupInv(G,f)
using comp_cont by auto
with A1 have IsContinuous(T,T{quotient by}r,GroupInv(G//r,F)0 {(b,r{b})}. b∈G) by auto
then have IsContinuous({quotient topology in}(∪T) // r{by}{{b, r {b}}. b ∈ ∪T}{from}T,T{quotient by}r,GroupInv(G//r,F))
using two_top_spaces0.cont_quotient_top[OF two quotient_proj_surj, of GroupInv(G//r,F)r] group0.Group_ZF_2_4_L8[OF group0.valid_in_tgroup assms(1)]
using total_quo_equi[OF eqT] unfolding r_def F_def by auto
then show thesis unfolding EviqQuo_def[OF eqT].
qed

Finally we can prove that quotient groups of topological groups are topological groups.

theorem (in topgroup) quotient_top_group:
assumes IsAnormalSubgroup(G,f,H)
defines r ≡ QuotientGroupRel(G,f,H)
defines F ≡ QuotientGroupOp(G,f,H)
shows IsAtopologicalGroup({quotient by}r,F)
unfolding IsAtopologicalGroup_def using total_quo_equi equiv_quo_is_top
Group_ZF_2_4_T1 Ggroup assms(1) quotient_top_group_INV_cont quotient_top_group_F_cont
Group_ZF_2_4_L8[OF group0.valid_in_tgroup assms(1)] unfolding r_def
F_def IsAnormalSubgroup_def
by auto

end

93 Topological groups 3

theory TopologicalGroup_ZF_3 imports Topology_ZF_10 TopologicalGroup_ZF_2 TopologicalGroup_ZF_1 Group_ZF_4
begin
This theory deals with topological properties of subgroups, quotient groups and relations between group theoretical properties and topological properties.

93.1 Subgroups topologies
The closure of a subgroup is a subgroup.

theorem (in topgroup) closure_subgroup:
assumes IsAsubgroup(H,f)
shows IsAsubgroup(cl(H),f)
proof-
have two:two_top_spaces0(ProductTopology(T,T),T,f) unfolding two_top_spaces0_def
using
from fcon have cont: IsContinuous(ProductTopology(T,T),T,f) by auto 
then have closed: \( \forall D.\ D\text{ is closed in } T \rightarrow f^{-1}(D)\text{ is closed in } T \) using two_top_spaces0.TopZF_2_1_L1 
  two by auto 
then have closure: \( \forall A \in \text{Pow}(\bigcup T).\ f(Closure(A,T))\subseteq cl(fA) \) using two_top_spaces0.TopZF_2_1_L1 
  two by force 
have sub1: \( H \subseteq G \) using group0.group0_3_L2 group0_valid_in_tgroup assms 
by force 
then have closure: \( \forall A \in \text{Pow}(\bigcup T).\ GroupInv(G,f)(cl(A))\subseteq cl(GroupInv(G,f)A) \) using two_top_spaces0.Top_ZF_2_1_L1 
  two by force 
moreover have GroupInv(H,restrict(f,H\times H)):H\times H\rightarrow H using assms unfolding IsAsubgroup_def 
with sub1 have Inv:GroupInv(G,f)(cl(H))\subseteq cl(GroupInv(G,f)H) by auto 
ultimately have A1: \( cl(H)\{\text{is closed under}\ f \) unfolding IsOpClosed_def by auto 
have two: two_top_spaces0(T,T,GroupInv(G,f)) unfolding two_top_spaces0_def using 
  topSpaceAssum Ggroup group0_2_T2 by auto 
from inv_cont have cont: IsContinuous(T,T,GroupInv(G,f)) by auto 
then have closed: \( \forall D.\ D\text{ is closed in } T \rightarrow GroupInv(G,f)^{-1}(D)\text{ is closed in } T \) using two_top_spaces0.TopZF_2_1_L1 
  two by auto 
then have closure: \( \forall A \in \text{Pow}(\bigcup T).\ GroupInv(G,f)(cl(A))\subseteq cl(GroupInv(G,f)A) \) using two_top_spaces0.Top_ZF_2_1_L1 
  two by force 
moreover have GroupInv(H,restrict(f,H\times H)):H\rightarrow H using assms unfolding IsAsubgroup_def
using group0_2_T2 by auto then 
  have GroupInv(H,restrict(f,H×H))H⊆H using func1_1_L6(2) by auto 
  then have restrict(\text{GroupInv}(G,f),H)H⊆H using group0.group0_3_T1 assms 
  group0_valid_in_tgroup by auto 
  then have sss:GroupInv(G,f)H⊆H using restrict_image by auto 
  then have H⊆G GroupInv(G,f)H⊆G using sub1 by auto 
  with sub1 sss have cl(GroupInv(G,f)H)⊆cl(H) using top_closure_mono 
  by auto ultimately 
  have \text{img}(\text{GroupInv}(G,f)(cl(H))⊆cl(H) by auto 
  \{ 
    \text{fix} x assume x∈cl(H) moreover 
    have GroupInv(G,f)(cl(H))={\text{GroupInv}(G,f)t. t∈cl(H)} using func_imagedef 
  \}
  G group0_2_T2
  clHG by force ultimately 
  have GroupInv(G,f)x∈GroupInv(G,f)(cl(H)) by auto 
  with \text{img} have GroupInv(G,f)x∈cl(H) by auto 
  \}
  then have A2:∀x∈cl(H). \text{GroupInv}(G,f)x∈cl(H) by auto 
  from assms have H≠∅ using group0.group0_3_L5 group0_valid_in_tgroup 
  by auto moreover 
  have H⊆cl(H) using cl_contains_set sub1 by auto ultimately 
  have cl(H)≠∅ by auto 
  with clHG A2 A1 show thesis using group0.group0_3_T3 group0_valid_in_tgroup 
  by auto 
  qed 

The closure of a normal subgroup is normal. 

\text{theorem (in topgroup) normal_subg:} 
\text{assumes IsAnormalSubgroup}(G,f,H) 
\text{shows IsAnormalSubgroup}(G,f,cl(H)) 
\text{proof-} 
\text{have A:IsASubgroup}(cl(H),f) using closure_subgroup assms unfolding IsAnormalSubgroup_def 
\text{by auto} 
\text{have sub1:H⊆G using group0.group0_3_L2 group0_valid_in_tgroup assms} 
\text{unfolding IsANormalSubgroup_def by auto} 
\text{then have sub2:cl(H)⊆G using Top_3_L11(1) by auto} 
\{ 
  \text{fix g assume g:}\subset G 
  \text{then have cl1:cl(g+H)=g+cl(H) using trans_closure sub1 by auto} 
  \text{have ss:g+cl(H)⊆G unfolding ltrans_def LeftTranslation_def by auto} 
  \text{have g+H⊆G unfolding ltrans_def LeftTranslation_def by auto} 
  \text{moreover from g have (-g)∈G using neg_in_tgroup by auto} 
  \text{ultimately have cl2:cl((g+H)+(-g))=cl(g+H)+(-g) using trans_closure2} 
  \text{by auto} 
  \text{with cl1 have clcon:cl((g+H)+(-g))=(g+(cl(H)))+(-g) by auto} 
  \{ 
    \text{fix r assume r∈(g+H)+(-g)} 
    \text{then obtain q where q:q∈g+H r=q+(-g) unfolding rtrans_def RightTranslation_def} 
    \text{by force} 
  \} 

1364
from q(1) obtain h where h∈H q=g+h unfolding ltrans_def LeftTranslation_def by auto
  with q(2) have r=(g+h)+(-g) by auto
  with ⟨h∈H⟩ ⟨g∈G⟩ ⟨(-g)∈G⟩ have r∈H using asms unfolding IsAnormalSubgroup_def
  grinv_def grop_def by auto
then have (g+H)+(-g)⊆H by auto
moreover then have (g+H)+(-g)⊆GH⊆G using sub1 by auto ultimately have c1((g+H)+(-g))⊆c1(H) using top_closure_mono by auto
with clcon have (g+(c1(H)))+(-g)⊆c1(H) by auto moreover
  have (g+(d-g))+(-g)∈H using assms unfolding IsAnormalSubgroup_def
  grsub_def grop_def by auto
ultimately have cl((g+H)+(-g))⊆cl(H) using top_closure_mono by auto
with clcon have (g+(cl(H)))+(-g)⊆cl(H) by auto
moreover
  fix b assume b∈{g+(d-g). d∈cl(H)}
then obtain d where d:d∈cl(H) b=g+(d-g) by auto moreover
then have g+d∈G using group0.group_op_closed[OF group0_valid_in_tgroup ⟨g∈G⟩] by auto
from d(2) have b:b=(g+d)-g using group0.group_oper_assoc[OF group0_valid_in_tgroup ⟨g∈G⟩ ⟨d∈G⟩ ⟨(-g)∈G⟩] unfolding grsub_def grop_def grinv_def by blast
have (g+d)=LeftTranslation(G,f,g)d using group0.group0_5_L2(2)[OF group0_valid_in_tgroup ⟨g∈G⟩ ⟨d∈G⟩ ⟨(-g)∈G⟩] unfolding ltrans_def using func_imagedef[OF group0_valid_in_tgroup ⟨g∈G⟩ ⟨d∈G⟩] by auto
with ⟨d∈cl(H)⟩ have g+d∈G+cl(H) unfolding ltrans_def using func_imagedef[OF group0_valid_in_tgroup ⟨g∈G⟩ ⟨d∈G⟩] by auto
moreover from b have b=RightTranslation(G,f,-g)(g+d) using group0.group0_5_L2(1)[OF group0_valid_in_tgroup ⟨g∈G⟩ ⟨d∈G⟩ ⟨(-g)∈G⟩] unfolding rtrans_def using func_imagedef[OF group0_valid_in_tgroup ⟨g∈G⟩ ⟨d∈G⟩ ⟨(-g)∈G⟩] ss] by force
ultimately have {g+(d-g). d∈cl(H)}⊆cl(H) by force
then show thesis using A group0.cont_conj_is_normal[OF group0_valid_in_tgroup, of cl(H)] unfolding grsub_def grinv_def grop_def by auto
qed

Every open subgroup is also closed.

theorem (in topgroup) open_subgroup_closed:
  assumes IsAsubgroup(H,f) H∈T
  shows H{is closed in}T
proof-
  from asms(1) have sub:H⊆G using group0.group0_3_L2 group0_valid_in_tgroup by force
  { fix t assume t∈G-H then have t∈H:t∈H and t∈G by auto


from assms(1) have sub:H⊆G using group0.group0_3_L2 group0_valid_in_tgroup by force
from assms(1) have nSubG:0∈H using group0.group0_3_L5 group0_valid_in_tgroup by auto

fix x assume x∈(t+H)∩H
then obtain u where x=t+u u∈H x∈H unfolding ltrans_def LeftTranslation_def by auto
then have u∈Gx∈Gt∈G using sub tG by auto
with ⟨x=t+u⟩ have x+(−u)=t using group0.group0_2_L18(1) group0_valid_in_tgroup unfolding grop_def grinv_def by auto
from ⟨u∈H⟩ have (−u)∈H unfolding grinv_def using assms(1) group0.group0_3_T3A group0_valid_in_tgroup by auto
with ⟨x+(−u)∈H⟩ have False using tnH by auto

then have (t+H)∩H=0 by auto moreover
have t+H⊆G unfolding ltrans_def LeftTranslation_def by auto ultimately
have (t+H)⊆G−H by auto
with tp P have ∃V∈T. t∈V ∧ V⊆G−H unfolding Bex_def by auto

then have ∀t∈G−H. ∃V∈T. t∈V ∧ V⊆G−H by auto
then have G−H∈T using open_neigh_open by auto
then show thesis unfolding IsClosed_def using sub by auto qed

Any subgroup with non-empty interior is open.

theorem (in topgroup) clopen_or_emptyInt:
assumes IsAsubgroup(H,f) int(H)≠0
shows H∈T
proof-
from assms(1) have sub:H⊆G using group0.group0_3_L2 group0_valid_in_tgroup by force

fix h assume h∈H
have intsub:int(H)⊆H using Top_2_L1 by auto
from assms(2) obtain u where u∈int(H) by auto
with intsub have u∈H by auto
then have (−u)∈H unfolding grinv_def using assms(1) group0.group0_3_T3A group0_valid_in_tgroup by auto
with ⟨h∈H⟩ have h−u∈H unfolding grop_def using assms(1) group0.group0_3_L6
In conclusion, a subgroup is either open or has empty interior.

corollary (in topgroup) emptyInterior_xor_op:
  assumes IsAsubgroup(H,f)
  shows (int(H)=0) Xor (H T{is connected})
  unfolding Xor_def using clopen_or_emptyInt assms Top_2.L3
  group0.group0_3_L5 group0_valid_in_tgroup by force

Then no connected topological groups has proper subgroups with non-empty interior.

corollary (in topgroup) connected_emptyInterior:
  assumes IsAsubgroup(H,f) T{is connected}
shows \((\text{int}(H)=0) \text{ Xor } (H=G)\)

proof-
  have \((\text{int}(H)=0) \text{ Xor } (H\in T)\) using emptyInterior_xor_op assms(1) by auto
  moreover
  \begin{enumerate}
  \item assume \(H\in T\) moreover
    then have \(H\text{ is closed in } T\) using open_subgroup_closed assms(1) by auto
    ultimately
    have \(H=0\lor H=G\) using assms(2) unfolding IsConnected_def by auto
    then have \(H=G\) using group0.group0_3_L5 group0_valid_in_tgroup assms(1)
    by auto
  \end{enumerate}
  } moreover
  have \(G\in T\) using topSpaceAssum unfolding IsATopology_def G_def by auto
  ultimately show thesis unfolding Xor_def by auto
qed

Every locally-compact subgroup of a \(T_0\) group is closed.

*theorem (in topgroup) loc_compact_T0_closed:
  assumes \(\text{IsAsubgroup}(H,f) \ (T\text{ restricted to } H)\{\text{is locally-compact} \ T\{\text{is } T_0}\)
  shows \(H\{\text{is closed in} \ T\)
proof-
  from assms(1) have clsub: \(\text{IsAsubgroup}(\text{cl}(H),f)\) using closure_subgroup by auto
  then have subcl: \(\text{cl}(H) \subseteq G\) using group0.group0_3_L2 group0_valid_in_tgroup by force
  from assms(1) have sub: \(H \subseteq G\) using group0.group0_3_L2 group0_valid_in_tgroup by force
  from assms(3) have \(T\{\text{is } T_2}\) using T1_imp_T2 neu_closed_imp_T1 T0_imp_neu_closed by auto
  then have \(\bigcup (T\text{ restricted to } H)\{\text{is } T_2}\) using T2_here sub by auto
  have tot: \(\bigcup (T\text{ restricted to } H) = H\) using unfolding RestrictedTo_def by auto
  with assms(2) have \(\forall x \in H. \exists A \in \text{Pow}(H). A \{\text{is compact in} \ (T\text{ restricted to } H) \wedge x \in \text{Interior}(A, (T\text{ restricted to } H))\) using topology0.locally_compact_exist_compact_neig[of T{restricted to}H]
  Top_1_L4 unfolding topology0_def
  by auto
  then obtain \(K\) where \(K:K \subseteq H\{\text{is compact in} \ (T\text{ restricted to } H)\)0\in\text{Interior}(K,(T\text{ restricted to } H))\)
  using group0.group0_3_L5 group0_valid_in_tgroup assms(1) unfolding gzero_def by force
  from \(K(1,2)\) have \(K\{\text{is compact in} \ T\) using compact_subspace_imp_compact by auto
  with \(\langle T\{\text{is } T_2}\rangle\) have \(Kc1:K\{\text{is closed in } T\) using in_t2_compact_is_cl by auto
  have \(\text{Interior}(K,(T\text{ restricted to } H))\subseteq(T\text{ restricted to } H)\) using topology0.Top_2_L2 unfolding topology0_def
  using Top_1_L4 by auto

1368
then obtain \( U \) where \( U \in \text{Interior}(K, (T\text{restricted to}H)) = H \cap U \) unfolding \( \text{RestrictedTo_def} \) by auto
then have \( H \cap U \subseteq K \) using topology0.Top_2_L1[of \( T\text{restricted to}H \)] unfolding topology0_def using Top_1_L4 by force
moreover have \( U_2 \subseteq U \cup K \) by auto
have \( ksub : K \subseteq H \) using tot K(2) unfolding IsCompact_def by auto
ultimately have \( int : H \cap (U \cup K) = K \) by auto
from \( U(2) \cap K(3) \) have \( 0 \in U \) by auto
with \( U(1) \) have \( 0 \in int(U \cup K) \) using Top_2_L6 by auto
then have \( U \cup K \in N_0 \) unfolding zerohoods_def by auto
then obtain \( V \) where \( V \subseteq U \cup K \) \( V \in N_0 \) \( V + V \subseteq U \) unfolding exists_procls_zerohood[of \( U \cup K \)] by auto

\[
\begin{align*}
\text{fix } h & \text{ assume } A:S : h \in c1(H) \\
\text{with } clsub & \text{ have } (-h) \in c1(H) \text{ using group0.group0_3_T3A group0_valid_in_tgroup} \\
\text{by auto moreover} & \text{ then have } (-h) \in G \text{ using subcl by auto} \\
\text{with } V(2) & \text{ have } (-h) \in int((-h) + V) \text{ using elem_in_int_ltrans by auto} \\
\text{ultimately} & \text{ have } (-h) \in (c1(H)) \cap (int((-h) + V)) \text{ by auto moreover} \\
\text{have } int((-h) + V) & \subseteq T \text{ using Top_2_L2 by auto moreover} \\
\text{note sub ultimately} & \text{ have } H \cap int((-h) + V) \neq 0 \text{ using cl_inter_neigh by auto moreover} \\
\text{from } & (-h) \in G \text{ V(2) have } int((-h) + V) = (-h) + int(V) \text{ unfolding zerohoods_def} \\
& \text{ using ltrans_interior by force} \\
\text{ultimately have } & H \cap int((-h) + V) \neq 0 \text{ by auto} \\
then obtain & y where y : y \in H \cap (-h) + int(V) \text{ by blast} \\
then obtain & v where v : v \in int(V) \text{ y = (-h) + v unfolding ltrans_def LeftTranslation_def by auto} \\
& \text{with } (-h) \in G \text{ V(2) y(1) sub have } v \in G \cap G \in G \text{ using Top_2_L1[of V]} \\
& \text{unfolding zerohoods_def by auto} \\
& \text{with } v(2) \text{ have } (-h) + y = y \text{ using group0.group0_2_L18(2) group0_valid_in_tgroup} \\
& \text{unfolding group_def grinv_def by auto moreover} \\
& \text{have } h \in G \text{ using AS subcl by auto} \\
& \text{then have } (-h) = h \text{ using group0.group_inv_of_inv group0_valid_in_tgroup by auto ultimately} \\
& \text{have } h + y = v \text{ by auto} \\
& \text{with } v(1) \text{ have hyV : h + y \in int(V) by auto} \\
& \text{have } y \in c1(H) \text{ using y(1) cl_contains_set sub by auto} \\
& \text{with AS have hycl : h + y \in c1(H) using c1sub group0.group0_3_L6 group0_valid_in_tgroup by auto} \\
& \text{with } W \subseteq W \in Th + y \subseteq W \text{ by auto moreover} \\
& \text{from } W(1) \text{ have } int(V) \cap W \subseteq T \text{ using Top_2_L2 topSpaceAssum unfolding IsATopology_def by auto moreover} \\
& \text{note hycl sub}
\end{align*}
\]
ultimately have \((\text{int} (V) \cap W) \cap H \neq 0\) using \text{cl_inter_neigh}[\text{of} \ \text{Hint}(V) \cap W] + y\) by auto
then have \(V \cap W \cap H \neq 0\) using \(\text{Top}_2\_L1\) by auto
with \(V(1)\) have \((U \cup K) \cap W \cap H \neq 0\) by auto
then have \(\langle H \rangle \cap (U \cup K) \cap W \neq 0\) by auto
with \(\text{int}\) have \(K \cap W \neq 0\) by auto
}
then have \(\forall W \in T. h+y \in W \rightarrow K \cap W \neq 0\) by auto
moreover have \(K \subseteq G h+y \in G\) using \(\text{sub}\) by auto
ultimately have \(h+y \in \text{cl}(K)\) using \(\text{inter_neigh_cl}[\text{of} \ K h+y]\)
unfolding \(\text{G_def}\) by force
then have \(h+y \in K\) using \(\text{Kcl}\ \text{Top}_3\_L8\) \(\langle K \subseteq G\rangle\) by auto
with \(\text{ksub}\) have \(h+y \in H\) by auto
moreover from \(y(1)\) have \((-y) \in H\) using \(\text{group0.group0_3_T3A assms}(1)\)
\(\text{group0_valid_in_tgroup}\) by auto
ultimately have \((h+y)-y \in H\) unfolding \(\text{grsub_def}\) using \(\text{group0.group0_3_L6}\)
\(\text{group0_valid_in_tgroup}\) \(\text{assms}(1)\) by auto
moreover have \((-y) \in G\) using \(\langle -y \in H\rangle\) sub by auto
then have \(h+(-y) = (h+y)-y\) using \(\langle y \in G \land h \in G\rangle\) \(\text{group0.group_oper_assoc}\)
\(\text{group0_valid_in_tgroup}\) unfolding \(\text{grsub_def}\) by auto
then have \(h=(h+y)-y\) using \(\text{group0.group0_2_L6}\) \(\text{group0_valid_in_tgroup}\)
\(\langle y \in G\rangle\) unfolding \(\text{grsub_def}\) \(\text{grinv_def}\) \(\text{gzero_def}\) by auto
then have \(h+0=(h+y)-y\) using \(\text{group0.group0_2_L6}\) \(\text{group0_valid_in_tgroup}\)
\(\langle y \in G\rangle\) unfolding \(\text{gzero_def}\) by auto
ultimately have \(h \in H\) by auto
}
then have \(\text{cl}(H) \subseteq H\) by auto
then have \(H = \text{cl}(H)\) using \(\text{cl_contains_set}\) sub by auto
then show thesis using \(\text{Top}_3\_L8\) sub by auto
qed

We can always consider a factor group which is \(T_2\).

\text{theorem (in topgroup)} \text{factor_haus}:  
shows \((\text{T(quotient by)QuotientGroupRel}(G,f,cl(\{0\}))){\text{is } T_2}\)
\text{proof-}
  let \(r=\text{QuotientGroupRel}(G,f,cl(\{0\}))\)
  let \(f=\text{QuotientGroupOp}(G,f,cl(\{0\}))\)
  let \(i=\text{GroupInv}(G//r,f)\)
  have \(\text{IsAnormalSubgroup}(G,f,\{0\})\) using \(\text{group0.trivial_normal_subgroup}\)
\(\text{Ggroup}\) unfolding \(\text{group0_def}\)
  by auto
  then have \(\text{normal}: \text{IsAnormalSubgroup}(G,f,cl(\{0\}))\) using \(\text{normal_subg}\) by auto
  then have \(\text{eq:equiv}(\bigcup T, r)\) using \(\text{group0.Group_ZF_2_4_L3}[\text{OF } \text{group0_valid_in_tgroup}]\)
  unfolding \(\text{IsAnormalSubgroup_def}\) by auto
  then have \(\text{tot:}\bigcup (\text{T(quotient by)}:r) = G//r\) using \(\text{total_quo_equi}\) by auto

1370
have neu: r(0) = \text{TheNeutralElement}(G//r) using Group_ZF_2_4_L5B[OF Ggroup normal] by auto
then have r(0) \in G//r using group0.group0_2_L2 Group_ZF_2_4_T1[OF Ggroup normal] unfolding group0_def by auto
then have sub: {r(0)} \subseteq G//r by auto
then have sub: {r(0)} \subseteq \bigcup (T\{\text{quotient by}\}r) using tot by auto
have zG: \emptyset \subseteq G//r using group0.group0_2_L2[OF group0_valid_in_tgroup] by auto
from zG have cla: r(0) \in G//r unfolding quotient_def by auto
let x = G//r-{r(0)}
{ fix s assume A: s \in \bigcup (G//r-{r(0)})
  then obtain U where s \in U \subseteq G//r-{r(0)} by auto
  then have U \subseteq G//r-{r(0)} by auto
  then have s \in \bigcup (G//r)-(r(0)) by auto
} moreover
{ fix s assume A: s \in (G//r)-(r(0))
  then obtain U where s \in U \subseteq G//r-{r(0)} by auto
  then have s \in \bigcup (G//r)-(r(0)) by auto
} ultimately have \bigcup (G//r-{r(0)}) = \bigcup (G//r)-(r(0)) by auto
then have A: \bigcup (G//r-(r(0))) = G-r(0) using Union_quotient eq by auto
{ fix s assume A: s \in (r(0))
  then have (0,s) \in r by auto
  then have (s,0) \in r using eq unfolding equiv_def sym_def by auto
  then have s \in cl({0}) using group0.Group_ZF_2_4_L5C[OF group0_valid_in_tgroup]
} unfolding QuotientGroupRel_def by auto
moreover
{ fix s assume A: s \in cl({0})
  then have (s,0) \in r using group0.Group_ZF_2_4_L5C[OF group0_valid_in_tgroup]
} A by auto
then have (0,s) \in r using eq unfolding equiv_def sym_def by auto
then have s \in r(0) by auto
ultimately have r(0) = cl({0}) by blast
with A have \bigcup (G//r-(r(0))) = G-cl({0}) by auto
moreover have cl({0}) \in \text{is closed in} T using cl_is_closed zG by auto
ultimately have \bigcup (G//r-(r(0))) \in T unfolding IsClosed_def by auto
then have (G//r-(r(0))) \in \{\text{quotient by}\}r using quotient_equiv_rel eq by auto
then have (\bigcup (T\{\text{quotient by}\}r)-(r(0))) \in \{\text{quotient by}\}r using total_quo_equi[OF
moreover from sub1 have \( \{r(0)\} \subseteq (\bigcup (T \text{quotient by} r)) \) using total_quo_equi[OF eq] by auto
ultimately have \( \{r(0)\}\{\text{is closed in}\}(T \text{quotient by} r) \) unfolding IsClosed_def by auto
then have \( \{\text{TheNeutralElement}(G//r,f)\}\{\text{is closed in}\}(T \text{quotient by} r) \) using neu by auto
then have \( T \{\text{quotient by} r}\{\text{is T}_1\} \) using topgroup.neu_closed_imp_T1[OF topGroupLocale[OF quotient_top_group[OF normal]]]
then show thesis using topgroup.T1_imp_T2[OF topGroupLocale[OF quotient_top_group[OF normal]]] by auto
qed

end

94 Metamath introduction

theory MMI_prelude imports Order_ZF_1

begin

Metamath's set.mm features a large (over 8000) collection of theorems proven in the ZFC set theory. This theory is part of an attempt to translate those theorems to Isar so that they are available for Isabelle/ZF users. A total of about 1200 assertions have been translated, 600 of that with proofs (the rest was proven automatically by Isabelle). The translation was done with the support of the mmisar tool, whose source is included in the IsarMathLib distributions prior to version 1.6.4. The translation tool was doing about 99 percent of work involved, with the rest mostly related to the difference between Isabelle/ZF and Metamath metalogics. Metamath uses Tarski-Megill metalogic that does not have a notion of bound variables (see http://planetx.cc.vt.edu/AsteroidMeta/Distinctors_vs_binders for details and discussion). The translation project is closed now as I decided that it was too boring and tedious even with the support of mmisar software. Also, the translated proofs are not as readable as native Isar proofs which goes against IsarMathLib philosophy.

94.1 Importing from Metamath - how is it done

We are interested in importing the theorems about complex numbers that start from the "recnt" theorem on. This is done mostly automatically by the mmisar tool that is included in the IsarMathLib distributions prior to version 1.6.4. The tool works as follows:
First it reads the list of (Metamath) names of theorems that are already imported to IsarMathlib ("known theorems") and the list of theorems that are intended to be imported in this session ("new theorems"). The new theorems are consecutive theorems about complex numbers as they appear in the Metamath database. Then mmisar creates a "Metamath script" that contains Metamath commands that open a log file and put the statements and proofs of the new theorems in that file in a readable format. The tool writes this script to a disk file and executes metamath with standard input redirected from that file. Then the log file is read and its contents converted to the Isar format. In Metamath, the proofs of theorems about complex numbers depend only on 28 axioms of complex numbers and some basic logic and set theory theorems. The tool finds which of these dependencies are not known yet and repeats the process of getting their statements from Metamath as with the new theorems. As a result of this process mmisar creates files new_thms.thy, new_deps.thy and new_known_thms.txt. The file new_thms.thy contains the theorems (with proofs) imported from Metamath in this session. As a result of this process mmisar creates files new_thms.thy, new_deps.thy and new_known_thms.txt. The file new_thms.thy contains the theorems (with proofs) imported from Metamath in this session. These theorems are added (by hand) to the current MMI_Complex_ZF_x.thy file. The file new_deps.thy contains the statements of new dependencies with generic proofs "by auto". These are added to the MMI_logic_and_sets.thy. Most of the dependencies can be proven automatically by Isabelle. However, some manual work has to be done for the dependencies that Isabelle cannot prove by itself and to correct problems related to the fact that Metamath uses a metalogic based on distinct variable constraints (Tarski-Megill metalogic), rather than an explicit notion of free and bound variables.

The old list of known theorems is replaced by the new list and mmisar is ready to convert the next batch of new theorems. Of course this rarely works in practice without tweaking the mmisar source files every time a new batch is processed.

94.2 The context for Metamath theorems

We list the Metamath’s axioms of complex numbers and define notation here.

The next definition is what Metamath $X \in V$ is translated to. I am not sure why it works, probably because Isabelle does a type inference and the "$=$" sign indicates that both sides are sets.

definition
IsASet :: i ⇒ o (_ isASet [90] 90) where
  IsASet_def[simp]: X isASet ≡ X = X

The next locale sets up the context to which Metamath theorems about complex numbers are imported. It assumes the axioms of complex numbers and defines the notation used for complex numbers.
One of the problems with importing theorems from Metamath is that Metamath allows direct infix notation for binary operations so that the notation \( afb \) is allowed where \( f \) is a function (that is, a set of pairs). To my knowledge, Isar allows only notation \( f(a, b) \) with a possibility of defining a syntax say \( a + b \) to mean the same as \( f(a, b) \) (please correct me if I am wrong here). This is why we have two objects for addition: one called \texttt{caddset} that represents the binary function, and the second one called \texttt{ca} which defines the \( a + b \) notation for \( \texttt{caddset}(a, b) \). The same applies to multiplication of real numbers.

Another difficulty is that Metamath allows to define sets with syntax \( \{ x \mid p \} \) where \( p \) is some formula that (usually) depends on \( x \). Isabelle allows the set comprehension like this only as a subset of another set i.e. \( \{ x \in A. p(x) \} \).

This forces us to have a slightly different definition of (complex) natural numbers, requiring explicitly that natural numbers is a subset of reals. Because of that, the proofs of Metamath theorems that reference the definition directly cannot be imported.

```isar
locale MMIsar0 = 
  fixes real (\( R \))
  fixes complex (\( \mathbb{C} \))
  fixes one (1)
  fixes zero (0)
  fixes iunit (i)
  fixes caddset (+)
  fixes cmulset (·)
  fixes lessrrel (\(<_R\))
  fixes ca (infixl + 69)
  defines ca_def: \( a + b \equiv +\langle a, b \rangle \)
  fixes cm (infixl · 71)
  defines cm_def: \( a \cdot b \equiv \cdot\langle a, b \rangle \)
  fixes sub (infixl - 69)
  defines sub_def: \( a - b \equiv \bigcup \{ x \in \mathbb{C}. b + x = a \} \)
  fixes cneg (-_ 95)
  defines cneg_def: \( - a \equiv 0 - a \)
  fixes cdiv (infixl / 70)
  defines cdiv_def: \( a / b \equiv \bigcup \{ x \in \mathbb{C}. b \cdot x = a \} \)
  fixes cpnf (+∞)
  defines cpnf_def: \(+\infty \equiv \mathbb{C} \)
  fixes cmnf (-∞)
  defines cmnf_def: \(-\infty \equiv \mathbb{C} \)
  fixes cxr (\( \ast \))
  defines cxr_def: \( \ast \equiv \bigcup \{ n \in \text{Pow}(\mathbb{N}).1 \in n \land (\forall n. n \in N \rightarrow n + 1 \in N) \} \)
  fixes cxn (N)
  defines cxn_def: \( N \equiv \bigcap \{ N \in \text{Pow}(R).1 \in N \land (\forall n. n \in N \rightarrow n + 1 \in N) \} \)
  fixes lessr (infix \(<_R\) 68)
  defines lessr_def: \( a <_R b \equiv \langle a, b \rangle \in <_R \)
  fixes cltrrset (<)
```
defines \( \text{cltrrset_def} : \) 
\[
\equiv (\lhd R \cap R \times R) \cup \{(-\infty, +\infty)\} \cup (R \times \{+\infty\}) \cup \{(-\infty) \times R\}
\]
fixes \( \text{cltrr} \) (infix \( \lhd \) 68) 
defines \( \text{cltrr_def} : \) \( a \lhd b \equiv (a, b) \in \) 
fixes \( \text{convcltrr} \) (infix \( > \) 68) 
defines \( \text{convcltrr_def} : \) \( a > b \equiv (a, b) \in \text{converse}(\lhd) \) 
fixes \( \text{lsq} \) (infix \( \leq \) 68) 
defines \( \text{lsq_def} : \) \( a \leq b \equiv \neg (b < a) \) 
fixes \( \text{two} \) (2) 
defines \( \text{two_def} : \) \( 2 \equiv 1 + 1 \) 
fixes \( \text{three} \) (3) 
defines \( \text{three_def} : \) \( 3 \equiv 2 + 1 \) 
fixes \( \text{four} \) (4) 
defines \( \text{four_def} : \) \( 4 \equiv 3 + 1 \) 
fixes \( \text{five} \) (5) 
defines \( \text{five_def} : \) \( 5 \equiv 4 + 1 \) 
fixes \( \text{six} \) (6) 
defines \( \text{six_def} : \) \( 6 \equiv 5 + 1 \) 
fixes \( \text{seven} \) (7) 
defines \( \text{seven_def} : \) \( 7 \equiv 6 + 1 \) 
fixes \( \text{eight} \) (8) 
defines \( \text{eight_def} : \) \( 8 \equiv 7 + 1 \) 
fixes \( \text{nine} \) (9) 
defines \( \text{nine_def} : \) \( 9 \equiv 8 + 1 \) 
assumes \( \text{MMI_pre_axlttri} : \) 
\( A \in R \land B \in R \implies (A \lhd B \iff \neg (A = B \lor B \lhd A)) \) 
assumes \( \text{MMI_pre_axlttrn} : \) 
\( A \in R \land B \in R \land C \in R \implies ((A \lhd B \land B \lhd C) \implies A \lhd C) \) 
assumes \( \text{MMI_pre_axltadd} : \) 
\( A \in R \land B \in R \land C \in R \implies (A \lhd B \implies C + A \lhd C + B) \) 
assumes \( \text{MMI_pre_axmulgt0} : \) 
\( A \in R \land B \in R \implies (0 \lhd A \land 0 \lhd B \implies 0 \lhd A \cdot B) \) 
assumes \( \text{MMI_pre_axsup} : \) 
\( A \subseteq R \land A \neq 0 \land \exists x \in R. \forall y \in A. y \lhd x \implies (\exists x \in R. \forall y \in A. \neg (x \lhd y)) \land (\forall x \land z \in R. (y \lhd x \implies (\exists z \in R. y \lhd z))) \) 
assumes \( \text{MMI_axresscn} : \) \( R \subseteq C \) 
assumes \( \text{MMI_axineq0} : \) \( 1 \neq 0 \) 
assumes \( \text{MMI_axcncx} : \) \( C \text{ isASet} \) 
assumes \( \text{MMI_axaddopr} : + : (C \times C) \to C \) 
assumes \( \text{MMI_axmulopr} : \cdot : (C \times C) \to C \) 
assumes \( \text{MMI_axmulcom} : A \in C \land B \in C \implies A \cdot B = B \cdot A \) 
assumes \( \text{MMI_axaddcl} : A \in C \land B \in C \implies A + B \in C \) 
assumes \( \text{MMI_axmulcl} : A \in C \land B \in C \implies A \cdot B \in C \) 
assumes \( \text{MMI_axdistr} : \) 
\( A \in C \land B \in C \land C \in C \implies A \cdot (B + C) = A \cdot B + A \cdot C \) 
assumes \( \text{MMI_axaddcom} : A \in C \land B \in C \implies A + B = B + A \) 
assumes \( \text{MMI_axaddass} : \)

1375
A ∈ C ∧ B ∈ C ∧ C ∈ C → A + B + C = A + (B + C)
assumes MMI_axmulass:
A ∈ C ∧ B ∈ C ∧ C ∈ C → A · B · C = A · (B · C)
assumes MMI_axre: 1 ∈ R
assumes MMI_ax1m1: i · i + 1 = 0
assumes MMI_ax0id: A ∈ C → A + 0 = A
assumes MMI_axicn: i ∈ C
assumes MMI_axnegex: A ∈ C → ( ∃ x ∈ C. ( A + x ) = 0 )
assumes MMI_axrecex: A ∈ C ∧ A ≠ 0 → ( ∃ x ∈ C. A · x = 1 )
assumes MMI_axid: A ∈ C → A · 1 = A
assumes MMI_axaddrcl: A ∈ R ∧ B ∈ R → A + B ∈ R
assumes MMI_axmulrcl: A ∈ R ∧ B ∈ R → A · B ∈ R
assumes MMI_axrnege: A ∈ R → ( ∃ x ∈ R. A + x = 0 )
assumes MMI_axrrecex: A ∈ R ∧ A ≠ 0 → ( ∃ x ∈ R. A · x = 1 )

end

95 Logic and sets in Metamath

theory MMI_logic_and_sets imports MMI_prelude
begin

95.1 Basic Metamath theorems

This section contains Metamath theorems that the more advanced theorems from MMI_sar.thy depend on. Most of these theorems are proven automatically by Isabelle, some have to be proven by hand and some have to be modified to convert from Tarski-Megill metalogic used by Metamath to one based on explicit notion of free and bound variables.

lemma MMI_ax_mp: assumes ϕ and ϕ → ψ shows ψ
using assms by auto

lemma MMI_sseli: assumes A1: A ⊆ B
shows C ∈ A → C ∈ B
using assms by auto

lemma MMI_sselii: assumes A1: A ⊆ B and
A2: C ∈ A
shows C ∈ B
using assms by auto

lemma MMI_syl: assumes A1: ϕ → ps and
A2: ps → ch
shows ϕ → ch
using assms by auto
lemma MMI_elimhyp: assumes A1: A = if ( ϕ , A , B ) −→ ( ϕ ←→ ψ ) and
    A2: B = if ( ϕ , A , B ) −→ ( ch ←→ ψ ) and
    A3: ch
    shows ψ
proof -
  { assume ϕ
    with A1 have ψ by simp }
moreover
  { assume ¬ϕ
    with A2 A3 have ψ by simp }
ultimately show ψ by auto
qed

lemma MMI_neeq1:
  shows A = B −→ ( A ≠ C ←→ B ≠ C )
by auto

lemma MMI_mp2: assumes A1: ϕ and
    A2: ψ and
    A3: ϕ −→ ( ψ −→ chi )
shows chi
using assms by auto

lemma MMI_xpex: assumes A1: A isASet and
    A2: B isASet
shows ( A × B ) isASet
using assms by auto

lemma MMI_fex:
shows
  A ∈ C −→ ( F : A → B −→ F isASet )
A isASet −→ ( F : A → B −→ F isASet )
by auto

lemma MMI_3eqtr4d: assumes A1: ϕ −→ A = B and
    A2: ϕ −→ C = A and
    A3: ϕ −→ D = B
shows ϕ −→ C = D
using assms by auto

lemma MMI_3coml: assumes A1: ( ϕ ∧ ψ ∧ chi ) −→ th
shows ( ψ ∧ chi ∧ ϕ ) −→ th
using assms by auto

lemma MMI_sylan: assumes A1: ( ϕ ∧ ψ ) −→ chi and
    A2: th −→ ϕ
shows ( th ∧ ψ ) −→ chi
using assms by auto
lemma MMI_3impa: assumes A1: ((ϕ ∧ ψ) ∧ chi) −→ th
  shows (ϕ ∧ ψ ∧ chi) −→ th
  using assms by auto

lemma MMI_3adant2: assumes A1: (ϕ ∧ ψ) −→ chi
  shows (ϕ ∧ th ∧ ψ) −→ chi
  using assms by auto

lemma MMI_3adant1: assumes A1: (ϕ ∧ ψ) −→ chi
  shows (th ∧ ϕ ∧ ψ) −→ chi
  using assms by auto

lemma (in MMIars0) MMI_opreq12d: assumes A1: ϕ −→ A = B and
  A2: ϕ −→ C = D
  shows
  ϕ −→ (A + C) = (B + D)
  ϕ −→ (A · C) = (B · D)
  ϕ −→ (A - C) = (B - D)
  ϕ −→ (A / C) = (B / D)
  using assms by auto

lemma MMI_mp2an: assumes A1: ϕ and
  A2: ψ and
  A3: (ϕ ∧ ψ) −→ chi
  shows chi
  using assms by auto

lemma MMI_mp3an: assumes A1: ϕ and
  A2: ψ and
  A3: ch and
  A4: (ϕ ∧ ψ ∧ ch) −→ θ
  shows θ
  using assms by auto

lemma MMI_eqeltrr: assumes A1: A = B and
  A2: A ∈ C
  shows B ∈ C
  using assms by auto

lemma MMI_eqtr: assumes A1: A = B and
  A2: B = C
  shows A = C
  using assms by auto

lemma MMI_impbi: assumes A1: ϕ −→ ψ and
  A2: ψ −→ ϕ
shows $\varphi \leftrightarrow \psi$

proof
assume $\varphi$ with A1 show $\psi$ by simp
next
assume $\psi$ with A2 show $\varphi$ by simp
qed

lemma MMI_mp3an3: assumes A1: ch and A2: $(\varphi \land \psi \land \text{ch}) \rightarrow \vartheta$
shows $(\varphi \land \psi) \rightarrow \vartheta$
using assms by auto

lemma MMI_eqeq12d: assumes A1: $\varphi \rightarrow A = B$ and A2: $\varphi \rightarrow C = D$
shows $\varphi \rightarrow (A = C \leftarrow\rightarrow B = D)$
using assms by auto

lemma MMI_mpan2: assumes A1: $\psi$ and A2: $(\varphi \land \psi) \rightarrow \text{ch}$
shows $\varphi \rightarrow \text{ch}$
using assms by auto

lemma (in MMIar0) MMI_opreq2:
shows $A = B \rightarrow (C + A) = (C + B)$
$A = B \rightarrow (C \cdot A) = (C \cdot B)$
$A = B \rightarrow (C - A) = (C - B)$
$A = B \rightarrow (C / A) = (C / B)$
by auto

lemma MMI_syl5bir: assumes A1: $\varphi \rightarrow (\psi \leftrightarrow \text{ch})$ and A2: $\vartheta \rightarrow \text{ch}$
shows $\varphi \rightarrow (\vartheta \rightarrow \psi)$
using assms by auto

lemma MMI_adantr: assumes A1: $\varphi \rightarrow \psi$
shows $(\varphi \land \text{ch}) \rightarrow \psi$
using assms by auto

lemma MMI_mpan: assumes A1: $\varphi$ and A2: $(\varphi \land \psi) \rightarrow \text{ch}$
shows $\psi \rightarrow \text{ch}$
using assms by auto

lemma MMI_eqeq1d: assumes A1: $\varphi \rightarrow A = B$
shows $\varphi \rightarrow (A = C \leftarrow\rightarrow B = C)$
using assms by auto

lemma (in MMIar0) MMI_opreq1:
shows
A = B \rightarrow ( A \cdot C ) = ( B \cdot C )
A = B \rightarrow ( A + C ) = ( B + C )
A = B \rightarrow ( A - C ) = ( B - C )
A = B \rightarrow ( A / C ) = ( B / C )
by auto

lemma MMI_syl6eq: assumes A1: \varphi \rightarrow A = B and
A2: B = C
shows \varphi \rightarrow A = C
using assms by auto

lemma MMI_syl6bi: assumes A1: \varphi \rightarrow ( \psi \leftrightarrow \text{ch} ) and
A2: \text{ch} \rightarrow \vartheta
shows \varphi \rightarrow ( \psi \rightarrow \vartheta )
using assms by auto

lemma MMI_imp: assumes A1: \varphi \rightarrow ( \psi \rightarrow \text{ch} )
shows ( \varphi \land \psi ) \rightarrow \text{ch}
using assms by auto

lemma MMI_syl1bd: assumes A1: \varphi \rightarrow ( \psi \rightarrow \text{ch} ) and
A2: \varphi \rightarrow ( \text{ch} \leftrightarrow \vartheta )
shows \varphi \rightarrow ( \psi \leftrightarrow \vartheta )
using assms by auto

lemma MMI_ex: assumes A1: ( \varphi \land \psi ) \rightarrow \text{ch}
shows \varphi \rightarrow ( \psi \rightarrow \text{ch} )
using assms by auto

lemma MMI_r19_23aiv: assumes A1: \forall x. ( x \in A \rightarrow ( \varphi(x) \rightarrow \psi ) )
shows ( \exists x \in A . \varphi(x) ) \rightarrow \psi
using assms by auto

lemma MMI_bitr: assumes A1: \varphi \leftrightarrow \psi and
A2: \psi \leftrightarrow \text{ch}
shows \varphi \leftrightarrow \text{ch}
using assms by auto

lemma MMI_eqeq12i: assumes A1: A = B and
A2: C = D
shows A = C \leftrightarrow B = D
using assms by auto

lemma MMI_dedth3h:
assumes A1: A = \text{if} ( \varphi , A , D ) \rightarrow ( \vartheta \leftrightarrow \text{ta} ) and
A2: B = \text{if} ( \psi , B , R ) \rightarrow ( \text{ta} \leftrightarrow \text{et} ) and
A3: C = \text{if} ( \text{ch} , C , S ) \rightarrow ( \text{et} \leftrightarrow \text{ze} ) and
A4: \text{ze}
shows $(\varphi \land \psi \land \text{ch}) \rightarrow \vartheta$
using assms by auto

lemma MMI_bibi1d: assumes A1: $\varphi \rightarrow (\psi \leftrightarrow \text{ch})$
shows $\varphi \rightarrow ((\psi \leftrightarrow \vartheta) \leftrightarrow (\text{ch} \leftrightarrow \vartheta))$
using assms by auto

lemma MMI_eqeq1:
shows $A = B \rightarrow (A = C \leftrightarrow B = C)$
by auto

lemma MMI_bibi12d: assumes A1: $\varphi \rightarrow (\psi \leftrightarrow \text{ch})$ and
A2: $\varphi \rightarrow (\vartheta \leftrightarrow \text{ta})$
shows $\varphi \rightarrow ((\psi \leftrightarrow \vartheta) \leftrightarrow (\text{ch} \leftrightarrow \text{ta}))$
using assms by auto

lemma MMI_eqeq2d: assumes A1: $\varphi \rightarrow A = B$
shows $\varphi \rightarrow (C = A \leftrightarrow C = B)$
using assms by auto

lemma MMI_eqeq2:
shows $A = B \rightarrow (C = A \leftrightarrow C = B)$
by auto

lemma MMI_elimel: assumes A1: $B \in C$
shows if $(A \in C, A, B) \in C$
using assms by auto

lemma MMI_3adant3: assumes A1: $(\varphi \land \psi) \rightarrow \text{ch}$
shows $(\varphi \land \psi \land \vartheta) \rightarrow \text{ch}$
using assms by auto

lemma MMI_bitr3d: assumes A1: $\varphi \rightarrow (\psi \leftrightarrow \text{ch})$ and
A2: $\varphi \rightarrow (\psi \leftrightarrow \vartheta)$
shows $\varphi \rightarrow (\text{ch} \leftrightarrow \vartheta)$
using assms by auto

lemma MMI_3eqtr3d: assumes A1: $\varphi \rightarrow A = B$ and
A2: $\varphi \rightarrow A = C$ and
A3: $\varphi \rightarrow B = D$
shows $\varphi \rightarrow C = D$
using assms by auto

lemma (in MMIIsar0) MMI_opreq1d: assumes A1: $\varphi \rightarrow A = B$
shows $\varphi \rightarrow (A + C) = (B + C)$
$\varphi \rightarrow (A - C) = (B - C)$

1381
\( \varphi \rightarrow (A \cdot C) = (B \cdot C) \)
\( \varphi \rightarrow (A / C) = (B / C) \)
using assms by auto

lemma MMI_3com12: assumes A1: \( (\varphi \land \psi \land \text{ch}) \rightarrow \vartheta \)
shows \( (\psi \land \varphi \land \text{ch}) \rightarrow \vartheta \)
using assms by auto

lemma (in MMIasar0) MMI_opreq2d: assumes A1: \( \varphi \rightarrow A = B \)
s shows 
\( \varphi \rightarrow (C + A) = (C + B) \)
\( \varphi \rightarrow (C - A) = (C - B) \)
\( \varphi \rightarrow (C \cdot A) = (C \cdot B) \)
\( \varphi \rightarrow (C / A) = (C / B) \)
using assms by auto

lemma MMI_3com23: assumes A1: \( (\varphi \land \psi \land \text{ch}) \rightarrow \vartheta \)
show s \( (\varphi \land \text{ch} \land \psi) \rightarrow \vartheta \)
using assms by auto

lemma MMI_3expa: assumes A1: \( (\varphi \land \psi \land \text{ch}) \rightarrow \vartheta \)
shows \( (\varphi \land (\psi \land \vartheta)) \rightarrow \vartheta \)
using assms by auto

lemma MMI_adantrr: assumes A1: \( (\varphi \land \psi) \rightarrow \text{ch} \)
s shows \( (\varphi \land (\psi \land \vartheta)) \rightarrow \text{ch} \)
using assms by auto

lemma MMI_3expb: assumes A1: \( (\varphi \land \psi \land \text{ch}) \rightarrow \vartheta \)
shows \( (\varphi \land (\psi \land \text{ch})) \rightarrow \vartheta \)
using assms by auto

lemma MMI_an4s: assumes A1: \( ((\varphi \land \psi) \land (\text{ch} \land \vartheta)) \rightarrow \tau \)
shows \( ((\varphi \land \text{ch}) \land (\psi \land \vartheta)) \rightarrow \tau \)
using assms by auto

lemma MMI_eqtrd: assumes A1: \( \varphi \rightarrow A = B \) and
A2: \( \varphi \rightarrow B = C \)
shows \( \varphi \rightarrow A = C \)
using assms by auto

lemma MMI_ad2ant2l: assumes A1: \( (\varphi \land \psi) \rightarrow \text{ch} \)
shows \( (\vartheta \land \varphi) \land (\tau \land \psi) \rightarrow \text{ch} \)
using assms by auto

lemma MMI_pm3_2i: assumes A1: \( \varphi \) and
A2: \( \psi \)
shows \( \varphi \land \psi \)
using assms by auto

1382
lemma (in MMI_sar0) MMI_opreq2i: assumes A1: A = B
  shows
  ( C + A ) = ( C + B )
  ( C - A ) = ( C - B )
  ( C - A ) = ( C - B )
  using assms by auto

lemma MMI_mpbir2an: assumes A1: \( \varphi \leftrightarrow ( \psi \land \mathrm{ch} ) \) and
  A2: \( \psi \) and
  A3: \( \mathrm{ch} \)
  shows \( \varphi \)
  using assms by auto

lemma MMI_reu4: assumes A1: \( \forall x \ y. \ x = y \rightarrow ( \varphi(x) \leftrightarrow \psi(y) ) \)
  shows ( \( \exists! \ x. \ x \in A \land \varphi(x) \) ) \( \leftrightarrow \)
  ( ( \( \exists x \in A . \ \varphi(x) \) ) \( \land \) ( \( \forall x \in A . \ \forall y \in A . \) 
  ( ( \( \varphi(x) \land \psi(y) \) ) \( 
  \rightarrow \ x = y ) ) )
  using assms by auto

lemma MMI_risset:
  shows A \( \in \) B \( \leftrightarrow \) ( \( \exists x \in B . \ x = A \) )
  by auto

lemma MMI_sylib: assumes A1: \( \varphi \rightarrow \psi \) and
  A2: \( \psi \leftrightarrow \mathrm{ch} \)
  shows \( \varphi \rightarrow \mathrm{ch} \)
  using assms by auto

lemma MMI_mp3an13: assumes A1: \( \varphi \) and
  A2: \( \psi \rightarrow \mathrm{ch} \) and
  A3: ( \( \varphi \land \psi \land \mathrm{ch} \) ) \( \rightarrow \) \( \vartheta \)
  shows \( \psi \rightarrow \vartheta \)
  using assms by auto

lemma MMI_eqcomd: assumes A1: \( \varphi \rightarrow A = B \)
  shows \( \varphi \rightarrow B = A \)
  using assms by auto

lemma MMI_sylan9eqr: assumes A1: \( \varphi \rightarrow A = B \) and
  A2: \( \psi \rightarrow B = C \)
  shows ( \( \psi \land \varphi \) ) \( \rightarrow \) A = C
  using assms by auto

lemma MMI_exp32: assumes A1: ( \( \varphi \land ( \psi \land \mathrm{ch} ) \) ) \( \rightarrow \) \( \vartheta \)
  shows \( \varphi \rightarrow ( \psi \rightarrow ( \mathrm{ch} \rightarrow \vartheta ) ) \)
  using assms by auto

1383
lemma MMI_impcom: assumes A1: $\phi \rightarrow (\psi \rightarrow \text{ch})$
shows $(\psi \land \phi) \rightarrow \text{ch}$
using assms by auto

lemma MMI_a1d: assumes A1: $\phi \rightarrow \psi$
shows $\phi \rightarrow (\text{ch} \rightarrow \psi)$
using assms by auto

lemma MMI_r19_21aiv: assumes A1: $\forall x. \phi \rightarrow (x \in A \rightarrow \psi(x))$
shows $\phi \rightarrow (\forall x \in A. \psi(x))$
using assms by auto

lemma MMI_r19_22:
shows $(\forall x \in A. (\phi(x) \rightarrow \psi(x))) \rightarrow$
$(\exists x \in A. \phi(x)) \rightarrow (\exists x \in A. \psi(x))$
by auto

lemma MMI_syl6: assumes A1: $\phi \rightarrow (\psi \rightarrow \text{ch})$ and
A2: $\text{ch} \rightarrow \vartheta$
shows $\phi \rightarrow (\psi \rightarrow \vartheta)$
using assms by auto

lemma MMI_mpid: assumes A1: $\phi \rightarrow \text{ch}$ and
A2: $\phi \rightarrow (\psi \rightarrow (\text{ch} \rightarrow \vartheta))$
shows $\phi \rightarrow (\psi \rightarrow \vartheta)$
using assms by auto

lemma MMI_eqtr3t:
shows $(A = C \land B = C) \rightarrow A = B$
by auto

lemma MMI_syl5bi: assumes A1: $\phi \rightarrow (\psi \leftrightarrow \text{ch})$ and
A2: $\vartheta \rightarrow \psi$
shows $\phi \rightarrow (\vartheta \rightarrow \text{ch})$
using assms by auto

lemma MMI_mp3an1: assumes A1: $\phi$ and
A2: $(\phi \land \psi \land \text{ch}) \rightarrow \vartheta$
shows $(\psi \land \text{ch}) \rightarrow \vartheta$
using assms by auto

lemma MMI_rgen2: assumes A1: $\forall x y. (x \in A \land y \in A) \rightarrow \phi(x,y)$
shows $\forall x \in A. \forall y \in A. \phi(x,y)$
using assms by auto

lemma MMI_ax_17: shows $\phi \rightarrow (\forall x. \phi)$ by simp
lemma MMI_3eqtr4g: assumes A1: \( \varphi \rightarrow A = B \) and
   A2: \( C = A \) and
   A3: \( D = B \)
shows \( \varphi \rightarrow C = D \)
using assms by auto

lemma MMI_3imtr4: assumes A1: \( \varphi \rightarrow \psi \) and
   A2: \( \chi \leftrightarrow \varphi \) and
   A3: \( \theta \leftrightarrow \psi \)
shows \( \chi \rightarrow \theta \)
using assms by auto

lemma MMI_eleq2i: assumes A1: \( A = B \)
shows \( C \in A \leftrightarrow C \in B \)
using assms by auto

lemma MMI_albii: assumes A1: \( \varphi \leftrightarrow \psi \)
shows \( (\forall x . \varphi ) \leftrightarrow (\forall x . \psi ) \)
using assms by auto

lemma MMI_reucl:
   shows \( (\exists ! x . x \in A \land \varphi(x) ) \rightarrow \bigcup \{ x \in A . \varphi(x) \} \in A \)
proof
   assume A1: \( \exists ! x . x \in A \land \varphi(x) \)
   then obtain a where I: a\( \in A \) and \( \varphi(a) \) by auto
   with A1 have \{ x \( \in A . \varphi(x) \} = \{a\} \) by blast
   with I show \( \bigcup \{ x \in A . \varphi(x) \} \in A \) by simp
qed

lemma MMI_dedth2h: assumes A1: \( A = \text{if} ( \varphi , A , C ) \rightarrow ( \chi \leftrightarrow \theta ) \) and
   A2: \( B = \text{if} ( \psi , B , D ) \rightarrow ( \theta \leftrightarrow \tau ) \) and
   A3: \( \tau \)
shows \( ( \varphi \land \psi ) \rightarrow \chi \)
using assms by auto

lemma MMI_eleq1d: assumes A1: \( \varphi \rightarrow A = B \)
shows \( \varphi \rightarrow ( A \in C \leftrightarrow B \in C ) \)
using assms by auto

lemma MMI_syl5eqel: assumes A1: \( \varphi \rightarrow A \in B \) and
   A2: \( C = A \)
shows $\varphi \rightarrow C \in B$
using assms by auto

lemma IML_euuni: assumes A1: $x \in A$ and A2: $\exists! t . t \in A \land \varphi(t)$
shows $\varphi(x) \leftrightarrow \bigcup \{ x \in A . \varphi(x) \} = x$
proof
assume $\varphi(x)$
with A1 A2 show $\bigcup \{ x \in A . \varphi(x) \} = x$ by auto
next assume A3: $\bigcup \{ x \in A . \varphi(x) \} = x$
from A2 obtain $y$ where $y\in A$ and I: $\varphi(y)$ by auto
with A2 A3 have $x = y$ by auto
with I show $\varphi(x)$ by simp
qed

lemma MMI_reuuni1:
shows $( x \in A \land ( \exists! x . x \in A \land \varphi(x) ) ) \rightarrow
( \varphi(x) \leftrightarrow \bigcup \{ x \in A . \varphi(x) \} = x )$
using IML_euuni by simp

lemma MMI_eeq1i: assumes A1: $A = B$
shows $A = C \leftrightarrow B = C$
using assms by auto

lemma MMI_syl6rbr: assumes A1: $\forall x. \varphi(x) \rightarrow ( \psi(x) \leftrightarrow \text{ch}(x) )$ and
A2: $\forall x. \varphi(x) \rightarrow \text{ch}(x)$
shows $\forall x. \varphi(x) \rightarrow ( \psi(x) \leftrightarrow \text{ch}(x) )$
using assms by auto

lemma MMI_syl6rbrA: assumes A1: $\varphi \rightarrow ( \psi \leftrightarrow \text{ch} )$ and
A2: $\varphi \rightarrow \text{ch}$
shows $\varphi \rightarrow ( \varphi \leftrightarrow \psi )$
using assms by auto

lemma MMI_vtoclga: assumes A1: $\forall x. x = A \rightarrow ( \varphi(x) \leftrightarrow \psi )$ and
A2: $\forall x. x \in B \rightarrow \varphi(x)$
shows $A \in B \rightarrow \psi$
using assms by auto

lemma MMI_3bitr4: assumes A1: $\varphi \leftrightarrow \psi$ and
A2: $\text{ch} \leftrightarrow \varphi$ and
A3: $\varphi \leftrightarrow \psi$
shows $\text{ch} \leftrightarrow \varphi$
using assms by auto
lemma MMI_mpbi2: assumes Amin: \( \psi \) and
  Amaj: \( \varphi \rightarrow ( \psi \leftrightarrow \text{ch} ) \)
  shows \( \varphi \rightarrow \text{ch} \)
  using assms by auto

lemma MMI_eqid:
  shows A = A
  by auto

lemma MMI_pm3_27:
  shows ( \( \varphi \land \psi \) ) \( \rightarrow \) \( \psi \)
  by auto

lemma MMI_pm3_26:
  shows ( \( \varphi \land \psi \) ) \( \rightarrow \) \( \varphi \)
  by auto

lemma MMI_ancoms: assumes A1: ( \( \varphi \land \psi \) ) \( \rightarrow \) \( \text{ch} \)
  shows ( \( \psi \land \varphi \) ) \( \rightarrow \) \( \text{ch} \)
  using assms by auto

lemma MMI_syl3anc: assumes A1: ( \( \varphi \land \psi \land \text{ch} \) ) \( \rightarrow \) \( \vartheta \) and
  A2: \( \tau \rightarrow \varphi \) and
  A3: \( \tau \rightarrow \psi \) and
  A4: \( \tau \rightarrow \text{ch} \)
  shows \( \tau \rightarrow \vartheta \)
  using assms by auto

lemma MMI_syl5eq: assumes A1: \( \varphi \rightarrow A = B \) and
  A2: C = A
  shows \( \varphi \rightarrow C = B \)
  using assms by auto

lemma MMI_eqcomi: assumes A1: A = B
  shows B = A
  using assms by auto

lemma MMI_3eqtr: assumes A1: A = B and
  A2: B = C and
  A3: C = D
  shows A = D
  using assms by auto

lemma MMI_mpbir: assumes Amin: \( \psi \) and
  Amaj: \( \varphi \leftrightarrow \psi \)
  shows \( \varphi \)
  using assms by auto
lemma MMI_syl3an3: assumes A1: ( \( \phi \land \psi \land \text{ch} \)) \( \rightarrow \) \( \theta \) and
A2: \( \tau \rightarrow \text{ch} \)
shows ( \( \phi \land \psi \land \tau \)) \( \rightarrow \) \( \theta \)
using assms by auto

lemma MMI_3eqtrd: assumes A1: \( \phi \rightarrow A = B \) and
A2: \( \phi \rightarrow B = C \) and
A3: \( \phi \rightarrow C = D \)
shows \( \phi \rightarrow A = D \)
using assms by auto

lemma MMI_syl5: assumes A1: \( \phi \rightarrow (\psi \rightarrow \text{ch}) \)
A2: \( \theta \rightarrow \psi \)
shows \( \phi \rightarrow (\theta \rightarrow \text{ch}) \)
using assms by auto

lemma MMI_exp3a: assumes A1: \( \phi \rightarrow (\psi \land \text{ch} \rightarrow \theta) \)
shows \( \phi \rightarrow (\psi \rightarrow (\text{ch} \rightarrow \theta)) \)
using assms by auto

lemma MMI_com12: assumes A1: \( \phi \rightarrow (\psi \rightarrow \text{ch}) \)
shows \( \psi \rightarrow (\phi \rightarrow \text{ch}) \)
using assms by auto

lemma MMI_3imp: assumes A1: \( \phi \rightarrow (\psi \rightarrow (\text{ch} \rightarrow \theta)) \)
shows ( \( \phi \land \psi \land \text{ch} \)) \( \rightarrow \) \( \theta \)
using assms by auto

lemma MMI_3eqtr3: assumes A1: A = B and
A2: A = C and
A3: B = D
shows C = D
using assms by auto

lemma (in MMIar0) MMI_opreqli: assumes A1: A = B
shows
( A + C ) = ( B + C )
( A - C ) = ( B - C )
( A \cdot C ) = ( B \cdot C )
using assms by auto

lemma MMI_eqtr3: assumes A1: A = B and
A2: A = C
shows B = C
using assms by auto
lemma MMI_dedth: assumes A1: A = if ( ϕ , A , B ) −→ ( ψ ←→ ch ) and
    A2: ch
  shows ϕ −→ ψ
  using assms by auto

lemma MMI_id:
  shows ϕ −→ ϕ
  by auto

lemma MMI_eqtr3d: assumes A1: φ −→ A = B and
    A2: φ −→ A = C
  shows φ −→ B = C
  using assms by auto

lemma MMI_sylan2: assumes A1: ( φ ∧ ψ ) −→ ch and
    A2: ϑ −→ ψ
  shows ( φ ∧ ϑ ) −→ ch
  using assms by auto

lemma MMI_adantl: assumes A1: φ −→ ψ
  shows ( ch ∧ φ ) −→ ψ
  using assms by auto

lemma (in MMIsar0) MMI_opreq12:
  shows
    ( A = B ∧ C = D ) −→ ( A + C ) = ( B + D )
    ( A = B ∧ C = D ) −→ ( A - C ) = ( B - D )
    ( A = B ∧ C = D ) −→ ( A · C ) = ( B · D )
    ( A = B ∧ C = D ) −→ ( A / C ) = ( B / D )
  by auto

lemma MMI_anidms: assumes A1: ( φ ∧ φ ) −→ ψ
  shows φ −→ ψ
  using assms by auto

lemma MMI_anabsan2: assumes A1: ( φ ∧ ( ψ ∧ ψ ) ) −→ ch
  shows ( φ ∧ ψ ) −→ ch
  using assms by auto

lemma MMI_3simp2:
  shows ( φ ∧ ψ ∧ ch ) −→ ψ
  by auto

lemma MMI_3simp3:
  shows ( φ ∧ ψ ∧ ch ) −→ ch
  by auto
lemma MMI_sylbir: assumes A1: \( \psi \leftrightarrow \varphi \) and
    
A2: \( \psi \rightsquigarrow \text{ch} \)

shows \( \varphi \rightsquigarrow \text{ch} \)
using assms by auto


definition of propositions

lemma MMI_3eqtr3g: assumes A1: \( \varphi \rightarrow A = B \) and
    
A2: \( A = C \) and
A3: \( B = D \)

shows \( \varphi \rightarrow C = D \)
using assms by auto

lemma MMI_3bitr: assumes A1: \( \varphi \leftrightarrow \psi \) and
    
A2: \( \psi \leftrightarrow \text{ch} \) and
A3: \( \text{ch} \leftrightarrow \vartheta \)

shows \( \varphi \leftrightarrow \vartheta \)
using assms by auto

lemma MMI_3bitr3: assumes A1: \( \varphi \leftrightarrow \psi \) and
    
A2: \( \varphi \leftrightarrow \text{ch} \) and
A3: \( \psi \leftrightarrow \vartheta \)

shows \( \text{ch} \leftrightarrow \vartheta \)
using assms by auto

lemma MMI_eqcom:
    
shows A = B \( \iff \) B = A
by auto

lemma MMI_syl6bb: assumes A1: \( \varphi \rightarrow ( \psi \leftrightarrow \text{ch} ) \) and
    
A2: \( \text{ch} \leftrightarrow \vartheta \)

shows \( \varphi \rightarrow ( \psi \leftrightarrow \vartheta ) \)
using assms by auto

lemma MMI_3bitr3d: assumes A1: \( \varphi \rightarrow ( \psi \leftrightarrow \text{ch} ) \) and
    
A2: \( \varphi \rightarrow ( \psi \leftrightarrow \vartheta ) \) and
A3: \( \text{ch} \leftrightarrow \tau \)

shows \( \varphi \rightarrow ( \vartheta \leftrightarrow \tau ) \)
using assms by auto

lemma MMI_syl3an2: assumes A1: ( \( \varphi \land \psi \land \text{ch} \), \( \varphi \land \psi \land \text{ch} \)) \( \rightarrow \) \( \psi \) and
    
A2: \( \tau \rightarrow \psi \)

shows ( \( \varphi \land \tau \land \text{ch} \), \( \varphi \land \tau \land \text{ch} \)) \( \rightarrow \) \( \psi \)
using assms by auto

lemma MMI_df_rex:
shows \((\exists x \in A . \varphi(x)) \iff (\exists x . (x \in A \wedge \varphi(x)))\)
by auto

lemma MMI_mpbi: assumes Amin: \(\varphi\) and
Amaj: \(\varphi \iff \psi\)
shows \(\psi\)
using assms by auto

lemma MMI_mp3an12: assumes A1: \(\varphi\) and
A2: \(\psi\) and
A3: \((\varphi \wedge \psi \wedge \text{ch}) \implies \vartheta\)
shows \(\text{ch} \implies \vartheta\)
using assms by auto

lemma MMI_syl5bb: assumes A1: \(\varphi \implies (\psi \iff \text{ch})\) and
A2: \(\vartheta \iff \psi\)
shows \(\varphi \implies (\vartheta \iff \text{ch})\)
using assms by auto

lemma MMI_eqeltrrd: assumes A1: \(\varphi \implies A = B\) and
A2: \(\varphi \implies A \in C\)
shows \(\varphi \implies B \in C\)
using assms by auto

lemma MMI_19_23aiv: assumes A1: \(\forall x . \varphi(x) \implies \psi\)
shows \((\exists x . \varphi(x)) \implies \psi\)
using assms by auto

lemma MMI_syl2an: assumes A1: \((\varphi \wedge \psi) \implies \text{ch}\) and
A2: \(\vartheta \implies \varphi\) and
A3: \(\tau \implies \psi\)
shows \((\vartheta \wedge \tau) \implies \text{ch}\)
using assms by auto

lemma MMI_adantrl: assumes A1: \((\varphi \wedge \psi) \implies \text{ch}\)
shows \((\varphi \wedge (\vartheta \wedge \psi)) \implies \text{ch}\)
using assms by auto

1391
lemma MMI_ad2ant2r: assumes A1: \((\varphi \land \psi) \rightarrow \chi\)
  shows \((\varphi \land \vartheta) \land (\psi \land \tau)\) \rightarrow \chi
  using assms by auto

lemma MMI_adantl1: assumes A1: \((\varphi \land \psi) \rightarrow \chi\)
  shows \((\vartheta \land \varphi) \land \psi\) \rightarrow \chi
  using assms by auto

lemma MMI_anandirs: assumes A1: \((\varphi \land \chi) \land (\psi \land \chi)\) \rightarrow \tau
  shows \((\varphi \land \psi) \land \chi\) \rightarrow \tau
  using assms by auto

lemma MMI_adantlr: assumes A1: \((\varphi \land \psi) \rightarrow \chi\)
  shows \((\varphi \land \vartheta) \land \psi\) \rightarrow \chi
  using assms by auto

lemma MMI_an42s: assumes A1: \((\varphi \land \psi) \land (\chi \land \vartheta)\) \rightarrow \tau
  shows \((\varphi \land \chi) \land (\vartheta \land \psi)\) \rightarrow \tau
  using assms by auto

lemma MMI_mp3an2: assumes A1: \(\psi\) and
  A2: \((\varphi \land \psi \land \chi) \rightarrow \vartheta\)
  shows \((\varphi \land \chi) \rightarrow \vartheta\)
  using assms by auto

lemma MMI_3simp1:
  shows \((\varphi \land \psi \land \chi) \rightarrow \varphi\)
  by auto

lemma MMI_3simpb: assumes A1: \((\varphi \land (\psi \land \chi)) \rightarrow \vartheta\)
  shows \((\varphi \land \psi \land \chi) \rightarrow \vartheta\)
  using assms by auto

lemma MMI_mpbird: assumes Amin: \(\varphi \rightarrow \chi\) and
  Amaj: \(\varphi \rightarrow (\psi \leftarrow \chi)\)
  shows \(\varphi \rightarrow \psi\)
  using assms by auto

lemma (in MMIsar0) MMI_opreq12i: assumes A1: A = B and
  A2: C = D
  shows
  \((A + C) = (B + D)\)
  \((A \cdot C) = (B \cdot D)\)
  \((A - C) = (B - D)\)
  using assms by auto

1392
lemma MMI_3eqtr4: assumes A1: A = B and
    A2: C = A and
    A3: D = B
shows C = D
    using assms by auto

lemma MMI_eqtr4d: assumes A1: ϕ \rightarrow A = B and
    A2: ϕ \rightarrow C = B
shows ϕ \rightarrow A = C
    using assms by auto

lemma MMI_3eqtr3rd: assumes A1: ϕ \rightarrow A = B and
    A2: ϕ \rightarrow A = C
    A3: ϕ \rightarrow B = D
shows ϕ \rightarrow D = C
    using assms by auto

lemma MMI_sylanc: assumes A1: (ϕ \land ψ) \rightarrow ch and
    A2: ϑ \rightarrow ϕ and
    A3: ϑ \rightarrow ψ
shows ϑ \rightarrow ch
    using assms by auto

lemma MMI_anim12i: assumes A1: ϕ \rightarrow ψ and
    A2: ch \rightarrow ϑ
shows (ϕ \land ch) \rightarrow (ψ \land ϑ)
    using assms by auto

lemma (in MMIisar0) MMI_opreqan12d: assumes A1: ϕ \rightarrow A = B and
    A2: ϕ \rightarrow C = D
shows
    (ϕ \land ψ) \rightarrow (A + C) = (B + D)
    (ϕ \land ψ) \rightarrow (A - C) = (B - D)
    (ϕ \land ψ) \rightarrow (A \cdot C) = (B \cdot D)
    using assms by auto

lemma MMI_sylanr2: assumes A1: (ϕ \land (ψ \land ch)) \rightarrow ϑ and
    A2: τ \rightarrow ch
shows (ϕ \land (ψ \land τ)) \rightarrow ϑ
    using assms by auto

lemma MMI_sylanl2: assumes A1: ((ϕ \land ψ) \land ch) \rightarrow ϑ and
    A2: τ \rightarrow ch
shows (ϕ \land (ψ \land τ)) \rightarrow ϑ
    using assms by auto
A2: \( \tau \rightarrow \psi \)
shows \((\varphi \land \tau) \land \text{ch}\) \(\rightarrow \vartheta\)
using assms by auto

lemma MMI_ancom2s: assumes A1: \((\varphi \land (\psi \land \text{ch}))\) \(\rightarrow \vartheta\)
shows \((\varphi \land (\text{ch} \land \psi))\) \(\rightarrow \vartheta\)
using assms by auto

lemma MMI_anandis: assumes A1: \(((\varphi \land \psi) \land (\varphi \land \text{ch}))\) \(\rightarrow \tau\)
shows \((\varphi \land (\psi \land \text{ch}))\) \(\rightarrow \tau\)
using assms by auto

lemma MMI_sylan9eq: assumes A1: \(\varphi \rightarrow A = B\) and
A2: \(\psi \rightarrow B = C\)
shows \((\varphi \land \psi) \rightarrow A = C\)
using assms by auto

lemma MMI_keephyp: assumes A1: \(A = \text{if } (\varphi, A, B) \rightarrow (\psi \leftrightarrow \vartheta)\)
and
A2: \(B = \text{if } (\varphi, A, B) \rightarrow (\text{ch} \leftrightarrow \vartheta)\) and
A3: \(\psi\) and
A4: \(\text{ch}\)
shows \(\vartheta\)
proof -
\{ assume \(\varphi\)
  with A1 A3 have \(\vartheta\) by simp \}
moreover
\{ assume \(\neg \varphi\)
  with A2 A4 have \(\vartheta\) by simp \}
ultimately show \(\vartheta\) by auto
qed

lemma MMI_eleq1:
shows \(A = B \rightarrow (A \in C \leftrightarrow B \in C)\)
by auto

lemma MMI_pm4_2i:
shows \(\varphi \rightarrow (\psi \leftrightarrow \psi)\)
by auto

lemma MMI_3anbi123d: assumes A1: \(\varphi \rightarrow (\psi \leftrightarrow \text{ch})\) and
A2: \(\varphi \rightarrow (\vartheta \leftrightarrow \tau)\) and
A3: \(\varphi \rightarrow (\eta \leftrightarrow \zeta)\)
shows \(\varphi \rightarrow ((\psi \land \vartheta \land \eta) \leftrightarrow (\text{ch} \land \tau \land \zeta))\)
using assms by auto

lemma MMI_imbi12d: assumes A1: \(\varphi \rightarrow (\psi \leftrightarrow \text{ch})\) and
A2: \( \varphi \rightarrow ( \vartheta \leftrightarrow \tau ) \)
shows \( \varphi \rightarrow ( ( \psi \rightarrow \vartheta ) \leftrightarrow ( \mathrm{ch} \rightarrow \tau ) ) \)
using \textit{assms} by \textit{auto}

\textbf{lemma MMI_bitrd:} assumes A1: \( \varphi \rightarrow ( \psi \leftrightarrow \mathrm{ch} ) \) and
A2: \( \varphi \rightarrow ( \mathrm{ch} \leftrightarrow \vartheta ) \)
shows \( \varphi \rightarrow ( \psi \leftrightarrow \vartheta ) \)
using \textit{assms} by \textit{auto}

\textbf{lemma MMI_df_ne:}
shows \( ( A \neq B \leftrightarrow \neg ( A = B ) ) \)
by \textit{auto}

\textbf{lemma MMI_3pm3_2i:} assumes A1: \( \varphi \) and
A2: \( \psi \) and
A3: \( \mathrm{ch} \)
shows \( \varphi \land \psi \land \mathrm{ch} \)
using \textit{assms} by \textit{auto}

\textbf{lemma MMI_eqeq2i:} assumes A1: \( A = B \)
shows \( C = A \leftrightarrow C = B \)
using \textit{assms} by \textit{auto}

\textbf{lemma MMI_syl5bbr:} assumes A1: \( \varphi \rightarrow ( \psi \leftrightarrow \mathrm{ch} ) \) and
A2: \( \psi \leftrightarrow \vartheta \)
shows \( \varphi \rightarrow ( \vartheta \leftrightarrow \mathrm{ch} ) \)
using \textit{assms} by \textit{auto}

\textbf{lemma MMI_biimpd:} assumes A1: \( \varphi \rightarrow ( \psi \leftrightarrow \mathrm{ch} ) \)
shows \( \varphi \rightarrow ( \psi \rightarrow \mathrm{ch} ) \)
using \textit{assms} by \textit{auto}

\textbf{lemma MMI_orrd:} assumes A1: \( \varphi \rightarrow ( \neg ( \psi ) \rightarrow \mathrm{ch} ) \)
shows \( \varphi \rightarrow ( \psi \lor \mathrm{ch} ) \)
using \textit{assms} by \textit{auto}

\textbf{lemma MMI_jaoi:} assumes A1: \( \varphi \rightarrow \psi \) and
A2: \( \mathrm{ch} \rightarrow \psi \)
shows \( \varphi \lor \mathrm{ch} \rightarrow \psi \)
using \textit{assms} by \textit{auto}

\textbf{lemma MMI_oridm:}
shows \( ( \varphi \lor \varphi ) \leftrightarrow \varphi \)
by \textit{auto}

\textbf{lemma MMI_orbld:} assumes A1: \( \varphi \rightarrow ( \psi \leftrightarrow \mathrm{ch} ) \)
shows \( \varphi \rightarrow ( ( \psi \lor \vartheta ) \leftrightarrow ( \mathrm{ch} \lor \vartheta ) ) \)
using \textit{assms} by \textit{auto}

1395
lemma MMI_orbi2d: assumes A1: $\varphi \rightarrow ( \psi \leftrightarrow \text{ch} )$
    shows $\varphi \rightarrow ( ( \vartheta \lor \psi ) \leftrightarrow ( \vartheta \lor \text{ch} ) )$
    using assms by auto

lemma MMI_3bitr4g: assumes A1: $\varphi \rightarrow ( \psi \leftrightarrow \text{ch} )$ and
    A2: $\vartheta \leftrightarrow \psi$ and
    A3: $\tau \leftrightarrow \text{ch}$
    shows $\varphi \rightarrow ( \vartheta \leftrightarrow \tau )$
    using assms by auto

lemma MMI_negbid: assumes A1: $\varphi \rightarrow ( \psi \leftrightarrow \text{ch} )$
    shows $\varphi \rightarrow ( \neg ( \psi ) \leftrightarrow \neg ( \text{ch} ) )$
    using assms by auto

lemma MMI_ioran: shows $\neg ( ( \varphi \lor \psi ) ) \leftrightarrow$
    $\neg ( \varphi ) \land \neg ( \psi )$
    by auto

lemma MMI_syl6rbb: assumes A1: $\varphi \rightarrow ( \psi \leftrightarrow \text{ch} )$ and
    A2: $\text{ch} \leftrightarrow \vartheta$
    shows $\varphi \rightarrow ( \vartheta \leftrightarrow \psi )$
    using assms by auto

lemma MMI_anbi12i: assumes A1: $\varphi \leftrightarrow \psi$ and
    A2: $\text{ch} \leftrightarrow \vartheta$
    shows $\varphi \land \text{ch} \leftrightarrow \psi \land \vartheta$
    using assms by auto

lemma MMI_keepel: assumes A1: $A \in C$ and
    A2: $B \in C$
    shows if $( \varphi , A , B ) \in C$
    using assms by auto

lemma MMI_imbi2d: assumes A1: $\varphi \rightarrow ( \psi \leftrightarrow \text{ch} )$
    shows $\varphi \rightarrow ( ( \vartheta \rightarrow \psi ) \leftrightarrow ( \vartheta \rightarrow \text{ch} ) )$
    using assms by auto

lemma MMI_eqeltr: assumes $A = B$ and $B \in C$
    shows $A \in C$ using assms by auto
lemma MMI_3impia: assumes A1: ( \( \varphi \land \psi \) ) \( \rightarrow \) ( ch \( \rightarrow \) \( \vartheta \) )
  shows ( \( \varphi \land \psi \land \text{ch} \) ) \( \rightarrow \) \( \vartheta \)
  using assms by auto

lemma MMI_eqneqd: assumes A1: \( \varphi \rightarrow ( A = B \leftrightarrow C = D ) \)
  shows \( \varphi \rightarrow ( A \neq B \leftrightarrow C \neq D ) \)
  using assms by auto

lemma MMI_3ad2ant2: assumes A1: \( \varphi \rightarrow \text{ch} \)
  shows ( \( \psi \land \varphi \land \vartheta \) ) \( \rightarrow \) \( \text{ch} \)
  using assms by auto

lemma MMI_mp3anl3: assumes A1: \( \text{ch} \) and
  \( \text{A2: ( ( \varphi \land \psi \land \text{ch} ) \land \vartheta ) } \rightarrow \tau \)
  shows ( ( \( \varphi \land \psi \) ) \( \land \vartheta \) ) \( \rightarrow \) \( \tau \)
  using assms by auto

lemma MMI_bitr4d: assumes A1: \( \varphi \rightarrow ( \psi \leftrightarrow \text{ch} ) \) and
  \( \text{A2: } \varphi \rightarrow ( \vartheta \leftrightarrow \text{ch} ) \)
  shows \( \varphi \rightarrow ( \psi \leftrightarrow \vartheta ) \)
  using assms by auto

lemma MMI_neeq1d: assumes A1: \( \varphi \rightarrow A = B \)
  shows \( \varphi \rightarrow ( A \neq C \leftrightarrow B \neq C ) \)
  using assms by auto

lemma MMI_3anim123i: assumes A1: \( \varphi \rightarrow \psi \) and
  \( \text{A2: } \text{ch} \rightarrow \vartheta \) and
  \( \text{A3: } \tau \rightarrow \eta \)
  shows ( \( \varphi \land \text{ch} \land \tau \) ) \( \rightarrow \) ( \( \psi \land \vartheta \land \eta \) )
  using assms by auto

lemma MMI_3exp: assumes A1: ( \( \varphi \land \psi \land \text{ch} \) ) \( \rightarrow \) \( \vartheta \)
  shows \( \varphi \rightarrow ( \psi \rightarrow ( \text{ch} \rightarrow \vartheta ) ) \)
  using assms by auto

lemma MMI_exp4a: assumes A1: \( \varphi \rightarrow ( \psi \rightarrow ( ( \text{ch} \land \vartheta ) \rightarrow \tau ) ) \)
  shows \( \varphi \rightarrow ( \psi \rightarrow ( \text{ch} \rightarrow ( \vartheta \rightarrow \tau ) ) ) \)
  using assms by auto

lemma MMI_3imp1: assumes A1: \( \varphi \rightarrow ( \psi \rightarrow ( \text{ch} \rightarrow ( \vartheta \rightarrow \tau ) ) ) \)
  shows ( ( \( \varphi \land \psi \land \text{ch} \) ) \( \land \vartheta \) ) \( \rightarrow \tau \)
  using assms by auto

1397
lemma MMI_anim1i: assumes A1: $\varphi \rightarrow \psi$
shows $(\varphi \land \text{ch}) \rightarrow (\psi \land \text{ch})$
using assms by auto

lemma MMI_3adantl1: assumes A1: $( (\varphi \land \psi) \land \text{ch} ) \rightarrow \vartheta$
shows $( (\tau \land \varphi \land \psi) \land \text{ch} ) \rightarrow \vartheta$
using assms by auto

lemma MMI_3adantl2: assumes A1: $( (\varphi \land \psi) \land \text{ch} ) \rightarrow \vartheta$
shows $( (\varphi \land \tau \land \psi) \land \text{ch} ) \rightarrow \vartheta$
using assms by auto

lemma MMI_3comr: assumes A1: $(\varphi \land \psi \land \text{ch}) \rightarrow \vartheta$
shows $(\text{ch} \land \varphi \land \psi) \rightarrow \vartheta$
using assms by auto

lemma MMI_bitr3: assumes A1: $\psi \leftrightarrow \varphi$ and
A2: $\psi \leftrightarrow \text{ch}$
shows $\varphi \leftrightarrow \text{ch}$
using assms by auto

lemma MMI_anbi12d: assumes A1: $\varphi \rightarrow (\psi \leftrightarrow \text{ch})$ and
A2: $\varphi \rightarrow (\vartheta \leftrightarrow \tau)$
shows $\varphi \rightarrow ((\psi \land \vartheta) \leftrightarrow (\text{ch} \land \tau))$
using assms by auto

lemma MMI_pm3_26i: assumes A1: $\varphi \land \psi$
shows $\varphi$
using assms by auto

lemma MMI_pm3_27i: assumes A1: $\varphi \land \psi$
shows $\psi$
using assms by auto

lemma MMI_anabsan: assumes A1: $( (\varphi \land \varphi) \land \psi) \rightarrow \text{ch}$
shows $(\varphi \land \psi) \rightarrow \text{ch}$
using assms by auto

lemma MMI_3eqtr4rd: assumes A1: $\varphi \rightarrow A = B$ and
A2: $\varphi \rightarrow C = A$ and
A3: $\varphi \rightarrow D = B$
shows $\varphi \rightarrow D = C$
using assms by auto
lemma MMI_syl3an1: assumes A1: ($\varphi \land \psi \land \text{ch}$) $\rightarrow$ $\vartheta$ and
A2: $\tau$ $\rightarrow$ $\varphi$
shows ($\tau \land \psi \land \text{ch}$) $\rightarrow$ $\vartheta$
using assms by auto

lemma MMI_syl3anl2: assumes A1: (($\varphi \land \psi \land \text{ch}$) $\land$ $\vartheta$) $\rightarrow$ $\tau$ and
A2: $\eta$ $\rightarrow$ $\psi$
shows (($\varphi \land \eta \land \text{ch}$) $\land$ $\vartheta$) $\rightarrow$ $\tau$
using assms by auto

lemma MMI_jca: assumes A1: $\varphi$ $\rightarrow$ $\psi$ and
A2: $\varphi$ $\rightarrow$ $\text{ch}$
shows $\varphi$ $\rightarrow$ ($\psi \land \text{ch}$)
using assms by auto

lemma MMI_3ad2ant3: assumes A1: $\varphi$ $\rightarrow$ $\text{ch}$
shows ($\varphi \land \text{ch}$) $\land$ ($\vartheta$ $\land$ $\varphi$) $\rightarrow$ $\text{ch}$
using assms by auto

lemma MMI_anmi2i: assumes A1: $\varphi$ $\rightarrow$ $\psi$
shows (ch $\land$ $\varphi$) $\rightarrow$ (ch $\land$ $\psi$)
using assms by auto

lemma MMI_ancom:
shows ($\varphi \land \psi$) $\leftrightarrow$ ($\psi \land \varphi$)
by auto

lemma MMI_anbiii: assumes Aaa: $\varphi$ $\leftrightarrow$ $\psi$
shows ($\varphi \land \text{ch}$) $\leftrightarrow$ ($\psi \land \text{ch}$)
using assms by auto

lemma MMI_an42:
shows (($\varphi \land \psi$) $\land$ (ch $\land$ $\vartheta$)) $\leftrightarrow$
($($\varphi \land \text{ch}$) $\land$ ($\vartheta$ $\land$ $\psi$))
by auto

lemma MMI_sylanb: assumes A1: ($\varphi \land \psi$) $\rightarrow$ ch and
A2: $\vartheta$ $\leftrightarrow$ $\varphi$
shows ($\vartheta \land \psi$) $\rightarrow$ ch
using assms by auto

lemma MMI_an4:
shows (($\varphi \land \psi$) $\land$ (ch $\land$ $\vartheta$)) $\leftrightarrow$
($($\varphi \land \text{ch}$) $\land$ ($\psi \land \vartheta$))
by auto
lemma MMI_syl2anb: assumes A1: $(\varphi \land \psi) \rightarrow \chi$ and
  A2: $\vartheta \leftrightarrow \varphi$ and
  A3: $\tau \leftrightarrow \psi$
  shows $(\vartheta \land \tau) \rightarrow \chi$
  using assms by auto

lemma MMI_eqtr2d: assumes A1: $\varphi \rightarrow A = B$ and
  A2: $\varphi \rightarrow B = C$
  shows $\varphi \rightarrow C = A$
  using assms by auto

lemma MMI_sylbid: assumes A1: $\varphi \rightarrow (\psi \leftrightarrow \chi)$ and
  A2: $\varphi \rightarrow (\chi \rightarrow \vartheta)$
  shows $\varphi \rightarrow (\psi \rightarrow \vartheta)$
  using assms by auto

lemma MMI_sylanl1: assumes A1: $(\varphi \land \psi) \land \chi \rightarrow \vartheta$ and
  A2: $\tau \rightarrow \varphi$
  shows $((\tau \land \psi) \land \chi) \rightarrow \vartheta$
  using assms by auto

lemma MMI_sylan2b: assumes A1: $(\varphi \land \psi) \rightarrow \chi$ and
  A2: $\vartheta \leftrightarrow \psi$
  shows $(\varphi \land \vartheta) \rightarrow \chi$
  using assms by auto

lemma MMI_pm3_22:
  shows $(\varphi \land \psi) \rightarrow (\psi \land \varphi)$
  by auto

lemma MMI_ancli: assumes A1: $\varphi \rightarrow \psi$
  shows $\varphi \rightarrow (\varphi \land \psi)$
  using assms by auto

lemma MMI_ad2antlr: assumes A1: $\varphi \rightarrow \psi$
  shows $(\chi \land \varphi) \land \vartheta \rightarrow \varphi$
  using assms by auto

lemma MMI_bimpa: assumes A1: $\varphi \rightarrow (\psi \leftrightarrow \chi)$
  shows $(\varphi \land \psi) \rightarrow \chi$
  using assms by auto

lemma MMI_sylan2i: assumes A1: $\varphi \rightarrow ((\psi \land \chi) \rightarrow \vartheta)$ and
  A2: $\tau \rightarrow \chi$
  shows $\varphi \rightarrow ((\psi \land \tau) \rightarrow \vartheta)$
  using assms by auto

lemma MMI_3jca: assumes A1: $\varphi \rightarrow \psi$ and
  A2: $\varphi \rightarrow \chi$ and

1400
A3: $\phi \rightarrow \theta$
shows $\phi \rightarrow (\psi \land \text{ch} \land \theta)$
using asms by auto

lemma MMI_com34: assumes A1: $\phi \rightarrow (\psi \rightarrow (\text{ch} \rightarrow (\theta \rightarrow \tau)))$
shows $\phi \rightarrow (\psi \rightarrow (\theta \rightarrow (\text{ch} \rightarrow \tau)))$
using asms by auto

lemma MMI_imp43: assumes A1: $\phi \rightarrow (\psi \rightarrow (\text{ch} \rightarrow (\theta \rightarrow \tau)))$
shows $(\phi \land \psi \land \text{ch}) \rightarrow \tau$
using asms by auto

lemma MMI_3anass:
shows $(\phi \land \psi \land \text{ch}) \iff (\phi \land (\psi \land \text{ch}))$
by auto

lemma MMI_3eqtr4r: assumes A1: $A = B$ and
A2: $C = A$ and
A3: $D = B$
shows $D = C$
using asms by auto

lemma MMI_jctl: assumes A1: $\psi$
shows $\phi \rightarrow (\psi \land \phi)$
using asms by auto

lemma MMI_sylibr: assumes A1: $\phi \rightarrow \psi$ and
A2: $\text{ch} \iff \psi$
shows $\phi \rightarrow \text{ch}$
using asms by auto

lemma MMI_mpanl1: assumes A1: $\phi$ and
A2: $(\phi \land \psi) \land \text{ch} \rightarrow \theta$
shows $(\psi \land \text{ch}) \rightarrow \theta$
using asms by auto

lemma MMI_a1i: assumes A1: $\psi$
shows $\phi \rightarrow \psi$
using asms by auto

lemma (in MMI_sar0) MMI_opreqan12rd: assumes A1: $\phi \rightarrow A = B$ and
A2: $\psi \rightarrow C = D$
shows
$(\psi \land \phi) \rightarrow (A + C) = (B + D)$
$(\psi \land \phi) \rightarrow (A \cdot C) = (B \cdot D)$
(ψ ∧ ϕ) → (A - C) = (B - D)
(ψ ∧ ϕ) → (A / C) = (B / D)

using assms by auto

lemma MMI_3adantl3: assumes A1: ((φ ∧ ψ) ∧ ch) → θ
  shows ((φ ∧ ψ ∧ τ) ∧ ch) → θ
using assms by auto

lemma MMI_sylbi: assumes A1: φ ↔ ψ and
A2: ψ → ch
shows φ → ch
using assms by auto

lemma MMI_eirr:
  shows ¬(A ∈ A)
by (rule mem_not_refl)

lemma MMI_eleqii: assumes A1: A = B
  shows A ∈ C ↔ B ∈ C
using assms by auto

lemma MMI_mtbir: assumes A1: ¬(ψ) and
A2: φ ↔ ψ
  shows ¬(ψ)
using assms by auto

lemma MMI_mto: assumes A1: ¬(ψ) and
A2: φ → ψ
  shows ¬(ψ)
using assms by auto

lemma MMI_df_nel:
  shows (A /∈ B ↔ ¬(A ∈ B))
by auto

lemma MMI_snid: assumes A1: A isASet
  shows A ∈ {A}
using assms by auto

lemma MMI_en2lp:
  shows ¬(A ∈ B ∧ B ∈ A)
proof
  assume A1: A ∈ B ∧ B ∈ A
  then have A ∈ B by simp
moreover
  { assume ¬(¬(A ∈ B ∧ B ∈ A))
    then have B ∈ A by auto
  }
ultimately have \( \neg ( A \in B \land B \in A ) \)
by (rule mem_asym)
with A1 show False by simp
qed

lemma MMI_imnan:
  shows ( \( \varphi \rightarrow \neg ( \psi ) \) ) \iff \( \neg ( ( \varphi \land \psi ) ) \)
by auto

lemma MMI_sseqtr4: assumes A1: A \subseteq B and
    A2: C = B
  shows A \subseteq C
using assms by auto

lemma MMI_ssun1:
  shows A \subseteq ( A \cup B )
by auto

lemma MMI_ibar:
  shows \( \varphi \rightarrow ( \psi \leftrightarrow ( \varphi \land \psi ) ) \)
by auto

lemma MMI_mtbiri: assumes Amin: \( \neg ( \text{ch} ) \) and
    Amaj: \( \varphi \rightarrow ( \psi \leftrightarrow \text{ch} ) \)
  shows \( \varphi \rightarrow \neg ( \psi ) \)
using assms by auto

lemma MMI_con2i: assumes Aa: \( \varphi \rightarrow \neg ( \psi ) \)
  shows \( \psi \rightarrow \neg ( \varphi ) \)
using assms by auto

lemma MMI_intnand: assumes A1: \( \varphi \rightarrow \neg ( \psi ) \)
  shows \( \varphi \rightarrow \neg ( ( \text{ch} \land \psi ) ) \)
using assms by auto

lemma MMI_intnanrd: assumes A1: \( \varphi \rightarrow \neg ( \psi ) \)
  shows \( \varphi \rightarrow \neg ( ( \psi \land \text{ch} ) ) \)
using assms by auto

lemma MMI_biorf:
  shows \( \neg ( \varphi ) \rightarrow ( \psi \leftrightarrow ( \varphi \lor \psi ) ) \)
by auto

lemma MMI_bitr2d: assumes A1: \( \varphi \rightarrow ( \psi \leftrightarrow \text{ch} ) \) and
    A2: \( \varphi \rightarrow ( \text{ch} \leftrightarrow \vartheta ) \)
  shows \( \varphi \rightarrow ( \vartheta \leftrightarrow \psi ) \)
using assms by auto

1403
lemma MMI_orass:
shows \((\varphi \lor \psi) \lor \text{ch}) \iff (\varphi \lor (\psi \lor \text{ch}))\)
by auto

lemma MMI_orcom:
shows \((\varphi \lor \psi) \iff (\psi \lor \varphi)\)
by auto

lemma MMI_3bitr4d: assumes A1: \(\varphi \rightarrow (\psi \leftrightarrow \text{ch})\)
A2: \(\varphi \rightarrow (\vartheta \leftrightarrow \psi)\)
A3: \(\varphi \rightarrow (\tau \leftrightarrow \text{ch})\)
shows \(\varphi \rightarrow (\vartheta \leftrightarrow \tau)\)
using assms by auto

lemma MMI_3imtr4d: assumes A1: \(\varphi \rightarrow (\psi \rightarrow \text{ch})\)
A2: \(\varphi \rightarrow (\vartheta \leftrightarrow \psi)\)
A3: \(\varphi \rightarrow (\tau \leftrightarrow \text{ch})\)
shows \(\varphi \rightarrow (\vartheta \rightarrow \tau)\)
using assms by auto

lemma MMI_3impdi: assumes A1: \((\varphi \land \psi) \land (\varphi \land \text{ch})\) \(\rightarrow \vartheta\)
shows \((\varphi \land \psi \land \text{ch}) \rightarrow \vartheta\)
using assms by auto

lemma MMI_bi2anan9: assumes A1: \(\varphi \rightarrow (\psi \leftrightarrow \text{ch})\)
A2: \(\vartheta \rightarrow (\vartheta \rightarrow \eta)\)
shows \((\varphi \land \vartheta) \rightarrow ((\varphi \land \tau) \leftrightarrow (\text{ch} \land \eta))\)
using assms by auto

lemma MMI_ssel2:
shows \((A \subseteq B \land C \in A) \rightarrow C \in B)\)
by auto

lemma MMI_an1rs: assumes A1: \((\varphi \land \psi) \land \text{ch})\) \(\rightarrow \vartheta\)
shows \((\varphi \land \text{ch}) \land \psi \rightarrow \vartheta\)
using assms by auto

lemma MMI_ralbidva: assumes A1: \(\forall x. (\varphi \land x \in A) \rightarrow (\psi(x) \leftrightarrow \text{ch}(x))\)
shows \(\varphi \rightarrow ((\forall x \in A. \psi(x)) \leftrightarrow (\forall x \in A. \text{ch}(x)))\)
using assms by auto
lemma MMI_rexbidva: assumes A1: \( \forall x. ( \varphi \land x \in A ) \rightarrow ( \psi(x) \leftrightarrow \text{ch}(x) ) \)
shows \( \varphi \rightarrow ( ( \exists x \in A . \psi(x) ) \leftrightarrow ( \exists x \in A . \text{ch}(x) ) ) \)
using assms by auto

lemma MMI_con2bid: assumes A1: \( \varphi \rightarrow ( \psi \leftrightarrow \neg ( \text{ch} ) ) \)
shows \( \varphi \rightarrow ( \text{ch} \leftrightarrow \neg ( \psi ) ) \)
using assms by auto

lemma MMI_so: assumes A1: \( \forall x y z. ( x \in A \land y \in A \land z \in A ) \rightarrow \)
\( ( ( (x,y) \in R \leftrightarrow \neg ( ( x = y \lor (y, x) \in R ) ) ) \land \)
\( ( ( (x,y) \in R \land (y,z) \in R ) \rightarrow (x,z) \in R ) ) \)
shows R Orders A
using assms StrictOrder_def by auto

lemma MMI_conlbid: assumes A1: \( \varphi \rightarrow ( \neg ( \psi ) \leftrightarrow \text{ch} ) \)
shows \( \varphi \rightarrow ( ( \neg ( \text{ch} ) \leftrightarrow \psi ) ) \)
using assms by auto

lemma MMI_sotrieq:
shows \( ( (R \text{ Orders A} ) \land ( B \in A \land C \in A ) ) \rightarrow \)
\( ( B = C \leftrightarrow \neg ( ( B,C ) \in R \lor ( C, B ) \in R ) ) ) \)
proof -
{ assume A1: R Orders A and A2: B \in A \land C \in A
from A1 have \( \forall x y z. ( x \in A \land y \in A \land z \in A ) \rightarrow \)
\( ( (x,y) \in R \leftrightarrow \neg (x=y \lor (y,x) \in R) ) \land \)
\( ( (x,y) \in R \land (y,z) \in R \rightarrow (x,z) \in R) \)
by (unfold StrictOrder_def)
then have \( \forall x y. x \in A \land y \in A \rightarrow ( (x,y) \in R \leftrightarrow \neg (x=y \lor (y,x) \in R) ) \)
by auto
with A2 have I: \( (B,C) \in R \leftrightarrow \neg (B=C \lor (C,B) \in R) \)
by blast
then have B = C \( \leftrightarrow \neg ( ( B,C ) \in R \lor ( C, B ) \in R ) ) \)
by auto
} then show \( ( (R \text{ Orders A}) \land ( B \in A \land C \in A ) ) \rightarrow \)
\( ( B = C \leftrightarrow \neg ( ( B,C ) \in R \lor ( C, B ) \in R ) ) \) by simp
qed

lemma MMI_bicomd: assumes A1: \( \varphi \rightarrow ( \psi \leftrightarrow \text{ch} ) \)
shows \( \varphi \rightarrow ( ( \text{ch} \leftrightarrow \psi ) ) \)
using assms by auto

lemma MMI_sotrieq2:
shows ( R Orders A \land ( B \in A \land C \in A ) ) \rightarrow
( B = C \leftrightarrow ( \neg ( \langle B, C \rangle \in R ) \land \neg ( \langle C, B \rangle \in R ) ) )
using MMI_sotrieq by auto

lemma MMI_orc:
  shows \phi \rightarrow ( \phi \lor \psi )
by auto

lemma MMI_syl6bbr: assumes A1: \phi \rightarrow ( \psi \leftrightarrow \text{ch} ) and
  A2: \theta \leftrightarrow \text{ch}
  shows \phi \rightarrow ( \psi \leftrightarrow \theta )
  using assms by auto

lemma MMI_orbi1i: assumes A1: \phi \leftrightarrow \psi
  shows ( \phi \lor \text{ch} ) \leftrightarrow ( \psi \lor \text{ch} )
  using assms by auto

lemma MMI_syl5rbbr: assumes A1: \phi \rightarrow ( \psi \leftrightarrow \text{ch} ) and
  A2: \psi \leftrightarrow \theta
  shows \phi \rightarrow ( \text{ch} \leftrightarrow \theta )
  using assms by auto

lemma MMI_anbi2d: assumes A1: \phi \rightarrow ( \psi \leftrightarrow \text{ch} )
  shows \phi \rightarrow ( ( \theta \land \psi ) \leftrightarrow ( \theta \land \text{ch} ) )
  using assms by auto

lemma MMI(ord): assumes A1: \phi \rightarrow ( \psi \lor \text{ch} )
  shows \phi \rightarrow ( \neg ( \psi ) \rightarrow \text{ch} )
  using assms by auto

lemma MMI_impbid: assumes A1: \phi \rightarrow ( \psi \rightarrow \text{ch} ) and
  A2: \phi \rightarrow ( \text{ch} \rightarrow \psi )
  shows \phi \rightarrow ( \psi \leftrightarrow \text{ch} )
  using assms by blast

lemma MMI_jcad: assumes A1: \phi \rightarrow ( \psi \rightarrow \text{ch} ) and
  A2: \phi \rightarrow ( \psi \rightarrow \theta )
  shows \phi \rightarrow ( \psi \rightarrow ( \text{ch} \land \theta ) )
  using assms by auto

lemma MMI_ax_1:
  shows \phi \rightarrow ( \psi \rightarrow \phi )
by auto

lemma MMI_pm2_24:
  shows \phi \rightarrow ( \neg ( \phi ) \rightarrow \psi )
by auto

1406
lemma MMI_imp3a: assumes $A1: \varphi \rightarrow (\psi \rightarrow (ch \rightarrow \vartheta))$
    shows $\varphi \rightarrow ((\psi \land ch) \rightarrow \vartheta)$
    using assms by auto

lemma (in MMIsar0) MMI_breq1:
    shows $A = B \rightarrow (A \leq C \leftrightarrow B \leq C)$
    $A = B \rightarrow (A < C \leftrightarrow B < C)$
    by auto

lemma MMI_bimplrd: assumes $A1: \varphi \rightarrow (\psi \leftrightarrow ch)$
    shows $\varphi \rightarrow (ch \rightarrow \psi)$
    using assms by auto

lemma MMI_jaod: assumes $A1: \varphi \rightarrow (\psi \rightarrow ch)$ and
    $A2: \varphi \rightarrow (\vartheta \rightarrow ch)$
    shows $\varphi \rightarrow ((\psi \lor \vartheta) \rightarrow ch)$
    using assms by auto

lemma MMI_com23: assumes $A1: \varphi \rightarrow (\psi \rightarrow (ch \rightarrow \vartheta))$
    shows $\varphi \rightarrow (ch \rightarrow (\psi \rightarrow \vartheta))$
    using assms by auto

lemma (in MMIsar0) MMI_breq2:
    shows $A = B \rightarrow (C \leq A \leftrightarrow C \leq B)$
    $A = B \rightarrow (C < A \leftrightarrow C < B)$
    by auto

lemma MMI_syld: assumes $A1: \varphi \rightarrow (\psi \rightarrow ch)$ and
    $A2: \varphi \rightarrow (ch \rightarrow \vartheta)$
    shows $\varphi \rightarrow (\psi \rightarrow \vartheta)$
    using assms by auto

lemma MMI_bimpred: assumes $A1: \varphi \rightarrow (\psi \leftrightarrow ch)$
    shows $\psi \rightarrow (\varphi \rightarrow ch)$
    using assms by auto

lemma MMI_mp2and: assumes $A1: \varphi \rightarrow \psi$ and
    $A2: \varphi \rightarrow ch$ and
    $A3: \varphi \rightarrow ((\psi \land ch) \rightarrow \vartheta)$
    shows $\varphi \rightarrow \vartheta$
    using assms by auto

lemma MMI_sonr:
    shows $(R \text{ Orders }A \land B \in A) \rightarrow \neg((B,B) \in R)$

1407
unfolding StrictOrder_def by auto

lemma MMI_orri: assumes A1: ¬ ( φ ) → ψ
    shows φ ∨ ψ
    using assms by auto

lemma MMI_mpbiri: assumes Amin: ch and
    Amaj: φ → ( ψ ↔ ch )
    shows φ → ψ
    using assms by auto

lemma MMI_pm2_46:
    shows ¬ ( ( φ ∨ ψ ) ) → ¬ ( ψ )
    by auto

lemma MMI_elun:
    shows A ∈ ( B ∪ C ) ↔ ( A ∈ B ∨ A ∈ C )
    by auto

lemma (in MMIar0) MMI_pnfxr:
    shows +∞ ∈ R’
    using cxr_def by simp

lemma MMI_elisseti: assumes A1: A ∈ B
    shows A isASet
    using assms by auto

lemma (in MMIar0) MMI_mnfxr:
    shows −∞ ∈ R’
    using cxr_def by simp

lemma MMI_elpr2: assumes A1: B isASet and
    A2: C isASet
    shows A ∈ { B , C } ↔ ( A = B ∨ A = C )
    using assms by auto

lemma MMI_orbi2i: assumes A1: φ ↔ ψ
    shows ( ch ∨ φ ) ↔ ( ch ∨ ψ )
    using assms by auto

lemma MMI_3orass:
    shows ( φ ∨ ψ ∨ ch ) ↔ ( φ ∨ ( ψ ∨ ch ) )
    by auto

lemma MMI_bitr4: assumes A1: φ ↔ ψ and
    A2: ch ↔ ψ
    shows φ ↔ ch
    using assms by auto
lemma MMI_eleq2:
  shows A = B \implies ( C \in A \iff C \in B )
  by auto

lemma MMI_nelneq:
  shows ( A \in C \land \neg ( B \in C ) ) \implies \neg ( A = B )
  by auto

lemma MMI_df_pr:
  shows \{ A , B \} = ( \{ A \} \cup \{ B \} )
  by auto

lemma MMI_ineq2i: assumes A1: A = B
  shows ( C \cap A ) = ( C \cap B )
  using assms by auto

lemma MMI_mt2: assumes A1: \psi and
  A2: \varphi \implies \neg ( \psi )
  shows \neg ( \varphi )
  using assms by auto

lemma MMI_disjsn:
  shows ( A \cap \{ B \} ) = 0 \iff \neg ( B \in A )
  by auto

lemma MMI_undisj2:
  shows ( A \cap B ) = 0 \land ( A \cap C ) = 0 \iff ( A \cap ( B \cup C ) ) = 0
  by auto

lemma MMI_disjssun:
  shows ( A \cap B ) = 0 \implies ( A \subseteq ( B \cup C ) \iff A \subseteq C )
  by auto

lemma MMI_uncom:
  shows ( A \cup B ) = ( B \cup A )
  by auto

lemma MMI_sseq2i: assumes A1: A = B
  shows ( C \subseteq A \iff C \subseteq B )
  using assms by auto

lemma MMI_disj:
  shows ( A \cap B ) = 0 \iff ( \forall x \in A . \neg ( x \in B ) )
  by auto
lemma MMI_syl5ibr: assumes A1: \( \varphi \rightarrow ( \psi \rightarrow \text{ch} ) \) and 
A2: \( \psi \leftrightarrow \vartheta \)
shows \( \varphi \rightarrow ( \vartheta \rightarrow \text{ch} ) \)
using assms by auto

lemma MMI_con3d: assumes A1: \( \varphi \rightarrow ( \psi \rightarrow \text{ch} ) \)
shows \( \varphi \rightarrow ( \neg ( \text{ch} ) \rightarrow \neg ( \psi ) ) \)
using assms by auto

lemma MMI_dfrex2:
shows \( ( \exists x \in A . \varphi(x) ) \leftrightarrow \neg ( ( \forall x \in A . \neg \varphi(x) ) ) \)
by auto

lemma MMI_visset:
shows \( x \text{ isASet} \)
by auto

lemma MMI_elpr: assumes A1: \( A \text{ isASet} \)
shows \( A \in \{ B , C \} \leftrightarrow ( A = B \lor A = C ) \)
using assms by auto

lemma MMI_rexbii: assumes A1: \( \forall x. \varphi(x) \leftrightarrow \psi(x) \)
shows \( ( \exists x \in A . \varphi(x) ) \leftrightarrow ( \exists x \in A . \psi(x) ) \)
using assms by auto

lemma MMI_orbi12i_orig: assumes A1: \( \varphi \leftrightarrow \psi \) and 
A2: \( \text{ch} \leftrightarrow \vartheta \)
shows \( \varphi \lor \text{ch} \leftrightarrow ( \psi \lor \vartheta ) \)
using assms by auto

lemma MMI_orbi12i: assumes A1: \( (\exists x. \varphi(x)) \leftrightarrow \psi \) and
A2: \((\exists x. \text{ch}(x)) \leftrightarrow \vartheta\)
shows \((\exists x. \varphi(x)) \lor (\exists x. \text{ch}(x)) \leftrightarrow (\psi \lor \vartheta)\)
using assms by auto

lemma MMI_syl6ib: assumes A1: \(\varphi \rightarrow (\psi \rightarrow \text{ch})\) and
A2: \(\text{ch} \leftrightarrow \vartheta\)
shows \(\varphi \rightarrow (\psi \rightarrow \vartheta)\)
using assms by auto

lemma MMI_intnan: assumes A1: \(\neg (\varphi)\)
shows \(\neg((\psi \land \varphi))\)
using assms by auto

lemma MMI_intnanr: assumes A1: \(\neg (\varphi)\)
shows \(\neg((\varphi \land \psi))\)
using assms by auto

lemma MMI_pm3_2ni: assumes A1: \(\neg (\varphi)\) and
A2: \(\neg (\psi)\)
shows \(\neg((\varphi \lor \psi))\)
using assms by auto

lemma (in MMIIsar0) MMI_breq12:
sows
\((A = B \land C = D) \rightarrow (A \leq C \leftrightarrow B < D)\)
\((A = B \land C = D) \rightarrow (A \leq C \leftrightarrow B \leq D)\)
by auto

lemma MMI_necom:
sows \(A \neq B \leftrightarrow B \neq A\)
by auto

lemma MMI_3jaoi: assumes A1: \(\varphi \rightarrow \psi\) and
A2: \(\text{ch} \rightarrow \psi\) and
A3: \(\vartheta \rightarrow \psi\)
shows \((\varphi \lor \text{ch} \lor \vartheta) \rightarrow \psi\)
using assms by auto

lemma MMI_jctr: assumes A1: \(\psi\)
shows \(\varphi \rightarrow (\varphi \land \psi)\)
using assms by auto

lemma MMI_olc:
sows \(\neg \varphi \rightarrow (\psi \lor \varphi)\)
by auto

lemma MMI_3syl: assumes A1: \(\varphi \rightarrow \psi\) and
A2: \(\psi \rightarrow \text{ch}\) and
A3: \(\text{ch} \rightarrow \vartheta\)
shows $\varphi \rightarrow \vartheta$
using assms by auto

lemma MMI_mtbird: assumes Amin: $\varphi \rightarrow \neg ( \text{ch} )$ and
    Amaj: $\varphi \rightarrow ( \psi \leftrightarrow \text{ch} )$
shows $\varphi \rightarrow \neg ( \psi )$
using assms by auto

lemma MMI_pm2_21d: assumes A1: $\varphi \rightarrow \neg ( \psi )$
    shows $\varphi \rightarrow ( \psi \rightarrow \text{ch} )$
    using assms by auto

lemma MMI_3jaodan: assumes A1: $( \varphi \land \psi ) \rightarrow \text{ch}$ and
    A2: $( \varphi \land \vartheta ) \rightarrow \text{ch}$ and
    A3: $( \varphi \land \tau ) \rightarrow \text{ch}$
shows $( \varphi \land ( \psi \lor \vartheta \lor \tau ) ) \rightarrow \text{ch}$
using assms by auto

lemma MMI_sylan2br: assumes A1: $( \varphi \land \psi ) \rightarrow \text{ch}$ and
    A2: $\psi \leftrightarrow \vartheta$
shows $( \varphi \land \vartheta ) \rightarrow \text{ch}$
using assms by auto

lemma MMI_3jaoian: assumes A1: $( \varphi \land \psi ) \rightarrow \text{ch}$ and
    A2: $( \vartheta \land \psi ) \rightarrow \text{ch}$ and
    A3: $( \tau \land \psi ) \rightarrow \text{ch}$
shows $( ( \varphi \lor \vartheta \lor \tau ) \land \psi ) \rightarrow \text{ch}$
using assms by auto

lemma MMI_mtbid: assumes Amin: $\varphi \rightarrow \neg ( \psi )$ and
    Amaj: $\varphi \rightarrow ( \psi \leftrightarrow \text{ch} )$
shows $\varphi \rightarrow \neg ( \text{ch} )$
using assms by auto

lemma MMI_con1d: assumes A1: $\varphi \rightarrow ( \neg ( \psi ) \rightarrow \text{ch} )$
    shows $\varphi \rightarrow ( \neg ( \text{ch} ) \rightarrow \psi )$
    using assms by auto

lemma MMI_pm2_21nd: assumes A1: $\varphi \rightarrow \psi$
    shows $\varphi \rightarrow ( \neg ( \psi ) \rightarrow \text{ch} )$
    using assms by auto

lemma MMI_syl3an1b: assumes A1: $( \varphi \land \psi \land \text{ch} ) \rightarrow \vartheta$ and
    A2: $\tau \leftrightarrow \varphi$
shows $( \tau \land \psi \land \text{ch} ) \rightarrow \vartheta$
using assms by auto

1412
lemma MMI_adantld: assumes A1: $\varphi \rightarrow (\psi \rightarrow \text{ch})$
  shows $\varphi \rightarrow ((\psi \land \vartheta) \rightarrow \text{ch})$
  using assms by auto

lemma MMI_adantrd: assumes A1: $\varphi \rightarrow (\psi \rightarrow \text{ch})$
  shows $\varphi \rightarrow ((\psi \land \vartheta) \rightarrow \text{ch})$
  using assms by auto

lemma MMI_anasss: assumes A1: $(\varphi \land \psi) \land \text{ch} \rightarrow \vartheta$
  shows $(\varphi \land (\psi \land \text{ch})) \rightarrow \vartheta$
  using assms by auto

lemma MMI_syl3an3b: assumes A1: $(\varphi \land \psi \land \text{ch}) \rightarrow \vartheta$
and A2: $\tau \leftrightarrow \text{ch}$
  shows $(\varphi \land \psi \land \tau) \rightarrow \vartheta$
  using assms by auto

lemma MMI_mpbid: assumes Amin: $\varphi \rightarrow \psi$ and
  Amaj: $\varphi \rightarrow (\psi \leftrightarrow \text{ch})$
  shows $\varphi \rightarrow \text{ch}$
  using assms by auto

lemma MMI_orbi12d: assumes A1: $\varphi \rightarrow (\psi \leftrightarrow \text{ch})$
and A2: $\varphi \rightarrow (\vartheta \leftrightarrow \tau)$
  shows $\varphi \rightarrow ((\psi \lor \vartheta) \leftrightarrow (\text{ch} \lor \tau))$
  using assms by auto

lemma MMI_ianor:
  shows $\neg(\varphi \land \psi) \leftrightarrow \neg\varphi \lor \neg\psi$
  by auto

lemma MMI_bitr2: assumes A1: $\varphi \leftrightarrow \psi$
and A2: $\psi \leftrightarrow \text{ch}$
  shows $\text{ch} \leftrightarrow \varphi$
  using assms by auto

lemma MMI_bimp: assumes A1: $\varphi \leftrightarrow \psi$
  shows $\varphi \rightarrow \psi$
  using assms by auto

lemma MMI_mpan2d: assumes A1: $\varphi \rightarrow \text{ch}$ and
  A2: $\varphi \rightarrow ((\psi \land \text{ch}) \rightarrow \vartheta)$
  shows $\varphi \rightarrow (\psi \rightarrow \vartheta)$
  using assms by auto

lemma MMI_ad2antrr: assumes A1: $\varphi \rightarrow \psi$

1413
shows ( ((\varphi \land \text{ch}) \land \theta) \rightarrow \psi)
using assms by auto

lemma MMI_biimpac: assumes A1: \varphi \rightarrow (\psi \leftrightarrow \text{ch})
shows (\psi \land \varphi) \rightarrow \text{ch}
using assms by auto

lemma MMI_con2bii: assumes A1: \varphi \leftrightarrow \neg(\psi)
shows \psi \leftrightarrow \neg(\varphi)
using assms by auto

lemma MMI_pm3_26bd: assumes A1: \varphi \leftrightarrow (\psi \land \text{ch})
shows \varphi \rightarrow \psi
using assms by auto

lemma MMI_bimpr: assumes A1: \varphi \leftrightarrow \psi
shows \psi \rightarrow \varphi
using assms by auto

lemma (in MMIIsar0) MMI_3brtr3g: assumes A1: \varphi \rightarrow A < B and
A2: A = C and
A3: B = D
shows \varphi \rightarrow C < D
using assms by auto

lemma (in MMIIsar0) MMI_breq12i: assumes A1: A = B and
A2: C = D
shows A < C \leftrightarrow B < D
A \leq C \leftrightarrow B \leq D
using assms by auto

lemma MMI_negbii: assumes Aa: \varphi \leftrightarrow \psi
shows \neg\varphi \leftrightarrow \neg\psi
using assms by auto

lemma (in MMIIsar0) MMI_breq1i: assumes A1: A = B
shows A < C \leftrightarrow B < C
A \leq C \leftrightarrow B \leq C
using assms by auto
lemma MMI_syl5eqr: assumes A1: \( \varphi \rightarrow A = B \) and  
A2: A = C  
shows \( \varphi \rightarrow C = B \)  
using assms by auto

lemma (in MMIIsar0) MMI_breq2d: assumes A1: \( \varphi \rightarrow A = B \)  
shows  
\( \varphi \rightarrow C < A \leftrightarrow C < B \)  
\( \varphi \rightarrow C \leq A \leftrightarrow C \leq B \)  
using assms by auto

lemma MMI_ccase: assumes A1: \( \varphi \land \psi \rightarrow \tau \) and  
A2: ch \land \psi \rightarrow \tau \) and  
A3: \( \varphi \land \vartheta \rightarrow \tau \) and  
A4: ch \land \vartheta \rightarrow \tau \)  
shows (\( \varphi \lor ch \land (\psi \lor \vartheta) \rightarrow \tau \)  
using assms by auto

lemma MMI_pm3_27bd: assumes A1: \( \varphi \leftrightarrow \psi \land ch \)  
shows \( \varphi \rightarrow ch \)  
using assms by auto

lemma MMI_nsy13: assumes A1: \( \varphi \rightarrow \neg \psi \) and  
A2: ch \rightarrow \psi  
shows ch \rightarrow \neg \varphi  
using assms by auto

lemma MMI_jctild: assumes A1: \( \varphi \rightarrow \psi \rightarrow ch \) and  
A2: \( \varphi \rightarrow \vartheta \)  
shows \( \varphi \rightarrow \)  
\( \psi \rightarrow \vartheta \land ch \)  
using assms by auto

lemma MMI_jctird: assumes A1: \( \varphi \rightarrow \psi \rightarrow ch \) and  
A2: \( \varphi \rightarrow \vartheta \)  
shows \( \varphi \rightarrow \)  
\( \psi \rightarrow ch \land \vartheta \)  
using assms by auto

lemma MMI_ccase2: assumes A1: \( \varphi \land \psi \rightarrow \tau \) and  
A2: ch \rightarrow \tau \) and  
A3: \( \vartheta \rightarrow \tau \)  
shows (\( \varphi \lor ch \land (\psi \lor \vartheta) \rightarrow \tau \)  
using assms by auto

lemma MMI_3bitr3r: assumes A1: \( \varphi \leftrightarrow \psi \) and
A2: \( \varphi \leftrightarrow \text{ch} \) and
A3: \( \psi \leftrightarrow \vartheta \)
shows \( \vartheta \leftrightarrow \text{ch} \)
using assms by auto

lemma (in MMIIsar0) MMI_syl6breq: assumes A1: \( \varphi \rightarrow A < B \) and
A2: \( B = C \)
shows
\( \varphi \rightarrow A < C \)
using assms by auto

lemma MMI_pm2_6ii: assumes A1: \( \varphi \rightarrow \psi \) and
A2: \( \neg \varphi \rightarrow \psi \)
shows \( \psi \)
using assms by auto

lemma MMI_syl6req: assumes A1: \( \varphi \rightarrow A = B \) and
A2: \( B = C \)
shows \( \varphi \rightarrow C = A \)
using assms by auto

lemma MMI_pm2_6id: assumes A1: \( \varphi \rightarrow \psi \rightarrow \text{ch} \) and
A2: \( \varphi \rightarrow \neg \psi \rightarrow \text{ch} \)
shows \( \varphi \rightarrow \text{ch} \)
using assms by auto

lemma MMI_orim1d: assumes A1: \( \varphi \rightarrow \psi \rightarrow \text{ch} \)
shows \( \varphi \rightarrow \psi \lor \vartheta \rightarrow \text{ch} \lor \vartheta \)
using assms by auto

lemma (in MMIIsar0) MMI_breq1d: assumes A1: \( \varphi \rightarrow A = B \)
shows
\( \varphi \rightarrow A < C \leftrightarrow B < C \)
\( \varphi \rightarrow A \leq C \leftrightarrow B \leq C \)
using assms by auto

lemma (in MMIIsar0) MMI_breq12d: assumes A1: \( \varphi \rightarrow A = B \) and
A2: \( \varphi \rightarrow C = D \)
shows
\( \varphi \rightarrow A < C \leftrightarrow B < D \)
\( \varphi \rightarrow A \leq C \leftrightarrow B \leq D \)
using assms by auto

lemma MMI_bibi2d: assumes A1: \( \varphi \rightarrow \psi \rightarrow \text{ch} \)
\( \psi \leftrightarrow \text{ch} \)
shows $\varphi \rightarrow$

$(\vartheta \leftrightarrow \psi) \leftrightarrow$

$\vartheta \leftrightarrow \text{ch}$

using assms by auto

lemma MMI_con4bid: assumes A1: $\varphi \rightarrow$

$\neg \psi \leftrightarrow \neg \text{ch}$

shows $\varphi \rightarrow$

$\psi \leftrightarrow \text{ch}$

using assms by auto

lemma MMI_3com13: assumes A1: $\varphi \land \psi \land \text{ch} \rightarrow \vartheta$

shows $\text{ch} \land \psi \land \varphi \rightarrow \vartheta$

using assms by auto

lemma MMI_3bitr3rd: assumes A1: $\varphi \rightarrow$

$\psi \leftrightarrow \text{ch}$ and

A2: $\varphi \rightarrow$

$\psi \leftrightarrow \vartheta$ and

A3: $\varphi \rightarrow$

$\text{ch} \leftrightarrow \tau$

shows $\varphi \rightarrow$

$\tau \leftrightarrow \vartheta$

using assms by auto

lemma MMI_3imtr4g: assumes A1: $\varphi \rightarrow \psi \rightarrow \text{ch}$ and

A2: $\vartheta \leftrightarrow \psi$ and

A3: $\tau \leftrightarrow \text{ch}$

shows $\varphi \rightarrow$

$\vartheta \rightarrow \tau$

using assms by auto

lemma MMI_expcom: assumes A1: $\varphi \land \psi \rightarrow \text{ch}$

shows $\psi \rightarrow \varphi \rightarrow \text{ch}$

using assms by auto

lemma (in MMIIsar0) MMI_breq2i: assumes A1: $A = B$

shows

$C < A \\leftrightarrow C < B$

$C \leq A \\leftrightarrow C \leq B$

using assms by auto

lemma MMI_3bitr2r: assumes A1: $\varphi \leftrightarrow \psi$ and

A2: $\text{ch} \leftrightarrow \psi$ and

A3: $\vartheta \leftrightarrow \varphi$

shows $\vartheta \leftrightarrow \varphi$
using assms by auto

lemma MMI_dedth4h: assumes A1: A = \( \text{if}(\varphi, A, R) \rightarrow \tau \) and
\( \eta \rightarrow \zeta \) and
A2: B = \( \text{if}(\psi, B, S) \rightarrow \eta \) and
\( \zeta \rightarrow \text{si} \) and
A3: C = \( \text{if}(\text{ch}, C, F) \rightarrow \zeta \) and
\( \text{si} \leftrightarrow \text{rh} \) and
A4: D = \( \text{if}(\vartheta, D, G) \rightarrow \text{si} \leftrightarrow \text{rh} \) and
A5: rh
shows \( (\varphi \land \psi) \land \text{ch} \land \vartheta \rightarrow \tau \)
using assms by auto

lemma MMI_anb1id: assumes A1: \( \varphi \rightarrow \psi \rightarrow \text{ch} \)
shows \( \varphi \rightarrow \psi \land \vartheta \rightarrow \text{ch} \land \vartheta \)
using assms by auto

lemma (in MMIIsar0) MMI_breqtrrd: assumes A1: \( \varphi \rightarrow A < B \) and
A2: \( \varphi \rightarrow C = B \)
shows \( \varphi \rightarrow A < C \)
using assms by auto

lemma MMI_syl3an: assumes A1: \( \varphi \land \psi \land \text{ch} \rightarrow \vartheta \) and
A2: \( \tau \rightarrow \varphi \) and
A3: \( \eta \rightarrow \psi \) and
A4: \( \zeta \rightarrow \text{ch} \)
shows \( \tau \land \eta \land \zeta \rightarrow \vartheta \)
using assms by auto

lemma MMI_3bitrd: assumes A1: \( \varphi \rightarrow \psi \rightarrow \text{ch} \) and
A2: \( \varphi \rightarrow \vartheta \) and
\( \text{ch} \leftrightarrow \vartheta \) and
A3: \( \varphi \rightarrow \vartheta \) and
\( \vartheta \leftrightarrow \tau \)
shows \( \varphi \rightarrow \psi \leftrightarrow \tau \)
using assms by auto

lemma (in MMIIsar0) MMI_breqtr: assumes A1: \( A < B \) and

1418
A2: B = C
shows A < C
using assms by auto

lemma MMI_mpi: assumes A1: ψ and
  A2: ϕ → ψ → ch
shows ϕ → ch
using assms by auto

lemma MMI_eqtr2: assumes A1: A = B and
  A2: B = C
shows C = A
using assms by auto

lemma MMI_eqeqi: assumes A1: A = B ↔ C = D
shows A ≠ B ↔ C ≠ D
using assms by auto

lemma (in MMIsar0) MMI_eqtrrd: assumes A1: ϕ → A = B and
  A2: ϕ → A < C
shows ϕ → B < C
using assms by auto

lemma MMI_mpd: assumes A1: ϕ → ψ and
  A2: ϕ → ψ → ch
shows ϕ → ch
using assms by auto

lemma MMI_mpdan: assumes A1: ϕ → ψ and
  A2: ϕ ∧ ψ → ch
shows ϕ → ch
using assms by auto

lemma (in MMIsar0) MMI_breqtrd: assumes A1: ϕ → A < B and
  A2: ϕ → B = C
shows ϕ → A < C
using assms by auto

lemma MMI_mpand: assumes A1: ϕ → ψ and
  A2: ϕ →
    ψ ∧ ch → ϑ
shows ϕ → ch → ϑ
using assms by auto

lemma MMI_imbi1d: assumes A1: ϕ →
ψ $\leftrightarrow$ ch
shows $\varphi \rightarrow$

$(\psi \rightarrow \vartheta) \leftrightarrow$

$(ch \rightarrow \vartheta)$

using asms by auto

lemma MMI_mtbii: assumes $\text{Amin}: \neg\psi$ and

$\text{Amaj}: \varphi \rightarrow$

$\psi \leftrightarrow ch$
shows $\varphi \rightarrow \neg ch$

using asms by auto

lemma MMI_sylan2d: assumes $A1: \varphi \rightarrow$

$\psi \land ch \rightarrow \vartheta$ and

$A2: \varphi \rightarrow \tau \rightarrow ch$
shows $\varphi \rightarrow$

$\psi \land \tau \rightarrow \vartheta$

using asms by auto

lemma (in MMIisar0) MMI_breqan12d: assumes $A1: \varphi \rightarrow A = B$ and

$A2: \psi \rightarrow C = D$
shows

$\varphi \land \psi \rightarrow A < C \leftrightarrow B < D$

$\varphi \land \psi \rightarrow A \leq C \leftrightarrow B \leq D$

using asms by auto

lemma MMI_a1dd: assumes $A1: \varphi \rightarrow \psi \rightarrow ch$
shows $\varphi \rightarrow$

$\psi \rightarrow \vartheta \rightarrow ch$

using asms by auto

lemma (in MMIisar0) MMI_3brtr3d: assumes $A1: \varphi \rightarrow A \leq B$ and

$A2: \varphi \rightarrow A = C$ and

$A3: \varphi \rightarrow B = D$
shows $\varphi \rightarrow C \leq D$

using asms by auto

lemma MMI_ad2antll: assumes $A1: \varphi \rightarrow \psi$
shows $\text{ch} \land \vartheta \land \varphi \rightarrow \psi$

using asms by auto
lemma MMI_adantrrl: assumes A1: \( \varphi \land \psi \land \chi \rightarrow \vartheta \)
shows \( \varphi \land \psi \land \tau \land \chi \rightarrow \vartheta \)
using assms by auto

lemma MMI_syl2ani: assumes A1: \( \varphi \rightarrow \psi \land \chi \rightarrow \vartheta \) and
A2: \( \tau \rightarrow \psi \) and
A3: \( \eta \rightarrow \chi \)
shows \( \varphi \rightarrow \tau \land \eta \rightarrow \vartheta \)
using assms by auto

lemma MMI_im2anan9: assumes A1: \( \varphi \rightarrow \psi \rightarrow \chi \) and
A2: \( \vartheta \rightarrow \eta \)
shows \( \varphi \land \vartheta \rightarrow \psi \land \tau \rightarrow \chi \land \eta \)
using assms by auto

lemma MMI_ancomsd: assumes A1: \( \varphi \rightarrow \psi \land \chi \rightarrow \vartheta \) and
A2: \( \varphi \rightarrow \chi \land \psi \rightarrow \vartheta \)
shows \( \varphi \rightarrow \chi \rightarrow \vartheta \)
using assms by auto

lemma MMI_mpani: assumes A1: \( \psi \land \chi \rightarrow \vartheta \) and
A2: \( \varphi \and \chi \rightarrow \vartheta \)
shows \( \varphi \rightarrow \chi \rightarrow \vartheta \)
using assms by auto

lemma MMI_syldan: assumes A1: \( \varphi \land \psi \rightarrow \chi \) and
A2: \( \varphi \land \chi \rightarrow \vartheta \)
shows \( \varphi \land \psi \rightarrow \vartheta \)
using assms by auto

lemma MMI_mp3anl1: assumes A1: \( \varphi \and \psi \land \chi \land \vartheta \rightarrow \tau \)
A2: \( (\varphi \land \psi \land \chi) \land \vartheta \rightarrow \tau \)
s shows \( (\psi \land \chi) \land \vartheta \rightarrow \tau \)
using assms by auto

lemma MMI_3ad2ant1: assumes A1: \( \varphi \rightarrow \chi \)
shows \( \varphi \land \psi \land \vartheta \rightarrow \chi \)
using assms by auto

lemma MMI_pm3_2:
shows \( \varphi \rightarrow \)
\( \psi \rightarrow \varphi \land \psi \)
by auto

**lemma MMI_pm2_43i**: assumes A1: \( \varphi \rightarrow \psi \)
shows \( \varphi \rightarrow \psi \)
using assms by auto

**lemma MMI_jctil**: assumes A1: \( \varphi \rightarrow \psi \) and
A2: ch
shows \( \varphi \rightarrow \text{ch} \land \psi \)
using assms by auto

**lemma MMI_mpanl12**: assumes A1: \( \varphi \land \psi \land \text{ch} \rightarrow \vartheta \)
shows \( \text{ch} \rightarrow \vartheta \)
using assms by auto

**lemma MMI_mpanr1**: assumes A1: \( \varphi \land \psi \land \text{ch} \rightarrow \vartheta \)
shows \( \varphi \land \text{ch} \rightarrow \vartheta \)
using assms by auto

**lemma MMI_ad2antrl**: assumes A1: \( \varphi \rightarrow \psi \)
shows \( \text{ch} \land \varphi \land \vartheta \rightarrow \psi \)
using assms by auto

**lemma MMI_3adant3r**: assumes A1: \( \varphi \land \psi \land \text{ch} \rightarrow \vartheta \)
shows \( \varphi \land \psi \land \text{ch} \land \tau \rightarrow \vartheta \)
using assms by auto

**lemma MMI_3adant1l**: assumes A1: \( \varphi \land \psi \land \text{ch} \rightarrow \vartheta \)
shows \( (\tau \land \varphi) \land \psi \land \text{ch} \rightarrow \vartheta \)
using assms by auto

**lemma MMI_3adant2r**: assumes A1: \( \varphi \land \psi \land \text{ch} \rightarrow \vartheta \)
shows \( \varphi \land (\psi \land \tau) \land \text{ch} \rightarrow \vartheta \)
using assms by auto

**lemma MMI_3bitr4rd**: assumes A1: \( \varphi \rightarrow \)
\( \psi \leftrightarrow \text{ch} \text{ and} \)
A2: \( \varphi \rightarrow \)
\( \vartheta \leftrightarrow \psi \text{ and} \)
A3: \( \varphi \rightarrow \)
\( \tau \leftrightarrow \text{ch} \)

shows \( \varphi \rightarrow \)

\( \tau \leftrightarrow \phi \)

using assms by auto

**lemma MMI_3anrev:**

shows \( \varphi \land \psi \land \text{ch} \leftrightarrow \text{ch} \land \psi \land \varphi \)

by auto

**lemma MMI_eqtr4:** assumes A1: A = B and

A2: C = B

shows A = C

using assms by auto

**lemma MMI_anidm:**

shows \( \varphi \land \varphi \leftrightarrow \varphi \)

by auto

**lemma MMI_bi2anan9r:** assumes A1: \( \varphi \rightarrow \psi \leftrightarrow \text{ch} \) and

A2: \( \phi \rightarrow \)

\( \tau \leftrightarrow \eta \)

shows \( \phi \land \varphi \rightarrow \)

\( \psi \land \tau \leftrightarrow \text{ch} \land \eta \)

using assms by auto

**lemma MMI_3imtr3g:** assumes A1: \( \varphi \rightarrow \psi \rightarrow \text{ch} \) and

A2: \( \psi \leftrightarrow \phi \leftrightarrow \)

A3: ch \leftrightarrow \tau

shows \( \varphi \rightarrow \)

\( \phi \rightarrow \tau \)

using assms by auto

**lemma MMI_a3d:** assumes A1: \( \varphi \rightarrow \)

\( \neg \psi \rightarrow \neg \text{ch} \)

shows \( \varphi \rightarrow \text{ch} \rightarrow \psi \)

using assms by auto

**lemma MMI_sylan9bbr:** assumes A1: \( \varphi \rightarrow \psi \leftrightarrow \text{ch} \) and

A2: \( \psi \leftrightarrow \phi \rightarrow \)

\( \text{ch} \leftrightarrow \tau \)

shows \( \phi \land \varphi \rightarrow \)

\( \psi \leftrightarrow \tau \)

using assms by auto

**lemma MMI_sylan9bb:** assumes A1: \( \varphi \rightarrow \)

\( \psi \leftrightarrow \text{ch} \) and

A2: \( \phi \rightarrow \)

1423
ch \leftrightarrow \tau
shows \varphi \land \vartheta \rightarrow 
\psi \leftrightarrow \tau
using assms by auto

lemma MMI_3bitr3g: assumes A1: \varphi \rightarrow
\psi \leftrightarrow ch and
A2: \psi \leftrightarrow \vartheta and
A3: ch \leftrightarrow \tau
shows \varphi \rightarrow 
\vartheta \leftrightarrow \tau
using assms by auto

lemma MMI_pm5_21:
shows \neg \varphi \land \neg \psi \rightarrow 
\varphi \leftrightarrow \psi
by auto

lemma MMI_an6:
shows (\varphi \land \psi \land ch) \land \vartheta \land \tau \land \eta \leftrightarrow 
(\varphi \land \vartheta) \land (\psi \land \tau) \land ch \land \eta
by auto

lemma MMI_syl3anl1: assumes A1: (\varphi \land \psi \land ch) \land \vartheta \rightarrow \tau and
A2: \eta \rightarrow \varphi
shows (\eta \land \psi \land ch) \land \vartheta \rightarrow \tau
using assms by auto

lemma MMI_imp4a: assumes A1: \varphi \rightarrow
\psi \rightarrow 
ch \rightarrow 
\vartheta \rightarrow \tau
shows \varphi \rightarrow 
\psi \rightarrow 
ch \land \vartheta \rightarrow \tau
using assms by auto

lemma (in MMIasr0) MMI_breqan12rd: assumes A1: \varphi \rightarrow A = B and
A2: \psi \rightarrow C = D
shows
\psi \land \varphi \rightarrow A \lessdot C \leftrightarrow B \lessdot D
\psi \land \varphi \rightarrow A \leq C \leftrightarrow B \leq D
using assms by auto

lemma (in MMIasr0) MMI_3brtr4d: assumes A1: \varphi \rightarrow A < B and
A2: \( \varphi \rightarrow C = A \) and
A3: \( \varphi \rightarrow D = B \)
shows \( \varphi \rightarrow C < D \)
using asms by auto

lemma MMI_adantrrr: assumes A1: \( \varphi \land \psi \land ch \rightarrow \vartheta \)
shows \( \varphi \land \psi \land ch \land \tau \rightarrow \vartheta \)
using asms by auto

lemma MMI_adantrlr: assumes A1: \( \varphi \land \psi \land ch \rightarrow \vartheta \)
shows \( \varphi \land (\psi \land \tau) \land ch \rightarrow \vartheta \)
using asms by auto

lemma MMI_indistani: assumes A1: \( \varphi \rightarrow \psi \rightarrow ch \)
shows \( \varphi \land \psi \rightarrow \varphi \land ch \)
using asms by auto

lemma MMI_anabss3: assumes A1: \( (\varphi \land \psi) \land ch \rightarrow \vartheta \)
shows \( \varphi \land \psi \rightarrow \varphi \land ch \)
using asms by auto

lemma MMI_mp3anl2: assumes A1: \( \psi \) and
A2: \( (\varphi \land \psi \land ch) \land \vartheta \rightarrow \tau \)
shows \( (\varphi \land ch) \land \vartheta \rightarrow \tau \)
using asms by auto

lemma MMI_mpanl2: assumes A1: \( \psi \) and
A2: \( (\varphi \land \psi) \land ch \rightarrow \vartheta \)
shows \( \varphi \land ch \rightarrow \vartheta \)
using asms by auto

lemma MMI_mpancom: assumes A1: \( \psi \rightarrow \varphi \) and
A2: \( \varphi \land \psi \rightarrow ch \)
shows \( \psi \rightarrow ch \)
using asms by auto

lemma MMI_or12:
shows \( \varphi \lor \psi \lor ch \leftrightarrow \psi \lor \varphi \lor ch \)
by auto

lemma MMI_rcla4ev: assumes A1: \( \forall x. x = A \rightarrow \varphi(x) \leftrightarrow \psi \)
shows \( A \in B \land \psi \rightarrow (\exists x \in B. \varphi(x)) \)
using asms by auto

lemma MMI_jctir: assumes A1: \( \varphi \rightarrow \psi \) and
A2: \( ch \)
shows \( \varphi \rightarrow \psi \land ch \)
using assms by auto

lemma MMI_ifffalse:  
  shows ¬ϕ −→ if(ϕ, A, B) = B  
  by auto

lemma MMI_iftrue:  
  shows ϕ −→ if(ϕ, A, B) = A  
  by auto

lemma MMI_pm2_61d2: assumes A1: ϕ −→ ¬ψ −→ ch and  
  A2: ψ −→ ch  
  shows ϕ −→ ch  
  using assms by auto

lemma MMI_pm2_61dan: assumes A1: ϕ ∧ ψ −→ ch and  
  A2: ϕ ∧ ¬ψ −→ ch  
  shows ϕ −→ ch  
  using assms by auto

lemma MMI_orcanai: assumes A1: ϕ −→ ψ ∨ ch  
  shows ϕ ∧ ¬ψ −→ ch  
  using assms by auto

lemma MMI_ifcl:  
  shows A ∈ C ∧ B ∈ C −→ if(ϕ, A, B) ∈ C  
  by auto

lemma MMI_imim2i: assumes A1: ϕ −→ ψ  
  shows (ch −→ ϕ) −→ ch −→ ψ  
  using assms by auto

lemma MMI_com13: assumes A1: ϕ −→  
  ψ −→ ch −→ ϑ  
  shows ch −→  
  ϕ −→ ϑ  
  using assms by auto

lemma MMI_rcla4v: assumes A1: ∀x. x = A −→ ϕ(x) ↔ ψ  
  shows A ∈ B −→ (∀x ∈ B. ϕ(x)) −→ ψ  
  using assms by auto

lemma MMI_syl5d: assumes A1: ϕ −→  
  ψ −→ ch −→ ϑ and  
  A2: ϕ −→ τ −→ ch  
  shows ϕ −→  
  ψ −→ 1426
\( \tau \rightarrow \emptyset \)
using assms by auto

lemma MMI_eqcoms: assumes A1: \( A = B \rightarrow \varphi \)
shows \( B = A \rightarrow \varphi \)
using assms by auto

lemma MMI_rgen: assumes A1: \( \forall x. x \in A \rightarrow \varphi(x) \)
shows \( \forall x \in A . \varphi(x) \)
using assms by auto

lemma (in MMIar0) MMI_reex:
shows \( R = R \)
by auto

lemma MMI_sstri: assumes A1: \( A \subseteq B \) and
\( A2: B \subseteq C \)
shows \( A \subseteq C \)
using assms by auto

lemma MMI_ssexi: assumes A1: \( B = B \) and
\( A2: A \subseteq B \)
shows \( A = A \)
using assms by auto

end

96 Complex numbers in Metamatah - introduction

theory MMI_Complex_ZF imports MMI_logic_and_sets

begin

This theory contains theorems (with proofs) about complex numbers im-
ported from the Metamath’s set.mm database. The original Metamath
proofs were mostly written by Norman Megill, see the Metamath Proof
Explorer pages for full attribution. This theory contains about 200 theorems
from ”recnt” to ”div11t”.

lemma (in MMIar0) MMI_recnt:
shows \( A \in R \rightarrow A \in C \)
proof -
  have S1: \( R \subseteq C \) by (rule MMI_axresscn)
  from S1 show \( A \in R \rightarrow A \in C \) by (rule MMI_sseli)
qed
lemma (in MMIsar0) MMI_recn: assumes A1: A ∈ R
shows A ∈ C
proof -
  have S1: R ⊆ C by (rule MMI_axresscn)
  from A1 have S2: A ∈ R.
  from S1 S2 show A ∈ C by (rule MMI_sselii)
qed

lemma (in MMIsar0) MMI_recnd: assumes A1: ϕ −→ A ∈ R
shows ϕ −→ A ∈ C
proof -
  from A1 have S1: ϕ −→ A ∈ R.
  have S2: A ∈ R −→ A ∈ C by (rule MMI_recnt)
  from S1 S2 show ϕ −→ A ∈ C by (rule MMI_syl)
qed

lemma (in MMIsar0) MMI_elimne0:
shows if ( A ≠ 0 , A , 1 ) ≠ 0
proof -
  have S1: A = if ( A ≠ 0 , A , 1 ) −→
          ( A ≠ 0 −→ if ( A ≠ 0 , A , 1 ) ≠ 0 ) by (rule MMI_neeq1)
  have S2: 1 = if ( A ≠ 0 , A , 1 ) −→
          ( 1 ≠ 0 −→ if ( A ≠ 0 , A , 1 ) ≠ 0 ) by (rule MMI_neeq1)
  have S3: 1 ≠ 0 by (rule MMI_ax1ne0)
  from S1 S2 S3 show if ( A ≠ 0 , A , 1 ) ≠ 0 by (rule MMI_elimhyp)
qed

lemma (in MMIsar0) MMI_addex:
shows + isASet
proof -
  have S1: C isASet by (rule MMI_axcnex)
  have S2: C isASet by (rule MMI_axcnex)
  from S1 S2 have S3: ( C × C ) isASet by (rule MMI_xpex)
  have S4: + : ( C × C ) −→ C by (rule MMI_axaddopr)
  have S5: ( C × C ) isASet −→
          ( + : ( C × C ) −→ C −→ + isASet ) by (rule MMI_fex)
  from S3 S4 S5 show + isASet by (rule MMI_mp2)
qed

lemma (in MMIsar0) MMI_mulex:
shows · isASet
proof -
  have S1: C isASet by (rule MMI_axcnex)
  have S2: C isASet by (rule MMI_axcnex)
  from S1 S2 have S3: ( C × C ) isASet by (rule MMI_xpex)
  have S4: · : ( C × C ) −→ C by (rule MMI_axmulopr)
  have S5: ( C × C ) isASet −→
          ( · : ( C × C ) −→ C −→ · isASet ) by (rule MMI_fex)
  from S3 S4 S5 show · isASet by (rule MMI_mp2)
lemma (in MMIsar0) MMI_adddirt:
  shows ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
  ( ( A + B ) · C ) = ( ( A · C ) + ( B · C ) )
proof -
  have S1: ( C ∈ C ∧ A ∈ C ∧ B ∈ C ) →
  ( C · ( A + B ) ) = ( ( C · A ) + ( C · B ) )
  by (rule MMI_axdistr)
  from S1 have S2: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
  ( C · ( A + B ) ) = ( ( C · A ) + ( C · B ) ) by (rule MMI_3coml)
  have S3: ( ( A + B ) ∈ C ∧ C ∈ C ) →
  ( ( A + B ) · C ) = ( C · ( A + B ) ) by (rule MMI_axmulcom)
  have S4: ( A ∈ C ∧ B ∈ C ) → ( A + B ) ∈ C by (rule MMI_axaddcl)
  from S3 S4 have S5: ( A ∈ C ∧ B ∈ C ) →
  ( ( A + B ) · C ) = ( C · ( A + B ) ) by (rule MMI_sylan)
  have S6: ( B ∈ C ∧ C ∈ C ) → ( B · C ) = ( C · B )
  by (rule MMI_axmulcom)
  from S5 have S7: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
  ( ( A + B ) · C ) = ( C · ( A + B ) ) by (rule MMI_3impa)
  have S8: ( A ∈ C ∧ C ∈ C ) → ( A · C ) = ( C · A )
  by (rule MMI_axmulcom)
  have S9: ( B ∈ C ∧ C ∈ C ) → ( B · C ) = ( C · B )
  by (rule MMI_3adant2)
  from S8 S9 have S10: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
  ( ( A + B ) · C ) = ( ( A · C ) + ( B · C ) )
  by (rule MMI_opreq12d)
  from S2 S6 S11 show ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
  ( ( A + B ) · C ) = ( ( A · C ) + ( B · C ) )
  by (rule MMI_3eqtr4d)
qed

lemma (in MMIsar0) MMI_addcl: assumes A1: A ∈ C and
  A2: B ∈ C
  shows ( A + B ) ∈ C
proof -
  from A1 have S1: A ∈ C.
  from A2 have S2: B ∈ C.
  have S3: ( A ∈ C ∧ B ∈ C ) → ( A + B ) ∈ C by (rule MMI_axaddcl)
  from S1 S2 S3 show ( A + B ) ∈ C by (rule MMI_mp2an)
qed

lemma (in MMIsar0) MMI_mulcl: assumes A1: A ∈ C and
  A2: B ∈ C
  shows ( A · B ) ∈ C
proof -
  from A1 have S1: A ∈ C.
  from A2 have S2: B ∈ C.
  have S3: ( A ∈ C ∧ B ∈ C ) → ( A · B ) ∈ C by (rule MMI_axmulcl)
  from S1 S2 S3 show ( A · B ) ∈ C by (rule MMI_mp2an)
qed

lemma (in MMIIsar0) MMI_addcom: assumes A1: A ∈ C and
  A2: B ∈ C
  shows ( A + B ) = ( B + A )
proof -
  from A1 have S1: A ∈ C.
  from A2 have S2: B ∈ C.
  have S3: ( A ∈ C ∧ B ∈ C ) → ( A + B ) = ( B + A )
    by (rule MMI_axaddcom)
  from S1 S2 S3 show ( A + B ) = ( B + A ) by (rule MMI_mp2an)
qed

lemma (in MMIIsar0) MMI_mulcom: assumes A1: A ∈ C and
  A2: B ∈ C
  shows ( A · B ) = ( B · A )
proof -
  from A1 have S1: A ∈ C.
  from A2 have S2: B ∈ C.
  have S3: ( A ∈ C ∧ B ∈ C ) → ( A · B ) = ( B · A )
    by (rule MMI_axmulcom)
  from S1 S2 S3 show ( A · B ) = ( B · A ) by (rule MMI_mp2an)
qed

lemma (in MMIIsar0) MMI_addass: assumes A1: A ∈ C and
  A2: B ∈ C and
  A3: C ∈ C
  shows ( ( A + B ) + C ) = ( A + ( B + C ) )
proof -
  from A1 have S1: A ∈ C.
  from A2 have S2: B ∈ C.
  from A3 have S3: C ∈ C.
  have S4: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) → ( ( A + B ) + C ) =
    ( A + ( B + C ) ) by (rule MMI_axaddass)
  from S1 S2 S3 S4 show ( ( A + B ) + C ) =
    ( A + ( B + C ) ) by (rule MMI_mp3an)
qed

lemma (in MMIIsar0) MMI_mulass: assumes A1: A ∈ C and
  A2: B ∈ C and
  A3: C ∈ C
  shows ( ( A · B ) · C ) = ( A · ( B · C ) )
proof -
  from A1 have S1: A ∈ C.
  1430
from A2 have S2: B ∈ C.
from A3 have S3: C ∈ C.
have S4: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) −→ ( ( A · B ) · C ) =
( A · ( B · C ) ) by (rule MMI_axmulass)
from S1 S2 S3 S4 show ( ( A · B ) · C ) = ( A · ( B · C ) )
by (rule MMI_mp3an)
qed

lemma (in MMIar0) MMI_adddi: assumes A1: A ∈ C and
A2: B ∈ C and
A3: C ∈ C
shows ( A · ( B + C ) ) = ( ( A · B ) + ( A · C ) )
proof -
from A1 have S1: A ∈ C.
from A2 have S2: B ∈ C.
from A3 have S3: C ∈ C.
have S4: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) −→ ( ( A · B ) + ( A · C ) ) =
( ( A · B ) + ( A · C ) ) by (rule MMI_axdistr)
from S1 S2 S3 S4 show ( A · ( B + C ) ) =
( ( A · B ) + ( A · C ) ) by (rule MMI_mp3an)
qed

lemma (in MMIar0) MMI_adddir: assumes A1: A ∈ C and
A2: B ∈ C and
A3: C ∈ C
shows ( ( A + B ) · C ) = ( ( A · C ) + ( B · C ) )
proof -
from A1 have S1: A ∈ C.
from A2 have S2: B ∈ C.
from A3 have S3: C ∈ C.
have S4: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) −→ ( ( A + B ) · C ) =
( ( A · C ) + ( B · C ) ) by (rule MMI_adddi)
from S1 S2 S3 S4 show ( ( A + B ) · C ) =
( ( A · C ) + ( B · C ) ) by (rule MMI_mp3an)
qed

lemma (in MMIar0) MMI_1cn: shows 1 ∈ C
proof -
have S1: 1 ∈ R by (rule MMI_axire)
from S1 show 1 ∈ C by (rule MMI_recn)
qed

lemma (in MMIar0) MMI_0cn: shows 0 ∈ C
proof -
have S1: ( i · i ) + 1 ) = 0 by (rule MMI_axi2m1)
have S2: i ∈ C by (rule MMI_axicn)
have S3: i ∈ C by (rule MMI_axicn)
from \( S2 \) \( S3 \) have \( S4: (i \cdot i) \in C \) by (rule MMI_mulcl)
have \( S5: 1 \in C \) by (rule MMI_1cn)
from \( S4 \) \( S5 \) have \( S6: (i \cdot i + 1) \in C \) by (rule MMI_addcl)
from \( S1 \) \( S6 \) show \( 0 \in C \) by (rule MMI_eqeltrr)
qed

**lemma** (in MMIsar0) MMI_addid1: assumes \( A1: A \in C \)
shows \( (A + 0) = A \)
proof -
from \( A1 \) have \( S1: A \in C. \)
have \( S2: A \in C \rightarrow (A + 0) = A \) by (rule MMI_ax0id)
from \( S1 \) \( S2 \) show \( (A + 0) = A \) by (rule MMI_ax_mp)
qed

**lemma** (in MMIsar0) MMI_addid2: assumes \( A1: A \in C \)
shows \( (0 + A) = A \)
proof -
have \( S1: 0 \in C \) by (rule MMI_0cn)
from \( A1 \) have \( S2: A \in C. \)
from \( S1 \) \( S2 \) have \( S3: (0 + A) = (A + 0) \) by (rule MMI_addcom)
from \( A1 \) have \( S4: A \in C. \)
from \( S4 \) have \( S5: (A + 0) = A \) by (rule MMI_addid1)
from \( S3 \) \( S5 \) show \( (0 + A) = A \) by (rule MMI_eqtr)
qed

**lemma** (in MMIsar0) MMI_mulid1: assumes \( A1: A \in C \)
shows \( (A \cdot 1) = A \)
proof -
from \( A1 \) have \( S1: A \in C. \)
have \( S2: A \in C \rightarrow (A \cdot 1) = A \) by (rule MMI_ax1id)
from \( S1 \) \( S2 \) show \( (A \cdot 1) = A \) by (rule MMI_ax_mp)
qed

**lemma** (in MMIsar0) MMI_mulid2: assumes \( A1: A \in C \)
shows \( (1 \cdot A) = A \)
proof -
have \( S1: 1 \in C \) by (rule MMI_1cn)
from \( A1 \) have \( S2: A \in C. \)
from \( S1 \) \( S2 \) have \( S3: (1 \cdot A) = (A \cdot 1) \) by (rule MMI_mulcom)
from \( A1 \) have \( S4: A \in C. \)
from \( S4 \) have \( S5: (A \cdot 1) = A \) by (rule MMI_mulid1)
from \( S3 \) \( S5 \) show \( (1 \cdot A) = A \) by (rule MMI_eqtr)
qed

**lemma** (in MMIsar0) MMI_negex: assumes \( A1: A \in C \)
shows \( \exists x \in C. (A + x) = 0 \)

1432
proof -
from A1 have S1: A ∈ C.
have S2: A ∈ C → (∃ x ∈ C . (A + x) = 0) by (rule MMI_axnegex)
from S1 S2 show ∃ x ∈ C . (A + x) = 0 by (rule MMI_ax_mp)
qed

lemma (in MMIsar0) MMI_reex: assumes A1: A ∈ C and
A2: A ≠ 0
shows ∃ x ∈ C . (A · x) = 1
proof -
from A1 have S1: A ∈ C.
from A2 have S2: A ≠ 0.
have S3: (A ∈ C ∧ A ≠ 0) → (∃ x ∈ C . (A · x) = 1) by (rule MMI_axreex)
from S1 S2 S3 show ∃ x ∈ C . (A · x) = 1 by (rule MMI_mp2an)
qed

lemma (in MMIsar0) MMI_readdcl: assumes A1: A ∈ R and
A2: B ∈ R
shows (A + B) ∈ R
proof -
from A1 have S1: A ∈ R.
from A2 have S2: B ∈ R.
have S3: (A ∈ R ∧ B ∈ R) → (A + B) ∈ R by (rule MMI_axaddrcl)
from S1 S2 S3 show (A + B) ∈ R by (rule MMI_mp2an)
qed

lemma (in MMIsar0) MMI_remulcl: assumes A1: A ∈ R and
A2: B ∈ R
shows (A · B) ∈ R
proof -
from A1 have S1: A ∈ R.
from A2 have S2: B ∈ R.
have S3: (A ∈ R ∧ B ∈ R) → (A · B) ∈ R by (rule MMI_axmulrcl)
from S1 S2 S3 show (A · B) ∈ R by (rule MMI_mp2an)
qed

lemma (in MMIsar0) MMI_addcan: assumes A1: A ∈ C and
A2: B ∈ C and
A3: C ∈ C
shows (A + B) = (A + C) ↔ B = C
proof -
from A1 have S1: A ∈ C.
from S1 have S2: ∃ x ∈ C . (A + x) = 0 by (rule MMI_negex)
from A1 have S3: A ∈ C.
from A2 have S4: B ∈ C.
{ fix x
  have S5: (x ∈ C ∧ A ∈ C ∧ B ∈ C) → ((x + A) + B) = (x + (A + B)) by (rule MMI_axaddass)
1433
from S4 S5 have S6: \(( x \in C \land A \in C ) \longrightarrow ( ( x + A ) + B ) = ( x + ( A + B ) )\) by (rule MMI_mp3an3)

from A3 have S7: \(C \in C\).

have S8: \(( x \in C \land A \in C \land C \in C ) \longrightarrow ( ( x + A ) + C ) = ( x + ( A + C ) )\) by (rule MMI_axaddass)

from S7 S8 have S9: \(( x \in C \land A \in C ) \longrightarrow ( ( x + A ) + C ) = ( x + ( A + C ) )\) by (rule MMI_mp3an3)

from S6 S9 have S10: \(( x \in C \land A \in C ) \longrightarrow ( ( x + A ) + C ) = ( x + ( A + C ) )\) by (rule MMI_mp3an3)

from S3 S10 have S11: \(( x \in C \land A \in C ) \longrightarrow ( ( x + A ) + B ) = ( ( x + A ) + C ) \longrightarrow ( x + ( A + B ) ) = ( x + ( A + C ) )\) by (rule MMI_eqeq12d)

from S11 S12 have S13: \(( x \in C \land ( A + x ) = 0 ) \longrightarrow ( ( A + B ) = ( A + C ) \longrightarrow ( ( ( x + A ) + B ) = ( ( x + A ) + C ) \longrightarrow ( x + ( A + B ) ) = ( x + ( A + C ) )\) by (rule MMI_syl5bin)

from S13 have S14: \(( x \in C \land ( A + x ) = 0 ) \longrightarrow ( ( A + B ) = ( A + C ) \longrightarrow ( ( ( x + A ) + B ) = ( ( x + A ) + C ) \longrightarrow ( x + ( A + B ) ) = ( x + ( A + C ) )\) by (rule MMI_syl6eq)

from S14 have S15: \(( A + x ) = 0 \longrightarrow ( ( ( x + A ) + B ) = ( ( x + A ) + C ) \longrightarrow ( x + ( A + B ) ) = ( x + ( A + C ) )\) by (rule MMI_addid2)

from S15 S16 have S17: \(( A + x ) = ( x + A )\) by (rule MMI_axaddcom)

from S17 have S18: \(( x + A ) = 0 \longrightarrow ( ( x + A ) + B ) = ( ( x + A ) + C ) \longrightarrow ( x + ( A + B ) ) = ( x + ( A + C ) )\) by (rule MMI_opreq1)

from S18 S19 have S20: \(( A + x ) = 0 \longrightarrow ( ( ( x + A ) + B ) = ( ( x + A ) + C ) \longrightarrow ( x + ( A + B ) ) = ( x + ( A + C ) )\) by (rule MMI_eqeq12d)

from S20 have S21: \(( x + A ) + C ) = C\) by (rule MMI_mp3an3)

from S21 have S22: \(( x + A ) = 0 \longrightarrow ( ( x + A ) + B ) = B\ by (rule MMI_axaddcom)

from S22 S23 have S24: \(( x + A ) = 0 \longrightarrow ( ( x + A ) + C ) = C\) by (rule MMI_mp3an3)

from S23 S24 have S25: \(( x + A ) = 0 \longrightarrow ( ( x + A ) + C ) = C\) by (rule MMI_mp3an3)

from S25 have S26: \(( x + A ) = 0 \longrightarrow ( ( x + A ) + C ) = C\) by (rule MMI_mp3an3)

from S26 S27 have S28: \(( x + A ) + B ) = ( ( x + A ) + C ) \longrightarrow ( x + ( A + B ) ) = ( x + ( A + C ) )\) by (rule MMI_syl5bin)

from S28 S29 have S30: \(( x + A ) + C ) = C\) by (rule MMI_mp3an3)
( ( ( x + A ) + B ) = ( ( x + A ) + C ) ) \iff B = C \)

by (rule MMI_imp)

from S14 S29 have S30: ( x \in \mathcal{C} \land ( A + x ) = 0 ) \implies
( ( A + B ) = ( A + C ) ) by (rule MMI_sylibd)

from S30 have x \in \mathcal{C} \implies ( ( A + x ) = 0 ) \implies
( ( A + B ) = ( A + C ) ) by (rule MMI_ex)

} then have S31: ( \forall x. ( x \in \mathcal{C} \implies ( ( A + x ) = 0 ) \implies
( ( A + B ) = ( A + C ) ) ) ) by auto

from S31 have S32: ( \exists x \in \mathcal{C}. ( A + x ) = 0 ) \implies
( ( A + B ) = ( A + C ) ) by (rule MMI_r19_23aiv)

from S2 S32 have S33: ( A + B ) = ( A + C ) \iff B = C

by (rule MMI_impbi)

qed

lemma (in MMIIsar0) MMI_addcan2: assumes A1: A \in \mathcal{C} and
A2: B \in \mathcal{C} and
A3: C \in \mathcal{C}
shows ( A + C ) = ( B + C ) \iff A = B

proof -

from A1 have S1: A \in \mathcal{C}.
from A3 have S2: C \in \mathcal{C}.

from S1 S2 have S3: ( A + C ) = ( C + A ) by (rule MMI_addcom)

from A2 have S4: B \in \mathcal{C}.
from A3 have S5: C \in \mathcal{C}.

from S4 S5 have S6: ( B + C ) = ( C + B ) by (rule MMI_addcom)

from S3 S6 have S7: ( A + C ) = ( B + C ) \iff
( C + A ) = ( C + B ) by (rule MMI_eqeq12i)

from A3 have S8: C \in \mathcal{C}.
from A1 have S9: A \in \mathcal{C}.
from A2 have S10: B \in \mathcal{C}.

from S8 S9 S10 have S11: ( A + C ) = ( C + B ) \iff A = B

by (rule MMI_addcan)

from S7 S11 show ( A + C ) = ( B + C ) \iff A = B by (rule MMI_bitr)

qed

lemma (in MMIIsar0) MMI_addcant:

shows ( A \in \mathcal{C} \land B \in \mathcal{C} \land C \in \mathcal{C} ) \implies
( ( A + B ) = ( A + C ) ) \iff B = C

proof -

have S1: A = \text{if } ( A \in \mathcal{C} , A , 0 ) \implies ( A + B ) = ( \text{if } ( A \in \mathcal{C} , A , 0 ) + B ) by (rule MMI_opreq1)

have S2: A = \text{if } ( A \in \mathcal{C} , A , 0 ) \implies
( A + C ) = ( \text{if } ( A \in \mathcal{C} , A , 0 ) + C ) by (rule MMI_opreq1)

from S1 S2 have S3: A = \text{if } ( A \in \mathcal{C} , A , 0 ) \implies
( ( A + B ) = ( A + C ) ) \iff ( \text{if } ( A \in \mathcal{C} , A , 0 ) + B ) = ( \text{if } ( A \in \mathcal{C} , A , 0 ) + C )

1435
by (rule MMI_eqeq12d)
from S3 have S4: A = if ( A ∈ C , A , 0 ) →
    ( ( ( A + B ) = ( A + C ) ←→ B = C ) ←→
    ( ( if ( A ∈ C , A , 0 ) + B ) = ( if ( A ∈ C , A , 0 ) + C )
        ←→ B = C ) ) by (rule MMI_bibi1d)
have S5: B = if ( B ∈ C , B , 0 ) →
    ( if ( A ∈ C , A , 0 ) + B ) =
    ( if ( A ∈ C , A , 0 ) + if ( B ∈ C , B , 0 ) ) by (rule MMI_opreq2)
from S5 have S6: B = if ( B ∈ C , B , 0 ) →
    ( ( if ( A ∈ C , A , 0 ) + B ) = ( if ( A ∈ C , A , 0 ) + C )
        ←→ ( if ( A ∈ C , A , 0 ) + if ( B ∈ C , B , 0 ) ) =
        ( if ( A ∈ C , A , 0 ) + C ) ) by (rule MMI_eqeq1d)
have S7: B = if ( B ∈ C , B , 0 ) → ( B = C ←→
    if ( B ∈ C , B , 0 ) = C ) by (rule MMI_eqeq1)
from S6 S7 have S8: B = if ( B ∈ C , B , 0 ) →
    ( ( ( if ( A ∈ C , A , 0 ) + B ) =
        ( if ( A ∈ C , A , 0 ) + C ) ←→ B = C ) ←→
        ( if ( A ∈ C , A , 0 ) + if ( B ∈ C , B , 0 ) ) =
        ( if ( A ∈ C , A , 0 ) + C ) ←→ if ( B ∈ C , B , 0 ) = C )
    by (rule MMI_bibi12d)
have S9: C = if ( C ∈ C , C , 0 ) → ( if ( A ∈ C , A , 0 ) + C
    =
    ( if ( A ∈ C , A , 0 ) + if ( C ∈ C , C , 0 ) )
    by (rule MMI_opreq2)
from S9 have S10: C = if ( C ∈ C , C , 0 ) →
    ( ( if ( A ∈ C , A , 0 ) + if ( B ∈ C , B , 0 ) ) =
        ( if ( A ∈ C , A , 0 ) + C ) ←→
        ( if ( A ∈ C , A , 0 ) + if ( B ∈ C , B , 0 ) ) =
        ( if ( A ∈ C , A , 0 ) + if ( C ∈ C , C , 0 ) )
    by (rule MMI_eqeq2d)
have S11: C = if ( C ∈ C , C , 0 ) → ( if ( B ∈ C , B , 0 ) = C
    ←→
    if ( B ∈ C , B , 0 ) = if ( C ∈ C , C , 0 ) ) by (rule MMI_eqeq2)
from S10 S11 have S12: C = if ( C ∈ C , C , 0 ) →
    ( ( if ( A ∈ C , A , 0 ) + if ( B ∈ C , B , 0 ) ) =
        ( if ( A ∈ C , A , 0 ) + C ) ←→ if ( B ∈ C , B , 0 ) = C ) ←→
        ( ( if ( A ∈ C , A , 0 ) + if ( B ∈ C , B , 0 ) ) =
            ( if ( A ∈ C , A , 0 ) + if ( C ∈ C , C , 0 ) ) ←→
            if ( B ∈ C , B , 0 ) = if ( C ∈ C , C , 0 ) ) by (rule MMI_bibi12d)
have S13: 0 ∈ C by (rule MMI_0cn)
from S13 have S14: if ( A ∈ C , A , 0 ) ∈ C by (rule MMI_elimel)
have S15: 0 ∈ C by (rule MMI_0cn)
from S15 have S16: if ( B ∈ C , B , 0 ) ∈ C by (rule MMI_elimel)
have S17: 0 ∈ C by (rule MMI_0cn)
from S17 have S18: if ( C ∈ C , C , 0 ) ∈ C by (rule MMI_elimel)
from S14 S16 S18 have S19:
( if ( A ∈ C , A , 0 ) + if ( B ∈ C , B , 0 ) ) =
( if ( A ∈ C , A , 0 ) + if ( C ∈ C , C , 0 ) ) ←→

1436
if \( (B \in f, B, 0) = (C \in C, C, 0) \) by (rule MMI_addcan)

from S4 S8 S12 S19 show \( (A \in C \land B \in C \land C \in C) \longrightarrow ((A + B) = (A + C) \iff B = C) \) by (rule MMI_dedth3h)

qed

lemma (in MMIar0) MMI_addcan2t:
shows \( (A \in f \land B \in f \land C \in f) \longrightarrow ((A + C) = (B + C) \iff A = B) \)
proof -
  have S1: \( (C \in C \land A \in C) \longrightarrow (C + A) = (A + C) \)
    by (rule MMI_axaddcom)
  from S1 have S2: \( (C \in C \land A \in C \land B \in C) \longrightarrow (C + A) = (A + C) \) by (rule MMI_3adant3)
  have S3: \( (C \in C \land B \in C) \longrightarrow (C + B) = (B + C) \)
    by (rule MMI_axaddcom)
  from S3 have S4: \( (C \in C \land A \in C \land B \in C) \longrightarrow (C + B) = (B + C) \) by (rule MMI_3adant2)
  from S2 S4 have S5: \( (C \in C \land A \in C \land B \in C) \longrightarrow ((C + A) = (C + B) \iff (A + C) = (B + C)) \)
    by (rule MMI_opreq1d)
  have S6: \( (C \in C \land A \in C \land B \in C) \longrightarrow ((C + A) = (C + B) \iff A = B) \) by (rule MMI_addcan)
  from S5 S6 have S7: \( (C \in C \land A \in C \land B \in C) \longrightarrow ((A + C) = (B + C) \iff A = B) \) by (rule MMI_3com1)
  qed

lemma (in MMIar0) MMI_add12t:
shows \( (A \in f \land B \in f \land C \in f) \longrightarrow (A + (B + C)) = (B + (A + C)) \)
proof -
  have S1: \( (A \in C \land B \in C) \longrightarrow (A + B) = (B + A) \)
    by (rule MMI_axaddcom)
  from S1 have S2: \( (A \in C \land B \in C) \longrightarrow ((A + B) + C) = ((B + A) + C) \) by (rule MMI_opreq1d)
  from S2 have S3: \( (A \in C \land B \in C \land C \in C) \longrightarrow ((A + B) + C) = ((B + A) + C) \)
    by (rule MMI_3adant3)
  have S4: \( (A \in C \land B \in C \land C \in C) \longrightarrow ((A + B) + C) = (A + (B + C)) \) by (rule MMI_axaddass)
  have S5: \( (B \in C \land A \in C \land C \in C) \longrightarrow ((B + A) + C) = (B + (A + C)) \) by (rule MMI_axaddass)
  from S5 have S6: \( (A \in C \land B \in C \land C \in C) \longrightarrow ((B + A) + C) = (B + (A + C)) \) by (rule MMI_3com12)
from S3 S4 S6 show \((A \in C \land B \in C \land C \in C) \longrightarrow (A + (B + C)) = (B + (A + C))\)
by (rule MMI_3eqtr3d)

qed

lemma (in MMIar0) MMI_add23t:
shows \((A \in C \land B \in C \land C \in C) \longrightarrow ((A + B) + C) = ((A + C) + B)\)

proof -
have S1: \((B \in C \land C \in C) \longrightarrow (B + C) = (C + B)\)
by (rule MMI_axaddcom)
from S1 have S2: \((B \in C \land C \in C) \longrightarrow (A + (B + C)) = (A + (C + B))\)
by (rule MMI_opreq2d)
from S2 have S3: \((A \in C \land B \in C \land C \in C) \longrightarrow (A + (B + C)) = (A + (C + B))\)
by (rule MMI_3adant1)
from S3 S4 S6 show \((A \in C \land B \in C \land C \in C) \longrightarrow (A + (B + C)) = (A + (C + B))\)
by (rule MMI_3eqtr4d)

qed

lemma (in MMIar0) MMI_add4t:
shows \(((A \in C \land B \in C) \land (C \in C \land D \in C)) \longrightarrow ((A + B) + (C + D)) = ((A + C) + (B + D))\)

proof -
have S1: \((A \in C \land B \in C \land C \in C) \longrightarrow ((A + B) + C) = ((A + C) + B)\ by (rule MMI_add23t)
from S1 have S2: \((A \in C \land B \in C \land C \in C) \longrightarrow ((A + B) + C) + D) = ((A + C) + B) + D)\ by (rule MMI_opreq1d)
from S2 have S3: \((A \in C \land B \in C) \land C \in C) \longrightarrow ((A + B) + C) + D) = ((A + C) + B) + D)\ by (rule MMI_3expa)
from S3 have S4: \((A \in C \land B \in C) \land (C \in C \land D \in C) \longrightarrow ((A + B) + C) + D) = ((A + C) + B) + D)\ by (rule MMI_adantrr)

have S5: \((A + B) \in C \land C \in C \land D \in C) \longrightarrow ((A + B) + C) + D) = ((A + C) + B) + D)\ by (rule MMI_axaddass)
from S5 have S6: \((A + B) \in C \land (C \in C \land D \in C) \longrightarrow ((A + B) + C) + D) = ((A + C) + B) + D)\ by (rule MMI_3expb)
have S7: \((A \in f \land B \in f) \rightarrow (A + B) \in f\) by (rule MMI_axaddcl)
from S6 S7 have S8: \((A \in f \land B \in f \land C \in f \land D \in f) \rightarrow ((A + B) + C + (B + D)) \in f\) by (rule MMI_axaddass)

have S9: \((A \in f \land B \in f \land C \in f \land D \in f) \rightarrow ((A + B) + C + (B + D)) = ((A + C) + (B + D))\) by (rule MMI_axaddcom)
from S8 have S10: \((A \in f \land B \in f \land C \in f \land D \in f) \rightarrow ((A + B) + C + (B + D)) = ((A + C) + (B + D))\) by (rule MMI_axaddass)

have S11: \((A \in f \land B \in f \land D \in f) \rightarrow ((A + B) + C + (B + D)) = ((A + C) + (B + D))\) by (rule MMI_axaddcom)
from S10 S11 have S12: \((A \in f \land B \in f \land C \in f \land D \in f) \rightarrow ((A + B) + C + (B + D)) = ((A + C) + (B + D))\) by (rule MMI_axaddass)

have S13: \((A \in f \land B \in f \land C \in f \land D \in f) \rightarrow ((A + B) + C + (B + D)) = ((A + C) + (B + D))\) by (rule MMI_axaddcom)
from S4 S8 S13 show \((A \in f \land B \in f \land C \in f \land D \in f) \rightarrow ((A + B) + C + (B + D)) = ((A + C) + (B + D))\) by (rule MMI_axaddass)

qed

lemma (in MMIIsar0) MMI_add42t:
shows \((A \in f \land B \in f \land C \in f \land D \in f) \rightarrow ((A + B) + (C + D)) = ((A + C) + (B + D))\)
proof -
have S1: \((A \in f \land B \in f \land C \in f \land D \in f) \rightarrow ((A + B) + (C + D)) = ((A + C) + (B + D))\) by (rule MMI_add4t)

have S2: \((B \in f \land C \in f \land D \in f) \rightarrow (B + D) = (D + B)\) by (rule MMI_axaddcom)
from S2 have S3: \((A \in f \land B \in f \land C \in f \land D \in f) \rightarrow ((A + B) + (C + D)) = ((A + C) + (B + D))\) by (rule MMI_axaddass)

have S4: \((A \in f \land B \in f \land C \in f \land D \in f) \rightarrow ((A + B) + (C + D)) = ((A + C) + (B + D))\) by (rule MMI_opreq2d)
from S1 S4 show \((A \in f \land B \in f \land C \in f \land D \in f) \rightarrow ((A + B) + (C + D)) = ((A + C) + (B + D))\) by (rule MMI_eqtrd)

qed

lemma (in MMIIsar0) MMI_add12: assumes A1: \(A \in C\) and A2: \(B \in C\) and
A3: C ∈ C
shows ( A + ( B + C ) ) = ( B + ( A + C ) )

proof -
from A1 have S1: A ∈ C.
from A2 have S2: B ∈ C.
from A3 have S3: C ∈ C.
have S4: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) → ( A + ( B + C ) ) =
( B + ( A + C ) ) by (rule MMI_add12t)
from S1 S2 S3 S4 show ( A + ( B + C ) ) =
( B + ( A + C ) ) by (rule MMI_mp3an)
qed

lemma (in MMIser0) MMI_add23: assumes A1: A ∈ C and
A2: B ∈ C and
A3: C ∈ C
shows ( ( A + B ) + C ) = ( ( A + C ) + B )

proof -
from A1 have S1: A ∈ C.
from A2 have S2: B ∈ C.
from A3 have S3: C ∈ C.
have S4: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( ( A + B ) + C ) = ( ( A + C ) + B ) by (rule MMI_add23t)
from S1 S2 S3 S4 show ( ( A + B ) + C ) =
( ( A + C ) + B ) by (rule MMI_mp3an)
qed

lemma (in MMIser0) MMI_add4: assumes A1: A ∈ C and
A2: B ∈ C and
A3: C ∈ C and
A4: D ∈ C
shows ( ( A + B ) + ( C + D ) ) =
( ( A + C ) + ( B + D ) )

proof -
from A1 have S1: A ∈ C.
from A2 have S2: B ∈ C.
from S1 S2 have S3: A ∈ C ∧ B ∈ C by (rule MMI_pm3_2i)
from A3 have S4: C ∈ C.
from A4 have S5: D ∈ C.
from S4 S5 have S6: C ∈ C ∧ D ∈ C by (rule MMI_pm3_2i)
have S7: ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →
( ( A + B ) + ( C + D ) ) =
( ( A + C ) + ( B + D ) ) by (rule MMI_add4t)
from S3 S6 S7 show ( ( A + B ) + ( C + D ) ) =
( ( A + C ) + ( B + D ) ) by (rule MMI_mp2an)
qed

lemma (in MMIser0) MMI_add42: assumes A1: A ∈ C and
A2: B ∈ C and
A3: C ∈ C and

1440
\( A4: \ D \in C \) shows \(( ( A + B ) + ( C + D ) ) = ( ( A + C ) + ( D + B ) ) \)

**proof** -
- from A1 have S1: \( A \in C \).
- from A2 have S2: \( B \in C \).
- from A3 have S3: \( C \in C \).
- from A4 have S4: \( D \in C \).
- from S1 S2 S3 S4 have S5: \(( ( A + B ) + ( C + D ) ) = ( ( A + C ) + ( D + B ) ) \) by (rule MMI_add4)
- from A2 have S6: \( B \in C \).
- from A4 have S7: \( D \in C \).
- from S6 S7 have S8: \(( B + D ) = ( D + B ) \) by (rule MMI_addcom)
- from S5 S8 have S9: \(( ( A + B ) + ( C + D ) ) = ( ( A + C ) + ( D + B ) ) \) by (rule MMI_opreq2i)
- from S5 S9 show \(( ( A + B ) + ( C + D ) ) = ( ( A + C ) + ( D + B ) ) \) by (rule MMI_eqtr)

**qed**

**lemma** (in MMIsar0) **MMI_addid2t**: shows \( A \in C \longrightarrow ( 0 + A ) = A \)

**proof** -
- have S1: \( 0 \in C \) by (rule MMI_0cn)
- have S2: \(( 0 \in C \land A \in C ) \longrightarrow ( 0 + A ) = ( A + 0 ) \) by (rule MMI_addcom)
- from S1 S2 have S3: \( A \in C \longrightarrow ( 0 + A ) = ( A + 0 ) \) by (rule MMI_mpan)
- have S4: \( A \in C \longrightarrow ( A + 0 ) = A \) by (rule MMI_ax0id)
- from S3 S4 show \( A \in C \longrightarrow ( 0 + A ) = A \) by (rule MMI_eqtrd)

**qed**

**lemma** (in MMIsar0) **MMI_peano2cn**: shows \( A \in C \longrightarrow ( A + 1 ) \in C \)

**proof** -
- have S1: \( 1 \in C \) by (rule MMI_1cn)
- have S2: \(( A \in C \land 1 \in C ) \longrightarrow ( A + 1 ) \in C \) by (rule MMI_axaddcl)
- from S1 S2 show \( A \in C \longrightarrow ( A + 1 ) \in C \) by (rule MMI_mpan2)

**qed**

**lemma** (in MMIsar0) **MMI_peano2re**: shows \( A \in R \longrightarrow ( A + 1 ) \in R \)

**proof** -
- have S1: \( 1 \in R \) by (rule MMI_ax1re)
- have S2: \(( A \in R \land 1 \in R ) \longrightarrow ( A + 1 ) \in R \) by (rule MMI_axaddrc1)
- from S1 S2 show \( A \in R \longrightarrow ( A + 1 ) \in R \) by (rule MMI_mpan2)

**qed**
lemma (in MMI_isar0) MMI_negeu: assumes A1: A ∈ C and
  A2: B ∈ C
  shows ∃! x . x ∈ C ∧ ( A + x ) = B
proof -
{ fix x y
  have S1: x = y ⟹ ( A + x ) = ( A + y ) by (rule MMI_opreq2)
  from S1 have x = y ⟹ ( ( A + x ) = B ⟷ ( A + y ) = B )
    by (rule MMI_eqeq1d)
} then have S2: ∀ x y . x = y ⟹ ( ( A + x ) = B ⟷ ( A + y ) = B ) by simp
from S2 have S3: ( ∃! x . x ∈ C ∧ ( A + x ) = B ) ⟷
  ( ( ∃ x ∈ C . ( A + x ) = B ) ∧
    ( ∀ x ∈ C . ∀ y ∈ C . ( ( ( A + x ) = B ∧ ( A + y ) = B ) ⟹
      x = y ) ) ) by (rule MMI_reu4)
from A1 have S4: A ∈ C.
from S4 have S5: ∃ y ∈ C . ( A + y ) = 0 by (rule MMI_negex)
from A2 have S6: B ∈ C.
{ fix y
  have S7: ( y ∈ C ∧ B ∈ C ) ⟹ ( y + B ) ∈ C by (rule MMI_axaddcl)
  from S6 S7 have S8: y ∈ C ⟹ ( y + B ) ∈ C by (rule MMI_mpan2)
  have S9: ( y + B ) ∈ C ⟷ ( ∃ x ∈ C . x = ( y + B ) )
    by (rule MMI_risset)
  from S8 S9 have S10: y ∈ C ⟹ ( ∃ x ∈ C . x = ( y + B ) )
    by (rule MMI_sylib)
  { fix x
    have S11: x = ( y + B ) ⟹ ( A + x ) =
      ( A + ( y + B ) ) by (rule MMI_opreq2)
    from A1 have S12: A ∈ C.
    from A2 have S13: B ∈ C.
    have S14: ( A ∈ C ∧ y ∈ C ∧ B ∈ C ) ⟹
      ( ( A + y ) + B ) = ( A + ( y + B ) )
    by (rule MMI_axaddass)
    from S12 S13 S14 have S15: y ∈ C ⟹
      ( ( A + y ) + B ) =
      ( A + ( y + B ) ) by (rule MMI_mp3an13)
    from S15 have S16: y ∈ C ⟹ ( A + ( y + B ) ) =
      ( ( A + y ) + B ) by (rule MMI_ecomd)
    from S11 S16 have S17: ( y ∈ C ∧ x = ( y + B ) )
      ⟹ ( A + x ) = ( ( A + y ) + B ) by (rule MMI_sylan9eqr)
    have S18: ( A + y ) = 0 ⟹
      ( ( A + y ) + B ) = ( 0 + B ) by (rule MMI_opreq1)
    from A2 have S19: B ∈ C.
    from S19 have S20: ( 0 + B ) = B by (rule MMI_addid2)
    from S18 S20 have S21: ( A + y ) = 0 ⟹
      ( ( A + y ) + B ) = B by (rule MMI_syl6eq)
    from S17 S21 have S22: ( ( A + y ) = 0 ∧ ( y ∈ C ∧ x =
      ( y + B ) ) ) ⟹ ( A + x ) = B by (rule MMI_sylan9eqr)
    from S22 have S23: ( A + y ) = 0 ⟹
      ( y ∈ C ⟷ ( x = ( y + B ) ) ⟹ ( A + x ) = B )
    by (rule MMI_exp32)
}
from S23 have S24: (y ∈ f ∧ (A + y) = 0) −→ (x = (y + B) −→ (A + x) = B) by (rule MMI_impcom)
from S24 have (y ∈ f ∧ (A + y) = 0) −→ (x ∈ C −→ (x = (y + B) −→ (A + x) = B)) by (rule MMI_a1d)

} then have S25: ∀x. (y ∈ f ∧ (A + y) = 0) −→ (x ∈ f −→ (x = (y + B) −→ (A + x) = B)) by auto
from S25 have S26: (y ∈ f ∧ (A + y) = 0) −→ (∀x ∈ f. (x = (y + B) −→ (A + x) = B)) by (rule MMI_r19_21aiv)
from S26 have S27: y ∈ f −→ ((A + y) = 0 −→ (∀x ∈ f. (x = (y + B) −→ (A + x) = B))) by (rule MMI_ex)
have S28: (∀x ∈ f. (x = (y + B) −→ (A + x) = B)) −→ ( (∃x ∈ f. A + x) = B ) by (rule MMI_r19_22)
from S27 S28 have S29: y ∈ f −→ (A + y) = 0 −→ (∃x ∈ f. (A + x) = B) by (rule MMI_ax_mp)

lemma (in MMIsubval) MMI_subval: assumes A ∈ C B ∈ C
shows \( A - B = \bigcup \{ x \in C . B + x = A \} \)
using sub_def by simp

lemma (in MMIIsar0) MMI_df_neg: shows \( - A = 0 - A \)
using cneg_def by simp

lemma (in MMIIsar0) MMI_negeq:
  shows \( A = B \longrightarrow (-A) = (-B) \)
proof -
  have S1: \( A = B \longrightarrow (0 - A) = (0 - B) \) by (rule MMI_opreq2)
  have S2: \( (-A) = (0 - A) \) by (rule MMI_df_neg)
  have S3: \( (-B) = (0 - B) \) by (rule MMI_df_neg)
  from S1 S2 S3 show \( A = B \longrightarrow (-A) = (-B) \) by (rule MMI_3eqtr4g)
qed

lemma (in MMIIsar0) MMI_negeqi: assumes A1: \( A = B \)
  shows \( (- A) = (-B) \)
proof -
  from A1 have S1: \( A = B \).
  have S2: \( A = B \longrightarrow (-A) = (-B) \) by (rule MMI_negeq)
  from S1 S2 show \( (-A) = (-B) \) by (rule MMI_ax_mp)
qed

lemma (in MMIIsar0) MMI_negeqd: assumes A1: \( \varphi \longrightarrow A = B \)
  shows \( \varphi \longrightarrow (-A) = (-B) \)
proof -
  from A1 have S1: \( \varphi \longrightarrow A = B \).
  have S2: \( A = B \longrightarrow (-A) = (-B) \) by (rule MMI_negeq)
  from S1 S2 show \( \varphi \longrightarrow (-A) = (-B) \) by (rule MMI_syl)
qed

lemma (in MMIIsar0) MMI_hbneg: assumes A1: \( y \in A \longrightarrow (\forall x . y \in A) \)
  shows \( y \in ((- A)) \longrightarrow (\forall x . (y \in ((- A))) \) )
using assms by auto

lemma (in MMIIsar0) MMI_minusex:
  shows \( (- A) \) isASet by auto

lemma (in MMIIsar0) MMI_subcl: assumes A1: \( A \in C \) and
A2: \( B \in C \)
shows \((A - B) \in C\)

proof -
  from A1 have S1: \(A \in C\).
  from A2 have S2: \(B \in C\).
  from S1 S2 have S3: \((A - B) = \bigcup \{x \in C . (B + x) = A\}\)
    by (rule MMI_subval)
  from A2 have S4: \(B \in C\).
  from A1 have S5: \(A \in C\).
  from S4 S5 have S6: \(\exists! \ x . x \in C \land (B + x) = A\) by (rule MMI_negeu)
    have S7: \(\bigcup \{x \in C . (B + x) = A\} \in C\) by (rule MMI_reucl)
  from S6 S7 have S8: \(\exists! \ x . x \in C \land (B + x) = A\) by (rule MMI_ax_mp)
    have S9: \((A - B) \in C\) by simp
  qed

lemma (in MMIsar0) MMI_subclt:
  shows \((A \in C \land B \in C) \longrightarrow (A - B) \in C\)

proof -
  have S1: \(A = \text{if} (A \in C, A, 0) \longrightarrow (A - B) = \)
    \((\text{if} (A \in C, A, 0) - B)\) by (rule MMI_opreq1)
  from S1 have S2: \(A = \text{if} (A \in C, A, 0) \longrightarrow ((A - B) \in C) \leftarrow\)
    \((\text{if} (A \in C, A, 0) - B) \in C\) by (rule MMI_equid)
  have S3: \(B = \text{if} (B \in C, B, 0) \longrightarrow (\text{if} (A \in C, A, 0) - B) = \)
    \((\text{if} (A \in C, A, 0) - \text{if} (B \in C, B, 0))\) by (rule MMI_opreq2)
  from S3 have S4: \(B = \text{if} (B \in C, B, 0) \longrightarrow (\text{if} (A \in C, A, 0) - B) = \)
    \((\text{if} (A \in C, A, 0) - \text{if} (B \in C, B, 0)) \in C\) by (rule MMI_equid)
  have S5: \(0 \in C\) by (rule MMI_0cn)
  from S5 have S6: \(A \in C\) by (rule MMI_elim)
  have S7: \(0 \in C\) by (rule MMI_0cn)
  from S7 have S8: \(B \in C\) by (rule MMI_elim)
  from S6 S8 have S9: \(\text{if} (A \in C, A, 0) - \text{if} (B \in C, B, 0) \in C\)
    by (rule MMI_subcl)
  from S2 S4 S9 show \((A \in C \land B \in C) \longrightarrow (A - B) \in C\)
    by (rule MMI_dedth2h)
  qed

lemma (in MMIsar0) MMI_negclt:
  shows \(A \in C \longrightarrow (\neg A) \in C\)

proof -
  have S1: \(0 \in C\) by (rule MMI_0cn)
  have S2: \(0 \in C \land A \in C\) \longrightarrow \((0 - A) \in C\) by (rule MMI_subcl)
  from S1 S2 have S3: \(A \in C \longrightarrow (0 - A) \in C\) by (rule MMI_mpan)
  have S4: \((\neg A) = (0 - A)\) by (rule MMI_df_neg)

  1445
from S3 S4 show A ∈ C → (¬ A) ∈ C by (rule MMI_syl5eqel)

qed

lemma (in MMIIsar0) MMI_negcl: assumes A1: A ∈ C
shows (¬ A) ∈ C
proof -
from A1 have S1: A ∈ C.
have S2: A ∈ C → (¬ A) ∈ C by (rule MMI_negclt)
from S1 S2 show (¬ A) ∈ C by (rule MMI_ax_mp)
qed

lemma (in MMIIsar0) MMI_subadd: assumes A1: A ∈ C and
A2: B ∈ C and
A3: C ∈ C
shows (A - B) = C ↔ (B + C) = A
proof -
from A3 have S1: C ∈ C.
{ fix x
  have S2: x = C → ((A - B) = x) ↔ (A - B) = C
  by (rule MMI_eqeq2)
  have S3: x = C → (B + x) = (B + C) by (rule MMI_opreq2)
  have S4: (x ∈ C → ((B + x) = A) ↔ ((A - B) = C) ↔ (B + C) = A)
  by (rule MMI_negeu)
  from A2 have S5: B ∈ C.
  from A1 have S6: A ∈ C.
  from S6 S7 have S8: ∃ ! x . x ∈ C ∧ (B + x) = A by (rule MMI_negeu)
  { fix x
    have S9: (x ∈ C ∧ (∃ ! x . x ∈ C ∧ (B + x) = A) ) →
      ((B + x) = A) ↔ ∪ {x ∈ C . (B + x) = A } = x
    by (rule MMI_reuuni1)
    from S8 S9 have x ∈ C → ((B + x) = A) ↔
      ∪ {x ∈ C . (B + x) = A } = x ) by (rule MMI_mpan2)
  } then have S10: ∀ x . x ∈ C → ((B + x) = A) ↔
    ∪ {x ∈ C . (B + x) = A } = x ) by blast
  from A1 have S11: A ∈ C.
  from A2 have S12: B ∈ C.
  from S11 S12 have S13: (A - B) = ∪ {x ∈ C . (B + x) = A}
    by (rule MMI_subval)
  from S13 have S14: ∀ x . (A - B) = x ↔
    ∪ {x ∈ C . (B + x) = A } = x by simp
  from S10 S14 have S15: ∀ x . x ∈ C → ((A - B) = x ↔
    (B + x) = A ) by (rule MMI_syl6rbbr)
  from S5 S15 have S16: C ∈ C → ((A - B) = C ↔

  1446
lemma (in MMIar0) MMI_subsub23: assumes A1: A ∈ C and
  A2: B ∈ C and
  A3: C ∈ C
  shows ( A - B ) = C ←→ ( A - C ) = B
proof -
  from A2 have S1: B ∈ C.
  from A3 have S2: C ∈ C.
  from S1 S2 have S3: ( B + C ) = ( C + B ) by (rule MMI_addcom)
  from S3 have S4: ( B + C ) = A ←→ ( C + B ) = A
      by (rule MMI_eqeq1i)
  from A1 have S5: A ∈ C.
  from S5 S6 S7 have S8: ( A - B ) = C ←→ ( B + C ) = A
      by (rule MMI_subadd)
  from S4 S8 S12 show ( A - B ) = C ←→ ( A - C ) = B
      by (rule MMI_3bitr4)
qed

lemma (in MMIar0) MMI_subaddt:
  shows ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) ←→ ( ( A - B ) = C ←→ ( B + C ) = A )
proof -
  have S1: A = if ( A ∈ C , A , 0 ) → ( A - B ) =
      ( if ( A ∈ C , A , 0 ) - B ) by (rule MMI_opreq1)
  from S1 have S2: A = if ( A ∈ C , A , 0 ) → ( ( A - B ) = C ←→ ( B + C ) = A )
  have S3: A = if ( A ∈ C , A , 0 ) → ( ( B + C ) = A ←→
      ( B + C ) = if ( A ∈ C , A , 0 ) ) by (rule MMI_eqeq2)
  from S2 S3 have S4: A = if ( A ∈ C , A , 0 ) →
      ( ( ( A - B ) = C ←→ ( B + C ) = A ) ←→
      ( ( if ( A ∈ C , A , 0 ) - B ) = C ←→ ( B + C ) =
      if ( A ∈ C , A , 0 ) ) ) by (rule MMI_bibi12d)
  have S5: B = if ( B ∈ C , B , 0 ) →
      ( if ( A ∈ C , A , 0 ) - B ) =
( if ( A ∈ C , A , 0 ) - if ( B ∈ C , B , 0 ) ) by (rule MMI_opreq2)

from S5 have S6: B = if ( B ∈ C , B , 0 ) →
( ( ( if ( A ∈ C , A , 0 ) - B ) = C ←→
( if ( A ∈ C , A , 0 ) - if ( B ∈ C , B , 0 ) ) = C )
by (rule MMI_eqeq1d)

have S7: B = if ( B ∈ C , B , 0 ) → ( B + C ) =
( if ( B ∈ C , B , 0 ) + C ) by (rule MMI_opreq1)
from S7 have S8: B = if ( B ∈ C , B , 0 ) →
( ( B + C ) = if ( A ∈ C , A , 0 ) ←→
( if ( B ∈ C , B , 0 ) + C ) = if ( A ∈ C , A , 0 ) )
by (rule MMI_eqeq1d)

from S6 S8 have S9: B = if ( B ∈ C , B , 0 ) →
( ( ( if ( A ∈ C , A , 0 ) - B ) = C ←→
( B + C ) = if ( A ∈ C , A , 0 ) ) ←→
( ( if ( A ∈ C , A , 0 ) - if ( B ∈ C , B , 0 ) ) = C ←→
( if ( B ∈ C , B , 0 ) + C ) = if ( A ∈ C , A , 0 ) )
by (rule MMI_bibi12d)

have S10: C = if ( C ∈ C , C , 0 ) →
( ( if ( A ∈ C , A , 0 ) - if ( B ∈ C , B , 0 ) ) = C ←→
( if ( A ∈ C , A , 0 ) - if ( B ∈ C , B , 0 ) ) =
if ( C ∈ C , C , 0 ) ) by (rule MMI_eqeq2)

have S11: C = if ( C ∈ C , C , 0 ) →
( if ( B ∈ C , B , 0 ) + C ) =
( if ( B ∈ C , B , 0 ) + if ( C ∈ C , C , 0 ) ) by (rule MMI_opreq2)

from S11 have S12: C = if ( C ∈ C , C , 0 ) →
( ( if ( A ∈ C , A , 0 ) - if ( B ∈ C , B , 0 ) ) = C ←→
( if ( B ∈ C , B , 0 ) + if ( C ∈ C , C , 0 ) ) =
if ( A ∈ C , A , 0 ) ) by (rule MMI_eqeq1d)

from S10 S12 have S13: C = if ( C ∈ C , C , 0 ) →
( ( ( if ( A ∈ C , A , 0 ) - if ( B ∈ C , B , 0 ) ) = C ←→
( if ( B ∈ C , B , 0 ) + C ) = if ( A ∈ C , A , 0 ) ) ←→
( ( if ( A ∈ C , A , 0 ) - if ( B ∈ C , B , 0 ) ) =
if ( C ∈ C , C , 0 ) ←→
( if ( B ∈ C , B , 0 ) + if ( C ∈ C , C , 0 ) ) =
if ( A ∈ C , A , 0 ) ) by (rule MMI_bibi12d)

have S14: 0 ∈ C by (rule MMI_0cn)

from S14 have S15: if ( A ∈ C , A , 0 ) ∈ C by (rule MMI_elimel)

have S16: 0 ∈ C by (rule MMI_0cn)

from S16 have S17: if ( B ∈ C , B , 0 ) ∈ C by (rule MMI_elimel)

have S18: 0 ∈ C by (rule MMI_0cn)

from S18 have S19: if ( C ∈ C , C , 0 ) ∈ C by (rule MMI_elimel)

from S15 S17 S19 have S20:
( if ( A ∈ C , A , 0 ) - if ( B ∈ C , B , 0 ) ) =
if ( C ∈ C , C , 0 ) ←→
( if ( B ∈ C , B , 0 ) + if ( C ∈ C , C , 0 ) ) =
if ( A ∈ C , A , 0 ) by (rule MMI_subadd)

from S4 S9 S13 S20 show ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( ( A - B ) = C ←→ ( B + C ) = A ) by (rule MMI_dedth3h)

qed
lemma (in MMIsar0) MMI_pncan3t:
  shows ( A ∈ C ∧ B ∈ C ) → ( A + ( B - A ) ) = B

proof -
  have S1: ( B - A ) = ( B - A ) by (rule MMI_eqid)
  have S2: ( B ∈ C ∧ A ∈ C ∧ ( B - A ) ∈ C ) →
        ( ( B - A ) = ( B - A ) ∈→ ( A + ( B - A ) ) = B )
        by (rule MMI_subaddt)
  have S3: ( A ∈ C ∧ B ∈ C ) → B ∈ C by (rule MMI_pm3_27)
  have S4: ( A ∈ C ∧ B ∈ C ) → A ∈ C by (rule MMI_pm3_26)
  have S5: ( B ∈ C ∧ A ∈ C ) → ( B - A ) ∈ C by (rule MMI_subclt)
  from S5 have S6: ( A ∈ C ∧ B ∈ C ) → ( B - A ) ∈ C
  by (rule MMI_ancoms)
  from S2 S3 S4 S6 have S7: ( A ∈ C ∧ B ∈ C ) →
        ( ( B - A ) = ( B - A ) ←→ ( A + ( B - A ) ) = B )
        by (rule MMI_syl3anc)
  from S1 S7 show ( A ∈ C ∧ B ∈ C ) →
        ( A + ( B - A ) ) = B by (rule MMI_mp2an)
qed

lemma (in MMIsar0) MMI_pncan3: assumes A1: A ∈ C and
  A2: B ∈ C
  shows ( A + ( B - A ) ) = B

proof -
  from A1 have S1: A ∈ C.
  from A2 have S2: B ∈ C.
  have S3: ( A ∈ C ∧ B ∈ C ) → ( A + ( B - A ) ) = B
  by (rule MMI_pncan3t)
  from S1 S2 S3 show ( A + ( B - A ) ) = B by (rule MMI_mp2an)
qed

lemma (in MMIsar0) MMI_negidt:
  shows A ∈ C → ( A + ( - A ) ) = 0

proof -
  have S1: 0 ∈ C by (rule MMI_0cn)
  have S2: ( A ∈ C ∧ 0 ∈ C ) → ( A + ( 0 - A ) ) = 0
  by (rule MMI_pncan3t)
  from S1 S2 have S3: A ∈ C → ( A + ( 0 - A ) ) = 0
  by (rule MMI_mp2an2)
  have S4: ( - A ) = ( 0 - A ) by (rule MMI_df_neg)
  from S4 have S5: ( A + ( - A ) ) = ( A + ( 0 - A ) )
  by (rule MMI_opreq2i)
  from S3 S5 show A ∈ C → ( A + ( - A ) ) = 0 by (rule MMI_syl15eq)
qed

lemma (in MMIsar0) MMI_negid: assumes A1: A ∈ C
  shows ( A + ( - A ) ) = 0

proof -
  from A1 have S1: A ∈ C.
  have S2: A ∈ C → ( A + ( - A ) ) = 0 by (rule MMI_negidt)
from S1 S2 show \((A + (\neg A)) = 0\) by (rule MMI_ax_mp)

qed

lemma (in MMIsar0) MMI_negsub: assumes \(A \in C\) and \(B \in C\)
shows \((A + (\neg B)) = (A - B)\)

proof -
  from A2 have S1: \(B \in C\).
  from A1 have S2: \(A \in C\).
  from A2 have S3: \(B \in C\).
  from S3 have S4: \((-B) \in C\) by (rule MMI_negcl).
  from S2 S4 have S5: \((A + ((-B))) \in C\) by (rule MMI_addcl).
  from S1 S5 have S6: \((B + (A + ((-B)))) = ((A + ((-B))) + B)\) by (rule MMI_addcom).
  from A1 have S7: \(A \in C\).
  from S4 have S8: \((-B) \in C\).
  from A2 have S9: \(B \in C\).
  from S7 S8 S9 have S10: \((A + ((-B))) + B) = (A + ((-B)) + B)\) by (rule MMI_addass).
  from A2 have S12: \(B \in C\).
  from S10 S12 have S13: \((-B) \in C\).
  from A1 have S14: \(B \in C\).
  from S14 have S15: \((B + ((-B))) = 0\) by (rule MMI_negid).
  from S13 S15 have S16: \(((-B)) + B) = 0\) by (rule MMI_eqtr).
  from S6 have S17: \((A + ((-B))) + B) = (A + 0)\) by (rule MMI_opreq2i).
  from A1 have S18: \(A \in C\).
  from S18 have S19: \((A + 0) = A\) by (rule MMI_addid1).
  from S10 S17 S19 have S20: \((A + ((-B))) + B) = A\) by (rule MMI_3eqtr).
  from S6 S20 have S21: \((B + (A + ((-B)))) = A\) by (rule MMI_eqtr).
  from A1 have S22: \(A \in C\).
  from A2 have S23: \(B \in C\).
  from S5 have S24: \((A + ((-B))) \in C\).
  from S22 S23 S24 have S25: \((A - B) = (A + ((-B))) \iff (B + (A + ((-B))) = A\) by (rule MMI_subadd).
  from S21 S25 have S26: \((A - B) = (A + ((-B)))\) by (rule MMI_mpbi).
  from S26 show \((A + ((-B))) = (A - B)\) by (rule MMI_eqcomi).
qed

lemma (in MMIsar0) MMI_negsubt: shows \((A \in C \land B \in C) \rightarrow (A + ((-B))) = (A - B)\)

proof -
  have S1: \(A = \text{if} (A \in C, A, 0) \rightarrow (A + ((-B))) = \text{if} (A \in C, A, 0) + ((-B))\) by (rule MMI_opreq1)
have S2: $A = \text{if} (A \in \mathcal{F}, A, 0) \rightarrow (A - B) =$

(by rule MMI_opreq1)

from S1 S2 have S3: $A = \text{if} (A \in \mathcal{F}, A, 0) \rightarrow$

(by rule MMI_eqeq12d)

have S4: $B = \text{if} (B \in \mathcal{F}, B, 0) \rightarrow$

(by rule MMI_negeq)

from S4 have S5: $B = \text{if} (B \in \mathcal{F}, B, 0) \rightarrow$

(by rule MMI_opreq2d)

have S6: $B = \text{if} (B \in \mathcal{F}, B, 0) \rightarrow$

(by rule MMI_opreq2)

from S5 S6 have S7: $B = \text{if} (B \in \mathcal{F}, B, 0) \rightarrow$

(by rule MMI_negsub)

from S3 S7 S12 show $A \in \mathcal{F} \land B \in \mathcal{F} \land C \in \mathcal{F} \rightarrow (A + (-B)) = (A - B)$

 qed

lemma (in MMI_isar0) MMI_addsubasst:

shows $A \in \mathcal{F} \land B \in \mathcal{F} \land C \in \mathcal{F} \rightarrow ((A + B) - C) = (A + (B - C))$

proof -

have S1: $(A \in \mathcal{F} \land B \in \mathcal{F} \land (-C) \in \mathcal{F}) \rightarrow$

(by rule MMI_axaddass)

have S2: $C \in \mathcal{F} \rightarrow (-C) \in \mathcal{F}$

(by rule MMI_negclt)

from S1 S2 have S3: $(A \in \mathcal{F} \land B \in \mathcal{F} \land C \in \mathcal{F}) \rightarrow$

(by rule MMI_syl3an3)

have S4: $(A + B) \in \mathcal{F} \land C \in \mathcal{F} \rightarrow$

(by rule MMI_negsubt)

have S5: $(A \in \mathcal{F} \land B \in \mathcal{F}) \rightarrow (A + B) \in \mathcal{F}$

(by rule MMI_axaddcl)
from S4 S5 have S6: ((A ∈ C ∧ B ∈ C) ∧ C ∈ C) →
((A + B) + (-C)) = ((A + B) - C)
by (rule MMI_sylan)

from S6 have S7: ((A ∈ C ∧ B ∈ C ∧ C ∈ C) →
((A + B) + (-C)) = ((A + B) - C)
by (rule MMI_3impa)

have S8: (B ∈ C ∧ C ∈ C) → (B + (-C)) = (B - C)
by (rule MMI_negsubt)

from S8 have S9: ((A ∈ C ∧ B ∈ C ∧ C ∈ C) →
(B + (-C)) = (B - C)
by (rule MMI_3adant1)

from S9 have S10: ((A ∈ C ∧ B ∈ C ∧ C ∈ C) →
(A + (B + (-C))) = (A + (B - C))
by (rule MMI_opreq2d)

from S3 S7 S10 show ((A ∈ C ∧ B ∈ C ∧ C ∈ C) →
((A + B) - C) = ((A + B) - C)
by (rule MMI_3eqtr3d)
qed

lemma (in MMIIsar0) MMI_addsubt:
shows ((A ∈ C ∧ B ∈ C ∧ C ∈ C) →
((A + B) - C) =
((A - C) + B))
proof -
have S1: ((A ∈ C ∧ B ∈ C) →
(A + B) = (B + A)
by (rule MMI_axaddcom)

from S1 have S2: ((A ∈ C ∧ B ∈ C) →
((A + B) - C) =
((B + A) - C)
by (rule MMI_opreqid)

from S2 have S3: ((A ∈ C ∧ B ∈ C ∧ C ∈ C) →
((A + B) - C) =
((B + A) - C)
by (rule MMI_3adant3)

have S4: (B ∈ C ∧ A ∈ C ∧ C ∈ C) →
((B + A) - C) =
(B + (A - C))
by (rule MMI_addsubst)

from S4 have S5: ((A ∈ C ∧ B ∈ C ∧ C ∈ C) →
((B + A) - C) =
(B + (A - C))
by (rule MMI_3com12)

have S6: (B ∈ C ∧ (A - C) ∈ C) →
(B + (A - C)) =
((A - C) + B)
by (rule MMI_axaddcom)

from S6 have S7: B ∈ C →
((A - C) ∈ C →
(B + (A - C)) =
((A - C) + B))
by (rule MMI_ex)

have S8: (A ∈ C ∧ C ∈ C) →
(A - C) ∈ C
by (rule MMI_subclt)

from S7 S8 have S9: B ∈ C →
((A ∈ C ∧ C ∈ C) →
(B + (A - C)) =
((A - C) + B))
by (rule MMI_axsy15)

from S9 have S10: B ∈ C →
(A ∈ C →
(C ∈ C →
(B + (A - C)) =
((A - C) + B)))
by (rule MMI_exp3a)

from S10 have S11: A ∈ C →
(B ∈ C →
(C ∈ C →
(B + (A - C)) =
((A - C) + B)))
by (rule MMI_com12)

from S11 have S12: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
(B + (A - C)) =
((A - C) + B)
by (rule MMI_3imp)

from S3 S5 S12 show ((A ∈ C ∧ B ∈ C ∧ C ∈ C) →

1452
\[(A + B) - C = (A - C) + B\] by (rule MMI_3eqtrd)

qed

lemma (in MMIar0) MMI_addsub12t:
shows \((A \in C \land B \in C \land C \in C) \rightarrow (A + (B - C)) = (B + (A - C))\)
proof -
have S1: \((A \in C \land B \in C) \rightarrow (A + B) = (B + A)\)
  by (rule MMI_axaddcom)
from S1 have S2: \((A \in C \land B \in C) \rightarrow ((A + B) - C) = ((B + A) - C)\)
  by (rule MMI_opreq1d)
from S2 have S3: \((A \in C \land B \in C \land C \in C) \rightarrow ((A + B) - C) = ((B + A) - C)\)
  by (rule MMI_3adant3)
  have S4: \((A \in C \land B \in C \land C \in C) \rightarrow ((A + B) - C) = (A + (B - C))\)
    by (rule MMI_addsubasst)
  from S4 have S5: \((A \in C \land B \in C \land C \in C) \rightarrow ((B + A) - C) = (B + (A - C))\)
    by (rule MMI_addsubasst)
  from S5 have S6: \((A \in C \land B \in C \land C \in C) \rightarrow ((B + A) - C) = (B + (A - C))\)
    by (rule MMI_3com12)
    from S3 S4 S6 show \((A \in C \land B \in C \land C \in C) \rightarrow (A + (B - C)) = (B + (A - C))\)
      by (rule MMI_3eqtr3d)
qed

lemma (in MMIar0) MMI_addsubass: assumes \(A1: A \in C\) and
  \(A2: B \in C\) and
  \(A3: C \in C\)
shows \(((A + B) - C) = (A + (B - C))\)
proof -
  from A1 have S1: \(A \in C\).
  from A2 have S2: \(B \in C\).
  from A3 have S3: \(C \in C\).
  have S4: \((A \in C \land B \in C \land C \in C) \rightarrow ((A + B) - C) = (A + (B - C))\)
    by (rule MMI_addsubasst)
  from S4 have S5: \((A \in C \land B \in C \land C \in C) \rightarrow ((B + A) - C) = (B + (A - C))\)
    by (rule MMI_addsubasst)
    from S1 S2 S3 S4 show \(((A + B) - C) = (B + (A - C))\)
      by (rule MMI_3com12)
    from S5 show \(((A + B) - C) = (B + (A - C))\)
      by (rule MMI_mp3an)
qed

lemma (in MMIar0) MMI_addsub: assumes \(A1: A \in C\) and
  \(A2: B \in C\) and
  \(A3: C \in C\)
shows \(((A + B) - C) = (A - C) + B\)
proof -
  from A1 have S1: \(A \in C\).
  from A2 have S2: \(B \in C\).
  from A3 have S3: \(C \in C\).
have S4: \((A \in \mathcal{F} \land B \in \mathcal{F} \land C \in \mathcal{F}) \rightarrow ((A + B) - C) = ((A - C) + B)\) by (rule MMI_addsubt)

from S1 S2 S3 S4 show \(((A + B) - C) = ((A - C) + B)\) by (rule MMI_mp3an)

qed

lemma (in MMIsar0) MMI_2addsubt:
shows \(((A \in \mathcal{F} \land B \in \mathcal{F} \land C \in \mathcal{F}) \land (C \in \mathcal{F} \land D \in \mathcal{F}) \rightarrow ((A + B) + C) = ((A + C) + B)\)

proof -

have S1: \(((A \in \mathcal{F} \land B \in \mathcal{F} \land C \in \mathcal{F}) \rightarrow ((A + B) + C) = ((A + C) + B)\)
  by (rule MMI_add23t)

from S1 have S2: \(((A \in \mathcal{F} \land B \in \mathcal{F}) \land C \in \mathcal{F} \rightarrow ((A + B) + C) = ((A + C) + B)\)
  by (rule MMI_3expa)

from S2 have S3: \(((A \in \mathcal{F} \land B \in \mathcal{F}) \land (C \in \mathcal{F} \land D \in \mathcal{F}) \rightarrow ((A + B) + C) = ((A + C) + B)\)
  by (rule MMI_adantrr)

from S3 have S4: \(((A \in \mathcal{F} \land B \in \mathcal{F}) \land (C \in \mathcal{F} \land D \in \mathcal{F}) \rightarrow ((A + B) + C) = ((A + C) + B)\)
  by (rule MMI_opreq1d)

have S5: \(((A + C) \in \mathcal{F} \land B \in \mathcal{F} \land D \in \mathcal{F}) \rightarrow ((A + C) + B) = ((A + C) - D) + B\)
  by (rule MMI_addsubt)

from S5 have S6: \(((A + C) \in \mathcal{F} \land B \in \mathcal{F} \land D \in \mathcal{F}) \rightarrow ((A + C) + B) = ((A + C) - D) + B\)
  by (rule MMI_3expb)

from S6 S7 have S8: \(((A \in \mathcal{F} \land C \in \mathcal{F}) \land (B \in \mathcal{F} \land D \in \mathcal{F}) \rightarrow ((A + C) + B) = ((A + C) - D) + B\)
  by (rule MMI_axaddcl)

from S4 S8 have S9: \(((A \in \mathcal{F} \land B \in \mathcal{F}) \land (C \in \mathcal{F} \land D \in \mathcal{F}) \rightarrow ((A + B) + C) = ((A + C) - D) + B\)
  by (rule MMI_areqrd)

qed

lemma (in MMIsar0) MMI_negneg: assumes A1: \(A \in \mathcal{C}\)
shows \((-((-A)))\) = \(A\)

proof -

from A1 have S1: \(A \in \mathcal{C}\).
from S1 have S2: \((-A) \in \mathcal{C}\) by (rule MMI_negcl)
from S2 have S3: \((-(-A)) + ((-(-A))) = 0\)
  by (rule MMI_negid)

qed
from S3 have S4: \(( A + ( ( - A ) ) + ( - ( - A ) ) )\) = 
\(( A + 0 )\) by (rule MMI_opreq2i)

from A1 have S5: \(A \in \mathbb{C}\).

from S5 have S6: \(( A + ( - A ) ) = 0\) by (rule MMI_negid)
from S6 have S7: \(( ( A + ( - A ) ) + ( - ( - A ) ) )\) = 
\(( 0 + ( - ( - A ) ) )\) by (rule MMI_opreq1i)

from A1 have S8: \(A \in \mathbb{C}\).

from S2 have S9: \(( - A ) \in \mathbb{C}\).

from S2 have S10: \(( - A ) \in \mathbb{C}\).

from S10 have S11: \(( - ( - A ) ) \in \mathbb{C}\) by (rule MMI_negcl)
from S8 S9 S11 have S12:
\(( A + ( - A ) + ( - ( - A ) ) )\) = 
\(( A + ( ( - A ) + ( - ( - A ) ) ) )\) by (rule MMI_addass)

from S11 have S13: \(( - ( - A ) ) \in \mathbb{C}\).

from S13 have S14: \(( 0 + ( - ( - A ) ) )\) = 
\(( - ( - A ) )\) by (rule MMI_addid2)
from S7 S12 S14 have S15:
\(( A + ( ( - A ) + ( - ( - A ) ) ) )\) = 
\(( - ( - A ) )\) by (rule MMI_3eqtr3)

from A1 have S16: \(A \in \mathbb{C}\).

from S16 have S17: \(( A + 0 ) = A\) by (rule MMI_addid1)
from S4 S15 S17 show \(( - ( - A ) ) = A\) by (rule MMI_3eqtr3)

qed

lemma (in MMIsar0) MMI_subid: assumes A1: \(A \in \mathbb{C}\)
shows \(( A - A ) = 0\)
proof -
from A1 have S1: \(A \in \mathbb{C}\).
from A1 have S2: \(A \in \mathbb{C}\).
from S1 S2 have S3: \(( A + ( - A ) ) = ( A - A )\) by (rule MMI_negsub)
from A1 have S4: \(A \in \mathbb{C}\).
from S4 have S5: \(( A + ( - A ) ) = 0\) by (rule MMI_negid)
from S3 S5 show \(( A - A ) = 0\) by (rule MMI_eqtr3)

qed

lemma (in MMIsar0) MMI_subid1: assumes A1: \(A \in \mathbb{C}\)
shows \(( A - 0 ) = A\)
proof -
from A1 have S1: \(A \in \mathbb{C}\).
from S1 have S2: \(( 0 + A ) = A\) by (rule MMI_addid2)
from A1 have S3: \(A \in \mathbb{C}\).
have S4: \(0 \in \mathbb{C}\) by (rule MMI_0cn)
from A1 have S5: \(A \in \mathbb{C}\).
from S3 S4 S5 have S6: \(( A - 0 ) = A \iff ( 0 + A ) = A\) by (rule MMI_subadd)
from S2 S6 show \(( A - 0 ) = A\) by (rule MMI_mpbir)

qed
lemma (in MMIIsar0) MMI_negnegt:
  shows \( A \in f \rightarrow (-( - A)) = A \)
proof -
  have S1: \( A = \text{if} (A \in C, A, 0) \rightarrow (-( - A)) = \)
  \[ ( - ( - \text{if} (A \in C, A, 0)) \) \] by (rule MMI_negeq)
  from S1 have S2: \( A = \text{if} (A \in C, A, 0) \rightarrow -( -( - A)) = \)
  \[ ( - ( - \text{if} (A \in C, A, 0)) \) \] by (rule MMI_negeqd)
  have S3: \( A = \text{if} (A \in C, A, 0) \rightarrow A = \text{if} (A \in C, A, 0) \)
  by (rule MMI_id)
  from S2 S3 have S4: \( A = \text{if} (A \in C, A, 0) \rightarrow -( -( - A)) = \)
  \( ( - ( - \text{if} (A \in C, A, 0)) ) \) by (rule MMI_eqeq12d)
  have S5: \( 0 \in f \) by (rule MMI_0cn)
  from S5 have S6: \( \text{if} (A \in C, A, 0) \in f \) by (rule MMI_elimel)
  from S6 have S7: \( -( -( - \text{if} (A \in C, A, 0)) ) = \)
  \( \text{if} (A \in C, A, 0) \) by (rule MMI_negneg)
  from S4 S7 show \( A \in f \rightarrow -( -( - A)) = A \) by (rule MMI_dedth)
qed

lemma (in MMIIsar0) MMI_subnegt:
  shows \( (A \in f \land B \in f) \rightarrow (A - (- B)) = (A + B) \)
proof -
  have S1: \( (A \in C \land ( - B) \in C) \rightarrow \)
  \[ (A + (- ( - B))) = A \rightarrow \]
  \[ -( ( - \text{if} (A \in C, A, 0))) = \text{if} (A \in C, A, 0) \)
  by (rule MMI_eqeq1d2)
  from S1 have S2: \( B \in C \rightarrow ( - ( - B)) \in C \) by (rule MMI_negclt)
  from S1 S2 have S3: \( (A \in C \land B \in C) \rightarrow \)
  \[ (A + (- ( - B))) = (A - (- B)) \)
  by (rule MMI_sylan2)
  have S4: \( B \in C \rightarrow ( - ( - B)) = B \) by (rule MMI_negneg)
  from S4 have S5: \( B \in C \rightarrow (A + (- ( - B))) = \)
  \( (A + B) \) by (rule MMI_opreq2d)
  from S5 have S6: \( (A \in C \land B \in C) \rightarrow \)
  \[ (A + (- ( - B))) = (A + B) \) by (rule MMI_adantl)
  from S3 S6 show \( (A \in C \land B \in C) \rightarrow (A - (- B)) = \)
  \( (A + B) \) by (rule MMI_eqtr3d)
qed

lemma (in MMIIsar0) MMI_subidt:
  shows \( A \in f \rightarrow (A - A) = 0 \)
proof -
  have S1: \( A = \text{if} (A \in C, A, 0) \land A = \text{if} (A \in C, A, 0) \)
  \[ \rightarrow (A - A) = (\text{if} (A \in C, A, 0) - \text{if} (A \in C, A, 0)) \]
  by (rule MMI_opreq12)
  from S1 have S2: \( A = \text{if} (A \in C, A, 0) \rightarrow \)
  \[ (A - A) = (\text{if} (A \in C, A, 0) - \text{if} (A \in C, A, 0)) \]
by (rule MMI_anidms)
from S2 have S3: A = if ( A ∈ C , A , 0 ) →
    ( A - A ) = 0 ↔
    ( if ( A ∈ C , A , 0 ) - if ( A ∈ C , A , 0 ) ) = 0 )
by (rule MMI_eqeqid)
have S4: 0 ∈ C by (rule MMI_0cn)
from S4 have S5: if ( A ∈ C , A , 0 ) ∈ C by (rule MMI_elimel)
from S5 have S6:
    ( if ( A ∈ C , A , 0 ) - if ( A ∈ C , A , 0 ) ) = 0
by (rule MMI_subid)
from S3 S6 show A ∈ C → ( A - A ) = 0 by (rule MMI_dedth)
qed

lemma (in MMIisar0) MMI_subidt:
shows A ∈ C → ( A - 0 ) = A
proof -
  have S1: A = if ( A ∈ C , A , 0 ) → ( A - 0 ) =
    ( if ( A ∈ C , A , 0 ) - 0 ) by (rule MMI_opreq1)
  have S2: A = if ( A ∈ C , A , 0 ) →
    A = if ( A ∈ C , A , 0 ) by (rule MMI_id)
  from S1 S2 have S3: A = if ( A ∈ C , A , 0 ) →
    ( A - 0 ) = A ↔ ( if ( A ∈ C , A , 0 ) - 0 ) =
    if ( A ∈ C , A , 0 ) by (rule MMI_eqeq12d)
  have S4: 0 ∈ C by (rule MMI_0cn)
  from S4 have S5: if ( A ∈ C , A , 0 ) ∈ C by (rule MMI_elimel)
  from S5 have S6: ( if ( A ∈ C , A , 0 ) - 0 ) =
    if ( A ∈ C , A , 0 ) by (rule MMI_subid1)
  from S3 S6 show A ∈ C → ( A - 0 ) = A by (rule MMI_dedth)
qed

lemma (in MMIisar0) MMI_pncant:
shows ( A ∈ C ∧ B ∈ C ) → ( ( A + B ) - B ) = A
proof -
  have S1: ( A ∈ C ∧ B ∈ C ∧ B ∈ C ) → ( ( A + B ) - B ) =
    ( A + ( B - B ) ) by (rule MMI_addsubasst)
  from S1 have S2: ( A ∈ C ∧ ( B ∈ C ∧ B ∈ C ) ) →
    ( ( A + B ) - B ) = ( A + ( B - B ) ) by (rule MMI_3expb)
  from S2 have S3: ( A ∈ C ∧ B ∈ C ) → ( ( A + B ) - B ) =
    ( A + ( B - B ) ) by (rule MMI_anabsan2)
  have S4: B ∈ C → ( B - B ) = 0 by (rule MMI_subidt)
  from S4 have S5: B ∈ C → ( A + ( B - B ) ) = ( A + 0 )
    by (rule MMI_opreq2d)
  have S6: A ∈ C → ( A + 0 ) = A by (rule MMI_ax0id)
  from S5 S6 have S7: ( A ∈ C ∧ B ∈ C ) → ( A + ( B - B ) ) = A
    by (rule MMI_sylan9eqr)
  from S3 S7 show ( A ∈ C ∧ B ∈ C ) → ( ( A + B ) - B ) = A
    by (rule MMI_eqtrd)
qed
lemma (in MMIar0) MMI_pncan2t:
  shows \(( A \in \mathcal{C} \land B \in \mathcal{C} ) \to ( ( A + B ) - A ) = B\)
proof -
  have S1: \(( B \in \mathcal{C} \land A \in \mathcal{C} ) \to ( B + A ) = ( A + B )\)
    by (rule MMI_axaddcom)
  from S1 have S2: \(( B \in \mathcal{C} \land A \in \mathcal{C} ) \to ( ( B + A ) - A ) = ( ( A + B ) - A ) = B\)
    by (rule MMI_opreq1d)
  have S3: \(( B \in \mathcal{C} \land A \in \mathcal{C} ) \to ( ( B + A ) - A ) = B\)
    by (rule MMI_pncant)
  from S2 S3 have S4: \(( B \in \mathcal{C} \land A \in \mathcal{C} ) \to ( ( A + B ) - A ) = B\)
    by (rule MMI_eqtr3d)
  from S4 show \(( A \in \mathcal{C} \land B \in \mathcal{C} ) \to ( ( A + B ) - A ) = B\)
    by (rule MMI_ancoms)
qed

lemma (in MMIar0) MMI_npcant:
  shows \(( A \in \mathcal{C} \land B \in \mathcal{C} ) \to ( ( A - B ) + B ) = A\)
proof -
  have S1: \(( A \in \mathcal{C} \land B \in \mathcal{C} \land B \in \mathcal{C} ) \to ( ( A + B ) - B ) = ( ( A - B ) + B )\)
    by (rule MMI_addsubt)
  have S2: \(( A \in \mathcal{C} \land B \in \mathcal{C} \land B \in \mathcal{C} ) \to ( ( A + B ) - B ) = ( ( A - B ) + B )\)
    by (rule MMI_3expb)
  from S2 have S3: \(( A \in \mathcal{C} \land B \in \mathcal{C} ) \to ( ( A + B ) - B ) = ( ( A - B ) + B )\)
    by (rule MMI_anabsan2)
  have S4: \(( A \in \mathcal{C} \land B \in \mathcal{C} ) \to ( ( A + B ) - B ) = A\)
    by (rule MMI_pncant)
  from S3 S4 have S5: \(( A \in \mathcal{C} \land B \in \mathcal{C} ) \to ( ( A + B ) - B ) = A\)
    by (rule MMI_eqtr3d)
  qed

lemma (in MMIar0) MMI_npncant:
  shows \(( A \in \mathcal{C} \land B \in \mathcal{C} \land C \in \mathcal{C} ) \to ( ( A - B ) + ( B - C ) ) = ( A - C )\)
proof -
  have S1: \(( ( A - B ) \in \mathcal{C} \land B \in \mathcal{C} \land C \in \mathcal{C} ) \to ( ( ( A - B ) + B ) - C ) = ( ( A - B ) + ( B - C ) )\)
    by (rule MMI_addsubasst)
  have S2: \(( A \in \mathcal{C} \land B \in \mathcal{C} ) \to ( ( A - B ) ) \in \mathcal{C} \to ( ( ( A - B ) + B ) - C ) = ( ( A - B ) + ( B - C ) )\)
    by (rule MMI_subclt)
  from S2 have S3: \(( A \in \mathcal{C} \land B \in \mathcal{C} \land C \in \mathcal{C} ) \to ( ( ( A - B ) + B ) - C ) = ( ( A - B ) + ( B - C ) )\)
    by (rule MMI_3adant3)
  have S4: \(( A \in \mathcal{C} \land B \in \mathcal{C} \land C \in \mathcal{C} ) \to B \in \mathcal{C} \to ( ( ( A - B ) + B ) - C ) = ( ( A - B ) + ( B - C ) )\)
    by (rule MMI_3simp2)
  have S5: \(( A \in \mathcal{C} \land B \in \mathcal{C} \land C \in \mathcal{C} ) \to C \in \mathcal{C} \to ( ( ( A - B ) + B ) - C ) = ( ( A - B ) + ( B - C ) )\)
    by (rule MMI_3simp3)
  from S1 S3 S4 S5 have S6: \(( A \in \mathcal{C} \land B \in \mathcal{C} \land C \in \mathcal{C} ) \to ( ( ( A - B ) + B ) - C ) = ( ( A - B ) + ( B - C ) )\)
    by (rule MMI_syl3anc)
  have S7: \(( A \in \mathcal{C} \land B \in \mathcal{C} ) \to ( ( ( A - B ) + B ) = A\)
  qed
by (rule MMI_npcant)
from S7 have S8: \((A \in C \land B \in C) \rightarrow ( ( (A - B) + B) - C) = (A - C)\)
  by (rule MMI_opreq1d)
from S8 have S9: \((A \in C \land B \in C \land C \in C) \rightarrow ( ( (A - B) + B) - C) = (A - C)\)
  by (rule MMI_3adant3)
from S6 S9 show \((A \in C \land B \in C \land C \in C) \rightarrow ( (A - B) + (B - C)) = (A - C)\)
  by (rule MMI_eqtr3d)
qed
lemma (in MMIar0) MMI_nppcant:
  shows \((A \in C \land B \in C \land C \in C) \rightarrow ( ( (A - B) + C) + B) = (A + C)\)
proof -
  have S1: \((A - B) \in C \land C \in C \land B \in C) \rightarrow ( (A - B) + C) + B = (A + C)\)
    by (rule MMI_add23t)
  have S2: \((A \in C \land B \in C) \rightarrow (A - B) \in C\) by (rule MMI_subclt)
  from S2 have S3: \((A \in C \land B \in C \land C \in C) \rightarrow (A - B) \in C\)
    by (rule MMI_3adant3)
  have S4: \((A \in C \land B \in C \land C \in C) \rightarrow C \in C\) by (rule MMI_3simp3)
  have S5: \((A \in C \land B \in C \land C \in C) \rightarrow B \in C\) by (rule MMI_3simp2)
  from S1 S3 S4 S5 have S6: \((A \in C \land B \in C \land C \in C) \rightarrow ( (A - B) + C) + B = A\)
    by (rule MMI_syl3anc)
  have S7: \((A \in C \land B \in C) \rightarrow ( (A - B) + B) = A\)
    by (rule MMI_npcant)
  from S7 have S8: \((A \in C \land B \in C) \rightarrow ( (A - B) + B) + C = (A + C)\)
    by (rule MMI_opreq1d)
  from S8 have S9: \((A \in C \land B \in C \land C \in C) \rightarrow ( (A - B) + B) + C = (A + C)\)
    by (rule MMI_3adant3)
  from S6 S9 show \((A \in C \land B \in C \land C \in C) \rightarrow ( (A - B) + C) + B = (A + C)\)
    by (rule MMI_eqtrd)
qed
lemma (in MMIar0) MMI_subneg: assumes A1: \(A \in C\) and
  A2: \(B \in C\)
  shows \(A - (\sim B) = (A + B)\)
proof -
  from A1 have S1: \(A \in C\).
  from A2 have S2: \(B \in C\).
  have S3: \((A \in C \land B \in C) \rightarrow (A - (\sim B)) = (A + B)\)
    by (rule MMI_subnegt)
  from S1 S2 S3 show \((A - (\sim B)) = (A + B)\)
    by (rule MMI_mp2an)
lemma (in MMIIsar0) MMI_subeq0: assumes A1: A ∈ C and
A2: B ∈ C
shows ( A - B ) = 0 ⇔ A = B

proof -
from A1 have S1: A ∈ C.
from A2 have S2: B ∈ C.
from S1 S2 have S3: ( A + ( ( - B ) ) ) = ( A - B )
  by (rule MMI_negsub)
from S3 have S4: ( A + ( ( - B ) ) ) = 0 ⇔ ( A - B ) = 0
  by (rule MMI_eqeq1i)
from S4 have S5: ( A + ( ( - B ) ) ) = 0
  by (rule MMI_opreq1ii)
from S5 have S6: ( ( A + ( ( - B ) ) + B ) = ( A - B ) ) by (rule MMI_negid)

from A1 have S7: A ∈ C.
from A2 have S8: B ∈ C.

from S8 have S9: ( ( - B ) ) ∈ C by (rule MMI_negcl)
from A2 have S10: B ∈ C.

from S7 S9 S10 have S11: ( ( A + ( ( - B ) ) + B ) =
  ( A + B ) + ( ( - B ) ) ) by (rule MMI_add23)
from A1 have S12: A ∈ C.
from A2 have S13: B ∈ C.

from S12 S13 have S14: ( - B ) ∈ C.
from S9 S14 have S15: ( ( A + B ) + ( ( - B ) ) ) =
  ( A + ( B + ( ( - B ) ) ) ) by (rule MMI_addass)
from A2 have S16: B ∈ C.

from S16 have S17: ( B + ( ( - B ) ) ) = 0 by (rule MMI_negid)
from S17 have S18: ( A + ( B + ( ( - B ) ) ) ) = ( A + 0 )
  by (rule MMI_addid1)
from S18 have S19: A ∈ C.
from S19 have S20: ( A + 0 ) = A by (rule MMI_addid1)
from S20 have S21: ( A + ( B + ( ( - B ) ) ) ) = A
  by (rule MMI_eqtr)
from S21 have S22: ( ( A + ( ( - B ) ) ) + B ) = A
  by (rule MMI_3eqtr)
from A2 have S23: B ∈ C.

from S23 have S24: ( 0 + B ) = B by (rule MMI_addid2)
from S24 have S25: ( A - B ) = 0
  by (rule MMI_syl6eq)
from S25 S29 show ( A - B ) = 0 ⇔ A = B by (rule MMI_impbi)

qed

lemma (in MMIIsar0) MMI_neg11: assumes A1: A ∈ C and
A2: B ∈ C
shows ( A + ( ( - B ) ) ) = 0 ⇔ ( A - B ) = 0

proof -
from A1 have S1: A ∈ C.
from A2 have S2: B ∈ C.

from S1 S2 have S3: ( A + ( ( - B ) ) ) = ( A - B )
  by (rule MMI_negsub)
from S3 have S4: ( A + ( ( - B ) ) ) = 0 ⇔ ( A - B ) = 0
  by (rule MMI_eqeq1i)
from S4 have S5: ( A + ( ( - B ) ) ) = 0
  by (rule MMI_opreq1ii)
from S5 have S6: ( ( A + ( ( - B ) ) + B ) = ( A - B ) ) by (rule MMI_negid)

from A1 have S7: A ∈ C.
from A2 have S8: B ∈ C.

from S8 have S9: ( ( - B ) ) ∈ C by (rule MMI_negcl)
from A2 have S10: B ∈ C.

from S7 S9 S10 have S11: ( ( A + ( ( - B ) ) + B ) =
  ( A + B ) + ( ( - B ) ) ) by (rule MMI_add23)
from A1 have S12: A ∈ C.
from A2 have S13: B ∈ C.

from S12 S13 have S14: ( - B ) ∈ C.
from S9 S14 have S15: ( ( A + B ) + ( ( - B ) ) ) =
  ( A + ( B + ( ( - B ) ) ) ) by (rule MMI_addass)
from A2 have S16: B ∈ C.

from S16 have S17: ( B + ( ( - B ) ) ) = 0 by (rule MMI_negid)
from S17 have S18: ( A + ( B + ( ( - B ) ) ) ) = ( A + 0 )
  by (rule MMI_addid1)
from S18 have S19: A ∈ C.
from S19 have S20: ( A + 0 ) = A by (rule MMI_addid1)
from S20 have S21: ( A + ( B + ( ( - B ) ) ) ) = A
  by (rule MMI_eqtr)
from S21 have S22: ( ( A + ( ( - B ) ) ) + B ) = A
  by (rule MMI_3eqtr)
from A2 have S23: B ∈ C.

from S23 have S24: ( 0 + B ) = B by (rule MMI_addid2)
from S24 have S25: ( A - B ) = 0
  by (rule MMI_syl6eq)
from S25 S29 show ( A - B ) = 0 ⇔ A = B by (rule MMI_impbi)

qed
A2: B ∈ C
shows ( (- A) ) = ( (- B) ) ⟷ A = B
proof -
  have S1: ( (- A) ) = ( 0 - A ) by (rule MMI_df_neg)
  have S2: ( (- B) ) = ( 0 - B ) by (rule MMI_df_neg)
  from S1 S2 have S3: ( (- A) ) = ( (- B) ) ⟷ ( 0 - A ) = ( 0 - B ) by (rule MMI_eqeq12i)
  have S4: 0 ∈ C by (rule MMI_0cn)
  from A1 have S5: A ∈ C.
  have S6: 0 ∈ C by (rule MMI_0cn)
  from A2 have S7: B ∈ C.
  from S6 S7 have S8: ( 0 - B ) ∈ C by (rule MMI_subcl)
  from S4 S5 S8 have S9: ( 0 - A ) = ( 0 - B ) ⟷ ( A + ( 0 - B ) ) = 0 by (rule MMI_subadd)
  from S2 have S10: ( - B ) = ( 0 - B ).
  from S10 have S11: ( A + ( - B ) ) = ( A + ( 0 - B ) )
    by (rule MMI_opreq2i)
  from A1 have S12: A ∈ C.
  from A2 have S13: B ∈ C.
  from S12 S13 have S14: ( A + ( - B ) ) = ( A - B )
    by (rule MMI_negsub)
  from S11 S14 have S15: ( A + ( 0 - B ) ) = ( A - B )
    by (rule MMI_eqtr3)
  from S15 have S16: ( A + ( 0 - B ) ) = 0 ⟷ ( A - B ) = 0
    by (rule MMI_eqeq1i)
  from A1 have S17: A ∈ C.
  from A2 have S18: B ∈ C.
  from S17 S18 have S19: ( A - B ) = 0 ⟷ A = B by (rule MMI_subeq0)
  from S16 S19 have S20: ( A + ( 0 - B ) ) = 0 ⟷ A = B
    by (rule MMI_bitr)
  from S3 S9 S20 show ( (- A) ) = ( (- B) ) ⟷ A = B by (rule MMI_3bitr)
  qed

lemma (in MMIar0) MMI_negcon1: assumes A1: A ∈ C and
  A2: B ∈ C
shows ( (- A) ) = B ⟷ ( (- B) ) = A
proof -
  from A1 have S1: A ∈ C.
  from S1 have S2: (- ( (- A) ) ) = A by (rule MMI_negneg)
  from S2 have S3: ( - ( (- A) ) ) = ( (- B) ) ⟷ A = ( (- B) )
    by (rule MMI_eqeq1i)
  from A1 have S4: A ∈ C.
  from S4 have S5: ( - ( A) ) ∈ C by (rule MMI_negcl)
  from A2 have S6: B ∈ C.
  from S5 S6 have S7: ( - ( (- A) ) ) = ( (- B) ) ⟷ ( (- A) ) = B by (rule MMI_eqeq1i)
have S8: \( A = ( -B) \leftrightarrow ( -B) = A \) by (rule MMI_eqcom)
from S3 S7 S8 show \(( -A) = B \leftrightarrow ( -B) = A \) by (rule MMI_3bitr3)
qed

lemma (in MMIIsar0) MMI_negcon2: assumes A1: \( A \in \mathcal{C} \) and
\( A2: B \in \mathcal{C} \)
shows \( A = ( -B) \leftrightarrow ( -B) = A \) by (rule MMI_eqcom)
proof -
from A2 have S1: \( B \in \mathcal{C} \).
from A1 have S2: \( A \in \mathcal{C} \).
from S1 S2 have S3: \( ( -B) = A \leftrightarrow ( -B) = A \) by (rule MMI_negcon1)
have S4: \( A = ( -B) \leftrightarrow ( -B) = A \) by (rule MMI_eqcom)
have S5: \( B = ( -A) \leftrightarrow ( -A) = B \) by (rule MMI_eqcom)
from S3 S4 S5 show \( A = ( -B) \leftrightarrow B = ( -A) \) by (rule MMI_3bitr4)
qed

lemma (in MMIIsar0) MMI_neg11t:
shows \( ( A \in \mathcal{C} \land B \in \mathcal{C} ) \rightarrow ( ( -A) = ( -B) \leftrightarrow A = B \) )
proof -
have S1: \( A = \text{if } ( A \in \mathcal{C} , A , 0 ) \rightarrow ( ( -A) = ( -if ( A \in \mathcal{C} , A , 0 ) ) \text{ by (rule MMI_negeq) } )
from S1 have S2: \( A = \text{if } ( A \in \mathcal{C} , A , 0 ) \rightarrow ( ( -B) ) \leftrightarrow ( -if ( A \in \mathcal{C} , A , 0 ) ) \leftrightarrow ( ( -B) ) \) by (rule MMI_eqeq1d)
have S3: \( A = \text{if } ( A \in \mathcal{C} , A , 0 ) \rightarrow ( A = B \leftrightarrow ( -if ( A \in \mathcal{C} , A , 0 ) ) \text{ by (rule MMI_eqeq1) } )
from S2 S3 have S4: \( A = \text{if } ( A \in \mathcal{C} , A , 0 ) \rightarrow ( ( ( -A) ) = ( ( -B) ) \leftrightarrow A = B ) \leftrightarrow ( ( -if ( A \in \mathcal{C} , A , 0 ) ) = ( ( -B) ) \leftrightarrow ( if ( A \in \mathcal{C} , A , 0 ) = B ) ) \) by (rule MMI_bibi12d)
have S5: \( B = \text{if } ( B \in \mathcal{C} , B , 0 ) \rightarrow ( ( -B) ) \leftrightarrow ( ( -if ( A \in \mathcal{C} , A , 0 ) ) = ( -if ( B \in \mathcal{C} , B , 0 ) ) \leftrightarrow ( ( -if ( A \in \mathcal{C} , A , 0 ) ) = ( -if ( B \in \mathcal{C} , B , 0 ) ) \leftrightarrow ( if ( A \in \mathcal{C} , A , 0 ) ) \leftrightarrow ( if ( B \in \mathcal{C} , B , 0 ) ) ) \) by (rule MMI_bibi12d)
have S6: \( B = \text{if } ( B \in \mathcal{C} , B , 0 ) \rightarrow ( ( ( -B) ) \leftrightarrow ( if ( A \in \mathcal{C} , A , 0 ) ) \leftrightarrow ( if ( A \in \mathcal{C} , A , 0 ) ) \leftrightarrow ( if ( B \in \mathcal{C} , B , 0 ) ) ) \) by (rule MMI_0cn)
have S7: \( 0 \in \mathcal{C} \) by (rule MMI_0cn)
from S6 S7 have S8: \( B = \text{if } ( B \in \mathcal{C} , B , 0 ) \rightarrow ( ( ( -if ( A \in \mathcal{C} , A , 0 ) ) = ( if ( A \in \mathcal{C} , A , 0 ) ) \leftrightarrow ( ( ( -if ( B \in \mathcal{C} , B , 0 ) ) = ( if ( A \in \mathcal{C} , A , 0 ) ) \leftrightarrow ( if ( B \in \mathcal{C} , B , 0 ) ) ) \leftrightarrow ( ( -if ( A \in \mathcal{C} , A , 0 ) ) = ( if ( B \in \mathcal{C} , B , 0 ) ) ) \leftrightarrow ( if ( B \in \mathcal{C} , B , 0 ) ) ) \leftrightarrow ( if ( B \in \mathcal{C} , B , 0 ) ) ) \) by (rule MMI_bibi12d)
have S9: \( 0 \in \mathcal{C} \) by (rule MMI_0cn)
from S9 have S10: \( \text{if } ( A \in \mathcal{C} , A , 0 ) \in \mathcal{C} \) by (rule MMI_elimel)
have S11: \( 0 \in \mathcal{C} \) by (rule MMI_0cn)
from S11 have S12: \( \text{if } ( B \in \mathcal{C} , B , 0 ) \in \mathcal{C} \) by (rule MMI_elimel)

1462
from S10 S12 have S13: (- if ( A ∈ C , A , 0 ) ) =
   (- if ( B ∈ C , B , 0 ) ) ←→ if ( A ∈ C , A , 0 ) =
   if ( B ∈ C , B , 0 ) by (rule MMI_neg11)
from S4 S8 S13 show ( A ∈ C ∧ B ∈ C ) → (( (- A ) ) =
   ( (- B ) ) ←→ A = B ) by (rule MMI_dedth2h)
qed

lemma (in MMIsar0) MMI_negcon1t:
  shows ( A ∈ C ∧ B ∈ C ) → (( (- A ) ) = B ←→ ( (- B ) ) = A )
proof -
  have S1: ( ( (- A ) ) ∈ C ∧ B ∈ C ) → (( ( - ( - A ) ) ) =
   ( (- B ) ) ←→ ( ( - A ) ) = B ) by (rule MMI_neg11t)
  have S2: A ∈ C → ( ( - A ) ) ∈ C by (rule MMI_negclt)
  from S1 S2 have S3: ( A ∈ C ∧ B ∈ C ) → (( ( - ( - A ) ) ) =
   ( (- B ) ) ←→ ( ( - A ) ) = B ) by (rule MMI_sylan)
  have S4: A ∈ C → ( ( - ( - A ) ) ) = A by (rule MMI_negnegt)
  from S4 have S5: A ∈ C → ( - ( (- A ) ) ) = A by (rule MMI_adantr)
  from S5 have S6: ( A ∈ C ∧ B ∈ C ) → (( ( - ( - A ) ) ) =
   ( (- B ) ) ←→ A = ( ( - B ) ) ) by (rule MMI_eqeq1d)
  from S3 S6 have S7: ( A ∈ C ∧ B ∈ C ) → (( ( - A ) ) = B ←→
   A = ( ( - B ) ) ) by (rule MMI_syl6bb)
qed

lemma (in MMIsar0) MMI_negcon2t:
  shows ( A ∈ C ∧ B ∈ C ) → ( A = ( ( - B ) ) ←→ B = ( ( - A ) ) )
proof -
  have S1: ( A ∈ C ∧ B ∈ C ) → (( ( - A ) ) = B ←→ ( ( - B ) ) =
   A ) by (rule MMI_negcon1t)
  have S2: A = ( ( - B ) ) ←→ ( ( - B ) ) = A by (rule MMI_eqcom)
  from S1 S2 have S3: ( ( - A ) ) = B ←→ A = ( ( - B ) ) ) by (rule MMI_syl6rbbra)
  from S3 S4 have S5: ( A ∈ C ∧ B ∈ C ) → (( ( - A ) ) = B ←→
   ( ( - A ) ) ) by (rule MMI_syl6bb)
qed

lemma (in MMIsar0) MMI_subcalt:
  shows ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) → (( A - B ) =
   ( A - C ) ←→ B = C )
proof -
  have S1: ( A ∈ C ∧ ( ( - B ) ) ∈ C ∧ ( ( - C ) ) ∈ C ) →
   (( ( A = ( ( - B ) ) ) = ( A = ( ( - C ) ) ) ←→
   ( ( - B ) ) = ( ( - C ) ) ) , by (rule MMI_addcalt)
have S2: C ∈ C → (¬C) ∈ C by (rule MMI_negclt)
from S1, S2 have S3: (A ∈ C ∧ (¬B)) ∈ C ∧ C ∈ C) →
  ((A + (¬B)) = (A + (¬C)) ↔
  (¬B) = (¬C) by (rule MMI_syl3an3)
have S4: B ∈ C → (¬B) ∈ C by (rule MMI_negclt)
from S3, S4 have S5: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
  ((A + (¬B)) = (A + (¬C)) ↔
  (¬B) = (¬C) by (rule MMI_syl3an2)
have S6: (A ∈ C ∧ B ∈ C) → (A + (¬B)) = (A − B)
by (rule MMI_negsubt)
from S6 have S7: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
  (A + (¬B)) = (A − B) by (rule MMI_3adant3)
have S8: (A ∈ C ∧ C ∈ C) → (A + (¬C)) = (A − C)
by (rule MMI_negsubt)
from S8 have S9: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
  (A + (¬C)) = (A − C) by (rule MMI_3adant2)
from S7, S9 have S10: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
  (A + (¬B)) = (A + (¬C)) ↔
  (A − B) = (A − C) by (rule MMI_eqeq12d)
have S11: (B ∈ C ∧ C ∈ C) → ((¬B) = (¬C) ↔ B = C)
by (rule MMI_neg11t)
from S11 have S12: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
  ((¬B) = (¬C) ↔ B = C) by (rule MMI_3adant1)
from S5, S10, S12 show (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
  ((A − B) = (A − C) ↔ B = C) by (rule MMI_3bitr3d)

qed

lemma (in MMIisar0) MMI_subcan2t:
  shows (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
  ((A − C) = (B − C) ↔ A = B)
proof -
  have S1: (A ∈ C ∧ C ∈ C) → (A + (¬C)) = (A − C)
  by (rule MMI_negsubt)
  from S1 have S2: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
  (A + (¬C)) = (A − C) by (rule MMI_3adant2)
  have S3: (B ∈ C ∧ C ∈ C) → (B + (¬C)) = (B − C)
  by (rule MMI_negsubt)
  from S3 have S4: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
  (B + (¬C)) = (B − C) by (rule MMI_3adant1)
  from S2, S4 have S5: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
  ((A + (¬C)) = (B + (¬C)) ↔ (A − C) =
  (B − C)) by (rule MMI_eqeq12d)
  have S6: (A ∈ C ∧ B ∈ C ∧ (¬C) ∈ C) →
  ((A + (¬C)) = (B + (¬C)) ↔ A = B)
  by (rule MMI_addcan2t)
  have S7: C ∈ C → (¬C) ∈ C by (rule MMI_negclt)
  from S6, S7 have S8: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
  ((A + (¬C)) = (B + (¬C)) ↔ A = B)

1464
by (rule MMI_syl3an3)
\[ S5 \quad S8 \quad \text{show} \quad (A \in C \land B \in C \land C \in C) \rightarrow \\
( (A - C) = (B - C) \iff A = B) \text{ by (rule MMI_bitr3d)} \]

qed

lemma (in MMIsar0) MMI_subcan: assumes A1: A \in C and 
\qquad A2: B \in C and 
\qquad A3: C \in C 
shows \[ (A - B) = (A - C) \iff B = C \]
proof - 
\quad from A1 have S1: A \in C.
\quad from A2 have S2: B \in C.
\quad from A3 have S3: C \in C.
\quad have S4: \[ (A \in C \land B \in C \land C \in C) \rightarrow \\
( (A - B) = (A - C) \iff B = C) \text{ by (rule MMI_subcan)} \]
\quad from S1 S2 S3 S4 show \[ (A - B) = (A - C) \iff B = C \]
\quad by (rule MMI_mp3an)

qed

lemma (in MMIsar0) MMI_subcan2: assumes A1: A \in C and 
\qquad A2: B \in C and 
\qquad A3: C \in C 
shows \[ (A - C) = (B - C) \iff A = B \]
proof - 
\quad from A1 have S1: A \in C.
\quad from A2 have S2: B \in C.
\quad from A3 have S3: C \in C.
\quad have S4: \[ (A \in C \land B \in C \land C \in C) \rightarrow \\
( (A - C) = (B - C) \iff A = B) \text{ by (rule MMI_subcan2)} \]
\quad from S1 S2 S3 S4 show \[ (A - C) = (B - C) \iff A = B \]
\quad by (rule MMI_mp3an)

qed

lemma (in MMIsar0) MMI_subeq0t: 
\quad shows \[ (A \in C \land B \in C) \rightarrow \\
( (A - B) = 0 \iff A = B) \]
proof - 
\quad have S1: A = if (A \in C, A, 0) \rightarrow (A - B) = \\
\qquad (if (A \in C, A, 0) - B) \text{ by (rule MMI_opreq1)} 
\quad from S1 have S2: A = if (A \in C, A, 0) \rightarrow (A - B) = 0 \iff \\
\qquad (if (A \in C, A, 0) - B) = 0 \iff 
\quad have S3: A = if (A \in C, A, 0) \rightarrow (A = B \iff \\
\qquad if (A \in C, A, 0) - B) \text{ by (rule MMI_eqeq1)} 
\quad from S2 S3 have S4: A = if (A \in C, A, 0) \rightarrow \\
\qquad ((A - B) = 0 \iff A = B) \iff 
\qquad ((if (A \in C, A, 0) - B) = 0 \iff 
\quad have S5: B = if (B \in C, B, 0) \rightarrow \\
\qquad (if (A \in C, A, 0) - B) =
( if ( A ∈ C , A , 0 ) - if ( B ∈ C , B , 0 ) )
by (rule MMI_opreq2)
from S5 have S6: B = if ( B ∈ C , B , 0 ) →
( ( if ( A ∈ C , A , 0 ) - B ) = 0 ⟷
( if ( A ∈ C , A , 0 ) - if ( B ∈ C , B , 0 ) ) = 0 )
by (rule MMI_eqeq1d)
have S7: B = if ( B ∈ C , B , 0 ) → ( if ( A ∈ C , A , 0 ) = B )
by (rule MMI_eqeq2)
from S6 S7 have S8: B = if ( B ∈ C , B , 0 ) →
( ( if ( A ∈ C , A , 0 ) - B ) = 0 ⟷
if ( A ∈ C , A , 0 ) = B ⟷
if ( A ∈ C , A , 0 ) = if ( B ∈ C , B , 0 ) )
by (rule MMI_bibi12d)
have S9: 0 ∈ C by (rule MMI_0cn)
from S9 have S10: if ( A ∈ C , A , 0 ) ∈ C by (rule MMI_elimel)
have S11: 0 ∈ C by (rule MMI_0cn)
from S11 have S12: if ( B ∈ C , B , 0 ) ∈ C by (rule MMI_elimel)
from S10 S12 have S13:
( if ( A ∈ C , A , 0 ) - if ( B ∈ C , B , 0 ) ) = 0 ⟷
if ( A ∈ C , A , 0 ) = if ( B ∈ C , B , 0 )
by (rule MMI_subeq0)
from S4 S8 S13 show ( A ∈ C ∧ B ∈ C ) →
( ( A - B ) = 0 ⟷ A = B ) by (rule MMI_dedth2h)
qed

lemma (in MMIsar0) MMI_neg0:
shows ( - 0 ) = 0
proof -
  have S1: ( - 0 ) = ( 0 - 0 ) by (rule MMI_df_neg)
  have S2: 0 ∈ C by (rule MMI_0cn)
  from S2 have S3: ( 0 - 0 ) = 0 by (rule MMI_subid)
  from S1 S3 show ( - 0 ) = 0 by (rule MMI_eqtr)
qed

lemma (in MMIsar0) MMI_renegcl: assumes A1: A ∈ R
shows ( - A ) ∈ R
proof -
  from A1 have S1: A ∈ R.
  have S2: A ∈ R → ( ∃ x ∈ R . ( A + x ) = 0 ) by (rule MMI_axrneqex)
  from S1 S2 have S3: ∃ x ∈ R . ( A + x ) = 0 by (rule MMI_ax_mp)
  have S4: ( ∃ x ∈ R . ( A + x ) = 0 ) ⟷
( ∃ x . ( x ∈ R ∧ ( A + x ) = 0 ) ) by (rule MMI_df_rex)
  from S3 S4 have S5: ∃ x . ( x ∈ R ∧ ( A + x ) = 0 )
  by (rule MMI_mpbi)
  { fix x

1466
have S6: \( x \in R \rightarrow x \in C \) by (rule MMI_recnt)

have S7: \( 0 \in C \) by (rule MMI_0cn)

deprecated

from A1 have S8: \( A \in R \).

from S8 have S9: \( A \in C \) by (rule MMI_recn)

have S10: \( ( 0 \in C \land A \in C \land x \in C ) \rightarrow ( \( 0 - A \) ) = x \leftrightarrow \( A + x \) = 0 ) by (rule MMI_subaddt)

from S7 S9 S10 have S11: \( x \in C \rightarrow ( \( 0 - A \) ) = x \leftrightarrow \( A + x \) = 0 ) by (rule MMI_mp3an12)

from S11 have S12: \( 0 \in C \rightarrow ( \( 0 - A \) ) = x \leftrightarrow \( A + x \) = 0 ) by (rule MMI_mp3an12)

have S13: \( ( - A ) = ( - A = x \leftrightarrow \( A + x \) = 0 ) \) by (rule MMI_eqeq1)

from S12 S13 have S14: \( ( - A ) = x \leftrightarrow \( A + x \) = 0 ) by (rule MMI_eqeq1)

from S14 have S15: \( ( - A ) = x \leftrightarrow \( A + x \) = 0 ) by (rule MMI_eqeq1)

have S16: \( x \in R \rightarrow ( ( - A ) ) \in R \)

from S16 have S17: \( x \in R \rightarrow ( ( - A ) ) \in R \)

by (rule MMI_renegcl)

from S17 have S18: \( \forall x . ( x \in R \land ( A + x ) = 0 ) \rightarrow ( ( - A ) ) \in R \)

by auto

from S18 have S19: \( \exists x . ( x \in R \land ( A + x ) = 0 ) \rightarrow ( ( - A ) ) \in R \)

by (rule MMI_ax_mp)

qed

lemma (in MMIar0) MMI_renegclt:

shows \( \forall x . ( x \in R \land ( A + x ) = 0 ) \rightarrow ( ( - A ) ) \in R \)

proof -

have S1: \( A = if ( A \in R , A , 1 ) \rightarrow ( ( - A ) ) = ( - if ( A \in R , A , 1 ) ) \) by (rule MMI_negeq)

from S1 have S2: \( A = if ( A \in R , A , 1 ) \rightarrow ( ( - A ) ) \in R \leftrightarrow \)

\( ( - if ( A \in R , A , 1 ) ) \in R \) by (rule MMI_eleq1)

have S3: \( 1 \in R \) by (rule MMI_axire)

from S3 have S4: \( if ( A \in R , A , 1 ) \in R \) by (rule MMI_elim)

from S4 have S5: \( ( - if ( A \in R , A , 1 ) ) \rightarrow ( ( - A ) ) \in R \) by (rule MMI_renegcl)

from S2 S5 show \( A \in R \rightarrow ( ( - A ) ) \in R \) by (rule MMI_dedth)

qed

lemma (in MMIar0) MMI_resubclt:

shows \( A \in R \rightarrow ( ( - A ) ) \in R \)

proof -

have S1: \( ( A \in R \land B \in R ) \rightarrow ( A - B ) \in R \)

by (rule MMI_recnt)
have $S_3$: $B \in R \rightarrow B \in C$ by (rule MMI_recnt)
from $S_1$ $S_2$ $S_3$ have $S_4$: $(A \in R \land B \in R) \rightarrow (A + (\neg B))$

= $(A - B)$ by (rule MMI_syl2an)

have $S_5$: $(A \in R \land (\neg B) \in R) \rightarrow (A + (\neg B)) \in R$
by (rule MMI_axaddrcl)

have $S_6$: $B \in R \rightarrow (\neg B) \in R$ by (rule MMI_renegclt)
from $S_5$ $S_6$ have $S_7$: $(A \in R \land B \in R) \rightarrow (A + (\neg B)) \in R$
by (rule MMI_sylan2)

from $S_4$ $S_7$ show $(A \in R \land B \in R) \rightarrow (A - B) \in R$
by (rule MMI_eqeltrrd)

qed

lemma (in MMIar0) MMI_resubcl: assumes $A_1$: $A \in R$ and
$A_2$: $B \in R$
shows $(A - B) \in R$
proof -
from $A_1$ have $S_1$: $A \in R$.
from $A_2$ have $S_2$: $B \in R$.

have $S_3$: $(A \in R \land B \in R) \rightarrow (A - B) \in R$ by (rule MMI_resubclt)
from $S_1$ $S_2$ $S_3$ show $(A - B) \in R$ by (rule MMI_mp2an)

qed

lemma (in MMIar0) MMI_0re:
says $0 \in R$
proof -
have $S_1$: $1 \in F$ by (rule MMI_1cn)
from $S_1$ have $S_2$: $(1 - 1) = 0$ by (rule MMI_subid)

have $S_3$: $1 \in R$ by (rule MMI_axire)
have $S_4$: $1 \in R$ by (rule MMI_axire)

from $S_3$ $S_4$ have $S_5$: $(1 - 1) \in R$ by (rule MMI_resubclt)
from $S_2$ $S_5$ show $0 \in R$ by (rule MMI_eqeltrrd)

qed

lemma (in MMIar0) MMI_mulid2t:
says $A \in F \rightarrow (1 \cdot A) = A$
proof -
have $S_1$: $1 \in C$ by (rule MMI_1cn)

have $S_2$: $(1 \in C \land A \in C) \rightarrow (1 \cdot A) = (A \cdot 1)$
by (rule MMI_axmulcom)
from $S_1$ $S_2$ have $S_3$: $A \in C \rightarrow (1 \cdot A) = (A \cdot 1)$ by (rule MMI_mpan)
have $S_4$: $A \in C \rightarrow (A \cdot 1) = A$ by (rule MMI_axiid)
from $S_3$ $S_4$ show $A \in C \rightarrow (1 \cdot A) = A$ by (rule MMI_eqtrd)

qed

lemma (in MMIar0) MMI_mul12t:
shows \((A \in C \land B \in C \land C \in C) \rightarrow (A \cdot (B \cdot C)) = (B \cdot (A \cdot C))\)

proof -

have S1: \((A \in C \land B \in C) \rightarrow (A \cdot B) = (B \cdot A)\)
by (rule MMI_axmulcom)

from S1 have S2: \((A \in C \land B \in C) \rightarrow ((A \cdot B) \cdot C) = ((B \cdot A) \cdot C)\)
by (rule MMI_opreq1d)

from S2 have S3: \((A \in C \land B \in C \land C \in C) \rightarrow ((A \cdot B) \cdot C) = ((B \cdot A) \cdot C)\)
by (rule MMI_3adant3)

have S4: \((A \in C \land B \in C \land C \in C) \rightarrow ((A \cdot B) \cdot C) = (A \cdot (B \cdot C))\)
by (rule MMI_axmulass)

have S5: \((B \in C \land A \in C \land C \in C) \rightarrow ((B \cdot A) \cdot C) = (B \cdot (A \cdot C))\)
by (rule MMI_axmulass)

from S5 have S6: \((B \in C \land A \in C \land C \in C) \rightarrow ((B \cdot A) \cdot C) = (B \cdot (A \cdot C))\)
by (rule MMI_3com12)

from S3 S4 S6 show \((A \in C \land B \in C \land C \in C) \rightarrow (A \cdot (B \cdot C)) = (B \cdot (A \cdot C))\)
by (rule MMI_3eqtr3d)

qed


lemma (in MMIar0) MMI_mul23t:
shows \((A \in C \land B \in C \land C \in C) \rightarrow ((A \cdot B) \cdot C) = ((A \cdot C) \cdot B)\)

proof -

have S1: \((B \in C \land C \in C) \rightarrow (B \cdot C) = (C \cdot B)\)
by (rule MMI_axmulcom)

from S1 have S2: \((B \in C \land C \in C) \rightarrow (A \cdot (B \cdot C)) = (A \cdot (C \cdot B))\)
by (rule MMI_opreq2d)

from S2 have S3: \((A \in C \land B \in C \land C \in C) \rightarrow (A \cdot (B \cdot C)) = (A \cdot (C \cdot B))\)
by (rule MMI_3adant1)

have S4: \((A \in C \land B \in C \land C \in C) \rightarrow ((A \cdot B) \cdot C) = (A \cdot (B \cdot C))\)
by (rule MMI_axmulass)

have S5: \((A \in C \land C \in C \land B \in C) \rightarrow ((A \cdot C) \cdot B) = (A \cdot (C \cdot B))\)
by (rule MMI_axmulass)

from S5 have S6: \((A \in C \land B \in C \land C \in C) \rightarrow (A \cdot (C \cdot B)) = (A \cdot (C \cdot B))\)
by (rule MMI_3com23)

from S3 S4 S6 show \((A \in C \land B \in C \land C \in C) \rightarrow ((A \cdot B) \cdot C) = ((A \cdot C) \cdot B)\)
by (rule MMI_3eqtr4d)

qed

lemma (in MMIar0) MMI_mul4t:
shows \(((A \in C \land B \in C) \land (C \in C \land D \in C)) \rightarrow ((A \cdot B) \cdot (C \cdot D)) = ((A \cdot C) \cdot (B \cdot D))\)

proof -

have S1: \((A \in C \land B \in C \land C \in C) \rightarrow ((A \cdot B) \cdot C) = (A \cdot (B \cdot C))\)
by (rule MMI_mul23t)

from S1 have S2: \((A \in C \land B \in C \land C \in C) \rightarrow ((A \cdot B) \cdot C) = ((A \cdot C) \cdot B)\)
by (rule MMI_opreq1d)

1469
from S2 have S3: ( ( A \in C \land B \in C ) \land C \in C ) \rightarrow \\
( ( ( A \cdot B ) \cdot C ) \cdot D ) = ( ( ( A \cdot C ) \cdot B ) \cdot D ) \\
by (rule MMI_3expa)

from S3 have S4: ( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \rightarrow \\
( ( ( A \cdot B ) \cdot C ) \cdot D ) = ( ( ( A \cdot C ) \cdot B ) \cdot D ) \\
by (rule MMI_adantrr)

have S5: ( ( A \cdot B ) \in C \land C \in C \land D \in C ) \rightarrow \\
( ( ( A \cdot B ) \cdot C ) \cdot D ) = ( ( A \cdot B ) \cdot ( C \cdot D ) ) \\
by (rule MMI_3expb)

from S5 have S6: ( ( ( A \cdot B ) ) \in C \land ( ( C \in C \land D \in C ) ) \rightarrow \\
( ( ( A \cdot B ) \cdot C ) \cdot D ) = ( ( A \cdot B ) \cdot ( C \cdot D ) ) \\
by (rule MMI_axmulass)

have S7: ( ( A \in C \land B \in C ) \rightarrow ( A \cdot B ) \in C ) by (rule MMI_axmulcl)

from S6 S7 have S8: ( ( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \rightarrow \\
( ( ( A \cdot B ) \cdot C ) \cdot D ) = ( ( A \cdot B ) \cdot ( C \cdot D ) ) \\
by (rule MMI_sylan)

have S9: ( ( ( A \cdot C ) \in C \land B \in C \land D \in C ) ) \rightarrow \\
( ( ( A \cdot C ) \cdot B ) \cdot D ) = ( ( A \cdot C ) \cdot ( B \cdot D ) ) \\
by (rule MMI_axmulass)

from S9 have S10: ( ( ( A \cdot C ) \in C \land ( B \in C \land D \in C ) ) \rightarrow \\
( ( ( A \cdot C ) \cdot B ) \cdot D ) = ( ( A \cdot C ) \cdot ( B \cdot D ) ) \\
by (rule MMI_3expb)

have S11: ( ( A \in C \land C \in C ) \rightarrow ( A \cdot C ) \in C ) by (rule MMI_axmulcl)

from S10 S11 have S12: ( ( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \rightarrow \\
( ( ( A \cdot C ) \cdot B ) \cdot D ) = ( ( A \cdot C ) \cdot ( B \cdot D ) ) \\
by (rule MMI_3eqtr3d)

qed

text

lemma (in MMIsar0) MMI_muladdt:

\text{shows } ( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \rightarrow \\
( ( A + B ) \cdot ( C + D ) ) = \\
( ( ( A \cdot C ) + ( D \cdot B ) ) + ( ( A \cdot D ) + ( C \cdot B ) ) )

proof -

have S1: ( ( ( A + B ) \in C \land C \in C \land D \in C ) ) \rightarrow \\
( ( A + B ) \cdot ( C + D ) ) = \\
( ( ( A + B ) \cdot C ) + ( ( A + B ) \cdot D ) ) \\
by (rule MMI_axdistr)

have S2: ( ( A \in C \land B \in C ) \rightarrow ( A + B ) \in C ) by (rule MMI_axaddcl)

from S2 have S3: ( ( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \rightarrow \\

\[(A + B) \in C\] by (rule MMI_adantr)

have S4: \((C \in C \land D \in C) \rightarrow C \in C\) by (rule MMI_pm3_26)

from S4 have S5: \((A \in C \land B \in C) \land (C \in C \land D \in C)\) \rightarrow \[
C \in C\]

by (rule MMI_adantl)

have S6: \((C \in C \land D \in C) \rightarrow D \in C\) by (rule MMI_pm3_27)

from S6 have S7: \((A \in C \land B \in C) \land (C \in C \land D \in C)\) \rightarrow \[
D \in C\]

by (rule MMI_adantl)

from S1 S3 S5 S7 have S8:
\[
((A \in C \land B \in C) \land (C \in C \land D \in C)) \rightarrow \\
((A + B) \cdot (C + D)) = \\
((A + B) \cdot C) + ((A + B) \cdot D)
\]

by (rule MMI_syl3anc)

have S9: \((A \in C \land B \in C \land C \in C) \rightarrow \\
(A + B) \cdot C = ((A \cdot C) + (B \cdot C))
\]

by (rule MMI_adddirt)

from S9 have S10: \((A \in C \land B \in C) \land C \in C\) \rightarrow \[
(A + B) \cdot C = ((A \cdot C) + (B \cdot C))
\]

by (rule MMI_3expa)

from S10 have S11: \((A \in C \land B \in C) \land (C \in C \land D \in C)\) \rightarrow \[
(A + B) \cdot C = ((A \cdot C) + (B \cdot C))
\]

by (rule MMI_adantr)

from S11 S14 have S15: \((A \in C \land B \in C) \land (C \in C \land D \in C)\) \rightarrow \[
((A + B) \cdot C) + ((A + B) \cdot D) = \\
((A \cdot C) + (B \cdot C)) + ((A \cdot D) + (B \cdot D))
\]

by (rule MMI_opreq12d)

have S16:
\[
((A \cdot C) \in C \land (B \cdot C) \in C \land \\
(A \cdot D) + (B \cdot D)) \in C
\]

by (rule MMI_adddirt)

from S16 have S17: \((A \in C \land C \in C) \rightarrow (A \cdot C) \in C\) by (rule MMI_axmulcl)

from S17 have S18: \((A \in C \land B \in C) \land (C \in C \land D \in C)\) \rightarrow \[
(A \cdot C) \in C\]

by (rule MMI_ad2ant2r)

1471
have S19: ( B ∈ C ∧ C ∈ C ) → ( B · C ) ∈ C by (rule MMI_axmulcl)
from S19 have S20: ( B ∈ C ∧ ( C ∈ C ∧ D ∈ C ) ) →
( B · C ) ∈ C by (rule MMI_adantrr)
from S20 have S21: ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →
( B · C ) ∈ C by (rule MMI_adantll)
have S22: ( ( A · D ) ∈ C ∧ ( B · D ) ∈ C ) →
( ( A · D ) + ( B · D ) ) ∈ C by (rule MMI_axaddcl)
have S23: ( A ∈ C ∧ D ∈ C ) → ( A · D ) ∈ C by (rule MMI_axmulcl)
have S24: ( B ∈ C ∧ D ∈ C ) → ( B · C ) ∈ C by (rule MMI_axmulcl)
from S22 S23 S24 have S25:
( ( A ∈ C ∧ D ∈ C ) ∧ ( B ∈ C ∧ D ∈ C ) ) →
( ( A · D ) + ( B · D ) ) ∈ C by (rule MMI_syl2an)
from S25 have S26: ( ( A ∈ C ∧ B ∈ C ) ∧ D ∈ C ) →
( ( A · D ) + ( B · D ) ) ∈ C by (rule MMI_anandirs)
from S26 have S27: ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →
( ( A · D ) + ( B · D ) ) ∈ C by (rule MMI_anadrll)
from S16 S18 S21 S27 have S28:
( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →
(( ( A · C ) + ( B · C ) ) + ( ( A · D ) + ( B · D ) )) =
(( ( A · C ) + ( A · D ) ) + ( B · D ) ) + ( B · C )
by (rule MMI_syl3anc)
have S29: ( B ∈ C ∧ D ∈ C ) → ( B · D ) = ( D · B )
by (rule MMI_axmulcom)
from S29 have S30: ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →
( B · D ) = ( D · B ) by (rule MMI_ad2ant2l)
from S30 have S31: ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →
(( ( A · C ) + ( A · D ) ) + ( B · D ) ) =
(( ( A · C ) + ( A · D ) ) + ( D · B ) )
by (rule MMI_opreq2d)
have S32: ( ( A · C ) ∈ C ∧ ( A · D ) ∈ C ∧ ( B · D ) ∈ C ) →
( ( A · C ) + ( A · D ) ) + ( B · D ) ) =
( A · C ) + ( ( A · D ) + ( B · D ) )
by (rule MMI_axadddass)
from S18 have S33:
( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) → ( A · C ) ∈ C.
from S23 have S34: ( A ∈ C ∧ D ∈ C ) → ( A · D ) ∈ C.
from S34 have S35: ( A ∈ C ∧ ( C ∈ C ∧ D ∈ C ) ) →
( A · D ) ∈ C by (rule MMI_adantrl)
from S35 have S36: ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →
( A · D ) ∈ C by (rule MMI_adantll)
from S24 have S37: ( B ∈ C ∧ D ∈ C ) → ( B · D ) ∈ C.
from S37 have S38: ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →
( B · D ) ∈ C by (rule MMI_ad2ant2l)

1472
from S32 S33 S36 S38 have S39:
( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →
( ( ( A · C ) + ( A · D ) ) + ( B · D ) ) =
( ( ( A · C ) + ( ( A · D ) + ( B · D ) ) ) by (rule MMI_syl3anc)
have S40: ( ( ( A · C ) ∈ C ∧ ( A · D ) ∈ C ∧ ( D · B ) ∈ C ) ) →
( ( ( A · C ) + ( ( A · D ) ) + ( D · B ) ) ) =
( ( ( A · C ) + ( D · B ) ) + ( A · D ) ) by (rule MMI_add23t)
from S18 have S41:
( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) → ( A · C ) ∈ C.
from S36 have S42: ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →
( A · D ) ∈ C.
have S43: ( D ∈ C ∧ B ∈ C ) → ( D · B ) ∈ C by (rule MMI_axmulcl)
from S43 have S44: ( ( B ∈ C ∧ D ∈ C ) → ( D · B ) ∈ C
by (rule MMI_ancoms)
from S44 have S45: ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →
( D · B ) ∈ C by (rule MMI_ad2ant21)
from S40 S41 S42 S45 have S46:
( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →
( ( ( A · C ) + ( A · D ) ) + ( D · B ) ) =
( ( ( A · C ) + ( D · B ) ) + ( A · D ) ) by (rule MMI_syl3anc)
from S31 S39 S46 have S47:
( ( ( A · C ) + ( A · D ) + ( B · D ) ) ) =
( ( ( A · C ) + ( D · B ) ) + ( A · D ) ) by (rule MMI_3eqtr3d)
have S48: ( B ∈ C ∧ C ∈ C ) → ( B · C ) = ( C · B )
by (rule MMI_axmulcom)
from S48 have S49: ( ( A ∈ C ∧ D ∈ C ) ∧ ( B ∈ C ∧ C ∈ C ) ) →
( B · C ) = ( C · B ) by (rule MMI_adantl)
from S49 have S50: ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →
( B · C ) = ( C · B ) by (rule MMI_an42s)
from S47 S50 have S51: ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C )
) →
( ( ( A · C ) + ( A · D ) + ( B · D ) ) ) + ( B · C ) =
( ( ( A · C ) + ( D · B ) ) + ( A · D ) ) + ( C · B )
by (rule MMI_opreq12d)
have S52:
( ( ( A · C ) + ( D · B ) ) ∈ C ∧ ( A · D ) ∈ C ∧
( C · B ) ∈ C ) →
( ( ( A · C ) + ( D · B ) ) + ( A · D ) ) + ( C · B ) =
( ( ( A · C ) + ( D · B ) ) + ( ( A · D ) + ( C · B ) ) )
by (rule MMI_axaddass)
have S53: ( ( A · C ) ∈ C ∧ ( D · B ) ∈ C ) →
( ( A · C ) + ( D · B ) ) ∈ C by (rule MMI_axaddcl)
from S17 have S54: ( A ∈ C ∧ C ∈ C ) → ( A · C ) ∈ C.
from S44 have S55: ( B ∈ C ∧ D ∈ C ) → ( D · B ) ∈ C.

1473
from S53 S54 S55 have S56: 
( ( A ∈ C ∧ C ∈ C ) ∧ ( B ∈ C ∧ D ∈ C ) ) → 
( ( A · C ) + ( D · B ) ) ∈ C by (rule MMI_syl2an)
from S56 have S57: 
( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) → 
( ( A · C ) + ( D · B ) ) ∈ C by (rule MMI_an4s)
from S36 have S58: 
( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) → 
( A · D ) ∈ C.
have S59: 
( A ∈ C ∧ B ∈ C ) → ( C · B ) ∈ C by (rule MMI_axmulcl)
from S59 have S60: 
( A ∈ C ∧ B ∈ C ) → ( C · B ) ∈ C
by (rule MMI_ancoms)
from S60 have S61: 
( ( A ∈ C ∧ D ∈ C ) ∧ ( B ∈ C ∧ C ∈ C ) ) → 
( C · B ) ∈ C by (rule MMI_adantl)
from S61 have S62: 
( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) → 
( C · B ) ∈ C by (rule MMI_an42s)
from S52 S57 S58 S62 have S63: 
( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) → 
( ( ( ( A · C ) + ( D · B ) ) + ( A · D ) ) + ( C · B ) ) = 
( ( ( A · C ) + ( D · B ) ) + ( ( A · D ) + ( C · B ) ) )
by (rule MMI_syl3anc)
from S28 S51 S63 have S64: 
( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) → 
( ( ( A · C ) + ( B · C ) ) + ( ( A · D ) + ( B · D ) ) ) = 
( ( ( A · C ) + ( D · B ) ) + ( ( A · D ) + ( C · B ) ) )
by (rule MMI_3eqtrd)
from S8 S15 S64 show 
( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) → 
( A + B ) · ( C + D ) = 
( ( A · C ) + ( D · B ) ) + ( ( A · D ) + ( C · B ) )
by (rule MMI_3eqtrd)
qed

lemma (in MMIIsar0) MMI_muladd11t:
shows 
( A ∈ C ∧ B ∈ C ) → ( ( 1 + A ) · ( 1 + B ) ) = 
( ( 1 + A ) + ( B + ( A · B ) ) )
proof -
have S1: 
1 ∈ C by (rule MMI_1cn)
have S2: 
( ( 1 + A ) ∈ C ∧ 1 ∈ C ∧ B ∈ C ) → 
( ( 1 + A ) · ( 1 + B ) ) = 
( ( ( 1 + A ) · 1 ) + ( ( 1 + A ) · B ) )
by (rule MMI_axdistr)
from S1 S2 have S3: 
( ( 1 + A ) ∈ C ∧ B ∈ C ) → 
( ( 1 + A ) · ( 1 + B ) ) = 
( ( ( 1 + A ) · 1 ) + ( ( 1 + A ) · B ) )

1474
\begin{verbatim}
by (rule MMI_mp3an2)
have S4: \(1 \in C\) by (rule MMI_1cn)
have S5: \((1 \in C \land A \in C) \rightarrow (1 + A) \in C\) by (rule MMI_axaddcl)
from S4 S5 have S6: \(A \in C \rightarrow (1 + A) \in C\) by (rule MMI_mpan)
from S3 S6 have S7: \((A \in C \land B \in C) \rightarrow ((1 + A) \cdot 1) = (1 + A)\)
by (rule MMI_ax1id)
from S8 have S9: \((1 + A) \in C \rightarrow ((1 + A) \cdot 1) = (1 + A)\)
by (rule MMI_syl)
from S10 have S11: \((A \in C \land B \in C) \rightarrow ((1 + A) \cdot 1) = (1 + A)\)
by (rule MMI_adantr)
have S12: \(1 \in C\) by (rule MMI_1cn)
have S13: \((1 \in C \land A \in C \land B \in C) \rightarrow ((1 + A) \cdot B) = ((1 \cdot B) + (A \cdot B))\) by (rule MMI_adddirt)
from S12 S13 have S14: \((A \in C \land B \in C) \rightarrow ((1 + A) \cdot B) = ((1 \cdot B) + (A \cdot B))\)
from S15 have S16: \((A \in C \land B \in C) \rightarrow (1 \cdot B) = B\)
by (rule MMI_mulid2t)
have S17: \((A \in C \land B \in C) \rightarrow ((1 + A) \cdot 1) = (1 + A)\)
by (rule MMI_adantl)
from S18 have S19: \((A \in C \land B \in C) \rightarrow ((1 + A) \cdot 1) = (1 + A)\)
by (rule MMI_eqtrd)
qed

lemma (in MMIsar0) MMI_mul12: assumes A1: \(A \in C\) and
A2: \(B \in C\) and
A3: \(C \in C\)
shows \((A \cdot (B \cdot C)) = (B \cdot (A \cdot C))\)
proof -
from A1 have S1: \(A \in C\).
from A2 have S2: \(B \in C\).
from S1 S2 have S3: \((A \cdot B) = (B \cdot A)\) by (rule MMI_mulcom)
from S3 have S4: \(((A \cdot B) \cdot C) = ((B \cdot A) \cdot C)\)
by (rule MMI_opreq1i)
from A1 have S5: \(A \in C\).
from A2 have S6: \(B \in C\).
\end{verbatim}
from A3 have S7: \( C \in \mathbb{F} \).
from S5 S6 S7 have S8: \( (A \cdot B) \cdot C = (A \cdot (B \cdot C)) \)
    by (rule MMI_mulass)
from A2 have S9: \( B \in \mathbb{F} \).
from A1 have S10: \( A \in \mathbb{F} \).
from A3 have S11: \( C \in \mathbb{F} \).
from S9 S10 S11 have S12: \( (B \cdot A) \cdot C = (B \cdot (A \cdot C)) \)
    by (rule MMI_mulass)
from S4 S8 S12 show \( (A \cdot (B \cdot C)) = (B \cdot (A \cdot C)) \)
    by (rule MMI_3eqtr3)
qed

lemma (in MMIIsar0) MMI_mul23: assumes A1: \( A \in \mathbb{F} \) and
A2: \( B \in \mathbb{F} \) and
A3: \( C \in \mathbb{F} \)
shows \( (A \cdot B) \cdot C = (A \cdot C) \cdot B \)
proof -
from A1 have S1: \( A \in \mathbb{F} \).
from A2 have S2: \( B \in \mathbb{F} \).
from A3 have S3: \( C \in \mathbb{F} \).
    have S4: \( (A \in \mathbb{F} \land B \in \mathbb{F} \land C \in \mathbb{F}) \rightarrow ((A \cdot B) \cdot C) = ((A \cdot C) \cdot B) \)
        by (rule MMI_mul23t)
from S1 S2 S3 S4 show \( (A \cdot B) \cdot C = (A \cdot C) \cdot B \)
    by (rule MMI_mp3an)
qed

lemma (in MMIIsar0) MMI_mul4: assumes A1: \( A \in \mathbb{F} \) and
A2: \( B \in \mathbb{F} \) and
A3: \( C \in \mathbb{F} \) and
A4: \( D \in \mathbb{F} \)
shows \( (A \cdot B) \cdot (C \cdot D) = (A \cdot C) \cdot (B \cdot D) \)
proof -
from A1 have S1: \( A \in \mathbb{F} \).
from A2 have S2: \( B \in \mathbb{F} \).
from S1 S2 have S3: \( A \in \mathbb{F} \land B \in \mathbb{F} \) by (rule MMI_pm3_2i)
from A3 have S4: \( C \in \mathbb{F} \).
from A4 have S5: \( D \in \mathbb{F} \).
from S4 S5 have S6: \( C \in \mathbb{F} \land D \in \mathbb{F} \) by (rule MMI_pm3_2i)
    have S7: \( ((A \in \mathbb{F} \land B \in \mathbb{F}) \land (C \in \mathbb{F} \land D \in \mathbb{F})) \rightarrow ((A \cdot B) \cdot (C \cdot D)) = ((A \cdot C) \cdot (B \cdot D)) \)
        by (rule MMI_mul4t)
from S3 S6 S7 show \( (A \cdot B) \cdot (C \cdot D) = (A \cdot C) \cdot (B \cdot D) \)
    by (rule MMI_mp2an)
qed

lemma (in MMIIsar0) MMI_muladd: assumes A1: \( A \in \mathbb{F} \) and
A2: \( B \in \mathbb{F} \) and
A3: \( C \in \mathbb{F} \) and
A4: \( D \in C \)
shows \(( (A + B) \cdot (C + D) ) = ((A \cdot C) + (D \cdot B)) + ((A \cdot D) + (C \cdot B))\)

proof -
from A1 have S1: \( A \in C \).
from A2 have S2: \( B \in C \).
from S1 S2 have S3: \( A \in C \land B \in C \) by (rule MMI_pm3_2i)
from A3 have S4: \( C \in C \).
from A4 have S5: \( D \in C \).
from S4 S5 have S6: \( C \in C \land D \in C \) by (rule MMI_pm3_2i)
have S7: \(( (A \in C \land B \in C) \land (C \in C \land D \in C)) \rightarrow ((A + B) \cdot (C + D)) = ((A \cdot C) + (D \cdot B)) + ((A \cdot D) + (C \cdot B))\)
by (rule MMI_muladdt)
from S3 S6 S7 show \(( (A + B) \cdot (C + D)) = ((A \cdot C) + (D \cdot B)) + ((A \cdot D) + (C \cdot B))\)
by (rule MMI_mp2an)

qed

lemma (in MMI_isar0) MMI_subdit:
shows \(( A \in C \land B \in C \land C \in C) \rightarrow (A \cdot (B - C)) = ((A \cdot B) - (A \cdot C))\)

proof -
have S1: \((A \in C \land C \in C \land (B - C) \in C) \rightarrow (A \cdot (C + (B - C))) = ((A \cdot C) + (A \cdot (B - C)))\) by (rule MMI_axdistr)
have S2: \((A \in C \land B \in C \land C \in C) \rightarrow A \in C\) by (rule MMI_3simp1)
have S3: \((A \in C \land B \in C \land C \in C) \rightarrow C \in C\) by (rule MMI_3simp3)
have S4: \((B \in C \land C \in C) \rightarrow (B - C) \in C\) by (rule MMI_subclt)
from S4 have S5: \((A \in C \land B \in C \land C \in C) \rightarrow (B - C) \in C\) by (rule MMI_3adant1)
from S1 S2 S3 S5 have S6: \((A \in C \land B \in C \land C \in C) \rightarrow (A + (B - C)) = ((A \cdot C) + (A \cdot (B - C)))\) by (rule MMI_syl3anc)
have S7: \((C \in C \land B \in C) \rightarrow (C + (B - C)) = B\) by (rule MMI_pncan3t)
from S7 have S8: \((B \in C \land C \in C) \rightarrow (C + (B - C)) = B\) by (rule MMI_ancoms)
from S8 have S9: \((A \in C \land B \in C \land C \in C) \rightarrow (C + (B - C)) = B\) by (rule MMI_3adant1)
from S9 have S10: \((A \in C \land B \in C \land C \in C) \rightarrow (A + (B - C)) = (A \cdot B)\) by (rule MMI_opreq2d)
from S6 S10 have S11: \((A \in C \land B \in C \land C \in C) \rightarrow ((A \cdot C) + (A \cdot (B - C))) = (A \cdot B)\) by (rule MMI_eqtr3d)
have S12: \((A \in C \land A \in C \land A \in C) \land (A \cdot (B - C)) \in C\) \rightarrow \(((A \cdot B) - (A \cdot C)) = (A \cdot (B - C))\) by (rule MMI_subaddt)

qed
have S13: ( A ∈ C ∧ B ∈ C ) → ( A · B ) ∈ C by (rule MMI_axmulcl)

from S13 have S14: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) → ( A · B ) ∈ C by (rule MMI_3adant3)

have S15: ( A ∈ C ∧ C ∈ C ) → ( A · C ) ∈ C by (rule MMI_axmulcl)

from S15 have S16: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) → ( A · C ) ∈ C by (rule MMI_3adant2)

have S17: ( A ∈ C ∧ ( B · C ) ∈ C ) → ( A · ( B · C ) ) ∈ C by (rule MMI_axmulcl)

from S4 have S18: ( B ∈ C ∧ C ∈ C ) → ( B · C ) ∈ C.

from S17 S18 have S19: ( A ∈ C ∧ ( B ∈ C ∧ C ∈ C ) ) → ( ( A · ( B · C ) ) ∈ C by (rule MMI_sylan2)

from S19 have S20: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) → ( ( A · ( B · C ) ) ∈ C by (rule MMI_3impb)

from S12 S14 S16 S20 have S21: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) → (( ( A · B ) - ( A · C ) ) = ( ( A · B · C ) ) ) by (rule MMI_axmulcom)

from S11 S21 have S22: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) → (( ( A · B ) - ( A · C ) ) = ( A · ( B · C ) ) ) by (rule MMI_mpbird)

from S22 show ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) → (( A · ( B · C ) ) = ( ( A · B ) - ( A · C ) ) ) by (rule MMI_eqcomd)

qed

lemma (in MMIIsar0) MMI_subdirt:
  shows ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) → (( ( A - B ) · C ) = ( ( A · C ) - ( B · C ) ) )

proof -
  have S1: ( C ∈ C ∧ A ∈ C ∧ B ∈ C ) → ( C · ( A - B ) ) = ( ( C · A ) - ( C · B ) ) by (rule MMI_subdirt)

  from S1 have S2: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) → ( C · ( A - B ) ) = ( ( C · A ) - ( C · B ) ) by (rule MMI_3coml)

  have S3: ( ( A - B ) ∈ C ∧ C ∈ C ) → ( ( A - B ) · C ) = ( C · ( A - B ) ) by (rule MMI_axmulcom)

  have S4: ( A ∈ C ∧ B ∈ C ) → ( A - B ) ∈ C by (rule MMI_subclt)

  from S3 S4 have S5: ( ( A ∈ C ∧ B ∈ C ) ∧ C ∈ C ) → ( ( A - B ) · C ) = ( C · ( A - B ) ) by (rule MMI_sylan)

  from S5 have S6: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) → ( ( A - B ) · C ) = ( C · ( A - B ) ) by (rule MMI_3imp)

  have S7: ( A ∈ C ∧ C ∈ C ) → ( A · C ) = ( C · A ) by (rule MMI_axmulcom)

  from S7 have S8: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) → ( A · C ) = ( C · A )

  by (rule MMI_3adant2)

  have S9: ( B ∈ C ∧ C ∈ C ) → ( B · C ) = ( C · B ) by (rule MMI_axmulcom)

  from S9 have S10: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) → ( B · C ) = ( C · B )

  by (rule MMI_3adant1)

  from S8 S10 have S11: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) → (( ( A · C ) - ( B · C ) ) = ( ( C · A ) - ( C · B ) ) ) by (rule MMI_opreqid2)

  from S2 S6 S11 show ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
\[(A - B) \cdot C = (A \cdot C) - (B \cdot C)\] by (rule MMI_3eqtr4d)

**qed**

**lemma** (in MMI_{ar0}) **MMI_subdi**: assumes
\[A1: A \in C \text{ and} A2: B \in C \text{ and} A3: C \in C\]
shows
\[(A \cdot (B - C)) = ((A \cdot B) - (A \cdot C))\]

**proof** -
from \(A1\) have \(S1: A \in C\).
from \(A2\) have \(S2: B \in C\).
from \(A3\) have \(S3: C \in C\).
have \(S4: (A \in C \land B \in C \land C \in C) \rightarrow ((A \cdot (B - C)) = ((A \cdot B) - (A \cdot C))\) by (rule MMI_subdit)
from \(S1\) \(S2\) \(S3\) \(S4\) show \((A \cdot (B - C)) = ((A \cdot B) - (A \cdot C))\) by (rule MMI_mp3an)

**qed**

**lemma** (in MMI_{ar0}) **MMI_subdir**: assumes
\[A1: A \in C \text{ and} A2: B \in C \text{ and} A3: C \in C\]
shows
\[(A - B) \cdot C = (A \cdot C) - (B \cdot C)\]

**proof** -
from \(A1\) have \(S1: A \in C\).
from \(A2\) have \(S2: B \in C\).
from \(A3\) have \(S3: C \in C\).
have \(S4: (A \in C \land B \in C \land C \in C) \rightarrow ((A - B) \cdot C) = (A \cdot C) - (B \cdot C)\) by (rule MMI_subdirt)
from \(S1\) \(S2\) \(S3\) \(S4\) show \((A - B) \cdot C) = (A \cdot C) - (B \cdot C)\) by (rule MMI_mp3an)

**qed**

**lemma** (in MMI_{ar0}) **MMI_mul01**: assumes \(A1: A \in C\)
shows
\[A \cdot 0 = 0\]

**proof** -
from \(A1\) have \(S1: A \in C\).
have \(S2: 0 \in C\) by (rule MMI_0cn)
have \(S3: 0 \in C\) by (rule MMI_0cn)
from \(S1\) \(S2\) \(S3\) have \(S4: (A \cdot (0 - 0)) = ((A \cdot 0) - (A \cdot 0))\) by (rule MMI_subdi)
have \(S5: 0 \in C\) by (rule MMI_0cn)
from \(S5\) have \(S6: (0 - 0) = 0\) by (rule MMI_subid)
from \(S6\) have \(S7: (A \cdot (0 - 0)) = (A \cdot 0)\) by (rule MMI_opreq2i)
from \(A1\) have \(S8: A \in C\).
have \(S9: 0 \in C\) by (rule MMI_0cn)
from \(S8\) \(S9\) have \(S10: (A \cdot 0) \in C\) by (rule MMI_mulcl)
from \(S10\) have \(S11: ((A \cdot 0) - (A \cdot 0)) = 0\) by (rule MMI_subid)

1479
from $S_4$ $S_7$ $S_{11}$ show $(A \cdot 0) = 0$ by (rule MMI_3eqtr3)

qed

lemma (in MMIsar0) MMI_mul02: assumes $A_1: A \in \mathbb{C}$
shows $(0 \cdot A) = 0$

proof -
  have $S_1$: $0 \in \mathbb{C}$ by (rule MMI_0cn)
  from $A_1$ have $S_2$: $A \in \mathbb{C}$.
  from $S_1$ $S_2$ have $S_3$: $(0 \cdot A) = (A \cdot 0)$ by (rule MMI_mulcom)
  from $A_1$ have $S_4$: $A \in \mathbb{C}$.
  from $S_4$ have $S_5$: $(A \cdot 0) = 0$ by (rule MMI_mul01)
  from $S_3$ $S_5$ show $(0 \cdot A) = 0$ by (rule MMI_eqtr)

qed

lemma (in MMIsar0) MMI_1p1times: assumes $A_1: A \in \mathbb{C}$
shows $((1 + 1) \cdot A) = (A + A)$

proof -
  have $S_1$: $1 \in \mathbb{C}$ by (rule MMI_1cn)
  have $S_2$: $1 \in \mathbb{C}$ by (rule MMI_1cn)
  from $A_1$ have $S_3$: $A \in \mathbb{C}$.
  from $S_1$ $S_2$ $S_3$ have $S_4$: $((1 + 1) \cdot A) = ((1 \cdot 1) \cdot A) = ((1 \cdot 1) + (1 \cdot 1) A)$
    by (rule MMI_adddir)
  from $A_1$ have $S_5$: $A \in \mathbb{C}$.
  from $S_5$ have $S_6$: $(1 \cdot A) = A$ by (rule MMI_mulid2)
  from $S_6$ have $S_7$: $(1 \cdot A) = A$.
  from $S_6$ $S_7$ have $S_8$: $((1 \cdot A) + (1 \cdot A)) = (A + A)$
    by (rule MMI_opreq12i)
  from $S_4$ $S_8$ show $((1 + 1) \cdot A) = (A + A)$
    by (rule MMI_eqtr)

qed

lemma (in MMIsar0) MMI_mul01t:
shows $A \in \mathbb{C} \rightarrow (A \cdot 0) = 0$

proof -
  have $S_1$: $A = \text{if} (A \in \mathbb{C}, A, 0) \rightarrow$
    $(A \cdot 0) = (\text{if} (A \in \mathbb{C}, A, 0) \cdot 0)$ by (rule MMI_opreq1)
  from $S_1$ have $S_2$: $A = \text{if} (A \in \mathbb{C}, A, 0) \rightarrow$
    $(A \cdot 0) = 0 \leftrightarrow (\text{if} (A \in \mathbb{C}, A, 0) \cdot 0) = 0$ by (rule MMI_eqeq1d)
  have $S_3$: $0 \in \mathbb{C}$ by (rule MMI_0cn)
  from $S_3$ have $S_4$: $\text{if} (A \in \mathbb{C}, A, 0) \in \mathbb{C}$ by (rule MMI_elime)
  from $S_4$ have $S_5$: $\text{if} (A \in \mathbb{C}, A, 0) \cdot 0 = 0$ by (rule MMI_mul01)
  from $S_2$ $S_5$ show $A \in \mathbb{C} \rightarrow (A \cdot 0) = 0$ by (rule MMI_dedth)

qed

lemma (in MMIsar0) MMI_mul02t:
shows $A \in \mathbb{C} \rightarrow (0 \cdot A) = 0$

proof -
  have $S_1$: $0 \in \mathbb{C}$ by (rule MMI_0cn)

1480
have \( S2: (0 \in \mathcal{F} \land A \in \mathcal{F}) \rightarrow (0 \cdot A) = (A \cdot 0) \) by (rule MMI_axmulcom)
from \( S1, S2 \) have \( S3: A \in \mathcal{F} \rightarrow (0 \cdot A) = (A \cdot 0) \) by (rule MMI_mpan)
have \( S4: A \in \mathcal{F} \rightarrow (A \cdot 0) = 0 \) by (rule MMI_mul01t)
from \( S3, S4 \) show \( A \in \mathcal{F} \rightarrow (0 \cdot A) = 0 \) by (rule MMI_eqtrd)
qed

lemma (in MMIar0) MMI_mulneg1: assumes \( A1: A \in \mathcal{F} \) and \( A2: B \in \mathcal{C} \)
shows \( ((-A) \cdot B) = (- (A \cdot B)) \)
proof -
from \( A2 \) have \( S1: B \in \mathcal{C}. \)
from \( S1 \) have \( S2: (B \cdot 0) = 0 \) by (rule MMI_mul01)
from \( A2 \) have \( S3: B \in \mathcal{C}. \)
from \( A1 \) have \( S4: A \in \mathcal{C}. \)
from \( S3, S4 \) have \( S5: (B \cdot A) = (A \cdot B) \) by (rule MMI_mulcom)
from \( S2, S5 \) have \( S6: ((B \cdot 0) - (B \cdot A)) = (0 - (A \cdot B)) \)
by (rule MMI_opreq12i)
have \( S7: ((-A)) = (0 - A) \) by (rule MMI_df_neg)
from \( S7 \) have \( S8: ((-A) \cdot B) = ((0 - A) \cdot B) \)
by (rule MMI_opreq1i)
from \( S9: 0 \in \mathcal{C} \) by (rule MMI_0cn)
from \( A1 \) have \( S10: A \in \mathcal{C}. \)
from \( S9, S10 \) have \( S11: (0 - A) \in \mathcal{C} \) by (rule MMI_subcl)
from \( A2 \) have \( S12: B \in \mathcal{C}. \)
from \( S11, S12 \) have \( S13: ((0 - A) \cdot B) = (B \cdot (0 - A)) \)
by (rule MMI_mulcom)
from \( A2 \) have \( S14: B \in \mathcal{C}. \)
from \( A1 \) have \( S16: A \in \mathcal{C}. \)
from \( S14, S15 \) have \( S17: (B \cdot (0 - A)) = ((B \cdot 0) - (B \cdot A)) \)
by (rule MMI_subdi)
from \( S8, S13 \) have \( S18: ((-A) \cdot B) = ((B \cdot 0) - (B \cdot A)) \) by (rule MMI_3eqtr)
have \( S19: (- (A \cdot B)) = (0 - (A \cdot B)) \) by (rule MMI_df_neg)
from \( S6, S18, S19 \) show \( ((-A) \cdot B) = (- (A \cdot B)) \)
by (rule MMI_3eqtr4)
qed

lemma (in MMIar0) MMI_mulneg2: assumes \( A1: A \in \mathcal{C} \) and \( A2: B \in \mathcal{C} \)
shows \( (A \cdot ((- B))) = \)
\( (- (A \cdot B)) \)
proof -
from \( A1 \) have \( S1: A \in \mathcal{C}. \)
from \( A2 \) have \( S2: B \in \mathcal{C}. \)
from \( S2 \) have \( S3: ((-B)) \in \mathcal{C} \) by (rule MMI_negcl)
from S1 S3 have S4: \((A \cdot ( (- B) ) ) = ( (- B) \cdot A )\) by (rule MMI_mulcom)
from A2 have S5: \(B \in C\).
from A1 have S6: \(A \in C\).
from S5 S6 have S7: \((- (A \cdot ( (- B) ) ) = ( (- B) \cdot A )\)
( - ( A · B ) ) by (rule MMI_mulneg1)
from A2 have S8: \(B \in C\).
from A1 have S9: \(A \in C\).
from S8 S9 have S10: \((B \cdot A) = (A \cdot B)\) by (rule MMI_mulcom)
from S10 have S11: \(- (B \cdot A) = (A \cdot B)\)
( - ( A · B ) ) by (rule MMI_negeqi)
from S4 S7 S11 have \((A \cdot ( (- B) ) = ( (- B) \cdot A)\)
( - ( A · B ) ) by (rule MMI_3eqtr)
qed

lemma (in MMIIsar0) MMI_mul2neg: assumes A1: \(A \in C\) and A2: \(B \in C\)
says \((((-A) \cdot ( (- B) ))) = (A \cdot B)\)
proof -
from A1 have S1: \(A \in C\).
from A2 have S2: \(B \in C\).
from S2 have S3: \((-B) \in C\) by (rule MMI_negcl)
from S1 S3 have S4: \(( ( (- A) ) \cdot ( (- B) ) ) = ( - ( A \cdot ( (- B) ) ) )\)
( - ( A · B ) ) by (rule MMI_mulneg1)
from A1 have S5: \(A \in C\).
from S3 have S6: \((-B) \in C\).
from S5 S6 have S7: \((A \cdot ( (- B) )) = ( ( (- B) ) \cdot A)\)
( - ( A · B ) ) by (rule MMI_mulcom)
from A2 have S8: \(B \in C\).
from A1 have S9: \(A \in C\).
from S8 S9 have S10: \(( ( (- B) ) \cdot A ) = ( - (B \cdot A) )\)
( - ( B · A ) ) by (rule MMI_mulneg1)
from S7 S10 have S11: \((A \cdot (I (- B))) = ( - (B \cdot A) )\)
( - ( A · B ) ) by (rule MMI_eqtr)
from S11 have S12: \(( - (B \cdot A) ) = ( (B \cdot A))\)
( - ( A · B ) ) by (rule MMI_negneg)
from A2 have S13: \(B \in C\).
from A1 have S14: \(A \in C\).
from S13 S14 have S15: \(( B \cdot A) \in C\) by (rule MMI_mulcl)
from S15 have S16: \((- (B \cdot A))\)
( B · A ) by (rule MMI_negneg)
from S4 S12 S16 have S17: \((- (A) \cdot ( (- B))) = (B \cdot A)\)
( B · A ) by (rule MMI_3eqtr)
from A2 have S18: \(B \in C\).
from A1 have S19: \(A \in C\).
from S18 S19 have S20: \((B \cdot A) = (A \cdot B)\) by (rule MMI_mulcom)
from S17 S20 have \((- (A) \cdot ( (- B))) = (A \cdot B)\)
( A · B ) by (rule MMI_eqtr)
lemma (in MMIsar0) MMI_negdi: assumes A1: A ∈ C and
A2: B ∈ C
shows ( - ( A + B ) ) =
( ( ( - A ) ) + ( ( - B ) ) )
proof -
  from A1 have S1: A ∈ C.
  from A2 have S2: B ∈ C.
  from S1 S2 have S3: ( A + B ) ∈ C by (rule MMI_addcl)
  from S3 have S4: ( 1 · ( A + B ) ) =
( A + B ) by (rule MMI_mulid2)
  from S4 have S5: ( - ( 1 · ( A + B ) ) ) =
( - ( A + B ) ) by (rule MMI_negeqi)
  have S6: 1 ∈ C by (rule MMI_1cn)
  from S6 have S7: ( - 1 ) ∈ C by (rule MMI_negcl)
  from A1 have S8: A ∈ C.
  from A2 have S9: B ∈ C.
  from S7 S8 S9 have S10: ( ( - 1 ) · ( A + B ) ) =
( ( - 1 ) · ( A + B ) ) by (rule MMI_adddi)
  from S3 have S11: 1 ∈ C by (rule MMI_1cn)
  from S11 S12 have S13: ( ( - 1 ) · ( A + B ) ) =
( - ( 1 · ( A + B ) ) ) by (rule MMI_mulneg1)
  have S14: 1 ∈ C by (rule MMI_1cn)
  from S14 S15 have S16: ( ( - 1 ) · A ) =
( - ( 1 · A ) ) by (rule MMI_mulneg1)
  from A1 have S17: A ∈ C.
  from S17 have S18: ( 1 · A ) = A by (rule MMI_mulid2)
  from S18 have S19: ( - ( 1 · A ) ) = ( ( - A ) ) by (rule MMI_negeqi)
  from S16 S19 have S20: ( ( - 1 ) · A ) = ( ( - A ) ) by (rule MMI_eqtr)
  from S20 have S21: 1 ∈ C by (rule MMI_1cn)
  from A2 have S22: B ∈ C.
  from S21 S22 have S23: ( ( - 1 ) · B ) =
( - ( 1 · B ) ) by (rule MMI_mulneg1)
  from A2 have S24: B ∈ C.
  from S24 have S25: ( 1 · B ) = B by (rule MMI_mulid2)
  from S25 have S26: ( - ( 1 · B ) ) = ( ( - B ) ) by (rule MMI_negeqi)
  from S23 S26 have S27: ( ( - 1 ) · B ) = ( ( - B ) ) by (rule MMI_eqtr)
  from S20 S27 have S28: ( ( ( - 1 ) · A ) + ( ( - 1 ) · B ) ) =
( ( ( - A ) ) + ( ( - B ) ) ) by (rule MMI_opreq12i)
  from S10 S13 S28 have S29: ( - ( 1 · ( A + B ) ) ) =
( ( ( - A ) ) + ( ( - B ) ) ) by (rule MMI_3eqtr3)
  from S5 S29 show ( - ( A + B ) ) =
( ( ( - A ) ) + ( ( - B ) ) ) by (rule MMI_eqtr3)
A2: B ∈ C
shows ( - ( A - B ) ) =
( ( - A ) + B )
proof -
from A1 have S1: A ∈ C.
from A2 have S2: B ∈ C.
from S2 have S3: ( - ( B ) ) ∈ C by (rule MMI_negcl)
from S1 S3 have S4: ( ( A + ( - B ) ) ) =
( ( - A ) ) + ( - ( - B ) ) ) by (rule MMI_negdi)
from A1 have S5: A ∈ C.
from A2 have S6: B ∈ C.
from S5 S6 have S7: ( A + ( - B ) ) = ( A - B ) by (rule MMI_negsub)
from S7 have S8: ( - ( A + ( - B ) ) ) =
( - ( A - B ) ) by (rule MMI_negeqi)
from A2 have S9: B ∈ C.
from S9 have S10: ( - ( - B ) ) = B by (rule MMI_negneg)
from S10 have S11: ( ( - A ) ) =
( ( - A ) ) + B ) by (rule MMI_opreq2i)
from S10 S8 S11 show ( - ( A - B ) ) =
( ( - A ) ) + B ) by (rule MMI_3eqtr3)
qed

lemma (in MMI_isar0) MMI_negsubdi2: assumes A1: A ∈ C and
A2: B ∈ C
shows ( - ( A - B ) ) = ( B - A )
proof -
from A1 have S1: A ∈ C.
from A2 have S2: B ∈ C.
from S1 S2 have S3: ( - ( A - B ) ) =
( ( - A ) ) + B ) by (rule MMI_negsubdi)
from A1 have S4: A ∈ C.
from S4 have S5: ( - ( A ) ) ∈ C by (rule MMI_negcl)
from A2 have S6: B ∈ C.
from S5 S6 have S7: ( ( - A ) ) + B ) =
( B + ( - A ) ) ) by (rule MMI_addcom)
from A2 have S8: B ∈ C.
from A1 have S9: A ∈ C.
from S8 S9 have S10: ( B + ( - A ) ) ) = ( B - A ) by (rule MMI_negsub)
from S3 S7 S10 show ( - ( A - B ) ) = ( B - A ) by (rule MMI_3eqtr)
qed

lemma (in MMI_isar0) MMI_mulneg1t:
shows ( A ∈ C ∧ B ∈ C ) →
( ( - A ) ) · B ) =
( - ( A · B ) )
proof -
have S1: A =
if ( A ∈ C , A , 0 ) →
( ( - A ) ) =

1484
\[
( - \text{if} ( A \in C, A, 0 ) ) \text{ by (rule MMI_negeq)}
\]
from S1 have S2: \( A = \)
if ( \( A \in C, A, 0 \) ) \rightarrow
\[
( ( - A ) ) \cdot B ) =
\]
( ( - \text{if} ( A \in C, A, 0 ) ) \cdot B ) \text{ by (rule MMI_opreq1d)}
have S3: \( A = \)
if ( \( A \in C, A, 0 \) ) \rightarrow
( A \cdot B ) =
( ( A \in C, A, 0 ) \cdot B ) \text{ by (rule MMI_opreq1)}
from S3 have S4: \( A = \)
if ( \( A \in C, A, 0 \) ) \rightarrow
( A \cdot B ) =
( ( - A ) ) \cdot B ) =
( ( - \text{if} ( A \in C, A, 0 ) ) \cdot B ) =
( ( - \text{if} ( A \in C, A, 0 ) ) \cdot B ) \text{ by (rule MMI_eqeq12d)}
have S6: \( B = \)
if ( \( B \in C, B, 0 \) ) \rightarrow
( ( - \text{if} ( A \in C, A, 0 ) ) \cdot B ) =
( - ( - A ) ) \cdot B ) =
( ( - \text{if} ( A \in C, A, 0 ) ) \cdot if ( B \in C, B, 0 ) ) \text{ by (rule MMI_opreq2)}
from S2 S4 have S5: \( A = \)
if ( \( A \in C, A, 0 \) ) \rightarrow
( - ( A \cdot B ) ) =
( ( - A ) ) \cdot B ) =
( ( - \text{if} ( A \in C, A, 0 ) ) \cdot if ( B \in C, B, 0 ) ) \text{ by (rule MMI_negeqd)}
have S7: \( B = \)
if ( \( B \in C, B, 0 \) ) \rightarrow
( ( - \text{if} ( A \in C, A, 0 ) ) \cdot B ) =
( ( \text{if} ( A \in C, A, 0 ) \cdot if ( B \in C, B, 0 ) ) \text{ by (rule MMI_opreq2)}
from S7 have S8: \( B = \)
if ( \( B \in C, B, 0 \) ) \rightarrow
( - ( - A ) ) \cdot B ) =
( - ( A \cdot B ) ) =
( - ( A \cdot B ) ) \text{ by (rule MMI_negeqd)}
from S6 S8 have S9: \( B = \)
if ( \( B \in C, B, 0 \) ) \rightarrow
( ( - A ) ) \cdot B ) =
( ( - A ) ) \cdot B ) =
( ( - \text{if} ( A \in C, A, 0 ) ) \cdot if ( B \in C, B, 0 ) ) \text{ by (rule MMI_eqeq12d)}
have S10: \( 0 \in C \) by (rule MMI_0cn)
from S10 have S11: \( if ( A \in C, A, 0 ) \in C \) by (rule MMI_slime1)
have S12: \( 0 \in C \) by (rule MMI_0cn)
from S12 have S13: \( if ( B \in C, B, 0 ) \in C \) by (rule MMI_slime1)
from S11 S13 have S14: \( ( - \text{if} ( A \in C, A, 0 ) ) \cdot if ( B \in C, B, 0 ) ) \text{ by (rule MMI_mulneg1)}
from S5 S9 S14 show \( ( A \in C \land B \in C ) \rightarrow
( ( - A ) ) \cdot B ) =
( - ( A \cdot B ) ) \text{ by (rule MMI_dedth2h)}
qed
lemma (in MMIsar0) MMI_mulneg2t:
  shows ( A ∈ C ∧ B ∈ C ) −→
  ( A · ( (− B ) ) ) =
  ( − ( A · B ) )
proof -
  have S1: ( B ∈ C ∧ A ∈ C ) −→
  ( (− B ) · A ) =
  ( − ( B · A ) ) by (rule MMI_mulneg1t)
  from S1 have S2: ( A ∈ C ∧ B ∈ C ) −→
  ( (− B ) · A ) =
  ( − ( B · A ) ) by (rule MMI_ancoms)
  have S3: ( A ∈ C ∧ (− B ) ∈ C ) −→
  ( A · ( (− B ) ) ) =
  ( (− B ) · A ) by (rule MMI_axmulcom)
  have S4: B ∈ C −→ (− B ) ∈ C by (rule MMI_negclt)
  from S3 S4 have S5: ( A ∈ C ∧ B ∈ C ) −→
  ( A · ( (− B ) ) ) =
  ( (− B ) · A ) by (rule MMI_sylan2)
  have S6: ( A ∈ C ∧ B ∈ C ) −→
  ( A · B ) = ( B · A ) by (rule MMI_axmulcom)
  from S6 have S7: ( A ∈ C ∧ B ∈ C ) −→
  ( A · ( (− B ) ) ) =
  ( − ( A · B ) ) by (rule MMI_negeqd)
  from S2 S4 S5 S7 show ( A ∈ C ∧ B ∈ C ) −→
  ( A · ( (− B ) ) ) =
  ( − ( A · B ) ) by (rule MMI_3eqtr4d)
qed

lemma (in MMIsar0) MMI_mulneg12t:
  shows ( A ∈ C ∧ B ∈ C ) −→
  ( (− A ) · B ) =
  ( A · ( (− B ) ) )
proof -
  have S1: ( A ∈ C ∧ B ∈ C ) −→
  ( (− A ) · B ) =
  ( − ( A · B ) ) by (rule MMI_mulneg1t)
  have S2: ( A ∈ C ∧ B ∈ C ) −→
  ( A · ( (− B ) ) ) =
  ( − ( A · B ) ) by (rule MMI_mulneg2t)
  from S1 S2 show ( A ∈ C ∧ B ∈ C ) −→
  ( (− A ) · B ) =
  ( − ( A · B ) ) by (rule MMI_eqtr4d)
qed

lemma (in MMIsar0) MMI_mul2negt:
  shows ( A ∈ C ∧ B ∈ C ) −→
  ( (− A ) · (− B ) ) =
  ( A · B )
proof -
have S1: \( A = \)
if \(( A \in \mathcal{F}, A, 0)\) \(\rightarrow\)
\((- A) = \)
\((- if ( A \in \mathcal{F}, A, 0))\) by (rule MMI_negeq)
from S1 have S2: \( A = \)
if \(( A \in \mathcal{F}, A, 0)\) \(\rightarrow\)
\((((- A)) \cdot ((- B))) = \)
\((if ( A \in \mathcal{F}, A, 0)) \cdot ((- B))\) by (rule MMI_opreq1d)
have S3: \( A = \)
if \(( A \in \mathcal{F}, A, 0)\) \(\rightarrow\)
\((A \cdot B) = \)
\((if ( A \in \mathcal{F}, A, 0) \cdot B))\) by (rule MMI_opreq1)
from S2 S3 have S4: \( A = \)
if \(( A \in \mathcal{F}, A, 0)\) \(\rightarrow\)
\((( (- A)) \cdot ((- B))) = \)
\((if ( A \in \mathcal{F}, A, 0) \cdot B))\) by (rule MMI_eqeq12d)
have S5: \( B = \)
if \(( B \in \mathcal{F}, B, 0)\) \(\rightarrow\)
\((- B) = \)
\((- if ( B \in \mathcal{F}, B, 0))\) by (rule MMI_negeq)
from S5 have S6: \( B = \)
if \(( B \in \mathcal{F}, B, 0)\) \(\rightarrow\)
\((( (- A)) \cdot ((- B))) = \)
\((( (- if ( A \in \mathcal{F}, A, 0)) \cdot ((- B))))\) by (rule MMI_opreq2d)
have S7: \( B = \)
if \(( B \in \mathcal{F}, B, 0)\) \(\rightarrow\)
\((A \cdot B) = \)
\((if ( A \in \mathcal{F}, A, 0) \cdot B))\) by (rule MMI_opreq2)
from S6 S7 have S8: \( B = \)
if \(( B \in \mathcal{F}, B, 0)\) \(\rightarrow\)
\((( (- if ( A \in \mathcal{F}, A, 0)) \cdot ((- B))) = \)
\((( (- if ( A \in \mathcal{F}, A, 0)) \cdot ((- if ( B \in \mathcal{F}, B, 0))))\) by (rule MMI_eqeq12d)
have S9: \( 0 \in \mathcal{F} \) by (rule MMI_0cn)
from S9 have S10: \( if ( A \in \mathcal{F}, A, 0)) \in \mathcal{F} \) by (rule MMI_elimel)
have S11: \( 0 \in \mathcal{F} \) by (rule MMI_0cn)
from S11 have S12: \( if ( B \in \mathcal{F}, B, 0)) \in \mathcal{F} \) by (rule MMI_elimel)
from S10 S12 have S13: \( (( (- if ( A \in \mathcal{F}, A, 0)) \cdot ((- if ( B \in \mathcal{F}, B, 0)))\) = \)
\((if ( A \in \mathcal{F}, A, 0)) \cdot if ( B \in \mathcal{F}, B, 0)))\) by (rule MMI_mul2neg)
from S4 S8 S13 show \(( A \in \mathcal{F} \land B \in \mathcal{F} \) \(\rightarrow\)
\((((- A)) \cdot ((- B))) = \)
\((A \cdot B))\) by (rule MMI_dedth2h)
qed
lemma (in MMIsar0) MMI_negdit:
  shows \((A \in C \land B \in C) \rightarrow\)
  \((- (A + B)) =\)
  \((- (-A)) + (- B))\)

proof -
  have S1: A =
  if (A ∈ C, A, 0) →
  (A + B) =
  (if (A ∈ C, A, 0) + B) by (rule MMI_opreq1)
  from S1 have S2: A =
  if (A ∈ C, A, 0) →
  (- (A + B)) =
  (- (if (A ∈ C, A, 0) + B)) by (rule MMI_negeqd)
  have S3: A =
  if (A ∈ C, A, 0) →
  (-A) =
  (- if (A ∈ C, A, 0)) by (rule MMI_neq)
  from S3 have S4: A =
  if (A ∈ C, A, 0) →
  (- (-A)) + (- B)) =
  (- if (A ∈ C, A, 0)) + (- B)) by (rule MMI_opreq1d)
  from S2 S4 have S5: A =
  if (A ∈ C, A, 0) →
  (- (if (A ∈ C, A, 0) + B)) =
  (if (A ∈ C, A, 0)) + (- B)) by (rule MMI_eqeq1d)
  have S6: B =
  if (B ∈ C, B, 0) →
  (if (A ∈ C, A, 0) + B) =
  (if (A ∈ C, A, 0) + if (B ∈ C, B, 0)) by (rule MMI_opreq2)
  from S6 have S7: B =
  if (B ∈ C, B, 0) →
  (- (if (A ∈ C, A, 0) + B)) =
  (- if (A ∈ C, A, 0) + if (B ∈ C, B, 0)) by (rule MMI_negeqd)
  have S8: B =
  if (B ∈ C, B, 0) →
  (-B) =
  (- if (B ∈ C, B, 0)) by (rule MMI_neq)
  from S8 have S9: B =
  if (B ∈ C, B, 0) →
  (- if (A ∈ C, A, 0)) + (- B)) =
  (- if (A ∈ C, A, 0) + (- if (B ∈ C, B, 0))) by (rule MMI_opreq2d)
  from S7 S9 have S10: B =
  if (B ∈ C, B, 0) →
  (- (if (A ∈ C, A, 0) + B)) =
  (- if (A ∈ C, A, 0)) + (- B)) =
  (if (A ∈ C, A, 0) + if (B ∈ C, B, 0)) =

1488
( ( - if ( A ∈ C , A , 0 ) ) + ( - if ( B ∈ C , B , 0 ) ) ) by (rule MMI_eqeq12d)

have S11: 0 ∈ C by (rule MMI_0cn)
from S11 have S12: if ( A ∈ C , A , 0 ) ∈ C by (rule MMI_elimel)
have S13: 0 ∈ C by (rule MMI_0cn)
from S13 have S14: if ( B ∈ C , B , 0 ) ∈ C by (rule MMI_elimel)
from S12 S14 have S15: ( - ( if ( A ∈ C , A , 0 ) + if ( B ∈ C , B , 0 ) ) ) =
( ( - if ( A ∈ C , A , 0 ) ) + ( - if ( B ∈ C , B , 0 ) ) ) by (rule MMI_negdi)
from S5 S10 S15 show ( A ∈ C ∧ B ∈ C ) →
( ( - A ) + ( - B ) ) =
by (rule MMI_negdi)

qed

lemma (in MMIsar0) MMI_negdi2t:
shows ( A ∈ C ∧ B ∈ C ) →
( - ( A + B ) ) = ( ( - A ) - B )
proof -
have S1: ( A ∈ C ∧ B ∈ C ) →
( ( - A ) + ( - B ) ) =
( ( - A ) ) + ( ( - B ) ) by (rule MMI_negdit)
have S2: ( ( ( - A ) ) ∈ C ∧ B ∈ C ) →
( ( ( - A ) ) + ( ( - B ) ) ) =
( ( - A ) ) + B ) by (rule MMI_negsubt)
have S3: A ∈ C → ( ( - A ) ) ∈ C by (rule MMI_negclt)
from S2 S3 have S4: ( A ∈ C ∧ B ∈ C ) →
( ( - A ) ) + ( ( - B ) ) ) =
( ( ( - A ) ) - B ) by (rule MMI_sylan)
from S1 S4 show ( A ∈ C ∧ B ∈ C ) →
( - ( A + B ) ) = ( ( ( - A ) ) - B )
by (rule MMI_eqtrd)

qed

lemma (in MMIsar0) MMI_negsubdit:
shows ( A ∈ C ∧ B ∈ C ) →
( ( - A - B ) ) = ( ( ( - A ) ) + B )
proof -
have S1: ( A ∈ C ∧ ( ( - B ) ) ∈ C ) →
( ( ( - A ) ) + ( ( - B ) ) ) ) =
( ( ( - A ) ) + ( ( ( - B ) ) ) ) by (rule MMI_negdit)
have S2: B ∈ C → ( ( - B ) ) ∈ C by (rule MMI_negclt)
from S1 S2 have S3: ( A ∈ C ∧ B ∈ C ) →
( ( ( ( A + ( ( - B ) ) ) ) ) =
( ( ( ( A ) ) + ( ( - ( B ) ) )) ) by (rule MMI_sylan2)
have S4: ( A ∈ C ∧ B ∈ C ) →
( A + ( ( ( - B ) ) ) ) = ( A - B ) by (rule MMI_negsubt)
from S4 have S5: \((A \in \mathcal{F} \land B \in \mathcal{C}) \rightarrow \)
\((- (A + (-(B)))) = \)
\((- (A - B))\) by (rule MMI_negeqd)

have S6: \(B \in \mathcal{C} \rightarrow (- (-(B))) = B\) by (rule MMI_negnegt)

from S6 have S7: \((A \in \mathcal{F} \land B \in \mathcal{C}) \rightarrow (- (-(B))) = B\)
by (rule MMI_adantl)

from S7 have S8: \((A \in \mathcal{F} \land B \in \mathcal{C}) \rightarrow (- (-(B))) = B\)
by (rule MMI_adantl)

from S3 S5 S8 show \((A \in \mathcal{F} \land B \in \mathcal{C}) \rightarrow \(- (A - B)\) = \((- (A) + B)\)
by (rule MMI_3eqtr3d)

qed

lemma (in MMI) MMI_negsubdi2t:
shows \((A \in \mathcal{F} \land B \in \mathcal{C}) \rightarrow \(- (A - B)\) = \((- (A) + B)\)
proof -

have S1: \((A \in \mathcal{F} \land B \in \mathcal{C}) \rightarrow \(- (A - B)\) = \((- (A) + B)\)
by (rule MMI_negsubdit)

have S2: \((A \in \mathcal{F} \land B \in \mathcal{C}) \rightarrow \((- (A) + B)\) = \((B + (-(A)))\)
by (rule MMI_axaddcom)

have S3: \((A \in \mathcal{F} \land B \in \mathcal{C}) \rightarrow (-(B)) \in \mathcal{C}\)
by (rule MMI_negclt)

from S2 S3 have S4: \((A \in \mathcal{F} \land B \in \mathcal{C}) \rightarrow \((- (A) + B)\) = \((B + (-(A)))\)
by (rule MMI_sylan)

have S5: \((B \in \mathcal{F} \land A \in \mathcal{C}) \rightarrow \((- (A) + B)\) = \((B - A)\)
by (rule MMI_3eqtr3d)

qed

lemma (in MMI) MMI_subsub2t:
shows \((A \in \mathcal{F} \land B \in \mathcal{C} \land C \in \mathcal{E}) \rightarrow \(- (A - (B - C))\) = \((- (A + (-(B))))\)
proof -

have S1: \((A \in \mathcal{F} \land B \in \mathcal{C} \land C \in \mathcal{E}) \rightarrow \((- (A - (B - C)))\) = \((- (A + (-(B))))\)
by (rule MMI_negsubt)

have S2: \((B \in \mathcal{F} \land C \in \mathcal{E}) \rightarrow \((- (B - C)))\) = \((B - C)\)
by (rule MMI_subclt)

from S1 S2 have S3: \((A \in \mathcal{F} \land B \in \mathcal{C} \land C \in \mathcal{E}) \rightarrow \((- (A - (B - C)))\) = \((- (A + (-(B))))\)
by (rule MMI_3impb)

have S5: \((B \in \mathcal{F} \land C \in \mathcal{E}) \rightarrow \)

( - ( B - C ) ) = ( C - B ) by (rule MMI_negsubd1t)
from S5 have S6: ( B ∈ C ∧ C ∈ C ) →
( A + ( - ( B - C ) ) ) =
( A + ( C - B ) ) by (rule MMI_opreqd)
from S6 have S7: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( A + ( - ( B - C ) ) ) =
( A + ( C - B ) ) by (rule MMI_3adant1)
from S4 S7 show ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( A - ( B - C ) ) = ( A + ( C - B ) )
by (rule MMI_eqtrd)
qed

lemma (in MMIar0) MMI_subsubt:
shows ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( A - ( B - C ) ) = ( ( A - B ) + C )
proof -
have S1: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( A - ( B - C ) ) = ( A + ( C - B ) ) by (rule MMI_subsub2t)
have S2: ( A ∈ C ∧ C ∈ C ∧ B ∈ C ) →
( ( A + C ) - B ) = ( A + ( C - B ) ) by (rule MMI_addsubasst)
have S3: ( A ∈ C ∧ C ∈ C ∧ B ∈ C ) →
( ( A + C ) - B ) = ( ( A - B ) + C ) by (rule MMI_addsubt)
from S2 S3 have S4: ( A ∈ C ∧ C ∈ C ∧ B ∈ C ) →
( A + ( C - B ) ) = ( ( A - B ) + C ) by (rule MMI_eqtr3d)
from S4 have S5: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( A - ( B - C ) ) = ( ( A - B ) + C )
by (rule MMI_eqtrd)
qed

lemma (in MMIar0) MMI_subsub3t:
shows ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( A - ( B - C ) ) = ( ( A + C ) - B )
proof -
have S1: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( A - ( B - C ) ) = ( A + ( C - B ) ) by (rule MMI_subsub2t)
have S2: ( A ∈ C ∧ C ∈ C ∧ B ∈ C ) →
( ( A + C ) - B ) = ( A + ( C - B ) ) by (rule MMI_addsubasst)
from S2 have S3: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( A + ( C - B ) ) = ( ( A - B ) + C ) by (rule MMI_3com23)
from S1 S3 have ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( A - ( B - C ) ) = ( ( A - B ) + C )
by (rule MMI_eqtrd)
qed

lemma (in MMIar0) MMI_subsub4t:
shows ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( ( A - B ) - C ) = ( A - ( B + C ) )
proof -

  have S1: \( A \in C \land B \in C \land ( - C ) \in C \) \( \rightarrow \)
  \( ( A - ( B - ( - C ) ) ) = ( ( A - B ) + ( - C ) ) \) by (rule MMI_subsubt)
  
  have S2: \( C \in C \) \( \rightarrow \) \( ( - C ) \in C \) by (rule MMI_negclt)
  from S1 S2 have S3: \( ( A \in C \land B \in C \land C \in C ) \) \( \rightarrow \)
  \( ( A - ( B - ( - C ) ) ) = ( ( A - B ) + ( - C ) ) \) by (rule MMI_syl3an3)

  have S4: \( B \in C \land C \in C \) \( \rightarrow \)
  \( ( B + C ) = ( B + C ) \) by (rule MMI_axaddcom)
  
  from S4 have S5: \( ( A \in C \land B \in C \land C \in C ) \) \( \rightarrow \)
  \( ( B + C ) = ( B + C ) \) by (rule MMI_3adant1)
  
  from S5 have S6: \( ( A \in C \land B \in C \land C \in C ) \) \( \rightarrow \)
  \( ( A - ( B - ( - C ) ) ) = ( ( A - B ) + ( - C ) ) \) by (rule MMI_syl3an3)

  have S7: \( ( A \in C \land B \in C \land C \in C ) \) \( \rightarrow \)
  \( ( A - ( B - ( - C ) ) ) = ( ( A - B ) + ( - C ) ) \) by (rule MMI_subsubt)
  
  have S8: \( B \in C \land C \in C \) \( \rightarrow \)
  \( ( B + C ) = ( B + C ) \) by (rule MMI_axaddcom)
  
  from S8 have S9: \( ( A \in C \land B \in C \land C \in C ) \) \( \rightarrow \)
  \( ( B + C ) = ( B + C ) \) by (rule MMI_3adant1)
  
  from S9 have S10: \( ( A \in C \land B \in C \land C \in C ) \) \( \rightarrow \)
  \( ( A - ( B - ( - C ) ) ) = ( ( A - B ) + ( - C ) ) \) by (rule MMI_syl3an3)

  from S7 S8 S9 S10 have S11: \( ( A \in C \land B \in C \land C \in C ) \) \( \rightarrow \)
  \( ( A - ( B - ( - C ) ) ) = ( ( A - B ) + ( - C ) ) \) by (rule MMI_3eqtr3rd)

qed

lemma (in MMIIsar0) MMI_sub23t:

  shows \( ( A \in C \land B \in C \land C \in C ) \) \( \rightarrow \)
  \( ( ( A - B ) - C ) = ( ( A - C ) - B ) \)

proof -

  have S1: \( ( B \in C \land C \in C ) \) \( \rightarrow \)
  \( ( B + C ) = ( B + C ) \) by (rule MMI_axaddcom)
  
  from S1 have S2: \( ( A \in C \land B \in C \land C \in C ) \) \( \rightarrow \)
  \( ( C + B ) = ( C + B ) \) by (rule MMI_3adant1)
  
  from S2 have S3: \( ( A \in C \land B \in C \land C \in C ) \) \( \rightarrow \)
  \( ( A - ( B + C ) ) = ( A - ( C + B ) ) \) by (rule MMI_opreq2d)
  
  have S4: \( ( A \in C \land B \in C \land C \in C ) \) \( \rightarrow \)
  \( ( A - ( B - ( - C ) ) ) = ( A - ( B - ( - C ) ) ) \) by (rule MMI_syl3an3)
  
  have S5: \( ( A \in C \land B \in C \land C \in C ) \) \( \rightarrow \)
  \( ( A - ( B - ( - C ) ) ) = ( A - ( B - ( - C ) ) ) \) by (rule MMI_syl3an3)
  
  from S4 S5 have S6: \( ( A \in C \land B \in C \land C \in C ) \) \( \rightarrow \)
  \( ( ( A - B ) - C ) = ( ( A - C ) - B ) \) by (rule MMI_3eqtr4d)

  1492
qed

lemma (in MMIIsar0) MMI_nnnccant:
  shows (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
    ((A - (B - C)) - C) = (A - B)
proof -
  have S1: (A ∈ C ∧ (B - C) ∈ C ∧ C ∈ C) →
    ((A - (B - C)) - C) =
    (A - ((B - C) + C)) by (rule MMI_subsub4t)
  have S2: (A ∈ C ∧ B ∈ C ∧ C ∈ C) → A ∈ C by (rule MMI_3simp1)
  have S3: (B ∈ C ∧ C ∈ C) → (B - C) ∈ C by (rule MMI_subclt)
  from S3 have S4: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
    (B - C) ∈ C by (rule MMI_3adant1)
  have S5: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
    (A - (B - C)) - C) =
    (A - ((B - C) + C)) by (rule MMI_syl3anc)
  have S6: (B ∈ C ∧ C ∈ C) →
    (B - C) + C = B by (rule MMI_npcant)
  from S6 have S7: (B ∈ C ∧ C ∈ C) →
    (A - (B - C)) - (A - C) =
    (A - ((B - C) + C)) by (rule MMI_negsubt)
  have S8: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
    (A - (B - C)) + (A - B) by (rule MMI_opreq2d)
  from S8 have S9: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
    (A - (B - C)) + (A - B) by (rule MMI_3adant1)
  from S9 have S10: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
    (A - (B - C)) - C) = (A - B)
  by (rule MMI_eqtrd)
qed

lemma (in MMIIsar0) MMI_nnnccant1:
  shows (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
    ((A - B) - (A - C)) = (C - B)
proof -
  have S1: ((A - B) ∈ C ∧ (A - C) ∈ C) →
    ((A - B) + ((A - C))) =
    ((A - B) - (A - C)) by (rule MMI_negsubt)
  have S2: ((A - B) ∈ C ∧ (A - C) ∈ C) →
    ((A - B) + ((A - C))) =
    ((- (A - C)) + (A - B)) by (rule MMI_axaddcom)
  have S3: (A - C) ∈ C →
    ((A - B) - (A - C)) =
    (-(A - C)) by (rule MMI_negclt)
  from S2 S3 have S4: ((A - B) ∈ C ∧ (A - C) ∈ C) →
    ((A - B) + ((A - C))) =
    ((- (A - C)) + (A - B)) by (rule MMI_syl2)
  from S1 S4 have S5: ((A - B) ∈ C ∧ (A - C) ∈ C) →
    ((A - B) - (A - C)) =
    ((- (A - C)) + (A - B)) by (rule MMI_eqtr3d)
  have S6: (A ∈ C ∧ B ∈ C) → (A - B) ∈ C by (rule MMI_subclt)
  from S6 have S7: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
    (A - B) ∈ C by (rule MMI_3adant3)
have S8: \(( A \in C \land C \in C ) \rightarrow ( A - C ) \in C\) by (rule MMI_subclt)

from S8 have S9: \(( A \in C \land B \in C \land C \in C ) \rightarrow ( A - B ) = ( A - C ) + ( A - B )\) by (rule MMI_sylanc)

have S11: \(( A \in C \land C \in C ) \rightarrow ( - ( A - C ) ) = ( C - A )\) by (rule MMI_negsubdi2t)

from S11 have S12: \(( A \in C \land B \in C \land C \in C ) \rightarrow ( - ( A - C ) ) + ( A - B ) = ( C - A ) + ( A - B )\) by (rule MMI_3coml)

from S10 S13 S15 show \(( A \in C \land B \in C \land C \in C ) \rightarrow ( ( - ( A - C ) ) + ( A - B ) ) = ( ( C - A ) + ( A - B ) )\) by (rule MMI_3eqtrd)

qed

lemma (in MMI_isar0) MMI_nncan2t:
  shows \(( A \in C \land B \in C \land C \in C ) \rightarrow ( ( A - C ) - ( B - C ) ) = ( A - B )\)
proof -
  have S1: \(( A \in C \land ( B - C ) \in C \land C \in C ) \rightarrow ( ( A - ( B - C ) ) - C ) = ( ( A - C ) - ( B - C ) )\) by (rule MMI_sub23t)

  have S2: \(( A \in C \land B \in C \land C \in C ) \rightarrow A \in C\) by (rule MMI_3simp1)

  have S3: \(( B \in C \land C \in C ) \rightarrow ( B - C ) \in C\) by (rule MMI_subclt)

  from S3 have S4: \(( A \in C \land B \in C \land C \in C ) \rightarrow ( ( A - ( B - C ) ) - C ) = ( ( A - C ) - ( B - C ) )\) by (rule MMI_3eqtr3d)

qed

lemma (in MMI_isar0) MMI_nncant:
  shows \(( A \in C \land B \in C ) \rightarrow ( A - ( A - B ) ) = B\)
proof -
have $S1$: $0 \in C$ by (rule MMI_0cn)

have $S2$: $(A \in C \land 0 \in C \land B \in C) \rightarrow ((A - 0) - (A - B)) = (B - 0)$ by (rule MMI_nncan1t)

from $S1 \ S2 \ S3$: $(A \in C \land B \in C \land C \in C) \rightarrow ((A - 0) - (A - B)) = (B - 0)$ by (rule MMI_mp3an2)

have $S4$: $A \in C \rightarrow (A - 0) = A$ by (rule MMI_subid1t)

have $S5$: $(A \in C \land B \in C \land C \in C) \rightarrow (A - 0) = A$ by (rule MMI_adantr)

from $S5$ have $S6$: $(A \in C \land B \in C \land C \in C) \rightarrow (A - (A - B)) = (A - B)$ by (rule MMI_3eqtr3d)

proof -

have $S1$: $(A \in C \land (B + C) \in C \land C \in C) \rightarrow (A - (B + C)) + C = (A - B)$

have $S2$: $(A \in C \land B \in C \land C \in C) \rightarrow A \in C$ by (rule MMI_3simp1)

have $S3$: $(B + C) \in C \rightarrow (B + C) \in C$ by (rule MMI_axaddcl)

from $S3$ have $S4$: $(A \in C \land B \in C \land C \in C) \rightarrow (B + C) \in C$ by (rule MMI_3adant1)

have $S5$: $(A \in C \land B \in C \land C \in C) \rightarrow C \in C$ by (rule MMI_3simp3)

from $S1 \ S2 \ S4 \ S5 \ S6$: $(A \in C \land B \in C \land C \in C) \rightarrow (A - (B + C)) + C = (A - B)$ by (rule MMI_eqtr3d)

qed

lemma (in MMIars0) MMI_nppcan2t:

shows $(A \in C \land B \in C \land C \in C) \rightarrow ((A - (B + C)) + C) = (A - B)$

proof -

have $S1$: $(A \in C \land (B + C) \in C \land C \in C) \rightarrow (A - (B + C)) + C = (A - B)$ by (rule MMI_syl3anc)

have $S2$: $(B + C) \in C \rightarrow (B + C) \in C$ by (rule MMI_pncant)

from $S7$ have $S8$: $(A \in C \land B \in C \land C \in C) \rightarrow (B + C) \in C$ by (rule MMI_3adant1)

from $S8$ have $S9$: $(A \in C \land B \in C \land C \in C) \rightarrow (A - (B + C)) - C = (A - B)$ by (rule MMI_opreq2d)

from $S6 \ S9 \ S8$ show $(A \in C \land B \in C \land C \in C) \rightarrow ((A - (B + C)) + C) = (A - B)$ by (rule MMI_eqtr3d)

qed

lemma (in MMIars0) MMI_mulm1t:

shows $A \in C \rightarrow ((-1) \cdot A) = ((-1) \cdot A)$

proof -

have $S1$: $1 \in C$ by (rule MMI_1cn)

have $S2$: $(1 \in C \land A \in C) \rightarrow ((-1) \cdot A) = ((-1) \cdot A)$ by (rule MMI_mulneg1t)

1495
from S1 S2 have S3: A ∈ C →
( ( - 1 ) · A ) = ( - ( 1 · A ) ) by (rule MMI_mpan)
have S4: A ∈ C → ( 1 · A ) = A by (rule MMI_mulid2t)
from S4 have S5: A ∈ C → ( - ( 1 · A ) ) = ( ( - A ) )
by (rule MMI_negeqd)
from S3 S5 show A ∈ C → ( ( - 1 ) · A ) = ( ( - A ) )
by (rule MMI_eqtrd)
qed

lemma (in MMI_isar0) MMI_mulm1: assumes A1: A ∈ C
shows ( ( - 1 ) · A ) = ( ( - A ) )
proof -
from A1 have S1: A ∈ C.
have S2: A ∈ C → ( ( - 1 ) · A ) = ( ( - A ) ) by (rule MMI_mulm1)
from S1 S2 show ( ( - 1 ) · A ) = ( ( - A ) ) by (rule MMI_ax_mp)
qed

lemma (in MMI_isar0) MMI_sub4t:
shows ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →
( ( A + B ) - ( C + D ) ) =
( ( A - C ) + ( B - D ) )
proof -
have S1: ( C ∈ C ∧ D ∈ C ) →
( - ( C + D ) ) =
( ( - C ) + ( - D ) ) by (rule MMI_negdit)
from S1 have S2: ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →
( ( - C ) + ( - D ) ) by (rule MMI_adantl)
from S2 have S3: ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →
( ( A + B ) + ( ( - C ) + ( - D ) ) ) =
( ( A + B ) + ( ( - C ) ) + ( - D ) ) ) by (rule MMI_opreq2d)
by (rule MMI_ax_mp)
have S4:
( ( A ∈ C ∧ B ∈ C ) ∧ ( ( - C ) ∈ C ∧ ( - D ) ∈ C ) ) →
( ( A + B ) + ( ( - C ) ) + ( - D ) ) ) =
( ( A + ( - C ) ) + ( B + ( - D ) ) ) by (rule MMI_add4t)
have S5: C ∈ C → ( - C ) ∈ C by (rule MMI_negclt)
have S6: D ∈ C → ( - D ) ∈ C by (rule MMI_negclt)
from S5 S6 have S7: ( C ∈ C ∧ D ∈ C ) →
( ( - C ) ∈ C ∧ ( - D ) ∈ C ) by (rule MMI_anim12i)
from S4 S7 have S8: ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →
( ( A + B ) + ( ( - C ) + ( - D ) ) ) =
( ( A + ( - C ) ) + ( B + ( - D ) ) ) by (rule MMI_sylan2)
from S3 S8 have S9: ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →
( ( A + B ) + ( ( - C + D ) ) ) =
( ( A + ( - C ) ) + ( B + ( - D ) ) ) by (rule MMI_eqtrd)
  have S10: ( ( A + B ) ∈ C ∧ ( C + D ) ∈ C ) ---->
  ( ( A + B ) + ( - ( C + D ) ) ) =
  ( ( A + B ) - ( C + D ) ) by (rule MMI_negsubt)
  have S11: ( A ∈ C ∧ B ∈ C ) ----> ( A + B ) ∈ C by (rule MMI_axaddcl)
  have S12: ( C ∈ C ∧ D ∈ C ) ----> ( C + D ) ∈ C by (rule MMI_axaddcl)
  from S10 S11 S12 have S13:
    ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) ---->
    ( ( A + B ) + ( - ( C + D ) ) ) =
    ( ( A + B ) - ( C + D ) ) by (rule MMI_syl2an)
  from S14 have S15: ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) ---->
    ( A + ( - C ) ) = ( A - C ) by (rule MMI_negsubt)
  from S16 have S17: ( ( A + B ) - ( C + D ) ) by (rule MMI_axaddcl)
  from S15 S17 have S18: ( ( A + B ) - ( C + D ) ) ---->
    ( ( A + B ) - ( C + D ) ) =
    ( ( A + B ) + ( - ( C + D ) ) ) =
    ( ( A + B ) + ( B - D ) ) by (rule MMI_opreq12d)
    from S9 S13 S18 show ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) ---->
    ( ( A + B ) - ( C + D ) ) =
    ( ( A - C ) + ( B - D ) ) by (rule MMI_3eqtr3d)
  qed

lemma (in MMIسار0) MMI_sub4: assumes A1: A ∈ C and
  A2: B ∈ C and
  A3: C ∈ C and
  A4: D ∈ C
  shows ( ( A + B ) - ( C + D ) ) =
  ( ( A - C ) + ( B - D ) )
proof -
  from A1 have S1: A ∈ C.
  from A2 have S2: B ∈ C.
  from S1 S2 have S3: A ∈ C ∧ B ∈ C by (rule MMI_pm3_2i)
  from A3 have S4: C ∈ C.
  from A4 have S5: D ∈ C.
  from S4 S5 have S6: C ∈ C ∧ D ∈ C by (rule MMI_pm3_2i)
  have S7: ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) ---->
    ( ( A + B ) - ( C + D ) ) =
    ( ( A - C ) + ( B - D ) ) by (rule MMI_sub4t)
    from S3 S6 S7 show ( ( A + B ) - ( C + D ) ) =
    ( ( A - C ) + ( B - D ) ) by (rule MMI_mp2an)
  qed

1497
lemma (in MMIsar0) MMI_mulsubt:
  shows \(( A \in C \land B \in C ) \land ( C \in C \land D \in C ) \) \(\rightarrow\)
  \(( ( A - B ) \cdot ( C - D ) ) =\)
  \(( ( ( A \cdot C ) + ( D \cdot B ) ) - ( ( A \cdot D ) + ( C \cdot B ) ) ) \)

proof -
  have S1: \(( A \in C \land B \in C ) \rightarrow\)
  \(( A + ( ( - B ) ) ) = ( A - B ) \) by (rule MMI_negsubt)
  have S2: \(( C \in C \land D \in C ) \rightarrow\)
  \(( C + ( - D ) ) = ( C - D ) \) by (rule MMI_negsubt)
  from S1 S2 have S3: \(( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \) \(\rightarrow\)
  \(( ( A + ( ( - B ) ) ) \cdot ( C + ( - D ) ) ) =\)
  \(( ( ( A \cdot C ) + ( ( - D ) \cdot ( ( - B ) ) ) + ( ( A \cdot ( - D ) ) + ( C \cdot ( ( - B ) ) ) ) \) by (rule MMI_opreqan12d)
  have S4: \(( ( A \in C \land ( ( - B ) ) \in C ) \land ( C \in C \land ( - D ) \in C ) \) \(\rightarrow\)
  \(( ( A + ( ( - B ) ) ) \cdot ( C + ( - D ) ) ) =\)
  \(( ( ( A \cdot C ) + ( ( - D ) \cdot ( ( - B ) ) ) ) +\)
  \(( ( ( A \cdot ( - D ) ) + ( C \cdot ( ( - B ) ) ) ) \) by (rule MMI_sylanr2)
  have S5: \(( D \in C \rightarrow ( - D ) \in C ) \) by (rule MMI_negclt)
  from S4 S5 have S6: \(( ( A \in C \land ( ( - B ) ) \in C ) \land ( C \in C \land D \in C ) \) \(\rightarrow\)
  \(( ( A + ( ( - B ) ) ) \cdot ( C + ( - D ) ) ) =\)
  \(( ( ( A \cdot C ) + ( ( - D ) \cdot ( ( - B ) ) ) ) +\)
  \(( ( ( A \cdot ( - D ) ) + ( C \cdot ( ( - B ) ) ) ) \) by (rule MMI_muladdt)
  have S7: \(( B \in C \rightarrow ( ( - B ) ) \in C ) \) by (rule MMI_negclt)
  from S6 S7 have S8: \(( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) \) \(\rightarrow\)
  \(( ( A + ( ( - B ) ) ) \cdot ( C + ( - D ) ) ) =\)
  \(( ( ( A \cdot C ) + ( ( - D ) \cdot ( ( - B ) ) ) ) +\)
  \(( ( ( A \cdot ( - D ) ) + ( C \cdot ( ( - B ) ) ) ) \) by (rule MMI_sylanl2)
  have S9: \(( D \in C \land B \in C ) \rightarrow\)
  \(( ( ( - D ) \cdot ( ( - B ) ) ) = ( D \cdot B ) \) by (rule MMI_mul2negt)
  from S9 have S10: \(( B \in C \land D \in C ) \rightarrow\)
  \(( ( - D ) \cdot ( ( - B ) ) ) = ( D \cdot B ) \) by (rule MMI_ancoms)
  from S10 have S11: \(( B \in C \land D \in C ) \rightarrow\)
  \(( ( A \cdot C ) + ( ( - D ) \cdot ( ( - B ) ) ) ) =\)
  \(( ( ( A \cdot C ) + ( ( - D ) \cdot ( ( - B ) ) ) ) \) by (rule MMI_opreq2d)
  from S11 have S12: \(( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) \) \(\rightarrow\)
  \(( ( A \cdot C ) + ( ( - D ) \cdot ( ( - B ) ) ) ) =\)
  \(( ( ( A \cdot C ) + ( ( - D ) \cdot ( ( - B ) ) ) ) \) by (rule MMI_ad2ant2l)
  have S13: \(( A \in C \land D \in C ) \rightarrow\)
  \(( A \cdot ( - D ) ) = ( - ( A \cdot D ) ) \) by (rule MMI_mulneg2t)
  have S14: \(( C \in C \land B \in C ) \rightarrow\)
  \(( C \cdot ( ( - B ) ) ) = ( - ( C \cdot B ) ) \) by (rule MMI_mulneg2t)
  from S13 S14 have S15: \(( ( A \in C \land D \in C ) \land ( C \in C \land B \in C ) \) \(\rightarrow\)
\(( ( A \cdot ( - D )) + ( C \cdot ( - B ))) = ( - ( A \cdot D )) + ( ( - ( C \cdot B )))\) by (rule MMI_opreqan12d)

have S16: \(( ( A \cdot D ) \in C \land ( C \cdot B ) \in C ) \rightarrow \)
\(( ( - ( A \cdot D ) ) + ( ( C \cdot B ))) = ( ( - ( A \cdot D )) + ( - ( C \cdot B )))\) by (rule MMI_negdit)

have S17: \(( A \in C \land D \in C ) \rightarrow ( A \cdot D ) \in C \) by (rule MMI_axmulcl)

have S18: \(( C \in C \land B \in C ) \rightarrow ( C \cdot B ) \in C \) by (rule MMI_axmulcl)

from S16 S17 S18 have S19:
\(( ( A \in C \land D \in C ) \land ( C \in C \land B \in C ) ) \rightarrow \)
\(( ( - ( A \cdot D ) ) + ( C \cdot B ))) \rightarrow \)
\(( - ( ( A \cdot D ) ) + ( ( - ( C \cdot B ))) ) \) by (rule MMI_syl2an)

from S15 S19 have S20: \(( ( A \in C \land D \in C ) \land ( C \in C \land B \in C ) \) \rightarrow \)
\(( ( A \cdot ( - D )) + ( C \cdot ( - B ))) = ( - ( ( A \cdot D ) ) + ( ( C \cdot B )))\) by (rule MMI_eqtr4d)

from S20 have S21: \(( ( A \in C \land D \in C ) \land ( B \in C \land C \in C ) ) \rightarrow \)
\(( ( A \cdot ( - D )) + ( C \cdot ( - B ))) = ( - ( ( A \cdot D ) ) + ( ( C \cdot B )))\) by (rule MMI_ancom2s)

from S21 have S22: \(( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \rightarrow \)
\(( ( A \cdot ( - D )) + ( C \cdot ( - B ))) = ( - ( ( A \cdot D ) ) + ( ( C \cdot B )))\) by (rule MMI_an42s)

from S12 S22 have S23: \(( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) \) \rightarrow \)
\(( ( ( A \cdot C ) + ( ( - D ) \cdot ( - B ))) ) + ( ( A \cdot D ) ) + ( ( C \cdot B ))) \) by (rule MMI_opreq12d)

have S24: \(( ( ( A \cdot C ) + ( ( D \cdot B ))) ) \in C \land ( A \cdot D ) + ( C \cdot B ) ) \in C \) \rightarrow \)
\(( ( ( A \cdot C ) + ( ( D \cdot B ))) + ( ( - ( A \cdot D ) ) + ( C \cdot B ))) ) \) by (rule MMI_negsult)

have S25: \(( ( A \cdot C ) + ( ( D \cdot B )) \in C \land ( A \cdot D ) + ( C \cdot B ) ) \in C \) \rightarrow \)
\(( ( ( A \cdot C ) + ( D \cdot B )) ) \in C \) by (rule MMI_axaddcl)

have S26: \(( A \in C \land C \in C ) \rightarrow ( A \cdot C ) \in C \) by (rule MMI_axmulcl)

have S27: \(( D \in C \land B \in C ) \rightarrow ( D \cdot B ) \in C \) by (rule MMI_axmulcl)

from S27 have S28: \(( B \in C \land D \in C ) \rightarrow ( D \cdot B ) \in C \) by (rule MMI_axcoms)

from S25 S26 S28 have S29:
\(( ( A \in C \land C \in C ) \land ( B \in C \land D \in C ) ) \rightarrow \)
\(( ( A \cdot C ) + ( ( D \cdot B ))) ) \in C \) by (rule MMI_syl2an)

from S29 have S30: \(( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \rightarrow \)
\(( ( A \cdot C ) + ( ( D \cdot B ))) ) \in C \) by (rule MMI_ancoms)

from S17 have S32: \(( A \in C \land D \in C ) \rightarrow ( A \cdot D ) \in C \).

1499
from S18 have S33: \(( C \in \mathbb{C} \land B \in \mathbb{C} ) \rightarrow ( C \cdot B ) \in \mathbb{C} \).
from S33 have S34: \(( B \in \mathbb{C} \land C \in \mathbb{C} ) \rightarrow ( C \cdot B ) \in \mathbb{C} \)
by (rule MMI_ancoms)
f
from S31 S32 S34 have S35:
\(( ( A \in \mathbb{C} \land D \in \mathbb{C} ) \land ( B \in \mathbb{C} \land C \in \mathbb{C} ) ) \rightarrow ( ( A \cdot D ) + ( C \cdot B ) ) \in \mathbb{C} \)
by (rule MMI_syl2an)
from S35 have S36: \(( ( A \in \mathbb{C} \land B \in \mathbb{C} ) \land ( C \in \mathbb{C} \land D \in \mathbb{C} ) ) \rightarrow ( ( A \cdot D ) + ( C \cdot B ) ) \in \mathbb{C} \)

\(( ( A \cdot D ) + ( C \cdot B ) ) \in \mathbb{C} \)
by (rule MMI_an42s)

from S24 S30 S36 have S37:
\(( ( A \in \mathbb{C} \land B \in \mathbb{C} ) \land ( C \in \mathbb{C} \land D \in \mathbb{C} ) ) \rightarrow ( ( ( A \cdot C ) + ( D \cdot B ) ) + ( - ( ( A \cdot D ) + ( C \cdot B ) ) ) ) = ( ( ( A \cdot C ) + ( D \cdot B ) ) - ( ( A \cdot D ) + ( C \cdot B ) ) ) \)

by (rule MMI_3eqtrd)
from S8 S23 S37 have S38: \(( ( A \in \mathbb{C} \land B \in \mathbb{C} ) \land ( C \in \mathbb{C} \land D \in \mathbb{C} ) ) \rightarrow ( ( A + ( - B ) ) \cdot ( C + ( - D ) ) ) = ( ( ( A \cdot C ) + ( D \cdot B ) ) - ( ( A \cdot D ) + ( C \cdot B ) ) ) \)
by (rule MMI_eqtr3d)

deduce (in MMIar0) MMI_pmpcant:
shows \(( A \in \mathbb{C} \land B \in \mathbb{C} \land C \in \mathbb{C} ) \rightarrow ( ( A + B ) - ( A + C ) ) = ( B - C ) \)

proof -
	note S1: \(( ( A \in \mathbb{C} \land B \in \mathbb{C} ) \land ( A \in \mathbb{C} \land C \in \mathbb{C} ) ) \rightarrow ( ( A + B ) - ( A + C ) ) = ( B - C ) \)

from S1 have S2: \(( A \in \mathbb{C} \land B \in \mathbb{C} \land C \in \mathbb{C} ) \rightarrow ( A + B ) - ( A + C ) = ( B - C ) \)

by (rule MMI_sub4t)
from S1 have S2: \(( A \in \mathbb{C} \land B \in \mathbb{C} \land C \in \mathbb{C} ) \rightarrow ( A + B ) - ( A + C ) = ( B - C ) \)

by (rule MMI_ananid)

have S3: \(( A \in \mathbb{C} \rightarrow ( A - A ) = 0 ) \)
by (rule MMI_subidt)

from S3 have S4: \(( A \in \mathbb{C} ) \rightarrow ( A - A ) = 0 ) \)

by (rule MMI_opreq1d)

have S5: \(( B \in \mathbb{C} \land C \in \mathbb{C} ) \rightarrow ( B - C ) \in \mathbb{C} \)
by (rule MMI_subclt)

have S6: \(( B - C ) \in \mathbb{C} \)

by (rule MMI_addid2t)

from S5 S6 have S7: \(( B \in \mathbb{C} \land C \in \mathbb{C} ) \rightarrow ( 0 + ( B - C ) ) = ( B - C ) \)
by (rule MMI_syl)

from S4 S7 have S8: \(( A \in \mathbb{C} \land B \in \mathbb{C} \land C \in \mathbb{C} ) \rightarrow ( ( A - A ) + ( B - C ) ) = ( B - C ) \)
by (rule MMI_sylan9eq)

from S2 S8 have S9: \(( A \in \mathbb{C} \land B \in \mathbb{C} \land C \in \mathbb{C} ) \rightarrow ( ( A + B ) - ( A + C ) ) = ( B - C ) \)
by (rule MMI_eqtrd)
from S9 show \(( A \in C \land B \in C \land C \in C ) \rightarrow \(( A + B ) - ( A + C ) ) = ( B - C )\) by (rule MMI_3impb)

qed

lemma (in MMIsar0) MMI_pnpcan2t:
shows \(( A \in C \land B \in C \land C \in C ) \rightarrow \(( A + C ) - ( B + C ) ) = ( A - B )\)

proof -
have S1: \(( A \in C \land C \in C ) \rightarrow \(( A + C ) = ( C + A )\) by (rule MMI_axaddcom)
from S1 have S2: \(( A \in C \land B \in C \land C \in C ) \rightarrow \(( A + C ) = ( C + A )\) by (rule MMI_3adant2)
have S3: \(( B \in C \land C \in C ) \rightarrow \(( B + C ) = ( C + B )\) by (rule MMI_axaddcom)
from S3 have S4: \(( A \in C \land B \in C \land C \in C ) \rightarrow \(( B + C ) = ( C + B )\) by (rule MMI_3adant1)
from S2 S4 have S5: \(( A \in C \land B \in C \land C \in C ) \rightarrow \(( ( A + C ) - ( B + C ) ) = ( ( C + A ) - ( C + B ) )\) by (rule MMI_opreq12d)
have S6: \(( C \in C \land A \in C \land B \in C ) \rightarrow \(( ( C + A ) - ( C + B ) ) = ( A - B )\) by (rule MMI_pnpcant)
from S6 have S7: \(( A \in C \land B \in C \land C \in C ) \rightarrow \(( ( A + B ) - ( A + ( - C ) ) ) = ( ( A + B ) - ( A - C ) )\) by (rule MMI_syl3an3)
have S8: \(( A \in C \land A \in C \land ( - C ) \in C ) \rightarrow \(( A + ( - C ) ) = ( A - C )\) by (rule MMI_negsubt)
from S8 have S9: \(( A \in C \land B \in C \land C \in C ) \rightarrow \(( A + ( - C ) ) = ( A - C )\) by (rule MMI_3adant2)
from S9 have S10: \(( A \in C \land B \in C \land C \in C ) \rightarrow \(( A + ( - C ) ) = ( A - C )\) by (rule MMI_opreq2d)
have S11: \(( B \in C \land C \in C ) \rightarrow \(( B - ( - C ) ) = ( B + C )\) by (rule MMI_subnegt)
from S11 have S12: \(( A \in C \land B \in C \land C \in C ) \rightarrow \(( B - ( - C ) ) = ( B + C )\) by (rule MMI_eqtrd)

qed

lemma (in MMIsar0) MMI_pnncant:
shows \(( A \in C \land B \in C \land C \in C ) \rightarrow \(( ( A + B ) - ( A - C ) ) = ( B + C )\)

proof -
have S1: \(( A \in C \land B \in C \land ( - C ) \in C ) \rightarrow \(( ( A + B ) - ( A + ( - C ) ) ) = ( B - ( - C ) )\) by (rule MMI_pnpcant)
from S1 have S2: \(( A \in C \land B \in C \land C \in C ) \rightarrow \(( ( A + B ) - ( A + ( - C ) ) ) = ( B - ( - C ) )\) by (rule MMI_syl3an3)
have S3: \(( ( A + B ) - ( A + ( - C ) ) ) = ( B - ( - C ) )\) by (rule MMI_axaddcom)
from S3 have S4: \(( ( A + B ) - ( A + ( - C ) ) ) = ( B - ( - C ) )\) by (rule MMI_3adant1)
from S4 have S5: \(( ( A + B ) - ( A + ( - C ) ) ) = ( B - ( - C ) )\) by (rule MMI_opreq2d)
have S6: \(( B \in C \land C \in C ) \rightarrow \(( ( A + B ) - ( A - C ) ) = ( B - ( - C ) )\) by (rule MMI_subnegt)
from S6 have S7: \(( ( A + B ) - ( A - C ) ) = ( B - ( - C ) )\) by (rule MMI_eqtrd)

qed
\[(B - ( - C )) = (B + C)\] by (rule MMI_3adant1)

from S3 S6 S8 show \(A \in C \land B \in C \land C \in C\) \(\rightarrow\)
\[(A + B) - (A - C) = (B + C)\] by (rule MMI_3eqtr3d)

qed

\textbf{lemma (in MMIar0) MMI_ppncant:}
\begin{align*}
&\text{shows } (A \in C \land B \in C \land C \in C) \rightarrow \\
&(\ (A + B) + (C - B) ) = (A + C)
\end{align*}

\textbf{proof -}
\begin{itemize}
  \item have S1: \((A \in C \land B \in C) \rightarrow ((A + B) - (B - C)) = (A + C)\) by (rule MMI_3adant3)
  \item have S2: \((A \in C \land B \in C \land C \in C) \rightarrow ((A + B) - (B - C)) = (A + C)\) by (rule MMI_opreq1d)
  \item from S1 have S3: \((A \in C \land B \in C \land C \in C) \rightarrow (A + B) = (B + A)\) by (rule MMI_axaddcom)
  \item from S3 have S4: \((A \in C \land B \in C \land C \in C) \rightarrow (A + B) = (B + A)\) by (rule MMI_3adant3)
  \item have S5: \((A \in C \land B \in C) \rightarrow (A + B) \in C\) by (rule MMI_axaddcl)
  \item from S5 have S6: \((A \in C \land B \in C \land C \in C) \rightarrow (A + B) \in C\) by (rule MMI_3adant3)
  \item have S7: \((A \in C \land B \in C \land C \in C) \rightarrow B \in C\) by (rule MMI_3simp2)
  \item have S8: \((A \in C \land B \in C \land C \in C) \rightarrow C \in C\) by (rule MMI_3simp3)
  \item from S4 S6 S7 S8 have S9: \((A \in C \land B \in C \land C \in C) \rightarrow ((A + B) - (B - C)) = (A + C)\) by (rule MMI_syl3anc)
\end{itemize}

have S10: \((B \in C \land A \in C \land C \in C) \rightarrow ((B + A) - (B - C)) = (A + C)\) by (rule MMI_syl3anc)

\item from S10 have S11: \((B + A) - (B - C) = (A + C)\) by (rule MMI_pnncant)

\item from S3 S9 S11 show \((A \in C \land B \in C \land C \in C) \rightarrow (A + B) + (C - B) = (A + C)\) by (rule MMI_3eqtr3d)

qed

\textbf{lemma (in MMIar0) MMI_pnncan:}
\begin{align*}
&\text{assumes } A1: A \in C \land A2: B \in C \land A3: C \in C \\
&\text{shows } (A + B) - (A - C) = (B + C)
\end{align*}

\textbf{proof -}
\begin{itemize}
  \item from A1 have S1: \(A \in C\).
  \item from A2 have S2: \(B \in C\).
  \item from A3 have S3: \(C \in C\).
  \item have S4: \((A \in C \land B \in C \land C \in C) \rightarrow ((A + B) - (A - C)) = (B + C)\) by (rule MMI_pnncant)
\end{itemize}

\item from S1 S2 S3 S4 show \((A + B) - (A - C) = (B + C)\) by (rule MMI_mp3an)

qed
lemma (in MMIsar0) MMI_mulcan: assumes A1: A ∈ C and
A2: B ∈ C and
A3: C ∈ C and
A4: A ≠ 0
shows \(( A \cdot B ) = ( A \cdot C ) \iff B = C\)
proof -
from A1 have S1: A ∈ C.
from A4 have S2: A ≠ 0.
from S1 S2 have S3: ∃ x ∈ C . \(( A \cdot x ) = 1\) by (rule MMI_recex)
from A1 have S4: A ∈ C.
from A2 have S5: B ∈ C.
from S5 have S6: ( x ∈ C ∧ A ∈ C ∧ B ∈ C ) \→
\(( ( x \cdot A ) \cdot B ) = ( x \cdot ( A \cdot B ) )\) by (rule MMI_axmulass)
from S3 S6 have S7: \(( x \cdot ( A \cdot B ) ) = ( x \cdot ( A \cdot C ) )\) by (rule MMI_axmulass)
from A3 have S8: C ∈ C.
from S7 S8 have S9: \(( x \cdot ( A \cdot C ) ) = ( x \cdot ( A \cdot C ) )\) by (rule MMI_mp3an3)
from S4 S9 have S10: \(( x \cdot ( A \cdot C ) ) = ( x \cdot ( A \cdot C ) )\) by (rule MMI_mp3an3)
from S8 S10 have S11: \(( x \cdot ( A \cdot C ) ) = ( x \cdot ( A \cdot C ) )\) by (rule MMI_mp3an3)
from S7 S11 have S12: \(( x \cdot ( A \cdot C ) ) = ( x \cdot ( A \cdot C ) )\) by (rule MMI_mp3an3)
from S6 have S13: \(( ( x \cdot A ) \cdot B ) \iff ( x \cdot ( A \cdot B ) )\) by (rule MMI_opreq2)
from S12 S13 have S14: \(( x \cdot ( A \cdot B ) ) = ( x \cdot ( A \cdot C ) )\) by (rule MMI_syl5bir)
from S14 have S15: \(( x \cdot ( A \cdot C ) ) = ( x \cdot ( A \cdot C ) )\) by (rule MMI_axmulcom)
from A1 have S16: A ∈ C.
from S16 have S17: \(( A \cdot x ) = ( x \cdot A )\) by (rule MMI_axmulcom)
from S16 S17 have S18: \(( x \cdot x ) = 1 \iff ( x \cdot A ) = 1\) by (rule MMI_eqeq1d)
from S18 have S19: \(( x \cdot x ) = 1 \iff ( x \cdot A ) = 1\) by (rule MMI_eqeq1d)
from S19 have S20: \(( x \cdot A ) = 1 \iff ( x \cdot A ) = 1\) by (rule MMI_mpan)
1 \rightarrow ((x \cdot A) \cdot B) = (1 \cdot B) \text{ by (rule MMI_opreq1)}

from A2 have S21: B \in C.

from S21 have S22: (1 \cdot B) = B by (rule MMI_mulid2)

from S20 S22 have S23: (x \cdot A) = 1 \rightarrow ((x \cdot A) \cdot B) = B

by (rule MMI_syl6eq)

have S24: (x \cdot A) = 1 \rightarrow ((x \cdot A) \cdot C) = (1 \cdot C)

by (rule MMI_opreq1)

from A3 have S25: C \in C.

from S25 have S26: (1 \cdot C) = C by (rule MMI_mulid2)

from S24 S26 have S27: (x \cdot A) = 1 \rightarrow ((x \cdot A) \cdot C) = C

by (rule MMI_syl6eq)

from S23 S27 have S28: (x \cdot A) = 1 \rightarrow ((x \cdot A) \cdot B) = B

by (rule MMI_syl6eq)

from S23 S27 have S29: (x \cdot A) = 1 \rightarrow ((x \cdot A) \cdot B) = B \rightarrow (x \cdot A) \cdot C) = (x \cdot A) \cdot C)

by (rule MMI_imp)

from S29 have S30: (x \in C \land (A \cdot x) = 1) \rightarrow ((A \cdot B) = (A \cdot C) \rightarrow B = C)

by (rule MMI_sylid)

from S31 have x \in C \rightarrow ((A \cdot x) = 1 \rightarrow ((A \cdot B) = (A \cdot C) \rightarrow B = C)

by (rule MMI_imp)

\{ then have S32: \forall x. x \in C \rightarrow ((A \cdot x) = 1 \rightarrow ((A \cdot B) = (A \cdot C) \rightarrow B = C)

by auto

from S32 have S33: (\exists x \in C. (A \cdot x) = 1) \rightarrow ((A \cdot B) = (A \cdot C) \rightarrow B = C)

by (rule MMI_r19_23aiv)

from S3 S33 have S34: (A \cdot B) = (A \cdot C) \rightarrow B = C

by (rule MMI_ax_mp)

have S35: B = C \rightarrow (A \cdot B) = (A \cdot C) \rightarrow (B = C) \rightarrow (rule MMI_impbi)

qed

lemma (in MMIsar0) MMI_mulcant2: assumes A1: A \neq 0

shows (A \in C \land B \in C \land C \in C) \rightarrow ((A \cdot B) = (A \cdot C) \rightarrow B = C)

proof

have S1: A =

if (A \in C, A, 1) \rightarrow

(A \cdot B) =

if (A \in C, A, 1) \cdot B) by (rule MMI_opreq1)

have S2: A =

if (A \in C, A, 1) \rightarrow

1504
\[(A \cdot C) =
\]

( if ( \( A \in C \), \( A \cdot 1 \) ) \( \cdot \) C ) by (rule MMI_opreq1)

from S1 S2 have S3: \( A = \)

if ( \( A \in C \), \( A \cdot 1 \) ) \( \rightarrow \)

( ( A \cdot B ) =

( A \cdot C ) \( \leftrightarrow \)

( if ( \( A \in C \), \( A \cdot 1 \) ) \( \cdot \) B ) =

( if ( \( A \in C \), \( A \cdot 1 \) ) \( \cdot \) C ) by (rule MMI_eqeq12d)

from S3 have S4: \( A = \)

if ( \( A \in C \), \( A \cdot 1 \) ) \( \rightarrow \)

( ( ( A \cdot B ) = ( A \cdot C ) \( \leftrightarrow \) B = C ) \( \leftrightarrow \)

( if ( \( A \in C \), \( A \cdot 1 \) ) \( \cdot \) B ) =

( if ( \( A \in C \), \( A \cdot 1 \) ) \( \cdot \) C ) \( \leftrightarrow \)

B = C ) ) by (rule MMI_bibi1d)

have S5: \( B = \)

if ( \( B \in C \), \( B \cdot 1 \) ) \( \rightarrow \)

( if ( \( A \in C \), \( A \cdot 1 \) ) \( \cdot \) B ) =

( if ( \( A \in C \), \( A \cdot 1 \) ) \( \cdot \) C ) \( \leftrightarrow \)

( if ( \( A \in C \), \( A \cdot 1 \) ) \( \cdot \) if ( \( B \in C \), \( B \cdot 1 \) ) ) by (rule MMI_opreq2)

from S5 have S6: \( B = \)

if ( \( B \in C \), \( B \cdot 1 \) ) \( \rightarrow \)

( ( if ( \( A \in C \), \( A \cdot 1 \) ) \( \cdot \) B ) =

( if ( \( A \in C \), \( A \cdot 1 \) ) \( \cdot \) C ) \( \leftrightarrow \)

( if ( \( A \in C \), \( A \cdot 1 \) ) \( \cdot \) if ( \( B \in C \), \( B \cdot 1 \) ) ) =

( if ( \( A \in C \), \( A \cdot 1 \) ) \( \cdot \) C ) by (rule MMI_eqeq1d)

have S7: \( B = \)

if ( \( B \in C \), \( B \cdot 1 \) ) \( \rightarrow \)

( B = C \( \leftrightarrow \) if ( \( B \in C \), \( B \cdot 1 \) ) = C ) by (rule MMI_eqeq1)

from S6 S7 have S8: \( B = \)

if ( \( B \in C \), \( B \cdot 1 \) ) \( \rightarrow \)

( ( ( ( A \in C \), \( A \cdot 1 \) ) \( \cdot \) B ) = ( if ( \( A \in C \), \( A \cdot 1 \) ) \( \cdot \) C ) \( \leftrightarrow \)

B = C ) \( \leftrightarrow \)

( ( ( ( A \in C \), \( A \cdot 1 \) ) \( \cdot \) B ) = ( if ( \( A \in C \), \( A \cdot 1 \) ) \( \cdot \) C ) \( \leftrightarrow \)

if ( \( B \in C \), \( B \cdot 1 \) ) = C ) ) by (rule MMI_bibi12d)

have S9: \( C = \)

if ( \( C \in C \), \( C \cdot 1 \) ) \( \rightarrow \)

( if ( \( A \in C \), \( A \cdot 1 \) ) \( \cdot \) C ) =

( if ( \( A \in C \), \( A \cdot 1 \) ) \( \cdot \) if ( \( C \in C \), \( C \cdot 1 \) ) ) by (rule MMI_opreq2)

from S9 have S10: \( C = \)

if ( \( C \in C \), \( C \cdot 1 \) ) \( \rightarrow \)

( ( if ( \( A \in C \), \( A \cdot 1 \) ) \( \cdot \) if ( \( B \in C \), \( B \cdot 1 \) ) ) =

( if ( \( A \in C \), \( A \cdot 1 \) ) \( \cdot \) C ) \( \leftrightarrow \)

( if ( \( A \in C \), \( A \cdot 1 \) ) \( \cdot \) if ( \( B \in C \), \( B \cdot 1 \) ) ) =

( if ( \( A \in C \), \( A \cdot 1 \) ) \( \cdot \) if ( \( C \in C \), \( C \cdot 1 \) ) ) by (rule MMI_eqeq2d)

have S11: \( C = \)

if ( \( C \in C \), \( C \cdot 1 \) ) \( \rightarrow \)

( if ( \( B \in C \), \( B \cdot 1 \) ) =

C \( \leftrightarrow \)

if ( \( B \in C \), \( B \cdot 1 \) ) =

1505
if ( C ∈ \mathcal{C}, C, 1 ) ) by (rule MMI_eqeq2)
   from S10 S11 have S12: C =
   if ( C ∈ \mathcal{C}, C, 1 ) −→
   ( ( ( if ( A ∈ \mathcal{C}, A, 1 ) · if ( B ∈ \mathcal{C}, B, 1 ) ) ) −→ )
   ( ( if ( A ∈ \mathcal{C}, A, 1 ) · if ( B ∈ \mathcal{C}, B, 1 ) ) ) −→
   ( ( if ( A ∈ \mathcal{C}, A, 1 ) · if ( C ∈ \mathcal{C}, C, 1 ) ) ) −→
   if ( B ∈ \mathcal{C}, B, 1 ) =
   if ( C ∈ \mathcal{C}, C, 1 ) ) ) by (rule MMI_bibi12d)
   have S13: 1 ∈ \mathcal{C} by (rule MMI_1cn)
   have S14: if ( A ∈ \mathcal{C}, A, 1 ) ∈ \mathcal{C} by (rule MMI_elimel)
   have S15: 1 ∈ \mathcal{C} by (rule MMI_1cn)
   have S16: if ( B ∈ \mathcal{C}, B, 1 ) ∈ \mathcal{C} by (rule MMI_elimel)
   have S17: 1 ∈ \mathcal{C} by (rule MMI_1cn)
   have S18: if ( C ∈ \mathcal{C}, C, 1 ) ∈ \mathcal{C} by (rule MMI_elimel)
   have S19: A =
   if ( A ∈ \mathcal{C}, A, 1 ) −→
   ( A ≠ 0 −→ if ( A ∈ \mathcal{C}, A, 1 ) ≠ 0 ) by (rule MMI_neeq1)
   have S20: 1 =
   if ( A ∈ \mathcal{C}, A, 1 ) −→
   ( 1 ≠ 0 −→ if ( A ∈ \mathcal{C}, A, 1 ) ≠ 0 ) by (rule MMI_neeq1)
   from A1 have S21: A ≠ 0.
   have S22: 1 ≠ 0 by (rule MMI_ax1ne0)
   from S19 S20 S21 S22 have S23: if ( A ∈ \mathcal{C}, A, 1 ) ≠ 0 by (rule
   MMI_keephyp)
   from S14 S16 S18 S23 have S24: ( if ( A ∈ \mathcal{C}, A, 1 ) · if ( B ∈ \mathcal{C}
   , B, 1 ) ) =
   ( ( if ( A ∈ \mathcal{C}, A, 1 ) · if ( C ∈ \mathcal{C}, C, 1 ) ) ) −→
   if ( B ∈ \mathcal{C}, B, 1 ) =
   if ( C ∈ \mathcal{C}, C, 1 ) by (rule MMI_mulcan)
   from S4 S8 S12 S24 show ( A ∈ \mathcal{C} ∧ B ∈ \mathcal{C} ∧ C ∈ \mathcal{C} ) −→
   ( ( A · B ) = ( A · C ) −→ B = C ) by (rule MMI_dedth3h).
qed

lemma (in MMIIsar0) MMI_mulcant:
  shows ( ( A ∈ \mathcal{C} ∧ B ∈ \mathcal{C} ∧ C ∈ \mathcal{C} ) ∧ A ≠ 0 ) −→
  ( ( A · B ) = ( A · C ) −→ B = C )
proof -
  have S1: A =
  if ( A ≠ 0 , A, 1 ) −→
  ( A ∈ \mathcal{C} −→ if ( A ≠ 0 , A, 1 ) ∈ \mathcal{C} ) by (rule MMI_eleq1)
  have S2: A =
  if ( A ≠ 0 , A, 1 ) −→
  ( B ∈ \mathcal{C} −→ B ∈ \mathcal{C} ) by (rule MMI_pm4_2i)
  have S3: A =
  if ( A ≠ 0 , A, 1 ) −→
  ( C ∈ \mathcal{C} −→ C ∈ \mathcal{C} ) by (rule MMI_pm4_2i)
  from S1 S2 S3 have S4: A =
  if ( A ≠ 0 , A, 1 ) −→

1506
( ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) \iff
  ( if ( A \neq 0 , A , 1 ) ∈ C ∧ B ∈ C ∧ C ∈ C ) ) by (rule MMI_3anbi123d)
  have S5: A =
  if ( A \neq 0 , A , 1 ) →
  ( A \cdot B ) =
  ( if ( A \neq 0 , A , 1 ) \cdot B ) by (rule MMI_opreq1)
  have S6: A =
  if ( A \neq 0 , A , 1 ) →
  ( A \cdot C ) =
  ( if ( A \neq 0 , A , 1 ) \cdot C ) by (rule MMI_opreq1)
  from S5 S6 have S7: A =
  if ( A \neq 0 , A , 1 ) →
  ( ( A \cdot B ) = ( A \cdot C ) ←→ B = C ) by (rule MMI_eqeq12d)
  from S7 have S8: A =
  if ( A \neq 0 , A , 1 ) →
  ( ( ( A \cdot B ) = ( A \cdot C ) ←→ B = C ) ) ←→
  ( ( if ( A \neq 0 , A , 1 ) \cdot B ) =
  ( if ( A \neq 0 , A , 1 ) \cdot C ) ←→
  B = C ) ) by (rule MMI_bibi1d)
  from S4 S8 have S9: A =
  if ( A \neq 0 , A , 1 ) →
  ( ( ( A \in C ∧ B ∈ C ∧ C ∈ C ) → ( A \cdot B ) = ( A \cdot C ) ←→ B = C ) ) ←→
  ( ( if ( A \neq 0 , A , 1 ) ∈ C ∧ B ∈ C ∧ C ∈ C ) →
  ( ( if ( A \neq 0 , A , 1 ) \cdot B ) =
  ( if ( A \neq 0 , A , 1 ) \cdot C ) ←→
  B = C ) ) ) by (rule MMI_elimne0)
  have S10: if ( A \neq 0 , A , 1 ) \neq 0 by (rule MMI_mulcan2t)
  from S10 have S11: if ( A \neq 0 , A , 1 ) ∈ C ∧ B ∈ C ∧ C ∈ C ) →
  ( ( if ( A \neq 0 , A , 1 ) \cdot B ) =
  ( if ( A \neq 0 , A , 1 ) \cdot C ) ←→
  B = C ) ) by (rule MMI_mulcan2t)
  from S9 S11 have S12: A \neq 0 →
  ( ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
  ( ( A \cdot B ) = ( A \cdot C ) ←→ B = C ) ) by (rule MMI_dedth)
  from S12 show ( ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) ∧ A \neq 0 ) →
  ( ( A \cdot B ) = ( A \cdot C ) ←→ B = C ) by (rule MMI_impcom)
  qed

lemma (in MMIsar0) MMI_mulcan2t:
  shows ( ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) ∧ C \neq 0 ) →
  ( ( A \cdot C ) = ( B \cdot C ) ←→ A = B )
proof -
  have S1: ( A ∈ C ∧ C ∈ C ) →
  ( A \cdot C ) = ( C \cdot A ) by (rule MMI_axmul)
  from S1 have S2: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
\[(A \cdot C) = (C \cdot A)\] by (rule MMI_3adant2)

have \(S3: (B \in C \land C \in C)\) \(\rightarrow\)

\[(B \cdot C) = (C \cdot B)\] by (rule MMI_axmulem)

from \(S3\) have \(S4: (A \in C \land B \in C \land C \in C)\) \(\rightarrow\)

\[(B \cdot C) = (C \cdot B)\] by (rule MMI_3adant1)

from \(S2\) \(S4\) have \(S5: (A \in C \land B \in C \land C \in C)\) \(\rightarrow\)

\[(A \cdot C) =\]

\[(B \cdot C) \leftrightarrow (C \cdot A) = (C \cdot B)\] by (rule MMI_eqeq12i)

from \(S5\) have \(S6: (A \in C \land B \in C \land C \in C)\) \(\land\) \(C \neq 0\) \(\rightarrow\)

\[(A \cdot C) =\]

\[(B \cdot C) \leftrightarrow (C \cdot A) = (C \cdot B)\] by (rule MMI_adantr)

have \(S7: (C \in C \land A \in C \land B \in C)\) \(\land\) \(C \neq 0\) \(\rightarrow\)

\[(C \cdot A) = (C \cdot B) \leftrightarrow A = B\] by (rule MMI_mulcant)

from \(S7\) have \(S8: (C \in C \land A \in C \land B \in C)\) \(\rightarrow\)

\[(C \neq 0)\]

\[(C \cdot A) = (C \cdot B) \leftrightarrow A = B\] by (rule MMI_ex)

from \(S8\) have \(S9: (A \in C \land B \in C \land C \in C)\) \(\rightarrow\)

\[(C \neq 0)\]

\[(C \cdot A) = (C \cdot B) \leftrightarrow A = B\] by (rule MMI_3coml)

from \(S9\) have \(S10: (A \in C \land B \in C \land C \in C)\) \(\land\) \(C \neq 0\) \(\rightarrow\)

\[(C \cdot A) = (C \cdot B) \leftrightarrow A = B\] by (rule MMI_imp)

from \(S6\) \(S10\) show \((A \in C \land B \in C \land C \in C)\) \(\land\) \(C \neq 0\) \(\rightarrow\)

\[(A \cdot C) = (B \cdot C) \leftrightarrow A = B\] by (rule MMI_bitrd)

qed

lemma (in MIMIsar0) MMI_mul0or: assumes \(A1: A \in C\) and

\(A2: B \in C\)

shows \((A \cdot B) = 0 \leftrightarrow (A = 0 \lor B = 0)\)

proof -

have \(S1: A \neq 0 \leftrightarrow \neg (A = 0)\) by (rule MMI_df_ne)

from \(A1\) have \(S2: A \in C\).

from \(A2\) have \(S3: B \in C\).

have \(S4: 0 \in C\) by (rule MMI_0cn)

from \(S2\) \(S3\) \(S4\) have \(S5: A \in C \land B \in C \land 0 \in C\) by (rule MMI_3pm3_2i)

have \(S6: (A \in C \land B \in C \land 0 \in C)\) \(\land\) \(A \neq 0\) \(\rightarrow\)

\[(A \cdot B) = (A \cdot 0) \leftrightarrow B = 0\] by (rule MMI_mulcant)

from \(S5\) \(S6\) have \(S7: A \neq 0\) \(\rightarrow\)

\[(A \cdot B) = (A \cdot 0) \leftrightarrow B = 0\] by (rule MMI_mpri)

from \(A1\) have \(S8: A \in C\).

from \(S8\) have \(S9: (A \cdot 0) = 0\) by (rule MMI_mul01)

from \(S9\) have \(S10: (A \cdot B) = (A \cdot 0) \leftrightarrow (A \cdot B) = 0\) by (rule MMI_eqeq1i)

from \(S7\) \(S10\) have \(S11: A \neq 0 \rightarrow ((A \cdot B) = 0 \leftrightarrow B = 0)\) by (rule MMI_syl15bri)

from \(S11\) have \(S12: A \neq 0 \rightarrow ((A \cdot B) = 0 \rightarrow B = 0)\) by (rule MMI_bimpd)

from \(S11\) \(S12\) have \(S13: \neg (A = 0) \rightarrow ((A \cdot B) = 0 \rightarrow B = 0)\) by (rule MMI_sylbri)

from \(S13\) have \(S14: (A \cdot B) = 0\)
\[ 0 \rightarrow ( \neg ( A = 0 ) \rightarrow B = 0 ) \text{ by (rule MMI_com12)} \]

from S14 have S15: \((A \cdot B) = 0 \rightarrow (A = 0 \lor B = 0)\) by (rule MMI_orrd)

have S16: \(A = 0 \rightarrow (A \cdot B) = (0 \cdot B)\) by (rule MMI_opreq1)

from A2 have S17: \(B \in C\).

from S17 have S18: \((0 \cdot B) = 0\) by (rule MMI_mul02)

from S16 S18 have S19: \(A = 0 \rightarrow (A \cdot B) = 0\) by (rule MMI_syl6eq)

have S20: \(B = 0 \rightarrow (A \cdot B) = (A \cdot 0)\) by (rule MMI_opreq2)

from S9 have S21: \((A \cdot 0) = 0\).

from S20 S21 have S22: \(B = 0 \rightarrow (A \cdot B) = 0\) by (rule MMI_syl6eq)

from S19 S22 have S23: \((A = 0 \lor B = 0) \rightarrow (A \cdot B) = 0\) by (rule MMI_jaoi)

from S15 S23 show \((A \cdot B) = 0 \leftrightarrow (A = 0 \lor B = 0)\) by (rule MMI_impbi)

Qed

lemma (in MMIar0) MMI_maq0: assumes A1: \(A \in C\)

shows \((A \cdot A) = 0 \leftrightarrow A = 0\)

proof -

from A1 have S1: \(A \in C\).

from A1 have S2: \(A \in C\).

from S1 S2 have S3: \((A \cdot A) = 0 \leftrightarrow (A = 0 \lor A = 0)\) by (rule MMI_mul0or)

have S4: \((A = 0 \lor A = 0) \leftrightarrow A = 0\) by (rule MMI_oridm)

from S3 S4 show \((A \cdot A) = 0 \leftrightarrow A = 0\) by (rule MMI_bitr)

Qed

lemma (in MMIar0) MMI_mul0ort:

shows \((A \in C \land B \in C) \rightarrow ((A \cdot B) = 0 \leftrightarrow (A = 0 \lor B = 0))\)

proof -

have S1: \(A = \)

if \((A \in C, A, 0) \rightarrow (A \cdot B) =\)

( if \((A \in C, A, 0) \cdot B)\) by (rule MMI_opreq1)

from S1 have S2: \(A =\)

if \((A \in C, A, 0) \rightarrow ((A \cdot B) =\)

( if \((A \in C, A, 0) \cdot B) = 0)\) by (rule MMI_syl6eq)

have S3: \(A =\)

if \((A \in C, A, 0) \rightarrow (A = 0 \leftrightarrow \text{if } (A \in C, A, 0) = 0)\) by (rule MMI_eqeq1)

from S3 have S4: \(A =\)

if \((A \in C, A, 0) \rightarrow ((A = 0 \lor B = 0) \leftrightarrow (if (A \in C, A, 0) = 0 \lor B = 0))\) by (rule MMI_orbiid)

from S2 S4 have S5: \(A =\)

if \((A \in C, A, 0) \rightarrow ((if (A \in C, A, 0) \cdot B) =\)

(0 \rightarrow (if (A \in C, A, 0) \rightarrow (A = 0 \lor B = 0)) \rightarrow \)

( if \((A \in C, A, 0) \cdot B) =\)

0 \rightarrow \)

1509
( if ( A ∈ C , A , 0 ) = 0 \lor B = 0 ) ) by (rule MMI_bibi12d)
  have S6: B =
  if ( B ∈ C , B , 0 ) →
  ( ( if ( A ∈ C , A , 0 ) \cdot B ) =
  ( if ( A ∈ C , A , 0 ) \cdot B ) ) by (rule MMI_opreq2)
  from S6 have S7: B =
  if ( B ∈ C , B , 0 ) →
  ( ( if ( A ∈ C , A , 0 ) \cdot B ) =
  0 ) by (rule MMI_equeq1d)
  have S8: B =
  if ( B ∈ C , B , 0 ) →
  ( B = 0 \iff ( B ∈ C , B , 0 ) = 0 ) by (rule MMI_equeq1)
  from S8 have S9: B =
  if ( B ∈ C , B , 0 ) →
  ( ( if ( A ∈ C , A , 0 ) = 0 \lor B = 0 ) \iff ( if ( A ∈ C , A , 0 ) =
  0 \lor if ( B ∈ C , B , 0 ) = 0 ) ) by (rule MMI_orbi2d)
  from S7 S9 have S10: B =
  if ( B ∈ C , B , 0 ) →
  ( ( ( if ( A ∈ C , A , 0 ) \cdot B ) = 0 \iff ( if ( A ∈ C , A , 0 ) = 0
  \lor B = 0 ) ) \iff ( ( if ( A ∈ C , A , 0 ) \cdot B ) =
  0 ) by (rule MMI_bibi12d)
  from S11 have S12: if ( A ∈ C , A , 0 ) \cdot B = 0 \iff ( if ( A ∈ C , A , 0 ) = 0
  \lor B = 0 ) ) \iff ( ( if ( A ∈ C , A , 0 ) \cdot B ) =
  0 by (rule MMI_elimel)
  from S12 S14 have S15: ( if ( A ∈ C , A , 0 ) \cdot B = 0 \iff ( if ( A ∈ C , A , 0 ) =
  0 \lor B = 0 ) ) by (rule MMI_dedth2h)
qed

lemma (in MMIIsar0) MMI_muln0bt:
  shows ( A ∈ C \land B ∈ C ) \iff
  ( A \neq 0 \land B \neq 0 ) \iff ( A \cdot B ) \neq 0 )
proof -
  have S1: ( A ∈ C \land B ∈ C ) →
  ( ( A \cdot B ) = 0 \iff ( A = 0 \lor B = 0 ) ) by (rule MMI_mul0ort)
from S1 have S2: \(( A \in C \land B \in C ) \rightarrow \\
\left( \neg \left( ( A \cdot B ) = 0 \right) \right) \leftrightarrow \neg \left( ( A = 0 \lor B = 0 ) \right) \) by (rule MMI_negbid)

have S3: \( \neg \left( ( A = 0 ) \land \neg \left( B = 0 \right) \right) \) by (rule MMI_ioran)

from S2 S3 have S4: \(( A \in C \land B \in C ) \rightarrow \\
\left( \neg \left( ( A \cdot B ) = 0 \right) \right) \leftrightarrow \neg \left( ( A = 0 \lor B = 0 ) \right) \) by (rule MMI_syl6rbb)

have S5: \( A \neq 0 \leftrightarrow \neg \left( A = 0 \right) \) by (rule MMI_df_ne)

have S6: \( B \neq 0 \leftrightarrow \neg \left( B = 0 \right) \) by (rule MMI_df_ne)

from S5 S6 have S7: \(( A \neq 0 \land B \neq 0 ) \leftrightarrow ( A \neq 0 \land B \neq 0 ) \leftrightarrow \neg \left( ( A \cdot B ) = 0 \right) \) by (rule MMI_df_ne)

from S4 S7 S8 show \(( A \in C \land B \in C ) \rightarrow \) by (rule MMI_3bitr4g)

qed

lemma (in MMIar0) MMI_muln0: assumes A1: \( A \in C \) and 
A2: \( B \in C \) and
A3: \( A \neq 0 \) and
A4: \( B \neq 0 \)
shows \( ( A \cdot B ) \neq 0 \)

proof -

from A1 have S1: \( A \in C \).
from A2 have S2: \( B \in C \).
from A3 have S3: \( A \neq 0 \).
from A4 have S4: \( B \neq 0 \).
from S3 S4 have S5: \( A \neq 0 \land B \neq 0 \) by (rule MMI_ps3_2i)

have S6: \(( A \in C \land B \in C ) \rightarrow \\
\left( ( A \neq 0 \land B \neq 0 ) \leftrightarrow ( A \cdot B ) \neq 0 \right) \) by (rule MMI_muln0bt)

from S5 S6 have S7: \(( A \in C \land B \in C ) \rightarrow ( A \cdot B ) \neq 0 \) by (rule MMI_mpbi)

from S1 S2 S7 show \(( A \cdot B ) \neq 0 \) by (rule MMI_mp2an)

qed

lemma (in MMIar0) MMI_receu: assumes A1: \( A \in C \) and 
A2: \( B \in C \) and
A3: \( A \neq 0 \)
shows \( \exists \! x . x \in C \land ( A \cdot x ) = B \)

proof -

{ fix \( x \) \( y \)

have S1: \( x = y \rightarrow ( A \cdot x ) = ( A \cdot y ) \) by (rule MMI_opreq2)

from S1 have S2: \( x = y \rightarrow ( A \cdot x ) = B \leftrightarrow ( A \cdot y ) = B \) \( \leftrightarrow \) (rule MMI_eqeq1d)

} then have S2: \( \forall x . x = y \rightarrow ( ( A \cdot x ) = B \leftrightarrow ( A \cdot y ) = B \) \)

by simp

from S2 have S3: \( ( \exists ! x . x \in C \land ( A \cdot x ) = B ) \leftrightarrow \)
( ∃ x ∈ C . ( A · x ) = B ) ∧
( ∀ x ∈ C . ∀ y ∈ C . ( ( ( A · x ) = B ) ∧ ( A · y ) = B ) → x = y ) )

by (rule MMI_reu4)
from A1 have S4: A ∈ C.
from A3 have S5: A ≠ 0.
from S4 S5 have S6: ∃ y ∈ C . ( A · y ) = 1 by (rule MMI_recex)
from A2 have S7: B ∈ C.
{ fix y have S8: ( y ∈ C ∧ B ∈ C ) → ( y · B ) ∈ C by (rule MMI_axmulcl)
from S7 S8 have S9: y ∈ C → ( y · B ) ∈ C by (rule MMI_mpan2)
have S10: ( y · B ) ∈ C ←→
( ∃ x ∈ C . x = ( y · B ) ) by (rule MMI_risset)
from S9 S10 have S11: y ∈ C → ( ∃ x ∈ C . x = ( y · B ) )
by (rule MMI_sylib)
{ fix x have S12: x = ( y · B ) →
( A · x ) = ( A · ( y · B ) ) by (rule MMI_opreq2)
from A1 have S13: A ∈ C.
from A2 have S14: B ∈ C.
have S15: ( A ∈ C ∧ y ∈ C ∧ B ∈ C ) →
( ( A · y ) · B ) = ( ( A · ( y · B ) ) by (rule MMI_axmulass)
from S13 S14 S15 have S16: y ∈ C →
( ( A · y ) · B ) = ( ( A · ( y · B ) ) by (rule MMI_mp3an13)
from S16 have S17: y ∈ C →
( A · ( y · B ) ) = ( ( A · y ) · B ) by (rule MMI_eqcomd)
from S12 S17 have S18: ( y ∈ C ∧ x =
( y · B ) ) →
( A · x ) = ( ( A · y ) · B ) by (rule MMI_sylan9eqr)
have S19: A · y ) =
1 → ( ( A · y ) · B ) = ( 1 · B ) by (rule MMI_opreq1)
from A2 have S20: B ∈ C.
from S20 have S21: ( 1 · B ) = B by (rule MMI_mulid2)
from S19 S21 have S22: ( A · y ) = 1 → ( ( A · y ) · B ) = B
by (rule MMI_syl6eq)
from S18 S22 have S23:
( ( A · y ) = 1 ∧ ( y ∈ C ∧ x =
( y · B ) ) ) → ( A · x ) = B by (rule MMI_sylan9eqr)
from S23 have S24:
( A · y ) = 1 → ( y ∈ C →
( x = ( y · B ) → ( A · x ) = B ) ) by (rule MMI_exp32)
from S24 have S25: ( y ∈ C ∧ ( A · y ) =
1 ) →
( x = ( y · B ) → ( A · x ) = B ) by (rule MMI_impcom)
from S25 have
( y ∈ C ∧ ( A · y ) = 1 ) → ( x ∈ C →
( x = ( y · B ) → ( A · x ) = B ) ) by (rule MMI_a1d)
} then have S26:
∀ x . ( y ∈ C ∧ ( A · y ) = 1 ) → ( x ∈ C →
1512
\[(x = (y \cdot B) \rightarrow (A \cdot x) = B)\] by simp

from S26 have S27:
\[(\forall x \in C. (x = (y \cdot B) \rightarrow (A \cdot x) = B))\] by (rule MMI_r19_21aiv)

from S27 have S28: \(y \in C\) \(\rightarrow\)
\[(\exists x \in C. (x = (y \cdot B) \rightarrow (A \cdot x) = B))\] by (rule MMI_ex)

have S29: \((\forall x \in C. (x = (y \cdot B) \rightarrow (A \cdot x) = B))\) by (rule MMI_r19_22)

from S28 S29 have S30:
\[(\exists x \in C. (A \cdot y) = 1) \rightarrow (\exists x \in C. (A \cdot x) = B)\] by (rule MMI_syl5bi)

from S3 have S31: \(\forall y . y \in\) \(\rightarrow\)
\[(\exists x \in C. (A \cdot y) = 1) \rightarrow (\exists x \in C. (A \cdot x) = B)\] by auto

from S31 have S32: \(\exists x \in C. (A \cdot y) = 1\) by (rule MMI_r19_23aiv)

from S32 have S33: \(\exists x \in C. (A \cdot x) = B\) by (rule MMI_ax_mp)

from A1 have S34: \(A \in C\).

from A3 have S35: \(A \neq 0\).

\{ fix x y \}

from S35 have S36: \((A \in C \land x \in C \land y \in C) \rightarrow\)
\[( (A \cdot x) = (A \cdot y) \leftrightarrow x = y)\] by (rule MMI_mulcant2)

have S37:
\[( (A \cdot x) = B \land (A \cdot y) = B) \rightarrow (A \cdot x) = (A \cdot y)\] by (rule MMI_eqtr3t)

from S36 S37 have S38: \((A \in C \land x \in C \land y \in C) \rightarrow\)
\[( (A \cdot x) = B \land (A \cdot y) = B) \rightarrow x = y\] by (rule MMI_mp3an1)

from S34 S38 have \((x \in C \land y \in C) \rightarrow\)
\[( (A \cdot x) = B \land (A \cdot y) = B) \rightarrow x = y\] by auto

from S39 have S40:
\[% x \in C . \forall y \in C . ( (A \cdot x) = B \land (A \cdot y) = B) \rightarrow x = y\] by (rule MMI_mpbir2an)

qed
lemma (in MMIas0) MMI_divval: assumes A ∈ C B ∈ C B ≠ 0
shows A / B = ∪ { x ∈ C . B · x = A }
using cdiv_def by simp

lemma (in MMIas0) MMI_divmul: assumes A1: A ∈ C and
A2: B ∈ C and
A3: C ∈ C and
A4: B ≠ 0
shows ( A / B ) = C ←→ ( B · C ) = A
proof -
  from A3 have S1: C ∈ C.
  { fix x
    have S2: x = C ----> ( ( A / B ) = x ----> ( A / B ) = C ) by (rule MMI_eqeq2)
    have S3: x = C ----> ( B · x ) = ( B · C ) by (rule MMI_opreq2)
    from S3 have S4: x = C ----> ( ( B · x ) = A ----> ( B · C ) = A ) by (rule MMI_eqeq1d)
    from S2 S4 have x = C ---->
       ( ( A / B ) = x ----> ( B · x ) = A ----> ( A / B ) = C ----> ( B · C ) = A )
       by (rule MMI_bibi12d)
  } then have S5: ∀ x. x = C ---->
       ( ( A / B ) = x ----> ( B · x ) = A ---->
       ( A / B ) = C ----> ( B · C ) = A )
       by simp
  from A2 have S6: B ∈ C.
  from A1 have S7: A ∈ C.
  from A4 have S8: B ≠ 0.
  from S6 S7 S8 have S9: ∃! x . x ∈ C ∧ ( B · x ) = A by (rule MMI_receu)
  { fix x
    have S10: ( x ∈ C ∧ ( ∃! x . x ∈ C ∧ ( B · x ) = A ) ) ---->
       ( ( B · x ) = A ----> ∪ { x ∈ C . ( B · x ) = A } = x ) by (rule MMI_reuuni1)
    from S9 S10 have
       x ∈ C ----> ( ( B · x ) = A ----> ∪ { x ∈ C . ( B · x ) = A } = x )
       by (rule MMI_mpan2)
  } then have S11:
    ∀ x. x ∈ C ----> ( ( B · x ) = A ----> ∪ { x ∈ C . ( B · x ) = A } = x )
    by blast
  from A1 have S12: A ∈ C.
  from A2 have S13: B ∈ C.
  from A4 have S14: B ≠ 0.
  from S12 S13 S14 have S15: ( A / B ) =
\[ \{ x \in C . ( B \cdot x ) = A \} \] by (rule MMI_divval)

from S15 have S16: \( \forall x . ( A / B ) = x \) \( \Longleftrightarrow \) \( \{ x \in C . ( B \cdot x ) = A \} = x \) by simp

from S11 S16 have S17: \( \forall x . x \in C \) \( \Rightarrow \) \( ( A / B ) = x \) \( \Longleftrightarrow \) \( ( B \cdot x ) = A \) by (rule MMI_syl6rrbbr)

from S5 S17 have S18: \( \forall x . x \in C \) \( \Rightarrow \) \( ( A / B ) = x \) \( \Longleftrightarrow \) \( ( B \cdot x ) = A \) by (rule MMI_vtoclga)

from S1 S18 show \( ( A / B ) = C \) \( 
\Longleftrightarrow \) \( ( B \cdot C ) = A \) by (rule MMI_ax_mp)

qed

lemma (in MMIIsar0) MMI_divmulz: assumes A1: \( A \in C \) and
A2: \( B \in C \) and
A3: \( C \in C \)
shows \( B \neq 0 \) \( \Rightarrow \) \( ( A / B ) = C \) \( 
\Longleftrightarrow \) \( ( B \cdot C ) = A \)
proof -

have S1: \( B = \)
if \( ( B \neq 0 , B , 1 ) \) \( \Rightarrow \)
( A / B ) =
( A / if \( ( B \neq 0 , B , 1 ) \) ) by (rule MMI_opreq2)

from S1 have S2: \( B = \)
if \( ( B \neq 0 , B , 1 ) \) \( \Rightarrow \)
( A / B ) =
C \( \Longleftrightarrow \) \( ( A / if \( ( B \neq 0 , B , 1 ) \) ) = C \) by (rule MMI_eqeq1d)

have S3: \( B = \)
if \( ( B \neq 0 , B , 1 ) \) \( \Rightarrow \)
( B \cdot C ) =
( if \( ( B \neq 0 , B , 1 ) \cdot C ) = A \) by (rule MMI_opreq1)

from S3 have S4: \( B = \)
if \( ( B \neq 0 , B , 1 ) \) \( \Rightarrow \)
( ( B \cdot C ) =
A \( \Longleftrightarrow \) \( ( ( A / B ) = C \) \( \Longleftrightarrow \) \( ( B \cdot C ) = A \) \( \Longleftrightarrow \)
( ( A / if \( ( B \neq 0 , B , 1 ) ) ) =
C \( \Longleftrightarrow \) \( ( ( B \neq 0 , B , 1 ) \cdot C ) = A \) \) by (rule MMI_bibi12d)

from A1 have S6: \( A \in C \).
from A2 have S7: \( B \in C \).

have S8: \( 1 \in C \) by (rule MMI_1cn)

from S7 S8 have S9: if \( ( B \neq 0 , B , 1 ) \in C \) by (rule MMI_keepel)

from A3 have S10: \( C \in C \).

have S11: if \( ( B \neq 0 , B , 1 ) \neq 0 \) by (rule MMI_elimne0)

from S6 S9 S10 S11 have S12: \( ( A / if \( ( B \neq 0 , B , 1 ) ) ) =
C \( \Longleftrightarrow \) \( ( ( B \neq 0 , B , 1 ) \cdot C ) = A \) by (rule MMI_divmul)

from S5 S12 show \( B \neq 0 \) \( \Rightarrow \)
( ( A / B ) = C \( \Longleftrightarrow \) \( ( B \cdot C ) = A \) ) by (rule MMI_dedth)

qed
lemma \textit{(in MMIsar0)} $\text{MMI_divmult}$:
\begin{itemize}
  \item shows $( ( A \in C \land B \in C \land C \in C ) \land B \neq 0 ) \rightarrow
  ( ( A / B ) = C \iff ( B \cdot C ) = A )$
\end{itemize}
proof
  have S1: $A =$
    \begin{itemize}
      \item if $( A \in C , A , 0 ) \rightarrow
        ( A / B ) =$
        \begin{itemize}
          \item if $( A \in C , A , 0 ) / B )$ by (rule $\text{MMI_opreq1}$)
        \end{itemize}
    \end{itemize}
  from S1 have S2: $A =$
    \begin{itemize}
      \item if $( A \in C , A , 0 ) \rightarrow
        ( ( A / B ) =$
        \begin{itemize}
          \item $C \iff ( A \in C , A , 0 ) / B ) = C )$ by (rule $\text{MMI_eqeq1d}$)
        \end{itemize}
    \end{itemize}
  have S3: $A =$
    \begin{itemize}
      \item if $( A \in C , A , 0 ) \rightarrow
        ( ( B \cdot C ) =$
        \begin{itemize}
          \item $A \iff ( ( A \in C , A , 0 ) / B ) =$
          \item $C \iff ( ( A \in C , A , 0 ) / B ) =$
        \end{itemize}
    \end{itemize}
  from S2 S3 have S4: $A =$
    \begin{itemize}
      \item if $( A \in C , A , 0 ) \rightarrow
        ( ( B \cdot C ) =$
        \begin{itemize}
          \item $if ( A \in C , A , 0 ) ) )$ by (rule $\text{MMI_bibi12d}$)
        \end{itemize}
    \end{itemize}
  from S4 have S5: $A =$
    \begin{itemize}
      \item if $( A \in C , A , 0 ) \rightarrow
        ( ( B \cdot C ) =$
        \begin{itemize}
          \item $if ( A \in C , A , 0 ) / B ) =$
        \end{itemize}
    \end{itemize}
  have S6: $B =$
    \begin{itemize}
      \item if $( B \in C , B , 0 ) \rightarrow
        ( B \neq 0 \iff if ( B \in C , B , 0 ) \neq 0 )$ by (rule $\text{MMI_neeq1}$)
    \end{itemize}
  have S7: $B =$
    \begin{itemize}
      \item if $( B \in C , B , 0 ) \rightarrow
        ( ( A \in C , A , 0 ) / B ) =$
        \begin{itemize}
          \item $if ( A \in C , A , 0 ) / if ( B \in C , B , 0 )$ by (rule $\text{MMI_opreq2}$)
        \end{itemize}
    \end{itemize}
  from S7 have S8: $B =$
    \begin{itemize}
      \item if $( B \in C , B , 0 ) \rightarrow
        ( ( if ( A \in C , A , 0 ) / B ) =$
        \begin{itemize}
          \item $if ( A \in C , A , 0 ) / if ( B \in C , B , 0 ) ) = C )$ by (rule $\text{MMI_eqe1d}$)
        \end{itemize}
    \end{itemize}
  have S9: $B =$
    \begin{itemize}
      \item if $( B \in C , B , 0 ) \rightarrow
        ( B \cdot C ) =$
        \begin{itemize}
          \item $if ( B \in C , B , 0 ) \cdot C )$ by (rule $\text{MMI_opreq1}$)
        \end{itemize}
    \end{itemize}
  from S9 have S10: $B =$
    \begin{itemize}
      \item if $( B \in C , B , 0 ) \rightarrow
        ( B \cdot C ) =$
        \begin{itemize}
          \item $if ( B \in C , B , 0 ) \cdot C )$ by (rule $\text{MMI_opreq1}$)
        \end{itemize}
    \end{itemize}
\[
( ( B \cdot C ) = \\
\text{if} ( A \in C, A, 0 ) \iff \\
( \text{if} ( B \in C, B , 0 ) \cdot C ) = \\
\text{if} ( A \in C, A, 0 ) ) \text{ by (rule MMI_eqeq1d)}
\]

from S8 S10 have S11: \( B = \\
\text{if} ( B \in C, B, 0 ) \rightarrow \\
( ( ( \text{if} ( A \in C, A, 0 ) / B ) = C \leftrightarrow ( B \cdot C ) = \text{if} ( A \in C, A, 0 ) ) \leftrightarrow \\
( \text{if} ( A \in C, A, 0 ) / \text{if} ( B \in C, B, 0 ) ) = \\
C \leftrightarrow \\
( \text{if} ( B \in C, B, 0 ) \cdot C ) = \\
\text{if} ( A \in C, A, 0 ) ) ) \text{ by (rule MMI_bibi12d)}

from S6 S11 have S12: \( B = \\
\text{if} ( B \in C, B, 0 ) \rightarrow \\
( ( B \neq 0 \rightarrow ( ( \text{if} ( A \in C, A, 0 ) / B ) = C \leftrightarrow ( B \cdot C ) = \text{if} ( A \in C, A, 0 ) ) ) \leftrightarrow \\
( \text{if} ( B \in C, B, 0 ) \neq 0 \rightarrow \\
( ( \text{if} ( A \in C, A, 0 ) / \text{if} ( B \in C, B, 0 ) ) = \\
C \leftrightarrow \\
( \text{if} ( B \in C, B, 0 ) \cdot C ) = \\
\text{if} ( A \in C, A, 0 ) ) ) ) \text{ by (rule MMI_imbi12d)}

have S13: \( C = \\
\text{if} ( C \in C, C, 0 ) \rightarrow \\
( ( \text{if} ( A \in C, A, 0 ) / \text{if} ( B \in C, B, 0 ) ) = \\
C \leftrightarrow \\
( \text{if} ( A \in C, A, 0 ) / \text{if} ( B \in C, B, 0 ) ) = \\
\text{if} ( C \in C, C, 0 ) ) ) \text{ by (rule MMI_eqeq2)}

have S14: \( C = \\
\text{if} ( C \in C, C, 0 ) \rightarrow \\
( ( \text{if} ( B \in C, B, 0 ) \cdot C ) = \\
( \text{if} ( B \in C, B, 0 ) \cdot \text{if} ( C \in C, C, 0 ) ) ) \text{ by (rule MMI_opreq2)}

from S14 have S15: \( C = \\
\text{if} ( C \in C, C, 0 ) \rightarrow \\
( ( \text{if} ( B \in C, B, 0 ) \cdot C ) = \\
( \text{if} ( A \in C, A, 0 ) \rightarrow \\
( \text{if} ( B \in C, B, 0 ) \cdot \text{if} ( C \in C, C, 0 ) ) = \\
\text{if} ( A \in C, A, 0 ) ) ) \text{ by (rule MMI_eqeq1d)}

from S13 S15 have S16: \( C = \\
( ( ( \text{if} ( A \in C, A, 0 ) / \text{if} ( B \in C, B, 0 ) ) = C \leftrightarrow ( \text{if} ( B \in C, B, 0 ) \cdot C ) = \text{if} ( A \in C, A, 0 ) ) \leftrightarrow \\
( \text{if} ( A \in C, A, 0 ) \rightarrow \\
( \text{if} ( C \in C, C, 0 ) \rightarrow \\
( \text{if} ( B \in C, B, 0 ) \cdot \text{if} ( C \in C, C, 0 ) ) = \\
\text{if} ( A \in C, A, 0 ) ) ) \text{ by (rule MMI_bibi12d)}

from S16 have S17: \( C = \\
\text{if} ( C \in C, C, 0 ) \rightarrow \\
( ( \text{if} ( B \in C, B, 0 ) \neq 0 \rightarrow ( ( \text{if} ( A \in C, A, 0 ) / \text{if} ( B \in C, B, 0 ) ) = C \leftrightarrow ( \text{if} ( B \in C, B, 0 ) \cdot C ) = \text{if} ( A \in C, A, 0 ) ) \leftrightarrow \\
( \text{if} ( B \in C, B, 0 ) \cdot \text{if} ( C \in C, C, 0 ) ) = \\
\text{if} ( A \in C, A, 0 ) ) ) \text{ by (rule MMI_bibi12d)}

1517
A , 0 ) ) \begin{proof}
\begin{itemize}
    \item if ( B ∈ C , 0 ) \neq 0 \quad → \quad if ( A ∈ C , 0 )
\end{itemize}
\end{proof}
from S1 S5 show \(( ( A \in C \land B \in C \land C \in C ) \land B \neq 0 ) \rightarrow \(( A / B ) = C \leftarrow\rightarrow A = ( C \cdot B ) )\) by (rule MMI_bitrd)

qed

lemma (in MMIIsar0) MMI_divcl: assumes A1: A \in C and 
A2: B \in C and 
A3: B \neq 0 
shows \(( A / B ) \in C\)

proof - 
from A1 have S1: A \in C.
from A2 have S2: B \in C.
from A3 have S3: B \neq 0.
from S1 S2 S3 have S4: \(( A / B ) = \bigcup \{ x \in C . ( B \cdot x ) = A \} \) by (rule MMI_divval)
from A2 have S5: B \in C.
from A1 have S6: A \in C.
from A3 have S7: B \neq 0.
from S5 S6 S7 have S8: \( \exists ! x . x \in C \land ( B \cdot x ) = A \) by (rule MMI_receu)
have S9: \( \exists ! x . x \in C \land ( B \cdot x ) = A \) \rightarrow \( \bigcup \{ x \in C . ( B \cdot x ) = A \} \) \in C by (rule MMI_reucl)
from S8 S9 have S10: \( \bigcup \{ x \in C . ( B \cdot x ) = A \} \) \in C by (rule MMI_ax_mp)
from S4 S10 show \(( A / B ) \in C\) by (rule MMI_eqeltr)

qed

lemma (in MMIIsar0) MMI_divclz: assumes A1: A \in C and 
A2: B \in C 
shows B \neq 0 \rightarrow \(( A / B ) \in C\)

proof - 
have S1: B =
if ( B \neq 0 , B , 1 ) \rightarrow
( A / B ) =
( A / if ( B \neq 0 , B , 1 ) ) by (rule MMI_opreq2)
from S1 have S2: B =
if ( B \neq 0 , B , 1 ) \rightarrow
( ( A / B ) \in C \leftarrow\rightarrow
( A / if ( B \neq 0 , B , 1 ) ) \in C ) by (rule MMI_numel2d)
from A1 have S3: A \in C.
from A2 have S4: B \in C.
have S5: 1 \in C by (rule MMI_1cn)
from S4 S5 have S6: if ( B \neq 0 , B , 1 ) \in C by (rule MMI_keepel)
have S7: if ( B \neq 0 , B , 1 ) \neq 0 by (rule MMI_elimne0)
from S3 S6 S7 have S9: \( ( A / if ( B \neq 0 , B , 1 ) ) \in C \) by (rule MMI_divcl)

from S2 S8 show B \neq 0 \rightarrow \(( A / B ) \in C\) by (rule MMI_dedth)

qed
lemma (in MMIsar0) MMI_divclt:
  shows \( (A \in C \land B \in C \land B \neq 0) \rightarrow (A / B) \in C \)
proof -
  have S1: A =
  \( \text{if} (A \in C, A, 0) \rightarrow \) 
  (A / B) =
  \( \text{if} (A \in C, A, 0) / B) \) by (rule MMI_opreq1)
  from S1 have S2: A =
  \( \text{if} (A \in C, A, 0) \rightarrow \) 
  ((A / B) \in C \leftarrow)
  \( \text{if} (A \in C, A, 0) / B) \) by (rule MMI_eleq1d)
  from S2 have S3: A =
  \( \text{if} (B \neq 0 \rightarrow (A / B) \in C) \leftarrow \) 
  \( B \neq 0 \rightarrow \) 
  \( \text{if} (A \in C, A, 0) / B \in C \) ) by (rule MMI_imbi2d)
  have S4: B =
  \( \text{if} (B \in C, B, 0) \rightarrow \) 
  \( B \neq 0 \rightarrow \text{if} (B \in C, B, 0) \neq 0 \) by (rule MMI_neeq1)
  have S5: B =
  \( \text{if} (B \in C, B, 0) \rightarrow \) 
  \( \text{if} (B \in C, B, 0) / B) = \) 
  \( \text{if} (B \in C, B, 0) / \text{id} (B \in C, B, 0) \) ) by (rule MMI_opreq2)
  from S5 have S6: B =
  \( \text{if} (B \in C, B, 0) \rightarrow \) 
  \( \text{if} (B \in C, B, 0) \rightarrow \) 
  \( \text{if} (B \in C, B, 0) / B) \in C \leftarrow \) 
  \( \text{if} (B \in C, B, 0) / \text{id} (B \in C, B, 0) \) ) by (rule MMI_eleq1d)
  from S4 S6 have S7: B =
  \( \text{if} (B \in C, B, 0) \rightarrow \) 
  \( \text{if} (B \in C, B, 0) \rightarrow \) 
  \( \text{if} (B \in C, B, 0) \rightarrow \) 
  \( \text{if} (B \in C, B, 0) / B = \) 
  \( \text{if} (B \in C, B, 0) / \text{id} (B \in C, B, 0) \) ) by (rule MMI_imbi12d)
  have S8: 0 \in C by (rule MMI_0cn)
  from S8 have S9: if (A \in C, A, 0) \in C by (rule MMI_elimel)
  have S10: 0 \in C by (rule MMI_0cn)
  from S10 have S11: if (B \in C, B, 0) \in C by (rule MMI_elimel)
  from S9 S11 have S12: if (B \in C, B, 0) \neq 0 \rightarrow 
  \( \text{if} (A \in C, A, 0) / \text{id} (B \in C, B, 0) \) \in C by (rule MMI_divclz)
  from S3 S7 S12 have S13: (A \in C \land B \in C) 
  \( \text{if} (B \neq 0 \rightarrow \) 
  \( \text{A} / B) \in C \) ) by (rule MMI_dedth2h)
  from S13 have S14: (A \in C \land B \in C \land B \neq 0) \rightarrow 
  \( \text{if} (A \in C, A, 0) / \text{id} (B \in C, B, 0) \) \in C by (rule MMI_3impia)
  have S15: 0 \in C by (rule MMI_0cn)
  from S15 have S16: if (A \in C, A, 0) \in C by (rule MMI_elimel)
  have S17: 0 \in C by (rule MMI_0cn)
  from S17 have S18: if (A \in C, A, 0) \neq 0 \rightarrow 
  \( \text{if} (A \in C, A, 0) / \text{id} (A \in C, A, 0) \) \in C by (rule MMI_divclz)
  from S3 S7 S18 have S19: (A \in C \land B \in C) 
  \( \text{if} (A \neq 0 \rightarrow \) 
  \( \text{A} / B) \in C \) ) by (rule MMI_dedth2h)
  from S19 have S20: (A \in C \land B \in C \land B \neq 0) \rightarrow 
  \( \text{if} (A \in C, A, 0) / \text{id} (A \in C, A, 0) \) \in C by (rule MMI_3impia)
  qed

lemma (in MMIsar0) MMI_reccl: assumes A1: A \in C and 
  A2: A \neq 0
  shows \( (1 / A) \in C \)
proof -
have S1: 1 ∈ C by (rule MMI_1cn)
from A1 have S2: A ∈ C.
from A2 have S3: A ≠ 0.
from S1 S2 S3 show ( 1 / A ) ∈ C by (rule MMI_divcl)
qed

lemma (in MMIsar0) MMI_reclz: assumes A1: A ∈ C
shows A ≠ 0 −→ ( 1 / A ) ∈ C
proof -
  have S1: 1 ∈ C by (rule MMI_1cn)
  from A1 have S2: A ∈ C.
  from S1 S2 show A ≠ 0 −→ ( 1 / A ) ∈ C by (rule MMI_divclz)
qed

lemma (in MMIsar0) MMI_reclt:
  shows ( A ∈ C ∧ A ≠ 0 ) −→ ( 1 / A ) ∈ C
proof -
  have S1: 1 ∈ C by (rule MMI_1cn)
  have S2: ( 1 ∈ C ∧ A ∈ C ∧ A ≠ 0 ) −→
         ( 1 / A ) ∈ C by (rule MMI_divclt)
  from S1 S2 show ( A ∈ C ∧ A ≠ 0 ) −→ ( 1 / A ) ∈ C by (rule MMI_mp3an1)
qed

lemma (in MMIsar0) MMI_divcan2: assumes A1: A ∈ C and
                             A2: B ∈ C and
                             A3: A ≠ 0
shows ( A · ( B / A ) ) = B
proof -
  have S1: ( B / A ) = ( B / A ) by (rule MMI_eqid)
  from A2 have S2: B ∈ C.
  from A1 have S3: A ∈ C.
  from A2 have S4: B ∈ C.
  from A1 have S5: A ∈ C.
  from A3 have S6: A ≠ 0.
  from S4 S5 S6 have S7: ( B / A ) ∈ C by (rule MMI_divcl)
  from A3 have S8: A ≠ 0.
  from S2 S3 S7 S8 have S9: ( B / A ) =
         ( B / A ) −→ ( A · ( B / A ) ) = B by (rule MMI_divmul)
  from S1 S9 show ( A · ( B / A ) ) = B by (rule MMI_mpbi)
qed

lemma (in MMIsar0) MMI_divcan1: assumes A1: A ∈ C and
                             A2: B ∈ C and
                             A3: A ≠ 0
shows ( ( B / A ) · A ) = B
proof -
  from A2 have S1: B ∈ C.
  from A1 have S2: A ∈ C.
  from A3 have S3: A ≠ 0.
from \(S_1\) \(S_2\) \(S_3\) have \(S_4\): \((B / A) \in \mathbb{C}\) by (rule MMI_divcl)
from \(A_1\) have \(S_5\): \(A \in \mathbb{C}\).
from \(S_4\) \(S_5\) have \(S_6\): \((B / A) \cdot A) = (A \cdot (B / A))\) by (rule MMI_mulcom)

from \(A_1\) have \(S_7\): \(A \in \mathbb{C}\).
from \(A_2\) have \(S_8\): \(B \in \mathbb{C}\).
from \(A_3\) have \(S_9\): \(A \neq 0\).
from \(S_7\) \(S_8\) \(S_9\) have \(S_{10}\): \((A \cdot (B / A)) = B\) by (rule MMI_divcan2)
from \(S_6\) \(S_{10}\) show \((B / A) \cdot A) = B\) by (rule MMI_eqtr)

qed

lemma (in MMIIsar0) MMI_divcan1z: assumes \(A_1\): \(A \in \mathbb{C}\) and 
\(A_2\): \(B \in \mathbb{C}\) shows \(A \neq 0 \rightarrow (B / A) \cdot A) = B\)
proof -
    have \(S_1\): \(A = \)
    if \((A \neq 0, A, 1) \rightarrow
    (B / A) = \)
    \((B / if (A \neq 0, A, 1))\) by (rule MMI_opreq2)
    have \(S_2\): \(A = \)
    if \((A \neq 0, A, 1) \rightarrow
    A = if (A \neq 0, A, 1)\) by (rule MMI_id)
    from \(S_1\) \(S_2\) have \(S_3\): \(A = \)
    if \((A \neq 0, A, 1) \rightarrow
    (B / A) \cdot A) = \)
    \((B / if (A \neq 0, A, 1)) \cdot if (A \neq 0, A, 1)\) by (rule MMI_opreq12d)
    from \(S_3\) have \(S_4\): \(A = \)
    if \((A \neq 0, A, 1) \rightarrow
    ((B / A) \cdot A) = \)
    \(B \leftarrow \)
    \((B / if (A \neq 0, A, 1)) \cdot if (A \neq 0, A, 1)\) = \(B\) by (rule MMI_eqeq1d)
    from \(A_1\) have \(S_5\): \(A \in \mathbb{C}\).
    have \(S_6\): \(1 \in \mathbb{C}\) by (rule MMI_1cn)
    from \(S_5\) \(S_6\) have \(S_7\): \(if (A \neq 0, A, 1) \in \mathbb{C}\) by (rule MMI_keepel)
    from \(A_2\) have \(S_8\): \(B \in \mathbb{C}\).
    have \(S_9\): \(if (A \neq 0, A, 1) \neq 0\) by (rule MMI_elimne0)
    from \(S_7\) \(S_8\) \(S_9\) have \(S_{10}\): \((B / if (A \neq 0, A, 1)) \cdot if (A \neq 0, A, 1)\) = \(B\)
    by (rule MMI_divcan1)
    from \(S_4\) \(S_{10}\) show \(A \neq 0 \rightarrow (B / A) \cdot A) = B\) by (rule MMI_dedth)
qed

lemma (in MMIIsar0) MMI_divcan2z: assumes \(A_1\): \(A \in \mathbb{C}\) and 
\(A_2\): \(B \in \mathbb{C}\) shows \(A \neq 0 \rightarrow (A \cdot (B / A)) = B\)
proof -
    have \(S_1\): \(A = \)
    if \((A \neq 0, A, 1) \rightarrow

1522
A = if ( A \neq 0 , A , 1 ) by (rule MMI_id)

have S2: A =
if ( A \neq 0 , A , 1 ) \rightarrow
(B / A ) =
(B / if ( A \neq 0 , A , 1 ) ) by (rule MMI_opreq2)
from S1 S2 have S3: A =
if ( A \neq 0 , A , 1 ) \rightarrow
(A \cdot ( B / A ) ) =
(if ( A \neq 0 , A , 1 ) \cdot ( B / if ( A \neq 0 , A , 1 ) ) ) by (rule MMI_opreq12d)
from S3 have S4: A =
if ( A \neq 0 , A , 1 ) \rightarrow
(( A \cdot ( B / A ) ) =
B ) by (rule MMI_eqeq1d)
from A1 have S5: A \in \mathcal{F}.

have S6: 1 \in \mathcal{F} by (rule MMI_1cn)
from S5 S6 have S7: if ( A \neq 0 , A , 1 ) \in \mathcal{C} by (rule MMI_keepel)
from A2 have S8: B \in \mathcal{C}.

have S9: if ( A \neq 0 , A , 1 ) \neq 0 by (rule MMI_elimne0)
from S7 S8 S9 have S10: if ( A \neq 0 , A , 1 ) \cdot ( B / if ( A \neq 0 , A , 1 ) ) =
B by (rule MMI_divcan2)
from S4 S10 show A \neq 0 \rightarrow ( A \cdot ( B / A ) ) = B by (rule MMI_dedth)
qed

lemma (in MMIsar0) MMI_divcan1t:
shows ( A \in \mathcal{C} \land B \in \mathcal{C} \land A \neq 0 ) \rightarrow
(( B / A ) \cdot A ) = B
proof -
have S1: A =
if ( A \in \mathcal{C} , A , 0 ) \rightarrow
(A \neq 0 \leftrightarrow if ( A \in \mathcal{C} , A , 0 ) \neq 0 ) by (rule MMI_neeq1)

have S2: A =
if ( A \in \mathcal{C} , A , 0 ) \rightarrow
(B / A ) =
(B / if ( A \in \mathcal{C} , A , 0 ) ) by (rule MMI_opreq2)

have S3: A =
if ( A \in \mathcal{C} , A , 0 ) \rightarrow
A = if ( A \in \mathcal{C} , A , 0 ) by (rule MMI_id)

from S2 S3 have S4: A =
if ( A \in \mathcal{C} , A , 0 ) \rightarrow
(( B / A ) \cdot A ) =
(( B / if ( A \in \mathcal{C} , A , 0 ) ) \cdot if ( A \in \mathcal{C} , A , 0 ) ) by (rule MMI_opreq12d)
from S4 have S5: A =
if ( A \in \mathcal{C} , A , 0 ) \rightarrow
(( ( B / A ) \cdot A ) =
B \leftarrow
d(( B / if ( A \in \mathcal{C} , A , 0 ) ) \cdot if ( A \in \mathcal{C} , A , 0 ) ) =

1523
B ) by (rule MMI_eqeq1d)
  from S1 S5 have S6: A =
  if ( A ∈ C , A , 0 ) →
  ( ( A ≠ 0 → ( ( B / A ) · A ) = B ) ↔
  ( if ( A ∈ C , A , 0 ) ≠ 0 →
  ( ( B / if ( A ∈ C , A , 0 ) ) · if ( A ∈ C , A , 0 ) ) =
  B ) ) by (rule MMI_imbi12d)
  have S7: B =
  if ( B ∈ C , B , 0 ) →
  ( B / if ( A ∈ C , A , 0 ) / if ( A ∈ C , A , 0 ) ) by (rule MMI_opreq1)
  from S7 have S8: B =
  if ( B ∈ C , B , 0 ) →
  ( ( B / if ( A ∈ C , A , 0 ) ) · if ( A ∈ C , A , 0 ) ) =
  ( ( if ( B ∈ C , B , 0 ) / if ( A ∈ C , A , 0 ) ) · if ( A ∈ C , A ,
  0 ) ) by (rule MMI_opreq1d)
  have S9: B =
  if ( B ∈ C , B , 0 ) →
  B = if ( B ∈ C , B , 0 ) by (rule MMI_id)
  from S8 S9 have S10: B =
  if ( B ∈ C , B , 0 ) →
  ( ( B / if ( A ∈ C , A , 0 ) ) · if ( A ∈ C , A , 0 ) ) =
  B →
  ( ( if ( B ∈ C , B , 0 ) / if ( A ∈ C , A , 0 ) ) · if ( A ∈ C , A ,
  0 ) ) =
  if ( B ∈ C , B , 0 ) ) by (rule MMI_eqeq12d)
  from S10 have S11: B =
  if ( B ∈ C , B , 0 ) →
  ( ( if ( A ∈ C , A , 0 ) ≠ 0 → ( ( B / if ( A ∈ C , A , 0 ) ) · if
  ( A ∈ C , A , 0 ) ) = B ) ↔
  ( if ( A ∈ C , A , 0 ) ≠ 0 →
  ( ( if ( B ∈ C , B , 0 ) / if ( A ∈ C , A , 0 ) ) · if ( A ∈ C , A ,
  0 ) ) =
  if ( B ∈ C , B , 0 ) ) ) by (rule MMI_imbi2d)
  have S12: 0 ∈ C by (rule MMI_0cn)
  from S12 have S13: if ( A ∈ C , A , 0 ) ∈ C by (rule MMI_elimel)
  have S14: 0 ∈ C by (rule MMI_0cn)
  from S14 have S15: if ( B ∈ C , B , 0 ) ∈ C by (rule MMI_elimel)
  from S13 S15 have S16: if ( A ∈ C , A , 0 ) ≠ 0 →
  ( ( if ( B ∈ C , B , 0 ) / if ( A ∈ C , A , 0 ) ) · if ( A ∈ C , A ,
  0 ) ) =
  if ( B ∈ C , B , 0 ) ) by (rule MMI_divcan1z)
  from S6 S11 S16 have S17: ( A ∈ C ∧ B ∈ C ) →
  ( A ≠ 0 → ( ( B / A ) · A ) = B ) by (rule MMI_dedth2h)
  from S17 show ( A ∈ C ∧ B ∈ C ∧ A ≠ 0 ) →
  ( ( B / A ) · A ) = B by (rule MMI_3impia)
qed

lemma (in MMIar0) MMI_divcan2t:
\[\text{shows } (A \in C \land B \in C \land A \neq 0) \rightarrow (A \cdot (B / A)) = B\]

proof -

\[\text{have } S1: A =\]

if \((A \in C, A, 0) \rightarrow\)

\((A \neq 0 \iff (A \in C, A, 0) \neq 0)\) by (rule MMI_neeq1)

\[\text{have } S2: A =\]

if \((A \in C, A, 0) \rightarrow\)

\(A = \text{if } (A \in C, A, 0)\) by (rule MM_id)

\[\text{have } S3: A =\]

if \((A \in C, A, 0) \rightarrow\)

\((B / A)\) = \((B / \text{if } (A \in C, A, 0))\) by (rule MMI_opreq2)

\[\text{from } S2 S3 \text{ have } S4: A =\]

if \((A \in C, A, 0) \rightarrow\)

\((A \cdot (B / A))\) = \((\text{if } (A \in C, A, 0) \cdot (B / \text{if } (A \in C, A, 0))\) by (rule MMI_opreq12d)

\[\text{from } S4 \text{ have } S5: A =\]

if \((A \in C, A, 0) \rightarrow\)

\((A \cdot (B / A))\) = \(B \iff (A \cdot (B / A)) = B\) by (rule MMI_imbi12d)

\[\text{from } S1 S5 \text{ have } S6: A =\]

if \((A \in C, A, 0) \rightarrow\)

\((A \neq 0 \iff (A \cdot (B / A)) = B) \iff\)

\((A \in C, A, 0) \neq 0 \rightarrow\)

\((A \cdot (B / A))\) = \((B / \text{if } (A \in C, A, 0))\) by (rule MMI_opreq12d)

\[\text{from } S6 \text{ have } S7: A =\]

if \((B \in C, B, 0) \rightarrow\)

\(B = \text{if } (B \in C, B, 0)\) by (rule MMI_id)

\[\text{from } S7 \text{ have } S8: B =\]

if \((B \in C, B, 0) \rightarrow\)

\((B / \text{if } (A \in C, A, 0))\) = \((B / \text{if } (A \in C, A, 0))\) by (rule MMI_opreq1)

\[\text{from } S8 \text{ have } S9: B =\]

if \((B \in C, B, 0) \rightarrow\)

\((B / \text{if } (A \in C, A, 0))\) = \((B / \text{if } (A \in C, A, 0))\) by (rule MMI_opreq2)

\[\text{from } S9 \text{ have } S10: B =\]

if \((B \in C, B, 0) \rightarrow\)

\(B = \text{if } (B \in C, B, 0)\) by (rule MM_id)

\[\text{from } S10 \text{ have } S11: B =\]

\(B = \text{if } (B \in C, B, 0)\) by (rule MMI_eqeq2d)
if (B ∈ C, B, O) →
(( if (A ∈ C, A, 0) ≠ 0 → ( if (A ∈ C, A, 0) · (B / if (A ∈ C, A, 0)) = B ) ↔
(if (A ∈ C, A, 0) ≠ 0 →
(if (A ∈ C, A, 0) · (if (B ∈ C, B, O) / if (A ∈ C, A, 0)) =
if (B ∈ C, B, O)) ) ) =
if (B ∈ C, B, O)) ) by (rule MMI_imbi2d)

have S12: 0 ∈ C by (rule MMI_0cn)
from S12 have S13: if (A ∈ C, A, 0) ∈ C by (rule MMI_eliml)
have S14: 0 ∈ C by (rule MMI_0cn)
from S14 have S15: if (B ∈ C, B, O) ∈ C by (rule MMI_eliml)
from S13 S15 have S16: if (A ∈ C, A, 0) ≠ 0 →
(if (A ∈ C, A, 0) · (if (B ∈ C, B, O) / if (A ∈ C, A, 0)) =
if (B ∈ C, B, O)) ) ) =
if (B ∈ C, B, O) by (rule MMI_divcan2z)
from S6 S11 S16 have S17: (A ∈ C ∧ B ∈ C) →
(A ≠ 0 → (A · (B / A)) = B) by (rule MMI_dedth2h)
from S17 show (A ∈ C ∧ B ∈ C ∧ A ≠ 0) →
(A · (B / A)) = B by (rule MMI_3impia)

definition (in MMIar0) MMI_divne0bt:
shows (A ∈ C ∧ B ∈ C ∧ B ≠ 0) →
(A ≠ 0 ↔ (A / B) ≠ 0)

proof -

have S1: B ∈ C → (B · 0) = 0 by (rule MMI_mul01t)
from S1 have S2: B ∈ C → ((B · 0) = A ↔ 0 = A) by (rule MMI_eqeq1d)
have S3: A = 0 ↔ 0 = A by (rule MMI_eqcom)
from S2 S3 have S4: B ∈ C → (A = 0 ↔ (B · 0) = A) by (rule
MMI_syl6rbrbA)
from S4 have S5: (A ∈ C ∧ B ∈ C ∧ B ≠ 0) →
(A = 0 ↔ (B · 0) = A) by (rule MMI_3add2ant2)
have S6: 0 ∈ C by (rule MMI_0cn)
have S7: ((A ∈ C ∧ B ∈ C ∧ 0 ∈ C) ∧ B ≠ 0) →
((A / B) = 0 ↔ (B · 0) = A) by (rule MMI_divmult)
from S6 S7 have S8: ((A ∈ C ∧ B ∈ C) ∧ B ≠ 0) →
((A / B) = 0 ↔ (B · 0) = A) by (rule MMI_mp3an13)
from S8 have S9: (A ∈ C ∧ B ∈ C ∧ B ≠ 0) →
((A / B) = 0 ↔ (B · 0) = A) by (rule MMI_3impia)
from S5 S9 have S10: (A ∈ C ∧ B ∈ C ∧ B ≠ 0) →
(A = 0 ↔ (A / B) = 0) by (rule MMI_bitr4d)
from S10 show (A ∈ C ∧ B ∈ C ∧ B ≠ 0) →
(A ≠ 0 ↔ (A / B) ≠ 0) by (rule MMI_eqeqd)

lemma (in MMIar0) MMI_divne0: assumes A1: A ∈ C and
A2: B ∈ C and

1526
A3: $A \neq 0$ and
A4: $B \neq 0$
shows $(A / B) \neq 0$

proof -

from A1 have S1: $A \in C$.
from A2 have S2: $B \in C$.
from A4 have S3: $B \neq 0$.
from A3 have S4: $A \neq 0$.

have S5: $(A \in C \land B \in C \land B \neq 0) \rightarrow (A \neq 0 \leftrightarrow (A / B) \neq 0)$ by (rule MMI_divne0bt)
from S4 S5 have S6: $(A \in C \land B \in C \land B \neq 0) \rightarrow (A / B) \neq 0$ by (rule MMI_mp3an)

have S7: $(1 / A) \neq 0 \leftrightarrow (A \neq 0 \leftrightarrow (1 / A) \neq 0)$ by (rule MMI_neeq1)
from S2 S7 S8 have S9: $(1 / if (A \neq 0, A, 1)) \neq 0$ by (rule MMI_divne0)

from S2 S9 show $A \neq 0 \rightarrow (1 / A) \neq 0$ by (rule MMI_dedth)

qed

 lemma (in MMIar0) MMI_recne0z: assumes A1: $A \in C$
 shows $A \neq 0 \rightarrow (1 / A) \neq 0$

proof -

have S1: $A =$
if ($A \neq 0, A, 1$) \rightarrow
(1 / A) =
(1 / if ($A \neq 0, A, 1$)) by (rule MMI_opreq2)
from S1 have S2: $A =$
if ($A \neq 0, A, 1$) \rightarrow
((1 / A) \neq 0 \leftrightarrow (1 / if ($A \neq 0, A, 1$)) \neq 0) by (rule MMI_neeq1d)
have S3: $1 \in C$ by (rule MMI_1cn)
from A1 have S4: $A \in C$.
have S5: $1 \in C$ by (rule MMI_1cn)
from S4 S5 have S6: $if (A \neq 0, A, 1) \in C$ by (rule MMI_keepel)
have S7: $1 \neq 0$ by (rule MMI_axine0)
have S8: $if (A \neq 0, A, 1) \neq 0$ by (rule MMI_elimne0)
from S3 S6 S7 S8 have S9: $(1 / if (A \neq 0, A, 1)) \neq 0$ by (rule MMI_divne0)

from S2 S9 show $A \neq 0 \rightarrow (1 / A) \neq 0$ by (rule MMI_dedth)

qed

 lemma (in MMIar0) MMI_recne0t:
 shows $(A \in C \land A \neq 0) \rightarrow (1 / A) \neq 0$

proof -

have S1: $A =$
if ($A \in C, A, 0$) \rightarrow
(A \neq 0 \leftrightarrow if ($A \in C, A, 0) \neq 0$) by (rule MMI_neeq1)

have S2: $A =$
if ($A \in C, A, 0$) \rightarrow
(1 / A) =
(1 / if ($A \in C, A, 0)) by (rule MMI_opreq2)
from S2 have S3: $A =$
if ($A \in C, A, 0$) \rightarrow
\( \left( \frac{1}{A} \right) \neq 0 \leftrightarrow \left( \frac{1}{\text{if} (A \in C, A, 0)} \right) \neq 0 \) by (rule MMI_neeq1d)

from S1 S3 have S4: \( A = \) if ( \( A \in C, A, 0 \) ) \( \rightarrow \) 
( \( A \neq 0 \rightarrow \left( \frac{1}{A} \right) \neq 0 \) ) \( \rightarrow \) 
( \( \frac{1}{\text{if} (A \in C, A, 0)} \) \( \neq 0 \) ) by (rule MMI_imbi12d)

have S5: \( 0 \in C \) by (rule MMI_0cn)

from S5 have S6: \( (A \in C, A, 0) \in C \) by (rule MMI_elimel)

from S6 have S7: \( (A \in C, A, 0) \neq 0 \rightarrow \) 
( \( 1 / \text{if} (A \in C, A, 0) \) \( \neq 0 \) ) by (rule MMI_recne0z)

from S4 S7 have S8: \( A \in C \rightarrow (A \neq 0 \rightarrow (1 / A) \neq 0) \) by (rule MMI_dedth)

from S8 show \( (A \in C \land A \neq 0) \rightarrow (1 / A) \neq 0 \) by (rule MMI_imp)

qed

lemma (in MMIIsar0) MMI_recid: assumes A1: \( A \in C \) and  
A2: \( A \neq 0 \)  
shows \( (A \cdot (1 / A)) = 1 \)  
proof -  
from A1 have S1: \( A \in C \).

have S2: \( 1 \in C \) by (rule MMI_1cn)

from A2 have S3: \( A \neq 0 \).

from S1 S2 S3 show \( (A \cdot (1 / A)) = 1 \) by (rule MMI_divcan2)

qed

lemma (in MMIIsar0) MMI_recidz: assumes A1: \( A \in C \)  
shows \( A \neq 0 \rightarrow (A \cdot (1 / A)) = 1 \)  
proof -  
from A1 have S1: \( A \in C \).

have S2: \( 1 \in C \) by (rule MMI_1cn)

from S1 S2 show \( A \neq 0 \rightarrow (A \cdot (1 / A)) = 1 \) by (rule MMI_divcan2z)

qed

lemma (in MMIIsar0) MMI_recidt:  
shows \( (A \in C \land A \neq 0) \rightarrow (A \cdot (1 / A)) = 1 \)  
proof -  

have S1: \( A = \) if ( \( A \in C, A, 0 \) ) \( 
( A \neq 0 \leftrightarrow \text{if} (A \in C, A, 0) \neq 0 \) ) by (rule MMI_neeq1)

have S2: \( A = \) if ( \( A \in C, A, 0 \) ) \( 
A = \text{if} (A \in C, A, 0) \) by (rule MMI_id)

have S3: \( A = \) if ( \( A \in C, A, 0 \) ) \( 
(1 / A) = \) ( \( 1 / \text{if} (A \in C, A, 0) \) ) by (rule MMI_opreq2)

from S2 S3 have S4: \( A = \)
if (A ∈ C, A, 0) →
(A · (1/A)) =
(if (A ∈ C, A, 0) · (1 / (A ∈ C, A, 0))) by (rule MMI_opreq12d)
from S4 have S5: A =
(if (A ∈ C, A, 0) →
((A · (1/A)) =
1) by (rule MMI_eiqeq1d)
from S1 S5 have S6: A =
(if (A ∈ C, A, 0) →
((A ≠ 0 → (A · (1/A)) = 1) ←→
(if (A ∈ C, A, 0) ≠ 0 →
(if (A ∈ C, A, 0) · (1 / (A ∈ C, A, 0))) =
1)) by (rule MMI_imp)
have S7: 0 ∈ C by (rule MMI_0cn)
from S7 have S8: if (A ∈ C, A, 0) ∈ C by (rule MMI_elimel)
from S8 have S9: if (A ∈ C, A, 0) ≠ 0 →
(if (A ∈ C, A, 0) · (1 / (A ∈ C, A, 0))) =
1 by (rule MMI_recidz)
from S6 S9 have S10: A ∈ C →
( (1/A) · A ) = 1 by (rule MMI_dedth)
from S10 show (A ∈ C ∧ A ≠ 0) →
( A · (1/A) ) = 1 by (rule MMI_imp)
qued

lemma (in MMIsar0) MMI_recid2t:
shows (A ∈ C ∧ A ≠ 0) →
( (1/A) · A ) = 1
proof -
have S1: ((1/A) ∈ C ∧ A ∈ C) →
((1/A) · A) = (A · (1/A)) by (rule MMI_axmulcom)
have S2: (A ∈ C ∧ A ≠ 0) → (1/A) ∈ C by (rule MMI_recclt)
have S3: (A ∈ C ∧ A ≠ 0) → A ∈ C by (rule MMI_pm3_26)
from S1 S2 S3 have S4: (A ∈ C ∧ A ≠ 0) →
((1/A) · A) = (A · (1/A)) by (rule MMI_sylanc)
have S5: (A ∈ C ∧ A ≠ 0) →
(A · (1/A)) = 1 by (rule MMI_recidt)
from S4 S5 show (A ∈ C ∧ A ≠ 0) →
((1/A) · A) = 1 by (rule MMI_eqtrd)
qued

lemma (in MMIsar0) MMI_divrec: assumes A1: A ∈ C and
A2: B ∈ C and
A3: B ≠ 0
shows (A/B) = (A · (1/B))
proof -
from A2 have S1: B ∈ C.
from A1 have S2: A ∈ C.
from A2 have S3: B ∈ C.
from A3 have S4: B ≠ 0.
from S3 S4 have S5: (1 / B) ∈ C by (rule MMI_recl)
from S2 S5 have S6: (A · (1 / B)) ∈ C by (rule MMI_mulcl)
from S1 S6 have S7: (B · (A · (1 / B))) =
((A · (1 / B)) · B) by (rule MMI_mulcom)
from A1 have S8: A ∈ C.
from S5 have S9: (1 / B) ∈ C.
from A2 have S10: B ∈ C.
from S8 S9 S10 have S11: (A · (1 / B)) · B =
(A · (1 / B)) by (rule MMI_mulass)
from A2 have S12: B ∈ C.
have S13: 1 ∈ C by (rule MMI_1cn)
from A3 have S14: B ≠ 0.
from S12 S13 S14 have S15: ((1 / B) · B) = 1 by (rule MMI_divcan1)
from S15 have S16: (A · ((1 / B) · B)) = (A · 1) by (rule MMI_opreq2i)
from A1 have S17: A ∈ C.
from S17 have S18: (A · 1) = A by (rule MMI_mulid1)
from S16 S18 have S19: (A · ((1 / B) · B)) = A by (rule MMI_sqtr)
from S7 S11 S19 have S20: (B · (A · (1 / B))) = A by (rule MMI_3eqtr)
from A1 have S21: A ∈ C.
from A2 have S22: B ∈ C.
from S6 have S23: (A · (1 / B)) ∈ C.
from A3 have S24: B ≠ 0.
from S21 S22 S23 S24 have S25: (A / B) =
(A · (1 / B)) ←→
(B · (A · (1 / B))) = A by (rule MMI_divmul)
from S20 S25 show (A / B) = (A · (1 / B)) by (rule MMI_mpbir)
qed

lemma (in MMIIsar0) MMI_divrecz: assumes A1: A ∈ C and
A2: B ∈ C
shows B ≠ 0 → (A / B) = (A · (1 / B))
proof -
  have S1: B =
  if (B ≠ 0, B, 1) →
  (A / B) =
  (A · if (B ≠ 0, B, 1)) by (rule MMI_opreq2)
  have S2: B =
  if (B ≠ 0, B, 1) →
  (1 / B) =
  (1 · if (B ≠ 0, B, 1)) by (rule MMI_opreq2)
  from S2 have S3: B =
  if (B ≠ 0, B, 1) →
  (A · (1 / B)) =
  (A · (1 · if (B ≠ 0, B, 1))) by (rule MMI_opreq2d)
  from S1 S3 have S4: B =
  if (B ≠ 0, B, 1) →
  ((A / B) =
( A · ( 1 / B ) ) ←→
( A / if ( B ≠ 0 , B , 1 ) ) =
( A · ( 1 / if ( B ≠ 0 , B , 1 ) ) ) by (rule MMI_eqeq12d)
from A1 have S5: A ∈ C.
from A2 have S6: B ∈ C.
have S7: 1 ∈ C by (rule MMI_1cn)
from S6 S7 have S8: if ( B ≠ 0 , B , 1 ) ∈ C by (rule MMI_keepel)
have S9: if ( B ≠ 0 , B , 1 ) ≠ 0 by (rule MMI_elimne0)
from S5 S8 S9 have S10: ( A / if ( B ≠ 0 , B , 1 ) ) =
( A · ( 1 / if ( B ≠ 0 , B , 1 ) ) ) by (rule MMI_divrec)
from S4 S10 show B ≠ 0 → ( A / B ) = ( A · ( 1 / B ) )
  by (rule MMI_dedth)

qed

lemma (in MMIIsar0) MMI_divrect:
  shows ( A ∈ C ∧ B ∈ C ∧ B ≠ 0 ) →
  ( A / B ) = ( A · ( 1 / B ) )
proof -
  have S1: A =
    if ( A ∈ C , A , 0 ) →
    ( A / B ) =
    ( if ( A ∈ C , A , 0 ) / B ) by (rule MMI_opreq1)
    have S2: A =
      if ( A ∈ C , A , 0 ) →
      ( A · ( 1 / B ) ) =
      ( if ( A ∈ C , A , 0 ) · ( 1 / B ) ) by (rule MMI_opreq1)
    from S1 S2 have S3: A =
      if ( A ∈ C , A , 0 ) →
      ( ( A / B ) =
      ( A · ( 1 / B ) ) ←→
      ( if ( A ∈ C , A , 0 ) / B ) =
      ( if ( A ∈ C , A , 0 ) · ( 1 / B ) ) ) by (rule MMI_eqeq12d)
    from S3 have S4: A =
      if ( A ∈ C , A , 0 ) →
      ( ( B ≠ 0 → ( A / B ) = ( A · ( 1 / B ) ) ) ←→
      ( B ≠ 0 →
      ( if ( A ∈ C , A , 0 ) / B ) =
      ( if ( A ∈ C , A , 0 ) · ( 1 / B ) ) ) ) by (rule MMI_imbi2d)
    have S5: B =
      if ( B ∈ C , B , 0 ) →
      ( B ≠ 0 ←→ if ( B ∈ C , B , 0 ) ≠ 0 ) by (rule MMI_neeq1)
    have S6: B =
      if ( B ∈ C , B , 0 ) →
      ( if ( A ∈ C , A , 0 ) / B ) =
      ( if ( A ∈ C , A , 0 ) / if ( B ∈ C , B , 0 ) ) by (rule MMI_opreq2)
    have S7: B =
      if ( B ∈ C , B , 0 ) →
\[
\begin{aligned}
(1 / B) &=
(1 / \text{if } (B \in C, B, 0)) \quad \text{by rule MMI_opreq2}
\end{aligned}
\]

from S7 have S8: B =
\[
\text{if } (B \in C, B, 0) \quad \rightarrow
\]
\[
(\text{if } (A \in C, A, 0) \cdot (1 / B)) =
\]
\[
(\text{if } (A \in C, A, 0) \cdot (1 / \text{if } (B \in C, B, 0))) \quad \text{by rule MMI_opreq2d}
\]

from S6 S8 have S9: B =
\[
\text{if } (B \in C, B, 0) \quad \rightarrow
\]
\[
((\text{if } (A \in C, A, 0) / B) =
\]
\[
(\text{if } (A \in C, A, 0) / \text{if } (B \in C, B, 0)) =
\]
\[
((\text{if } (A \in C, A, 0) \cdot (1 / \text{if } (B \in C, B, 0)))) \quad \text{by rule MMI_eqeq12d}
\]

from S5 S9 have S10: B =
\[
\text{if } (B \in C, B, 0) \quad \rightarrow
\]
\[
((B \neq 0 \quad \rightarrow (\text{if } (A \in C, A, 0) / B) = (\text{if } (A \in C, A, 0)) \cdot (1 / B))) \quad \rightarrow
\]
\[
(\text{if } (B \in C, B, 0) \neq 0 \quad \rightarrow
\]
\[
(\text{if } (A \in C, A, 0) / \text{if } (B \in C, B, 0)) =
\]
\[
(\text{if } (A \in C, A, 0) \cdot (1 / \text{if } (B \in C, B, 0))) \quad \text{by rule MMI_imbi12d}
\]

have S11: 0 \in C by (rule MMI_0cn)
from S11 have S12: if (A \in C, A, 0) \in C by (rule MMI_elim)

have S13: 0 \in C by (rule MMI_0cn)
from S13 have S14: if (B \in C, B, 0) \in C by (rule MMI_elim)

from S12 S14 have S15: if (B \in C, B, 0) \neq 0 \quad \rightarrow
\]
\[
(\text{if } (A \in C, A, 0) / \text{if } (B \in C, B, 0)) =
\]
\[
((\text{if } (A \in C, A, 0) \cdot (1 / \text{if } (B \in C, B, 0))) \quad \text{by rule MMI_divrecz}
\]

from S4 S10 S15 have S16: (A \in C \land B \in C) \quad \rightarrow
\]
\[
(B \neq 0 \quad \rightarrow
\]
\[
(A / B) = (A \cdot (1 / B)) \quad \text{by rule MMI_dedth2h}
\]

from S16 show (A \in C \land B \in C \land B \neq 0) \quad \rightarrow
\]
\[
(A / B) = (A \cdot (1 / B)) \quad \text{by rule MMI_3impia}
\]

qed

lemma (in MMIisar0) MMI_divrec2t:

shows (A \in C \land B \in C \land B \neq 0) \quad \rightarrow

(A / B) = ((1 / B) \cdot A)

proof -

have S1: (A \in C \land B \in C \land B \neq 0) \quad \rightarrow
\]
\[
(A / B) = (A \cdot (1 / B)) \quad \text{by rule MMI_divrect}
\]

have S2: (A \in C \land (1 / B) \in C) \quad \rightarrow
\]
\[
(A \cdot (1 / B)) = ((1 / B) \cdot A) \quad \text{by rule MMI_axmulcom}
\]

have S3: (A \in C \land B \in C \land B \neq 0) \quad \rightarrow A \in C \quad \text{by rule MMI_3simp1}

have S4: (B \in C \land B \neq 0) \quad \rightarrow (1 / B) \in C \quad \text{by rule MMI_recclt}

from S4 have S5: (A \in C \land B \in C \land B \neq 0) \quad \rightarrow
\]
\[
(1 / B) \in C \quad \text{by rule MMI_3adant1}
\]

from S2 S3 S5 have S6: (A \in C \land B \in C \land B \neq 0) \quad \rightarrow
\]
\[
(A \cdot (1 / B)) = ((1 / B) \cdot A) \quad \text{by rule MMI_sylanc}
\]

from S1 S6 show (A \in C \land B \in C \land B \neq 0) \quad \rightarrow

1532
\[
\frac{A}{B} = \left( \frac{1}{B} \right) \cdot A \text{ by (rule MMI_eqtrd)}
\]

**lemma** (in MMIsar0) **MMI_divasst**:

shows \((A \in C \land B \in C \land C \in C) \land C \neq 0\) \(\longrightarrow\)
\((A \cdot B) / C = (A \cdot (B / C))\)

**proof** -

have S1: \(A \in C \longrightarrow A \in C\) by (rule MMI_id)
have S2: \(B \in C \longrightarrow B \in C\) by (rule MMI_id)
have S3: \((C \in C \land C \neq 0) \longrightarrow (1 / C) \in C\) by (rule MMI_recclt)
from S1 S2 S3 have S4: \((A \in C \land B \in C \land (C \in C \land C \neq 0)) \longrightarrow\)
\((A \in C \land B \in C \land (1 / C) \in C)\) by (rule MMI_3anim123i)
from S4 have S5: \(A \in C \longrightarrow\)
\((B \in C)\)
\((C \in C \land C \neq 0) \longrightarrow\)
\((A \in C \land B \in C \land (1 / C) \in C)\) by (rule MMI_3exp)
from S5 have S6: \(A \in C \longrightarrow\)
\((B \in C)\)
\((C \in C)\)
\((C \neq 0) \longrightarrow\)
\((A \in C \land B \in C \land (1 / C) \in C)\) by (rule MMI_exp4a)
from S6 have S7: \((A \in C \land B \in C \land (1 / C) \in C) \land C \neq 0\) \(\longrightarrow\)
\((A \in C \land B \in C \land (1 / C) \in C)\) by (rule MMI_3simp1)
have S8: \((A \in C \land B \in C \land (1 / C) \in C) \longrightarrow\)
\((A \cdot B) \cdot (1 / C)\) by (rule MMI_3expa)
from S7 S8 have S9: \((A \in C \land B \in C \land C \in C) \land C \neq 0\) \(\longrightarrow\)
\((A \cdot B) \cdot (1 / C)\) by (rule MMI_syl)
have S10: \((A \cdot B) \in C \land C \neq 0\) \(\longrightarrow\)
\((A \cdot B) \cdot (1 / C)\) by (rule MMI_divrect)
from S10 have S11: \((A \cdot B) \in C \land C \neq 0\) \(\longrightarrow\)
\((A \cdot B) \cdot (1 / C)\) by (rule MMI_3expa)
have S12: \((A \in C \land B \in C) \longrightarrow (A \cdot B) \in C\) by (rule MMI_axmulcl)
from S12 have S13: \((A \in C \land B \in C) \land C \in C\) \(\longrightarrow\)
\((A \cdot B) \in C \land C \in C\) by (rule MMI_axmulcl)
from S13 have S14: \((A \in C \land B \in C \land C \in C) \longrightarrow\)
\((A \cdot B) \in C \land C \in C\) by (rule MMI_3imp)
from S11 S14 have S15: \((A \in C \land B \in C \land C \in C) \land C \neq 0\) \(\longrightarrow\)
\((A \cdot B) \cdot (1 / C)\) by (rule MMI_syl)

\[
(\frac{A}{B}) = (\frac{A \cdot B}{C}) \text{ by (rule MMI_syl)}
\]

have S16: \((B \in C \land C \in C \land C \neq 0) \longrightarrow\)
\((B / C) = (B \cdot (1 / C))\) by (rule MMI_divrect)
from S16 have S17: \((B \in C \land C \in C) \land C \neq 0\) \(\longrightarrow\)
\((B / C) = (B \cdot (1 / C))\) by (rule MMI_3expa)
from S17 have S18: ( ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) ∧ C ≠ 0 ) →
( B / C ) = ( B · ( 1 / C ) ) by (rule MMI_3adantl1)
from S18 have S19: ( ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) ∧ C ≠ 0 ) →
( A · ( B / C ) ) =
( A · ( B · ( 1 / C ) ) ) by (rule MMI_opreq2d)
from S9 S15 S19 show ( ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) ∧ C ≠ 0 ) →
( ( A · B ) / C ) = ( A · ( B / C ) ) by (rule MMI_3eqtr4d)
qed

lemma (in MMI_isar0) MMI_div23t:
  shows ( ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) ∧ C ≠ 0 ) →
( ( A · B ) / C ) = ( ( A / C ) · B )
proof -
  have S1: ( A ∈ C ∧ B ∈ C ) →
( A · B ) = ( B · A ) by (rule MMI_axmulcom)
  from S1 have S2: ( ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( A · B ) = ( B · A ) by (rule MMI_3adant3)
  from S2 have S3: ( ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) ∧ C ≠ 0 ) →
( A · B ) = ( B · A ) by (rule MMI_adantr)
  from S3 have S4: ( ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) ∧ C ≠ 0 ) →
( ( A · B ) / C ) = ( ( B · A ) / C ) by (rule MMI_opreq1d)
  have S5: ( ( B ∈ C ∧ A ∈ C ∧ C ∈ C ) ∧ C ≠ 0 ) →
( ( B · A ) / C ) = ( B · ( A / C ) ) by (rule MMI_divasst)
  from S5 have S6: ( B ∈ C ∧ A ∈ C ∧ C ∈ C ) →
( C ≠ 0 →
( ( B · A ) / C ) =
( B · ( A / C ) ) ) by (rule MMI_ex)
  from S6 have S7: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( C ≠ 0 →
( ( B · A ) / C ) =
( B · ( A / C ) ) ) by (rule MMI_3com12)
  from S7 have S8: ( ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) ∧ C ≠ 0 ) →
( ( B · A ) / C ) = ( B · ( A / C ) ) by (rule MMI_imp)
  have S9: ( B ∈ C ∧ ( A / C ) ∈ C ) →
( B · ( A / C ) ) = ( ( A / C ) · B ) by (rule MMI_axmulcom)
  have S10: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) → B ∈ C by (rule MMI_3simp2)
  from S10 have S11: ( ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) ∧ C ≠ 0 ) →
B ∈ C by (rule MMI_adantr)
  have S12: ( A ∈ C ∧ C ∈ C ∧ C ≠ 0 ) →
( A / C ) ∈ C by (rule MMI_divclt)
  from S12 have S13: ( ( A ∈ C ∧ C ∈ C ) ∧ C ≠ 0 ) →
( A / C ) ∈ C by (rule MMI_3expa)
  from S13 have S14: ( ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) ∧ C ≠ 0 ) →
( A / C ) ∈ C by (rule MMI_3adant12)
  from S9 S11 S14 have S15: ( ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) ∧ C ≠ 0 ) →
( B · ( A / C ) ) = ( ( A / C ) · B ) by (rule MMI_sylanc)
  from S4 S8 S15 show ( ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) ∧ C ≠ 0 ) →
proof

lemma (in MMIIsar0) MMI_div13t:
  shows \(( ( A / B ) \cdot C ) = ( ( C / B ) \cdot A )\)
proof
  have S1: \(( A \in C \land C \in C ) \rightarrow\)
  \(( A ) \cdot C \) = \(( C \cdot A )\) by (rule MMI_axmulcom)
  from S1 have S2: \(( A \in C \land C \in C ) \rightarrow\)
  \(( ( A \cdot C ) / B ) = ( ( C \cdot A ) / B )\) by (rule MMI_opreq1d)
  from S2 have S3: \(( A \in C \land B \in C \land C \in C ) \rightarrow\)
  \(( ( A \cdot C ) / B ) = ( ( C \cdot A ) / B )\) by (rule MMI_3adant2)
  from S3 have S4: \(( ( A \in C \land B \in C \land C \in C ) \land B \neq 0 ) \rightarrow\)
  \(( ( A \cdot C ) / B ) = ( ( C \cdot A ) / B )\) by (rule MMI_adantr)
  have S5: \(( ( A \in C \land C \in C \land B \in C ) \land B \neq 0 ) \rightarrow\)
  \(( ( A \cdot C ) / B ) = ( ( A / B ) \cdot C )\) by (rule MMI_div23t)
  from S5 have S6: \(( A \in C \land C \in C \land B \in C ) \rightarrow\)
  \(( B \neq 0 \rightarrow\)
  \(( ( A \cdot C ) / B ) = ( ( A / B ) \cdot C )\) by (rule MMI_3com23)
  from S6 have S7: \(( A \in C \land B \in C \land C \in C ) \rightarrow\)
  \(( B \neq 0 \rightarrow\)
  \(( ( C \cdot A ) / B ) = ( ( C / B ) \cdot A )\) by (rule MMI_imp)
  have S8: \(( ( C \in C \land A \in C \land B \in C ) \land B \neq 0 ) \rightarrow\)
  \(( ( C \cdot A ) / B ) = ( ( C / B ) \cdot A )\) by (rule MMI_div23t)
  from S8 have S9: \(( ( C \cdot A ) / B ) = ( ( A / B ) \cdot C )\) by (rule MMI_imp)
  have S10: \(( ( A \in C \land A \in C \land B \in C ) \land B \neq 0 ) \rightarrow\)
  \(( ( A \cdot B ) \cdot C ) = ( ( C / B ) \cdot A )\) by (rule MMI_3eqtr3d)
qed

lemma (in MMIIsar0) MMI_div12t:
  shows \(( ( A \in C \land B \in C \land C \in C ) \land C \neq 0 ) \rightarrow\)
  \(( A \cdot ( B / C ) ) = ( B \cdot ( A / C ) )\)
proof
  have S1: \(( A \in C \land ( B / C ) \in C ) \rightarrow\)
  \(( A \cdot ( B / C ) ) = ( ( B / C ) \cdot A )\) by (rule MMI_axmulcom)
have S2: \(( A \in C \land B \in C \land C \in C \) \(\rightarrow\) \( A \in C \)) by (rule MMI_3simp1)
from S2 have S3: \(( ( A \in C \land B \in C \land C \in C ) \land C \neq 0 ) \rightarrow\)
\( A \in C \) by (rule MMI_adantr)
    have S4: \(( B \in C \land C \in C \land C \neq 0 ) \rightarrow\)
    \(( B / C ) \in C \) by (rule MMI_divclt)
    from S4 have S5: \(( ( B \in C \land C \in C ) ) \land C \neq 0 ) \rightarrow\)
    \(( B / C ) \in C \) by (rule MMI_3expa)
    from S5 have S6: \(( ( A \in C \land B \in C ) \land C \neq 0 ) \rightarrow\)
    \(( B / C ) \in C \) by (rule MMI_3adantl1)
    from S1 S3 S6 have S7: \(( ( A \in C \land B \in C \land C \in C ) ) \land C \neq 0 ) \rightarrow\)

\(( A \cdot ( B / C ) ) = ( ( B / C ) \cdot A ) \) by (rule MMI_sylanc)
    have S8: \(( ( B \in C \land C \in C \land A \in C ) ) \land C \neq 0 ) \rightarrow\)
    \(( B / C ) \cdot A ) = ( ( A / C ) \cdot B ) \) by (rule MMI_div13t)
    from S8 have S9: \(( ( B \in C \land C \in C \land A \in C ) ) \rightarrow\)
    \( C \neq 0 \rightarrow\)
    \(( B / C ) \cdot A ) =
    \(( A / C ) \cdot B ) \) by (rule MMI_3comr)
    from S10 have S11: \(( ( A \in C \land B \in C \land C \in C ) ) \land C \neq 0 ) \rightarrow\)
    \(( A / C ) \cdot A ) = ( ( A / C ) \cdot B ) \) by (rule MMI_imp)
    have S12: \(( ( A / C ) \in C \land B \in C ) \rightarrow\)
    \(( A / C ) \cdot B ) = ( B \cdot ( A / C ) ) \) by (rule MMI_axmulcom)
    have S13: \(( A \in C \land C \in C \land C \neq 0 ) \rightarrow\)
    \(( A / C ) \in C \) by (rule MMI_divclt)
    from S14 have S15: \(( ( A \in C \land B \in C \land C \in C ) ) \land C \neq 0 ) \rightarrow\)
    \(( A / C ) \in C \) by (rule MMI_3expa)
    from S14 have S15: \(( ( A \in C \land B \in C \land C \in C ) ) \land C \neq 0 ) \rightarrow\)
    \(( A / C ) \in C \) by (rule MMI_3adantl2)
    have S16: \(( A \in C \land B \in C \land C \in C ) \rightarrow B \in C \) by (rule MMI_3simp2)
    from S16 have S17: \(( ( A \in C \land B \in C \land C \in C ) ) \land C \neq 0 ) \rightarrow\)
    B \in C by (rule MMI_adantr)
    from S12 S15 S17 have S18: \(( ( A \in C \land B \in C \land C \in C ) ) \land C \neq 0 ) \rightarrow\)

\(( A \cdot ( B / C ) ) = ( B \cdot ( A / C ) ) \) by (rule MMI_3eqtrd)

qed

lemma (in MMIIsar0) MMI_divassz: assumes A1: \( A \in C \) and
A2: \( B \in C \) and
A3: \( C \in C \)
shows \( C \neq 0 \rightarrow\)
\(( ( A \cdot B ) / C ) = ( A \cdot ( B / C ) ) \)

proof -
from $A_1$ have $S_1: A \in C$.
from $A_2$ have $S_2: B \in C$.
from $A_3$ have $S_3: C \in C$.
from $S_1$ $S_2$ $S_3$ have $S_4: A \in C \land B \in C \land C \in C$ by (rule MMI_3pm3_2i)
have $S_5: ( ( A \land B \in C \land C \in C ) \land C \neq 0 ) \rightarrow$
$(( A \land B ) / C ) = ( A \cdot ( B / C ) )$ by (rule MMI_divasst)
from $S_4$ $S_5$ show $C \neq 0 \rightarrow$
$(( A \land B ) / C ) = ( A \cdot ( B / C ) )$ by (rule MMI_mpan)
qed

lemma (in MMIsar0) MMI_divass: assumes $A_1: A \in C$ and
$A_2: B \in C$ and
$A_3: C \in C$ and
$A_4: C \neq 0$
shows $(( A \cdot B ) / C ) = ( A \cdot ( B / C ) )$
proof -
from $A_4$ have $S_1: C \neq 0$.
from $A_1$ have $S_2: A \in C$.
from $A_2$ have $S_3: B \in C$.
from $A_3$ have $S_4: C \in C$.
from $S_3$ $S_4$ have $S_5: C \neq 0 \rightarrow$
$(( A \cdot B ) / C ) = ( A \cdot ( B / C ) )$ by (rule MMI_divassz)
from $S_1$ $S_2$ $S_5$ show $(( A \cdot B ) / C ) = ( A \cdot ( B / C ) )$ by (rule MMI_ax_mp)
qued

lemma (in MMIsar0) MMI_divdir: assumes $A_1: A \in C$ and
$A_2: B \in C$ and
$A_3: C \in C$ and
$A_4: C \neq 0$
shows $(( A + B ) / C ) = ((( A / C ) + ( B / C ) )$)
proof -
from $A_1$ have $S_1: A \in C$.
from $A_2$ have $S_2: B \in C$.
from $A_3$ have $S_3: C \in C$.
from $A_4$ have $S_4: C \neq 0$.
from $S_3$ $S_4$ have $S_5: ( 1 / C ) \in C$ by (rule MMI_reccl)
from $S_1$ $S_2$ $S_5$ have $S_6: ( ( A + B ) \cdot ( 1 / C ) ) =$
$(( A \cdot ( 1 / C ) ) + ( B \cdot ( 1 / C ) ) )$ by (rule MMI_adddir)
from $A_1$ have $S_7: A \in C$.
from $A_2$ have $S_8: B \in C$.
from $S_7$ $S_8$ have $S_9: ( A + B ) \in C$ by (rule MMI_addcl)
from $A_3$ have $S_{10}: C \in C$.
from $A_4$ have $S_{11}: C \neq 0$.
from $S_9$ $S_{10}$ $S_{11}$ have $S_{12}: ( ( A + B ) / C ) =$
$(( A + B ) \cdot ( 1 / C ) )$ by (rule MMI_divrec)
from $A_1$ have $S_{13}: A \in C$.
from $A_3$ have $S_{14}: C \in C$.
from $A_4$ have $S_{15}: C \neq 0$. 

1537
lemma (in MMIsar0) MMI_div23: assumes A1: A ∈ C and  
A2: B ∈ C and  
A3: C ∈ C and  
A4: C ≠ 0  
shows ( ( A · B ) / C ) = ( ( A / C ) · B )  
proof -  
from A1 have S1: A ∈ C.  
from A2 have S2: B ∈ C.  
from S1 S2 have S3: ( A · B ) = ( B · A ) by (rule MMI_mulcom)  
from S3 have S4: ( ( A · B ) / C ) = ( ( B · A ) / C )  
by (rule MMI_opreq1i)  
from A2 have S5: B ∈ C.  
from A1 have S6: A ∈ C.  
from A3 have S7: C ∈ C.  
from A4 have S8: C ≠ 0.  
from S5 S6 S7 S8 have  
S9: ( ( B · A ) / C ) = ( B · ( A / C ) ) by (rule MMI_divass)  
from A2 have S10: B ∈ C.  
from A1 have S11: A ∈ C.  
from A3 have S12: C ∈ C.  
from A4 have S13: C ≠ 0.  
from S11 S12 S13 have S14: ( A / C ) ∈ C by (rule MMI_divcl)  
from S10 S14 have S15: ( B · ( A / C ) ) = ( ( A / C ) · B )  
by (rule MMI_mulcom)  
from S4 S9 S15 show ( ( A · B ) / C ) = ( ( A / C ) · B )  
by (rule MMI_3eqtr)  
qed

lemma (in MMIsar0) MMI_divdirz: assumes A1: A ∈ C and  
A2: B ∈ C and  
A3: C ∈ C  
shows C ≠ 0 ⟹  
( ( A + B ) / C ) =
( ( A / C ) + ( B / C ) )

proof -
  have S1: C = 
  if ( C ≠ 0 , C , 1 ) →
  (( A + B ) / C ) = 
  ( ( A + B ) / if ( C ≠ 0 , C , 1 ) ) by (rule MMI_opreq2)
  have S2: C = 
  if ( C ≠ 0 , C , 1 ) →
  ( A / C ) = 
  ( A / if ( C ≠ 0 , C , 1 ) ) by (rule MMI_opreq2)
  have S3: C = 
  if ( C ≠ 0 , C , 1 ) →
  ( B / C ) = 
  ( B / if ( C ≠ 0 , C , 1 ) ) by (rule MMI_opreq2)
  from S2 S3 have S4: C =
  (( A / C ) + ( B / C ) ) = 
  ( ( A / if ( C ≠ 0 , C , 1 ) ) + ( B / if ( C ≠ 0 , C , 1 ) ) ) by
  (rule MMI_opreq12d)
  from S1 S4 have S5: C =
  if ( C ≠ 0 , C , 1 ) →
  ( ( ( A + B ) / C ) = 
  ( ( A / C ) + ( B / C ) ) ↔
  ( ( A + B ) / if ( C ≠ 0 , C , 1 ) ) = 
  ( ( A / if ( C ≠ 0 , C , 1 ) ) + ( B / if ( C ≠ 0 , C , 1 ) ) ) by
  (rule MMI_eqeq12d)
  from A1 have S6: A ∈ C.
  from A2 have S7: B ∈ C.
  from A3 have S8: C ∈ C.
  have S9: 1 ∈ C by (rule MMI_1cn)
  from S8 S9 have S10: if ( C ≠ 0 , C , 1 ) ∈ C by (rule MMI_keepel)
  have S11: if ( C ≠ 0 , C , 1 ) ≠ 0 by (rule MMI_elimne0)
  from S6 S7 S10 S11 have S12: ( ( A + B ) / if ( C ≠ 0 , C , 1 ) )
  = 
  ( ( A / if ( C ≠ 0 , C , 1 ) ) + ( B / if ( C ≠ 0 , C , 1 ) ) ) by
  (rule MMI_divdir)
  from S5 S12 show C ≠ 0 →
  ( ( A + B ) / C ) = 
  ( ( A / C ) + ( B / C ) ) by (rule MMI_dedth)
qed

lemma (in MMIasar0) MMI_divdirt:
  shows ( ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) ∧ C ≠ 0 ) →
  ( ( A + B ) / C ) = 
  ( ( A / C ) + ( B / C ) )
proof -
  have S1: A = 
  if ( A ∈ C , A , 0 ) →
  ( A + B ) =

1539
( if ( A ∈ C , A , 0 ) + B ) by (rule MMI_opreq1)

from S1 have S2: A =
if ( A ∈ C , A , 0 ) →
( ( A + B ) / C ) =
( ( if ( A ∈ C , A , 0 ) + B ) / C ) by (rule MMI_opreq1d)

have S3: A =
if ( A ∈ C , A , 0 ) →
( A / C ) =
( if ( A ∈ C , A , 0 ) / C ) by (rule MMI_opreq1)

from S3 have S4: A =
if ( A ∈ C , A , 0 ) →
( ( A + B ) / C ) =
( ( A / C ) + ( B / C ) ) by (rule MMI_opreq1d)

from S2 S4 have S5: A =
if ( A ∈ C , A , 0 ) →
( ( ( A + B ) / C ) = ( ( A / C ) + ( B / C ) ) ) ←→
( C ≠ 0 →
( ( if ( A ∈ C , A , 0 ) + B ) / C ) =
( ( if ( A ∈ C , A , 0 ) / C ) + ( B / C ) ) ) by (rule MMI_eqeq12d)

have S7: B =
if ( B ∈ C , B , 0 ) →
( ( if ( A ∈ C , A , 0 ) + B ) =
( if ( A ∈ C , A , 0 ) + if ( B ∈ C , B , 0 ) ) by (rule MMI_opreq2)

from S7 have S8: B =
if ( B ∈ C , B , 0 ) →
( ( if ( A ∈ C , A , 0 ) + B ) / C ) =
( ( if ( A ∈ C , A , 0 ) / C ) + ( B / C ) ) by (rule MMI_opreq1d)

have S9: B =
if ( B ∈ C , B , 0 ) →
( B / C ) =
( if ( B ∈ C , B , 0 ) / C ) by (rule MMI_opreq1)

from S9 have S10: B =
if ( B ∈ C , B , 0 ) →
( ( if ( A ∈ C , A , 0 ) + B ) / C ) =
( ( if ( A ∈ C , A , 0 ) / C ) + ( if ( B ∈ C , B , 0 ) / C ) ) by (rule MMI_opreq2d)

from S8 S10 have S11: B =
if ( B ∈ C , B , 0 ) →
( ( ( if ( A ∈ C , A , 0 ) + B ) / C ) =
( ( if ( A ∈ C , A , 0 ) / C ) + ( B / C ) ) ←→
( ( if ( A ∈ C , A , 0 ) + if ( B ∈ C , B , 0 ) / C ) ) =
( ( if ( A ∈ C , A , 0 ) / C ) + ( if ( B ∈ C , B , 0 ) / C ) ) ) by (rule MMI_eqeq12d)
from S11 have S12: B =
if ( B ∈ C , B , 0 ) →
( ( C ≠ 0 → ( ( if ( A ∈ C , A , 0 ) + B ) / C ) ) = (( if ( A ∈ C, A , 0 ) / C ) + ( B / C ) ) )
( C ≠ 0 →
( ( if ( A ∈ C , A , 0 ) + if ( B ∈ C , B , 0 ) ) / C ) )
( ( if ( A ∈ C , A , 0 ) / C ) + ( if ( B ∈ C , B , 0 ) / C ) )
by (rule MMI_imbi2d)

have S13: C =
if ( C ∈ C , C , 0 ) →
( C ≠ 0 ↔ if ( C ∈ C , C , 0 ) ) by (rule MMI_neeq1)

have S14: C =
if ( A ∈ C , A , 0 ) / C =
if ( A ∈ C , A , 0 ) / ( if ( C ∈ C , C , 0 ) )
( rule MMI_opreq2)

have S15: C =
if ( B ∈ C , B , 0 ) / C =
if ( B ∈ C , B , 0 ) / ( if ( C ∈ C , C , 0 ) )
by (rule MMI_opreq2)

from S15 S16 have S17: C =
if ( C ∈ C , C , 0 ) →
( ( if ( A ∈ C , A , 0 ) / C ) + ( if ( B ∈ C , B , 0 ) / C ) ) =
( ( if ( A ∈ C , A , 0 ) / ( if ( C ∈ C , C , 0 ) ) ) + ( if ( B ∈ C , B , 0 ) / ( if ( C ∈ C , C , 0 ) ) ) )
by (rule MMI_opreq12d)

from S14 S17 have S18: C =
if ( C ∈ C , C , 0 ) →
( ( ( if ( A ∈ C , A , 0 ) + if ( B ∈ C , B , 0 ) ) / C ) ) =
( ( if ( A ∈ C , A , 0 ) / C ) + ( if ( B ∈ C , B , 0 ) / C ) )
( rule MMI_eqeq12d)

have S20: 0 ∈ C by (rule MMI_0cn)
from S20 have S21: if \( A \in \mathcal{F}, A, 0 \) \( \in \mathcal{C} \) by (rule MMI_elimel)

have S22: 0 \( \in \mathcal{C} \) by (rule MMI_0cn)

from S22 have S23: if \( B \in \mathcal{C}, B, 0 \) \( \in \mathcal{C} \) by (rule MMI_elimel)

have S24: 0 \( \in \mathcal{C} \) by (rule MMI_0cn)

from S21 S23 S25 have S26: if \( C \in \mathcal{C}, C, 0 \) \( \neq 0 \) \( \rightarrow \) \( ( if \ ( A \in \mathcal{C}, A, 0 ) + if \ ( B \in \mathcal{C}, B, 0 ) ) / if \ ( C \in \mathcal{C}, C, 0 ) ) = \)

\( ( ( if \ ( A \in \mathcal{C}, A, 0 ) / if \ ( C \in \mathcal{C}, C, 0 ) ) + ( if \ ( B \in \mathcal{C}, B, 0 ) / if \ ( C \in \mathcal{C}, C, 0 ) ) ) \) by (rule MMI_divdirz)

from S6 S12 S19 S26 have S27: \( ( A \in \mathcal{C} \land B \in \mathcal{C} \land C \in \mathcal{C} ) \rightarrow \)

( C \( \neq 0 \) \( \rightarrow \) \( ( ( A + B ) / C ) = \)

\( ( ( A / C ) + ( B / C ) ) \) ) by (rule MMI_dedth3h)

from S27 show \( ( ( A \in \mathcal{C} \land B \in \mathcal{C} \land C \in \mathcal{C} ) \land C \neq 0 ) \rightarrow \)

( ( A + B ) / C ) = \( ( ( A / C ) + ( B / C ) ) \) by (rule MMI_imp)

deduced

lemma (in MMIsar0) MMI_divcan3: assumes \( A1: A \in \mathcal{C} \) and

\( A2: B \in \mathcal{C} \) and

\( A3: A \neq 0 \)

shows \( ( ( A \cdot B ) / A ) = B \)

proof -

from A1 have S1: A \( \in \mathcal{C} \).

from A2 have S2: B \( \in \mathcal{C} \).

from A1 have S3: A \( \in \mathcal{C} \).

from A3 have S4: A \( \neq 0 \).

from S1 S2 S3 S4 have S5: \( ( ( A \cdot B ) / A ) = ( A \cdot ( B / A ) ) \) by (rule MMI_divass)

from A1 have S6: A \( \in \mathcal{C} \).

from A2 have S7: B \( \in \mathcal{C} \).

from A3 have S8: A \( \neq 0 \).

from S6 S7 S8 have S9: \( ( A \cdot ( B / A ) ) = B \) by (rule MMI_divcan2)

from S5 S9 show \( ( ( A \cdot B ) / A ) = B \) by (rule MMI_eqtr)

deduced

lemma (in MMIsar0) MMI_divcan4: assumes \( A1: A \in \mathcal{C} \) and

\( A2: B \in \mathcal{C} \) and

\( A3: A \neq 0 \)

shows \( ( ( B \cdot A ) / A ) = B \)

proof -

from A2 have S1: B \( \in \mathcal{C} \).

from A1 have S2: A \( \in \mathcal{C} \).

from S1 S2 have S3: \( ( B \cdot A ) = ( A \cdot B ) \) by (rule MMI_mulcom)

from S3 have S4: \( ( ( B \cdot A ) / A ) = ( ( A \cdot B ) / A ) \) by (rule MMI_opreq1i)

from A1 have S5: A \( \in \mathcal{C} \).

from A2 have S6: B \( \in \mathcal{C} \).

from A3 have S7: A \( \neq 0 \).
from S5 S6 S7 have S8: ( ( A · B ) / A ) = B by (rule MMI_divcan3)
from S4 S8 show ( ( B · A ) / A ) = B by (rule MMI_eqtr)
qed

lemma (in MMIIsar0) MMI_divcan3z: assumes A1: A ∈ C and
A2: B ∈ C
shows A ≠ 0 → ( ( A · B ) / A ) = B
proof -
  have S1: A =
    if ( A ≠ 0 , A , 1 ) →
    ( A · B ) =
    ( ( if ( A ≠ 0 , A , 1 ) · B ) ) by (rule MMI_opreq1)
    have S2: A =
      if ( A ≠ 0 , A , 1 ) →
      A = if ( A ≠ 0 , A , 1 ) by (rule MMI_id)
      from S1 S2 have S3: A =
        if ( A ≠ 0 , A , 1 ) →
        ( ( A · B ) / A ) =
        ( ( if ( A ≠ 0 , A , 1 ) · B ) ) by (rule MMI_opreq12d)
      from S3 have S4: A =
        if ( A ≠ 0 , A , 1 ) →
        ( ( A · B ) / A ) =
        ( if ( A ≠ 0 , A , 1 ) ) =
        B by (rule MMI_eqeq1d)
    from S1 S4 have S5: A ∈ C.
    have S6: 1 ∈ C by (rule MMI_1cn)
    from S5 S6 have S7: if ( A ≠ 0 , A , 1 ) ∈ C by (rule MMI_keepel)
    from A2 have S8: B ∈ C.
    have S9: if ( A ≠ 0 , A , 1 ) ≠ 0 by (rule MMI_elimne0)
    from S7 S8 S9 have S10: ( ( if ( A ≠ 0 , A , 1 ) · B ) ) / ( A ≠ 0 , A , 1 ) =
        B by (rule MMI_divcan3)
    from S4 S10 show A ≠ 0 → ( ( A · B ) / A ) = B by (rule MMI_dedth)
qed

lemma (in MMIIsar0) MMI_divcan4z: assumes A1: A ∈ C and
A2: B ∈ C
shows A ≠ 0 → ( ( B · A ) / A ) = B
proof -
  from A1 have S1: A ∈ C.
  from A2 have S2: B ∈ C.
  from S1 S2 have S3: A ≠ 0 → ( ( A · B ) / A ) = B by (rule MMI_divcan3z)
  from A2 have S4: B ∈ C.
  from A1 have S5: A ∈ C.
  from S4 S5 have S6: ( B · A ) = ( A · B ) by (rule MMI_mulcom)
  from S6 have S7: ( ( B · A ) / A ) = ( ( A · B ) / A ) by (rule MMI_opreq1i)
  from S3 S7 show A ≠ 0 → ( ( B · A ) / A ) = B by (rule MMI_syl5eq)
qed
lemma (in MMIsar0) MMI_divcan3t:
  shows ( A ∈ C ∧ B ∈ C ∧ A ≠ 0 ) →
  ( ( A · B ) / A ) = B
proof -
  have S1: A =
    if ( A ∈ C , A , 0 ) →
    ( A ≠ 0 ←→ if ( A ∈ C , A , 0 ) ≠ 0 ) by (rule MMI_neeq1)
  have S2: A =
    if ( A ∈ C , A , 0 ) →
    ( A · B ) =
    ( if ( A ∈ C , A , 0 ) · B ) by (rule MMI_opreq1)
  have S3: A =
    if ( A ∈ C , A , 0 ) →
    A = if ( A ∈ C , A , 0 ) by (rule MMI_id)
  from S2 S3 have S4: A =
    if ( A ∈ C , A , 0 ) →
    ( ( A · B ) / A ) =
    ( if ( A ∈ C , A , 0 ) · B ) / if ( A ∈ C , A , 0 ) by (rule MMI_opreq12d)
  from S4 have S5: A =
    if ( A ∈ C , A , 0 ) →
    ( ( A · B ) / A ) = B
    ( ( if ( A ∈ C , A , 0 ) · B ) / if ( A ∈ C , A , 0 ) ) =
    ( if ( A ∈ C , A , 0 ) · B ) by (rule MMI_opreq1d)
  from S1 S5 have S6: A =
    if ( A ∈ C , A , 0 ) →
    ( ( A ≠ 0 → ( ( A · B ) / A ) = B ) ←→
    ( if ( A ∈ C , A , 0 ) ≠ 0 →
    ( ( if ( A ∈ C , A , 0 ) · B ) / if ( A ∈ C , A , 0 ) ) =
    B ) ) by (rule MMI_imbi12d)
  have S7: B =
    if ( B ∈ C , B , 0 ) →
    ( ( if ( A ∈ C , A , 0 ) · B ) =
    ( if ( A ∈ C , A , 0 ) · if ( B ∈ C , B , 0 ) ) by (rule MMI_opreq2)
  from S7 have S8: B =
    if ( B ∈ C , B , 0 ) →
    ( ( if ( A ∈ C , A , 0 ) · B ) / if ( A ∈ C , A , 0 ) ) =
    ( ( if ( A ∈ C , A , 0 ) · if ( B ∈ C , B , 0 ) ) / if ( A ∈ C , A , 0 ) ) by (rule MMI_opreq1d)
  have S9: B =
    if ( B ∈ C , B , 0 ) →
    B = if ( B ∈ C , B , 0 ) by (rule MMI_id)
  from S8 S9 have S10: B =
    if ( B ∈ C , B , 0 ) →
    ( ( if ( A ∈ C , A , 0 ) · B ) / if ( A ∈ C , A , 0 ) ) =
    B
    ( ( if ( A ∈ C , A , 0 ) · if ( B ∈ C , B , 0 ) ) / if ( A ∈ C , A , 0 ) ) =

1544
if \((B \in C, B, 0)\) by (rule MMI_eqeq12d)

from S10 have S11: \(B = \)

if \((B \in C, B, 0)\) \((\ if \((A \in C, A, 0) \neq 0 \rightarrow (\ if \((A \in C, A, 0) \cdot B) / \ if \((A \in C, A, 0) = B) \leftrightarrow \)\) \)

if \((A \in C, A, 0) \neq 0 \rightarrow \)

if \((A \in C, A, 0) \cdot if \((B \in C, B, 0)\) / \ if \((A \in C, A, 0)\)\) = \(if \((B \in C, B, 0)\)\) \(by\) (rule MMI_imbi2d)

have S12: \(0 \in C\) \(by\) (rule MMI_0cn)

from S12 have S13: \(\ if \((A \in \ C, A, 0) \in \ C\) \(by\) (rule MMI_elimel)

have S14: \(0 \in C\) \(by\) (rule MMI_0cn)

from S14 have S15: \(if \((B \in C, B, 0)\) \(\in\) \(\ C\) \(by\) (rule MMI_elimel)

from S13 S15 have S16: \(\ if \((A \in \ C, A, 0) \neq 0 \rightarrow \)

\((\ if \((A \in C, A, 0) \cdot if \((B \in C, B, 0)\) / \ if \((A \in C, A, 0)\)\) = \(if \((B \in C, B, 0)\)\) \(by\) (rule MMI_divcan3z)

from S6 S11 S16 have S17: \((A \in C \ Land B \in C) \rightarrow \)

\((A \neq 0 \rightarrow ((A \cdot B) / A) = B) \ by\ (rule\ MMI_dedth2h)

from S17 show \((A \in C \ Land B \in C \ Land A \neq 0) \rightarrow \)

\((A \cdot B) / A) = B \ by\ (rule\ MMI_3impia)\)

qed

lemma (in MMIar0) MMI_divcan4t:

shows \((A \in C \ Land B \in C \ Land A \neq 0) \rightarrow \)

\((B \cdot A) / A) = B\)

proof -

have S1: \((A \in C \ Land B \in C) \rightarrow \)

\((A \cdot B) = (B \cdot A)\) \(by\) (rule MMI_axmulcom)

from S1 have S2: \((A \in C \ Land B \in C) \rightarrow \)

\((A \cdot B) / A) = (B \cdot A) / A\) \(by\) (rule MMI_opreq1d)

from S2 have S3: \((A \in C \ Land B \in C \ Land A \neq 0) \rightarrow \)

\((A \cdot B) / A) = (B \cdot A) / A\) \(by\) (rule MMI_3adant3)

have S4: \((A \in C \ Land B \in C \ Land A \neq 0) \rightarrow \)

\((A \cdot B) / A) = B\) \(by\) (rule MMI_divcan3t)

from S3 S4 show \((A \in C \ Land B \in C \ Land A \neq 0) \rightarrow \)

\((B \cdot A) / A) = B\) \(by\) (rule MMI_eqtr3d)

qed

lemma (in MMIar0) MMI_div11: assumes A1: \(A \in C\) and

A2: \(B \in C\) and

A3: \(C \in C\) and

A4: \(C \neq 0\)

shows \((A / C) = (B / C) \leftrightarrow A = B\)

proof -

from A3 have S1: \(C \in C\).

from A1 have S2: \(A \in C\).

from A3 have S3: \(C \in C\).

from A4 have S4: \(C \neq 0\).
from S2 S3 S4 have S5: ( A / C ) ∈ C by (rule MMI_divcl)
from A2 have S6: B ∈ C.
from A3 have S7: C ∈ C.
from A4 have S8: C ≠ 0.
from S6 S7 S8 have S9: ( B / C ) ∈ C by (rule MMI_divcl)
from A4 have S10: C ≠ 0.
from S1 S5 S9 S10 have S11: ( C · ( A / C ) ) =
( ( C · ( B / C ) ) ←→
( A / C ) = ( B / C ) ) by (rule MMI_mulcan)
from A3 have S12: C ∈ C.
from A1 have S13: A ∈ C.
from A4 have S14: C ≠ 0.
from S12 S13 S14 have S15: ( C · ( A / C ) ) = A by (rule MMI_divcan2)
from A3 have S16: C ∈ C.
from A2 have S17: B ∈ C.
from A4 have S18: C ≠ 0.
from S16 S17 S18 have S19: ( C · ( B / C ) ) = B by (rule MMI_divcan2)
from S15 S19 have S20: ( ( A / C ) = ( B / C ) ←→
A = B ) by (rule MMI_bitr3)
qed

lemma (in MMIIsar0) MMI_div11t:
shows ( ( A ∈ C ∧ B ∈ C ∧ ( C ∈ C ∧ C ≠ 0 ) ) ) −→
( ( A / C ) = ( B / C ) ←→ A = B )
proof -
  have S1: A =
    if ( A ∈ C , A , 1 ) −→
    ( A / C ) =
    ( if ( A ∈ C , A , 1 ) / C ) by (rule MMI_opreq1)
  from S1 have S2: A =
    if ( A ∈ C , A , 1 ) −→
    ( ( A / C ) =
    ( B / C ) ←→
    ( if ( A ∈ C , A , 1 ) / C ) =
    ( B / C ) ) by (rule MMI_eqeq1d)
  have S3: A =
    if ( A ∈ C , A , 1 ) −→
    ( A = B ←→ if ( A ∈ C , A , 1 ) = B ) by (rule MMI_eqeq1)
  from S2 S3 have S4: A =
    if ( A ∈ C , A , 1 ) −→
    ( ( ( A / C ) = ( B / C ) ←→ A = B ) ←→
    ( ( if ( A ∈ C , A , 1 ) / C ) =
    ( B / C ) ←→
    if ( A ∈ C , A , 1 ) = B ) ) by (rule MMI_bibi12d)
  have S5: B =
    if ( B ∈ C , B , 1 ) −→
    ( B / C ) =
    ( if ( B ∈ C , B , 1 ) / C ) by (rule MMI_opreq1)
from S5 have S6: B =
if ( B ∈ f, B , 1 ) →
(( if ( A ∈ C , A , 1 ) / C ) =
( ( B / C ) )
( if ( A ∈ C , A , 1 ) / C ) =
( if ( B ∈ C , B , 1 ) / C ) ) by (rule MMI_eqeq2d)
have S7: B =
if ( B ∈ C , B , 1 ) →
( if ( A ∈ C , A , 1 ) =
B )
if ( ( A ∈ C , A , 1 ) ) by (rule MMI_eqeq2)
from S6 S7 have S8: B =
if ( B ∈ C , B , 1 ) →
(( if ( A ∈ C , A , 1 ) / C ) = ( B / C ) ←→ if ( A ∈ C , A , 1 )
) =
if ( ( A ∈ C , A , 1 ) / C ) =
if ( ( B ∈ C , A , 1 ) =
if ( B ∈ C , B , 1 ) ) ) by (rule MMI_bibi12d)
have S9: C =
if ( ( C ∈ C ∧ C ≠ 0 ), C , 1 ) →
( if ( A ∈ C , A , 1 ) / C ) =
( if ( B ∈ C , B , 1 ) / if ( ( C ∈ C ∧ C ≠ 0 ), C , 1 ) ) ) by (rule
MMI_opreq2)
have S10: C =
if ( ( C ∈ C ∧ C ≠ 0 ), C , 1 ) →
( if ( B ∈ C , B , 1 ) / C ) =
( if ( B ∈ C , C , 1 ) / if ( ( C ∈ C ∧ C ≠ 0 ), C , 1 ) ) ) by (rule
MMI_opreq2)
from S9 S10 have S11: C =
if ( ( C ∈ C ∧ C ≠ 0 ), C , 1 ) →
( if ( A ∈ C , A , 1 ) / C ) =
( if ( B ∈ C , B , 1 ) / if ( ( C ∈ C ∧ C ≠ 0 ), C , 1 ) ) =
( if ( B ∈ C , B , 1 ) / if ( ( C ∈ C ∧ C ≠ 0 ), C , 1 ) ) ) by (rule
MMI_eqeq12d)
from S11 have S12: C =
if ( ( C ∈ C ∧ C ≠ 0 ), C , 1 ) →
( if ( A ∈ C , A , 1 ) / C ) =
if ( A ∈ C , A , 1 ) = if ( B ∈ C , B , 1 ) ) ←→
( if ( A ∈ C , A , 1 ) / if ( ( C ∈ C ∧ C ≠ 0 ), C , 1 ) ) =
( if ( B ∈ C , B , 1 ) / if ( ( C ∈ C ∧ C ≠ 0 ), C , 1 ) ) ←→
if ( A ∈ C , A , 1 ) =
if ( B ∈ C , B , 1 ) ) ) by (rule MMI_bibi1d)
have S13: 1 ∈ f by (rule MMI_1cn)
from S13 have S14: if ( A ∈ C , A , 1 ) ∈ C by (rule MMI_elimel)
have S15: 1 ∈ C by (rule MMI_1cn)
from S15 have S16: if ( B ∈ C , B , 1 ) ∈ C by (rule MMI_elimel)
have S17: C =
if ( ( C ∈ C ∧ C \neq 0 ) , C , 1 ) \rightarrow
( C ∈ C \leftarrow
if ( ( C ∈ C ∧ C \neq 0 ) , C , 1 ) ∈ C ) by (rule MMI_eleq1)
have S18: C =
if ( ( C ∈ C ∧ C \neq 0 ) , C , 1 ) \rightarrow
( C \neq 0 \leftarrow
if ( ( C ∈ C ∧ C \neq 0 ) , C , 1 ) \neq 0 ) by (rule MMI_neeq1)
from S17 S18 have S19: C =
if ( ( C ∈ C ∧ C \neq 0 ) , C , 1 ) \rightarrow
( ( C ∈ C ∧ C \neq 0 ) \leftarrow
( if ( ( C ∈ C ∧ C \neq 0 ) , C , 1 ) ∈ C ∧ if ( ( C ∈ C ∧ C \neq 0 ) ,
C , 1 ) \neq 0 ) ) by (rule MMI_anbi12d)
have S20: 1 =
if ( ( C ∈ C ∧ C \neq 0 ) , C , 1 ) \rightarrow
( 1 ∈ C \leftarrow
if ( ( C ∈ C ∧ C \neq 0 ) , C , 1 ) ∈ C ) by (rule MMI_eleq1)
have S21: 1 =
if ( ( C ∈ C ∧ C \neq 0 ) , C , 1 ) \rightarrow
( 1 \neq 0 \leftarrow
if ( ( C ∈ C ∧ C \neq 0 ) , C , 1 ) \neq 0 ) by (rule MMI_neeq1)
from S20 S21 have S22: 1 =
if ( ( C ∈ C ∧ C \neq 0 ) , C , 1 ) \rightarrow
( ( 1 ∈ C ∧ 1 \neq 0 ) \leftarrow
( if ( ( C ∈ C ∧ C \neq 0 ) , C , 1 ) ∈ C ∧ if ( ( C ∈ C ∧ C \neq 0 ) ,
C , 1 ) \neq 0 ) ) by (rule MMI_anbi12d)
have S23: 1 ∈ C by (rule MMI_1cn)
have S24: 1 \neq 0 by (rule MMI_ax1ne0)
from S23 S24 have S25: 1 ∈ C ∧ 1 \neq 0 by (rule MMI_pm3_2i)
from S19 S22 S25 have S26: if ( ( C ∈ C ∧ C \neq 0 ) , C , 1 ) ∈ C
∧ if ( ( C ∈ C ∧ C \neq 0 ) , C , 1 ) \neq 0 by (rule MMI_elimhyp)
from S26 have S27: if ( ( C ∈ C ∧ C \neq 0 ) , C , 1 ) ∈ C by (rule
MMI_pm3_26i)
from S26 have S28: if ( ( C ∈ C ∧ C \neq 0 ) , C , 1 ) ∈ C ∧ if ( ( C ∈
C ∧ C \neq 0 ) , C , 1 ) \neq 0 .
from S28 have S29: if ( ( C ∈ C ∧ C \neq 0 ) , C , 1 ) \neq 0 by (rule
MMI_pm3_27i)
from S14 S16 S27 S29 have S30: ( if ( A ∈ C , A , 1 ) / if ( ( C ∈
C ∧ C \neq 0 ) , C , 1 ) ) =
( if ( B ∈ C , B , 1 ) / if ( ( C ∈ C ∧ C \neq 0 ) , C , 1 ) ) \leftarrow
if ( A ∈ C , A , 1 ) =
if ( B ∈ C , B , 1 ) by (rule MMI_div11)
from S4 S9 S12 S30 show ( A ∈ C ∧ B ∈ C ∧ ( C ∈ C ∧ C \neq 0 ) ) \rightarrow
( ( A / C ) = ( B / C ) \leftarrow A = B ) by (rule MMI_dedth3h)
qed
97 Metamath examples

theory MMI_examples imports MMI_Complex_ZF

begin

This theory contains 10 theorems translated from Metamath (with proofs).
It is included in the proof document as an illustration of how a translated
Metamath proof looks like. The "known_theorems.txt" file included in the
IsarMathLib distribution provides a list of all translated facts.

lemma (in MMIIsar0) MMI_dividt:
shows ( A ∈ C ∧ A ≠ 0 ) → ( A / A ) = 1

proof -
  have S1: ( A ∈ C ∧ A ∈ C ∧ A ≠ 0 ) →
    ( A / A ) = ( A · ( 1 / A ) ) by (rule MMI_divrect)
  from S1 have S2: ( ( A ∈ C ∧ A ∈ C ) ∧ A ≠ 0 ) →
    ( A / A ) = ( A · ( 1 / A ) ) by (rule MMI_3expa)
  from S2 have S3: ( A ∈ C ∧ A ≠ 0 ) →
    ( A / A ) = ( A · ( 1 / A ) ) by (rule MMI_anabsan)
    have S4: ( A ∈ C ∧ A ≠ 0 ) →
      ( A · ( 1 / A ) ) = 1 by (rule MMI_recidt)
  from S3 S4 show ( A ∈ C ∧ A ≠ 0 ) → ( A / A ) = 1 by (rule MMI_eqtrd)
qed

lemma (in MMIIsar0) MMI_div0t:
shows ( A ∈ C ∧ A ≠ 0 ) → ( 0 / A ) = 0

proof -
  have S1: 0 ∈ C by (rule MMI_0cn)
  have S2: ( 0 ∈ C ∧ A ∈ C ∧ A ≠ 0 ) →
    ( 0 / A ) = ( 0 · ( 1 / A ) ) by (rule MMI_divrect)
  from S1 S2 have S3: ( A ∈ C ∧ A ≠ 0 ) →
    ( 0 / A ) = ( 0 · ( 1 / A ) ) by (rule MMI_mp3an1)
    have S4: ( A ∈ C ∧ A ≠ 0 ) → ( 1 / A ) ∈ C by (rule MMI_reclt)
    have S5: ( 1 / A ) ∈ C → ( 0 · ( 1 / A ) ) = 0
      by (rule MMI_mul02t)
  from S4 S5 have S6: ( A ∈ C ∧ A ≠ 0 ) →
    ( 0 · ( 1 / A ) ) = 0 by (rule MMI_syl)
  from S3 S6 show ( A ∈ C ∧ A ≠ 0 ) → ( 0 / A ) = 0 by (rule MMI_eqtrd)
qed

lemma (in MMIIsar0) MMI_diveq0t:
shows ( A ∈ C ∧ C ∈ C ∧ C ≠ 0 ) →
( ( A / C ) = 0 ↔ A = 0 )

proof -
  have S1: ( C ∈ C ∧ C ≠ 0 ) → ( 0 / C ) = 0 by (rule MMI_div0t)
  from S1 have S2: ( C ∈ C ∧ C ≠ 0 ) →
    ( A / C ) =
    ( 0 / C ) ↔ ( A / C ) = 0 by (rule MMI_equeq2d)
  from S2 have S3: ( A ∈ C ∧ C ∈ C ∧ C ≠ 0 ) →
\[(A / C) =
(0 / C) \iff (A / C) = 0\] by (rule MMI_3adant1)

have S4: \(0 \in C\) by (rule MMI_0cn)

have S5: \((A \in C \land 0 \in C \land (C \in C \land C \neq 0)) \implies (A / C) = (0 / C) \iff A = 0\) by (rule MMI_div11t)

from S4 S5 have S6: \((A \in C \land (C \in C \land C \neq 0)) \implies (A / C) = (0 / C) \iff A = 0\) by (rule MMI_mp3an2)

from S6 have S7: \((A \in C \land C \in C \land C \neq 0) \implies (A / C) = (0 / C) \iff A = 0\) by (rule MMI_3impb)

from S3 S7 have S8: \((A \in C \land C \in C \land C \neq 0) \implies (A / C) = (0 / C) \iff A = 0\) by (rule MMI_bitr3d)

qed

lemma (in MMIIsar0) MMI_recrec: assumes A1: \(A \in C\) and
A2: \(A \neq 0\)
shows \(1 / (1 / A)\) = \(A\)

proof -

from A1 have S1: \(A \in C\).
from A2 have S2: \(A \neq 0\).

from S1 S2 have S3: \((1 / A) \in C\) by (rule MMI_reccl)

have S4: \(1 \in C\) by (rule MMI_1cn)

from A1 have S5: \(A \in C\).

have S6: \(1 \neq 0\) by (rule MMI_axine0)

from A2 have S7: \(A \neq 0\).

from S4 S5 S6 S7 have S8: \((1 / A) \neq 0\) by (rule MMI_divne0)

from S3 S8 have S9: \((1 / A) \cdot (1 / (1 / A))\) = \(1\)

by (rule MMI_recid)

from S9 have S10: \((A \cdot (1 / A) \cdot (1 / (1 / A)))\) = \((A \cdot 1)\) by (rule MMI_opreq2i)

from A1 have S11: \(A \in C\).

from A2 have S12: \(A \neq 0\).

from S11 S12 have S13: \((A \cdot (1 / A)) = 1\) by (rule MMI_recid)

from S13 have S14: \((A \cdot (1 / A)) \cdot (1 / (1 / A))\) = \((1 \cdot (1 / (1 / A)))\) by (rule MMI_opreq1i)

from A1 have S15: \(A \in C\).

from S3 have S16: \((1 / A) \in C\).

from S3 have S17: \((1 / A) \in C\).

from S8 have S18: \((1 / A) \neq 0\).

from S17 S18 have S19: \((1 / (1 / A)) \in C\) by (rule MMI_reccl)

from S15 S16 S19 have S20:

\((A \cdot (1 / A) \cdot (1 / (1 / A)))\) = \((A \cdot ((1 / A) \cdot (1 / (1 / A))))\) by (rule MMI_mulass)

from S19 have S21: \((1 / (1 / A)) \in C\).

from S21 have S22: \((1 \cdot (1 / (1 / A)))\) = \((1 / (1 / A))\) by (rule MMI_mulid2)

from S14 S20 S22 have S23:

\((A \cdot ((1 / A) \cdot (1 / (1 / A))))\) = \((1 / (1 / A))\) by (rule MMI_3eqtr3)

from A1 have S24: \(A \in C\).
from S24 have S25: (A · 1) = A by (rule MMI_mulid1)
from S10 S23 S25 show (1 / (1 / A)) = A by (rule MMI_3eqtr3)

qed

lemma (in MMIsar0) MMI_divid: assumes A1: A ∈ C and
a2: A ≠ 0
shows (A / A) = 1
proof -
  from A1 have S1: A ∈ C.
  from A1 have S2: A ∈ C.
  from A2 have S3: A ≠ 0.
  from S1 S2 S3 have S4: (A / A) = (A · (1 / A)) by (rule MMI_divrec)
  from A1 have S5: A ∈ C.
  from A2 have S6: A ≠ 0.
  from S5 S6 have S7: (A · (1 / A)) = 1 by (rule MMI_recid)
  from S4 S7 show (A / A) = 1 by (rule MMI_eqtr)
qed

lemma (in MMIsar0) MMI_div0: assumes A1: A ∈ C and
a2: A ≠ 0
shows (0 / A) = 0
proof -
  from A1 have S1: A ∈ C.
  from A2 have S2: A ≠ 0.
  have S3: (A ∈ C ∧ A ≠ 0) → (0 / A) = 0 by (rule MMI_div0t)
  from S1 S2 S3 show (0 / A) = 0 by (rule MMI_mp2an)
qed

lemma (in MMIsar0) MMI_div1: assumes A1: A ∈ C
shows (A / 1) = A
proof -
  from A1 have S1: A ∈ C.
  from S1 have S2: (1 · A) = A by (rule MMI_mulid2)
  from A1 have S3: A ∈ C.
  have S4: 1 ∈ C by (rule MMI_1cn)
  from A1 have S5: A ∈ C.
  have S6: 1 ≠ 0 by (rule MMI_axine0)
  from S3 S4 S5 S6 have S7: (A / 1) = A ↔ (1 · A) = A
    by (rule MMI_divmul)
  from S2 S7 show (A / 1) = A by (rule MMI_mpbir)
qed

lemma (in MMIsar0) MMI_div1t:
  shows A ∈ C → (A / 1) = A
proof -
  have S1: A =
    if (A ∈ C, A, 1) →
    (A / 1) =
    (if (A ∈ C, A, 1) / 1) by (rule MMI_opreq1)
have S2: A = if ( A ∈ ƒ, A, 1 ) —> A = if ( A ∈ ƒ, A, 1 ) by (rule MMI_id)

from S1 S2 have S3: A = if ( A ∈ ƒ, A, 1 ) —> 
( ( A / 1 ) = 
A —>
( if ( A ∈ C, A, 1 ) / 1 ) =
if ( A ∈ C, A, 1 ) by (rule MMI_eqeql2d)

have S4: 1 ∈ ƒ by (rule MMI_1cn)

from S4 have S5: if ( A ∈ ƒ, A, 1 ) ∈ ƒ by (rule MMI_elimel)

from S5 have S6: ( if ( A ∈ ƒ, A, 1 ) / 1 ) = if ( A ∈ ƒ, A, 1 ) by (rule MMI_div1)

from S3 S6 show A ∈ ƒ —> ( A / 1 ) = A by (rule MMI_dedth)

qed

lemma (in MMIIsar0) MMI_divnegt:
shows ( A ∈ C ∧ B ∈ C ∧ B ≠ 0 ) —> 
( ( A / B ) ) = ( ( - A ) / B )

proof -

have S1: ( A ∈ C ∧ ( 1 / B ) ∈ C ) —> 
( ( - A ) · ( 1 / B ) ) = 
( ( - A ) · ( 1 / B ) ) by (rule MMI_mulneg1t)

have S2: ( B ∈ C ∧ B ≠ 0 ) —> ( 1 / B ) ∈ C by (rule MMI_reclt)

from S1 S2 have S3: ( A ∈ C ∧ ( B ∈ C ∧ B ≠ 0 ) ) —> 
( ( - A ) · ( 1 / B ) ) = 
( ( - A ) · ( 1 / B ) ) by (rule MMI_sylan2)

from S3 have S4: ( A ∈ C ∧ B ∈ C ∧ B ≠ 0 ) —> 
( ( - A ) · ( 1 / B ) ) = 
( ( - A ) · ( 1 / B ) ) by (rule MMI_3impb)

have S5: ( ( - A ) ∈ C ∧ B ∈ C ∧ B ≠ 0 ) —> 
( ( - A ) / B ) = 
( ( - A ) · ( 1 / B ) ) by (rule MMI_divrect)

have S6: A ∈ C —> ( - A ) ∈ C by (rule MMI_negclt)

from S5 S6 have S7: ( A ∈ C ∧ B ∈ C ∧ B ≠ 0 ) —> 
( ( - A ) / B ) = 
( ( - A ) . ( 1 / B ) ) by (rule MMI_syl3an1)

have S8: ( A ∈ C ∧ B ∈ C ∧ B ≠ 0 ) —> 
( A / B ) = ( A · ( 1 / B ) ) by (rule MMI_divrect)

from S8 have S9: ( A ∈ C ∧ B ∈ C ∧ B ≠ 0 ) —> 
( - ( A / B ) ) = 
( ( - A · ( 1 / B ) ) ) by (rule MMI_negeqd)

from S4 S7 S9 show ( A ∈ C ∧ B ∈ C ∧ B ≠ 0 ) —> 
( - ( A / B ) ) = ( ( - A ) / B ) by (rule MMI_3eqtr4rd)

qed

lemma (in MMIIsar0) MMI_divsubdirt:
shows ( ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) ∧ C ≠ 0 ) —> 
( ( A - B ) / C ) =
proof -

have S1: \((A \in C \land (B \in C \land C \in C) \land C \neq 0)\) \(\longrightarrow\)
\((A + (-B)) / C) = \((A / C) + ((-B) / C))\) by \(\text{rule MMI_divdirt}\)

have S2: \(B \in C \longrightarrow (-B) \in C\) by \(\text{rule MMI_negclt}\)
from S1 S2 have S3: \(((A \in C \land B \in C \land C \in C) \land C \neq 0)\) \(\longrightarrow\)
\((A + (-B)) / C) = \((A / C) + ((-B) / C))\) by \(\text{rule MMI_syl3anl2}\)

have S4: \((A \in C \land B \in C)\) \(\longrightarrow\)
\((A + (-B)) / C) = \((A - B)\) by \(\text{rule MMI_negsubt}\)
from S4 have S5: \((A \in C \land B \in C \land C \in C)\) \(\longrightarrow\)
\((A + (-B)) = (A - B)\) by \(\text{rule MMI_3adant3}\)
from S5 have S6: \((A \in C \land B \in C \land C \in C)\) \(\longrightarrow\)
\((A + (-B)) / C) = \((A - B) / C\) by \(\text{rule MMI_opreq1d}\)
from S6 have S7: \(((A \in C \land B \in C \land C \in C) \land C \neq 0)\) \(\longrightarrow\)
\((A + (-B)) / C) = \((A - B) / C\) by \(\text{rule MMI_adantr}\)

have S8: \((B \in C \land C \in C \land C \neq 0)\) \(\longrightarrow\)
\((B / C) = ((-B) / C)\) by \(\text{rule MMI_divnegt}\)
from S8 have S9: \(((B \in C \land C \in C) \land C \neq 0)\) \(\longrightarrow\)
\((B / C) = ((-B) / C)\) by \(\text{rule MMI_3expa}\)
from S9 have S10: \(((A \in C \land B \in C \land C \in C) \land C \neq 0)\) \(\longrightarrow\)
\((B / C) = ((-B) / C)\) by \(\text{rule MMI_3adant1}\)
from S10 have S11: \(((A \in C \land B \in C \land C \in C) \land C \neq 0)\) \(\longrightarrow\)
\((A / C) + ((-B) / C)) = \((A / C) + ((-B) / C))\) by \(\text{rule MMI_opreq2d}\)

have S12: \(((A / C) \in C \land (B / C) \in C)\) \(\longrightarrow\)
\((A / C) + ((-B) / C)) = \((A / C) - (B / C))\) by \(\text{rule MMI_negsubt}\)

have S13: \((A \in C \land C \in C \land C \neq 0)\) \(\longrightarrow\)
\((A / C) \in C\) by \(\text{rule MMI_divclt}\)
from S13 have S14: \(((A \in C \land C \in C) \land C \neq 0)\) \(\longrightarrow\)
\((A / C) \in C\) by \(\text{rule MMI_3expa}\)
from S14 have S15: \(((A \in C \land B \in C \land C \in C) \land C \neq 0)\) \(\longrightarrow\)
\((A / C) \in C\) by \(\text{rule MMI_3adant1}\)

have S16: \((B \in C \land C \in C \land C \neq 0)\) \(\longrightarrow\)
\((B / C) \in C\) by \(\text{rule MMI_divclt}\)
from S16 have S17: \(((B \in C \land C \in C) \land C \neq 0)\) \(\longrightarrow\)
\((B / C) \in C\) by \(\text{rule MMI_3expa}\)
from S17 have S18: \(((A \in C \land B \in C \land C \in C) \land C \neq 0)\) \(\longrightarrow\)
\((B / C) \in C\) by \(\text{rule MMI_3adant1}\)
from S12 S15 S18 have S19: \(((A \in C \land B \in C \land C \in C) \land C \neq 0)\) \(\longrightarrow\)
\((A / C) + ((-B) / C)) = \((A / C) - (B / C))\) by \(\text{rule MMI_sylanc}\)
from S11 S19 have S20: \(((A \in C \land B \in C \land C \in C) \land C \neq 0)\) \(\longrightarrow\)

1553
\[
\left( \frac{A}{C} \right) + \left( \frac{-B}{C} \right) = \\
\left( \frac{A}{C} \right) - \left( \frac{B}{C} \right)
\]

by (rule MMI_eqtr3d)

from S3 S7 S20 show \((A \in C \land B \in C \land C \neq 0) \implies \\
\left( \frac{A - B}{C} \right) = \\
\left( \frac{A}{C} \right) - \left( \frac{B}{C} \right)\)

by (rule MMI_3eqtr3d)

qed

end

98 Metamath interface

theory Metamath_Interface imports Complex_ZF MMI_prelude

begin

This theory contains some lemmas that make it possible to use the theorems translated from Metamath in a the complex0 context.

98.1 MMIsar0 and complex0 contexts.

In the section we show a lemma that the assumptions in complex0 context imply the assumptions of the MMIsar0 context. The Metamath_sampler theory provides examples how this lemma can be used.

The next lemma states that we can use the theorems proven in the MMIsar0 context in the complex0 context. Unfortunately we have to use low level Isabelle methods "rule" and "unfold" in the proof, simp and blast fail on the order axioms.

lemma (in complex0) MMIsar_valid:
  shows MMIsar0(R,C,1,0,i,CplxAdd(R,A),CplxMul(R,A,M), 
                   StrictVersion(CplxROrder(R,A,r)))

proof -
  let real = R
  let complex = C
  let zero = 0
  let one = 1
  let iunit = i
  let caddset = CplxAdd(R,A)
  let cmulset = CplxMul(R,A,M)
  let lessrrel = StrictVersion(CplxROrder(R,A,r))
  have (\forall a b. a \in real \land b \in real ---
       (a, b) \in lessrrel \iff \neg (a = b \lor (b, a) \in lessrrel))
  proof -
    have I:
      \forall a b. a \in R \land b \in R --- (a <_R b \iff \neg(a=b \lor b <_R a))
    using pre_axlttri by blast

1554
\{ \text{fix a b assume } a \in \text{real} \land b \in \text{real} \\
\text{with I have } (a \lt_R b \iff \neg (a = b \lor b \lt_R a)) \}

\text{by blast}

\text{hence}

\langle a, b \rangle \in \text{lessrrel} \iff \neg (a = b \lor \langle b, a \rangle \in \text{lessrrel})

\text{by simp}

\}

\text{thus } (\forall a \ b. \ a \in \text{real} \land b \in \text{real} \to

\langle a, b \rangle \in \text{lessrrel} \iff \neg (a = b \lor \langle b, a \rangle \in \text{lessrrel}))

\text{by blast}

\text{qed}

\text{moreover have } (\forall a \ b \ c. \ a \in \text{real} \land b \in \text{real} \land c \in \text{real} \to

\langle a, b \rangle \in \text{lessrrel} \land \langle b, c \rangle \in \text{lessrrel} \to \langle a, c \rangle \in \text{lessrrel})

\text{by blast}

\text{hence}

\langle a, b \rangle \in \text{lessrrel} \land \langle b, c \rangle \in \text{lessrrel} \to \langle a, c \rangle \in \text{lessrrel}

\text{by simp}

\}

\text{thus } (\forall a \ b \ c. \ a \in \text{real} \land b \in \text{real} \land c \in \text{real} \to

\langle a, b \rangle \in \text{lessrrel} \land \langle b, c \rangle \in \text{lessrrel} \to \langle a, c \rangle \in \text{lessrrel})

\text{by blast}

\text{qed}

\text{moreover have } (\forall A \ B \ C. \ A \in \text{real} \land B \in \text{real} \land C \in \text{real} \to

\langle A, B \rangle \in \text{lessrrel} \to

\langle \text{caddset } \langle C, A \rangle, \text{caddset } \langle C, B \rangle \rangle \in \text{lessrrel})

\text{by simp}

\text{moreover have } (\forall A \ B. \ A \in \text{real} \land B \in \text{real} \to

\langle \text{zero}, A \rangle \in \text{lessrrel} \land \langle \text{zero}, B \rangle \in \text{lessrrel} \to

\langle \text{zero}, \text{cmulset } \langle A, B \rangle \rangle \in \text{lessrrel})

\text{by simp}

\text{moreover have } (\forall S. \ S \subseteq \text{real} \land S \neq 0 \land \langle \exists x \in \text{real}. \ \forall y \in S. \ (y, x) \in \text{lessrrel} \rangle \to

\langle \exists x \in \text{real}. \ \forall y \in S. \ (x, y) \notin \text{lessrrel} \rangle \land

\langle \forall y \in \text{real}. \ (y, x) \in \text{lessrrel} \to \langle \exists z \in S. \ (y, z) \notin \text{lessrrel} \rangle \rangle)

\text{by simp}

\text{moreover have } R \subseteq C \text{ using axresscn by simp}

\text{moreover have } 1 \neq 0 \text{ using axne0 by simp}

\text{moreover have } C \text{ isASet by simp}

\text{moreover have } \text{CplxAdd}(R, A) : C \times C \to C

\text{by simp}
moreover have \( \text{CplxMul}(R, A, M) : C \times C \rightarrow C \)
using \text{axmulopr} by \text{simp}
moreover have
\[
\forall a. b. a \in C \land b \in C \implies a \cdot b = b \cdot a
\]
using \text{axmulcom} by \text{simp}
hence \( \forall a. b. a \in C \land b \in C \implies \text{cmulset} \langle a, b \rangle = \text{cmulset} \langle b, a \rangle \)
) by \text{simp}
moreover have \( \forall a. b. a \in C \land b \in C \implies a + b \in C \)
using \text{axaddcl} by \text{simp}
hence \( \forall a. b. a \in C \land b \in C \implies \text{caddset} \langle a, b \rangle \in C \)
) by \text{simp}
moreover have \( \forall a. b. a \in C \land b \in C \implies a \cdot b \in C \)
using \text{axmulcl} by \text{simp}
hence \( \forall a. b. a \in C \land b \in C \implies \text{cmulset} \langle a, b \rangle \in C \)
moreover have \( \forall a. b. a \in C \land b \in C \land C \in C \implies a \cdot (b + C) = a \cdot b + a \cdot C \)
using \text{axdistr} by \text{simp}
hence \( \forall a. b. C. a \in C \land b \in C \land C \in C \implies \text{cmulset} \langle a, \text{caddset} \langle b, C \rangle \rangle = \text{caddset} \langle \text{cmulset} \langle a, b \rangle, \text{cmulset} \langle a, C \rangle \rangle \)
) by \text{simp}
moreover have \( \forall a. b. a \in C \land b \in C \implies a + b = b + a \)
using \text{axaddcom} by \text{simp}
hence \( \forall a. b. a \in C \land b \in C \implies \text{caddset} \langle a, b \rangle = \text{caddset} \langle b, a \rangle \)
) by \text{simp}
moreover have \( \forall a. b. C. a \in C \land b \in C \land C \in C \implies a + b + C = a + (b + C) \)
using \text{axaddass} by \text{simp}
hence \( \forall a. b. C. a \in C \land b \in C \land C \in C \implies \text{caddset} \langle \text{caddset} \langle a, b \rangle, C \rangle = \text{caddset} \langle a, \text{caddset} \langle b, C \rangle \rangle \)
) by \text{simp}
moreover have \( \forall a. b. c. a \in C \land b \in C \land c \in C \implies a \cdot b \cdot c = a \cdot (b \cdot c) \)
using \text{axmulass} by \text{simp}
hence \( \forall a. b. C. a \in C \land b \in C \land C \in C \implies \text{cmulset} \langle \text{cmulset} \langle a, b \rangle, C \rangle = \text{cmulset} \langle a, \text{cmulset} \langle b, C \rangle \rangle \)
) by \text{simp}
moreover have \( 1 \in R \) using \text{axire} by \text{simp}
moreover have \( ii + 1 = 0 \)
using axi2m1 by simp
hence caddset ⟨cmulset ⟨i, i⟩, 1⟩ = 0 by simp
moreover have ∀a. a ∈ C → a + 0 = a
  using ax0id by simp
hence ∀a. a ∈ C → caddset ⟨a, 0⟩ = a by simp
moreover have i ∈ C using axicn by simp
moreover have ∀a. a ∈ C → (∃x∈C. a + x = 0)
  using axnegex by simp
hence ∀a. a ∈ C →
  (∃x∈C. caddset ⟨a, x⟩ = 0) by simp
moreover have ∀a. a ∈ C ∧ a ≠ 0 → (∃x∈C. a · x = 1)
  using axrnegex by simp
hence ∀a. a ∈ C ∧ a ≠ 0 →
  ( ∃x∈C. cmulset ⟨a, x⟩ = 1 ) by simp
moreover have ∀a. a ∈ C → a1 = a
  using axid by simp
hence ∀a. a ∈ C →
  cmulset ⟨a, 1⟩ = a by simp
moreover have ∀a b. a ∈ R ∧ b ∈ R → a + b ∈ R
  using axaddrcl by simp
hence ∀a b. a ∈ R ∧ b ∈ R →
  caddset ⟨a, b⟩ ∈ R by simp
moreover have ∀a b. a ∈ R ∧ b ∈ R → a · b ∈ R
  using axmulrcl by simp
hence ∀a b. a ∈ R ∧ b ∈ R →
  cmulset ⟨a, b⟩ ∈ R by simp
moreover have ∀a. a ∈ R → (∃x∈R. a + x = 0)
  using axrnegex by simp
hence ∀a. a ∈ R →
  ( ∃x∈R. caddset ⟨a, x⟩ = 0 ) by simp
moreover have ∀a. a ∈ R ∧ a ≠ 0 → (∃x∈R. a · x = 1)
  using axrrecex by simp
hence ∀a. a ∈ R ∧ a ≠ 0 →
  ( ∃x∈R. cmulset ⟨a, x⟩ = 1 ) by simp
ultimately have

\{ ( ∀a b.
   a ∈ R ∧ b ∈ R →
   ⟨a, b⟩ ∈ lessrrel ←→
   ¬ (a = b ∨ ⟨b, a⟩ ∈ lessrrel)
 ) \} ∧

\{ ( ∀a b C.
   a ∈ R ∧ b ∈ R ∧ C ∈ R →
   ⟨a, b⟩ ∈ lessrrel ∧
   ⟨b, C⟩ ∈ lessrrel →
 ) \}
\( \langle a, C \rangle \in \text{lessrrel} \)

\( \wedge \)

\( (\forall a \ b \ C. \ a \in R \wedge b \in R \wedge C \in R \rightarrow \langle a, b \rangle \in \text{lessrrel} \rightarrow \langle \text{caddset} \ (C, a), \text{caddset} \ (C, b) \rangle \in \text{lessrrel} ) \wedge \)

\( (\forall a \ b. \ a \in R \wedge b \in R \rightarrow \langle 0, a \rangle \in \text{lessrrel} \wedge \langle 0, b \rangle \in \text{lessrrel} \rightarrow \langle 0, \text{cmulset} \ (a, b) \rangle \in \text{lessrrel} ) \wedge \)

\( (\forall S. S \subseteq R \wedge S \neq 0 \wedge \ (\exists x \in R. \ \forall y \in S. \ \langle y, x \rangle \in \text{lessrrel} ) \rightarrow \ (\exists x \in R. \ (\forall y \in S. \ \langle y, x \rangle \notin \text{lessrrel} ) \wedge \ (\forall y \in R. \ \langle y, x \rangle \in \text{lessrrel} \rightarrow \ (\exists z \in S. \ \langle y, z \rangle \in \text{lessrrel} ) \) \)

\( ) \wedge \)

\( R \subseteq C \wedge 1 \neq 0 \)

\( ) \wedge \)

\( (C \text{ isASet} \wedge \text{caddset} \in C \times C \rightarrow C \wedge \text{cmulset} \in C \times C \rightarrow C) \wedge \)

\( ) \wedge \)

\( (\forall a \ b. \ a \in C \wedge b \in C \rightarrow \text{cmulset} \ (a, b) = \text{cmulset} \ (b, a) ) \wedge \)

\( (\forall a \ b. \ a \in C \wedge b \in C \rightarrow \)

1558
\[\text{caddset} \ (a, b) \in C \]
\]
\[\wedge\]
\[\left(\forall a \ b. \ a \in C \land b \in C \rightarrow \text{cmulset} \ (a, b) \in C\right) \]
\[\wedge\]
\[\left(\forall a \ b \ c. \ a \in C \land b \in C \land c \in C \rightarrow \text{cmulset} \ (a, \text{caddset} \ (b, c)) = \text{caddset} \ \left(\text{cmulset} \ (a, b), \text{cmulset} \ (a, c)\right)\right) \]
\[\wedge\]
\[\left(\forall a \ b \ c. \ a \in C \land b \in C \land c \in C \rightarrow \text{caddset} \ \left(\text{caddset} \ (a, b), c\right) = \text{caddset} \ \left(\text{caddset} \ (a, b), \text{caddset} \ (a, c)\right)\right) \]
\[\wedge\]
\[\left(\forall a \ b. \ a \in C \land b \in C \rightarrow \text{caddset} \ (a, b) = \text{caddset} \ (b, a)\right) \]
\[\wedge\]
\[\left(\forall a \ b \ c. \ a \in C \land b \in C \land c \in C \rightarrow \text{caddset} \ \left(\text{caddset} \ (a, b), c\right) = \text{caddset} \ \left(\text{caddset} \ (a, b), \text{caddset} \ (a, c)\right)\right) \]
\[\wedge\]
\[\left(\forall a \ b \ c. \ a \in C \land b \in C \land c \in C \rightarrow \text{caddset} \ \left(\text{caddset} \ (a, b), c\right) = \text{caddset} \ \left(\text{caddset} \ (a, b), \text{caddset} \ (a, c)\right)\right) \]
\[\wedge\]
\[\left(\forall a. \ a \in C \rightarrow \text{caddset} \ (a, 0) = a\right) \]
\[\wedge\]
\[i \in C\]
\[\wedge\]
\[ (\forall a. a \in C \rightarrow (\exists x \in C. \text{caddset } \langle a, x \rangle = 0) ) \wedge \]

\[ (\forall a. a \in C \wedge a \neq 0 \rightarrow (\exists x \in C. \text{cmulset } \langle a, x \rangle = 1) ) \wedge \]

\[ (\forall a. a \in C \rightarrow \text{cmulset } \langle a, 1 \rangle = a) \wedge \]

\[ (\forall a. a \in R \rightarrow (\exists x \in R. \text{caddset } \langle a, x \rangle = 0) ) \wedge \]

\[ (\forall a. a \in R \wedge a \neq 0 \rightarrow (\exists x \in R. \text{cmulset } \langle a, x \rangle = 1) ) \]

by blast
then show MMIsar0(R,C,1,0,i,CplxAdd(R,A),CplxMul(R,A,M), StrictVersion(CplxROrder(R,A,r))) unfolding MMIsar0_def by blast qed

end

99 Metamath sampler

theory Metamath_Sampler imports Metamath_Interface MMI_Complex_ZF_2 begin
The theorems translated from Metamath reside in the MMI_Complex_ZF, MMI_Complex_ZF_1 and MMI_Complex_ZF_2 theories. The proofs of these theorems are very verbose and for this reason the theories are not shown in the proof document or the FormaMath.org site. This theory file contains some examples of theorems translated from Metamath and formulated in the complex0 context. This serves two purposes: to give an overview of the material covered in the translated theorems and to provide examples of how to take a translated theorem (proven in the MMIsar0 context) and transfer it to the complex0 context. The typical procedure for moving a theorem from MMIsar0 to complex0 is as follows: First we define certain aliases that map names defined in the complex0 to their corresponding names in the MMIsar0 context. This makes it easy to copy and paste the statement of the theorem as displayed with ProofGeneral. Then we run the Isabelle from ProofGeneral up to the theorem we want to move. When the theorem is verified ProofGeneral displays the statement in the raw set theory notation, stripped from any notation defined in the MMIsar0 locale. This is what we copy to the proof in the complex0 locale. After that we just can write “then have ?thesis by simp” and the simplifier translates the raw set theory notation to the one used in complex0.

99.1 Extended reals and order

In this section we import a couple of theorems about the extended real line and the linear order on it.

Metamath uses the set of real numbers extended with \( +\infty \) and \( -\infty \). The \( +\infty \) and \( -\infty \) symbols are defined quite arbitrarily as \( \mathbb{C} \) and \( \{\mathbb{C}\} \), respectively. The next lemma that corresponds to Metamath’s renfdisj states that \( +\infty \) and \( -\infty \) are not elements of \( \mathbb{R} \).

**lemma (in complex0) renfdisj:** shows \( \mathbb{R} \cap \{+\infty,-\infty\} = 0 \)

**proof**

\[
\begin{align*}
&\text{let } \text{real} = \mathbb{R} \\
&\text{let } \text{complex} = \mathbb{C} \\
&\text{let } \text{one} = 1 \\
&\text{let } \text{zero} = 0 \\
&\text{let } \text{iunit} = i \\
&\text{let } \text{caddset} = \text{CplxAdd}(\mathbb{R},\mathbb{A}) \\
&\text{let } \text{cmulset} = \text{CplxMul}(\mathbb{R},\mathbb{A},\mathbb{M}) \\
&\text{let } \text{lessrrel} = \text{StrictVersion}(\text{CplxROrder}(\mathbb{R},\mathbb{A},\mathbb{r})) \\
&\text{have } \text{MMIsar0} \\
&\quad (\text{real, complex, one, zero, iunit, caddset, cmulset, lessrrel}) \\
&\quad \text{using } \text{MMIsar_valid by simp} \\
&\quad \text{then have } \text{real} \cap \{\text{complex}, \{\text{complex}\}\} = 0 \\
&\quad \text{by } (\text{rule MMIsar0.MMI_renfdisj}) \\
&\text{thus } \mathbb{R} \cap \{+\infty,-\infty\} = 0 \text{ by simp} \\
\end{align*}
\]

qed
The order relation used most often in Metamath is defined on the set of complex reals extended with $+\infty$ and $-\infty$. The next lemma allows to use Metamath’s xrltso that states that the $<$ relations is a strict linear order on the extended set.

**Lemma** (in complex0) xrltso: shows $<\text{Orders}\ R^*$

**Proof**

1. Let $\text{real} = R$
2. Let $\text{complex} = C$
3. Let $\text{one} = 1$
4. Let $\text{zero} = 0$
5. Let $\text{iunit} = i$
6. Let $\text{caddset} = \text{CplxAdd}(R,A)$
7. Let $\text{cmulset} = \text{CplxMul}(R,A,M)$
8. Let $\text{lessrrel} = \text{StrictVersion}(\text{CplxROrder}(R,A,r))$
9. Have $\text{MMIsar0}$
   - $\langle \text{real, complex, one, zero, iunit, caddset, cmulset, lessrrel} \rangle$
   - Using $\text{MMIsar\ valid}$ by simp
10. Then have
    - $(\text{lessrrel} \cap \text{real} \times \text{real} \cup \langle \{\text{complex}\}, \text{complex} \rangle \cup \text{real} \times \{\text{complex}\} \cup \{\text{complex}\} \times \text{real}) \text{Orders} (\text{real} \cup \{\text{complex}, \{\text{complex}\}\})$
    - By rule $\text{MMIsar0.MM\_xrltso}$
11. Moreover have $\text{lessrrel} \cap \text{real} \times \text{real} = \text{lessrrel}$
    - Using $\text{cplx\_strict\_ord\_on\_cplx\_reals}$ by auto
12. Ultimately show $<\text{Orders}\ R^*$ by simp
13. Qed

Metamath defines the usual $<$ and $\leq$ ordering relations for the extended real line, including $+\infty$ and $-\infty$.

**Lemma** (in complex0) xrrebndt: assumes A1: $x \in R^*$

**Proof**

1. Let $\text{real} = R$
2. Let $\text{complex} = C$
3. Let $\text{one} = 1$
4. Let $\text{zero} = 0$
5. Let $\text{iunit} = i$
6. Let $\text{caddset} = \text{CplxAdd}(R,A)$
7. Let $\text{cmulset} = \text{CplxMul}(R,A,M)$
8. Let $\text{lessrrel} = \text{StrictVersion}(\text{CplxROrder}(R,A,r))$
9. Have $\text{MMIsar0}$
   - $\langle \text{real, complex, one, zero, iunit, caddset, cmulset, lessrrel} \rangle$
   - Using $\text{MMIsar\ valid}$ by simp
10. Then have $x \in R \cup \{C, \{C\}\} \rightarrow$
    - $x \in R \leftrightarrow \langle \{C\}, x \rangle \in \text{lessrrel} \cap \text{R} \times \text{R} \cup \langle \{\{C\}, C\} \cup \text{R} \times \{\{C\}\} \cup \{\text{C}\} \times \{\text{C}\} \cup \{\{\text{C}\}\} \times \{\text{C}\}\}$
    - $\langle x, C \rangle \in \text{lessrrel} \cap \text{R} \times \text{R} \cup \langle \{\{C\}, C\} \cup \text{R} \times \{\{C\}\} \cup \{\text{C}\} \cup \{\{\text{C}\}\} \times \text{R}$

1562
A quite involved inequality.

**Lemma (in complex0) lt2mul2divt:**

assumes $A_1$: $a \in \mathbb{R}$, $b \in \mathbb{R}$, $c \in \mathbb{R}$, $d \in \mathbb{R}$ and

$A_2$: $0 < b \leq d$

shows $a/2 < c/d \iff a/b < c/d$

**Proof** -

- let real = $\mathbb{R}$
- let complex = $\mathbb{C}$
- let one = 1
- let iunit = i
- let caddset = CplxAdd(R,A)
- let cmulset = CplxMul(R,A,M)
- let lessrrel = StrictVersion(CplxROrder(R,A,r))

have MMIsar0

(by rule MMIsar0.MMI_xrrebdnt)

then have $x \in \mathbb{R}^+ \implies (x \in \mathbb{R} \iff -\infty < x \land x < +\infty)$

(by simp)

with $A_1$ show thesis by simp

qed

A real number is smaller than its half iff it is positive.

**Lemma (in complex0) halfpos:** assumes $A_1$: $a \in \mathbb{R}$

shows $0 < a \iff a/2 < a$

**Proof** -

- let real = $\mathbb{R}$
- let complex = $\mathbb{C}$
- let one = 1
- let zero = 0
let iunit = i
let caddset = CplxAdd(R,A)
let cmulset = CplxMul(R,A,M)
let lessrrel = StrictVersion(CplxROrder(R,A,r))

from A1 have MMIsar0
  (real, complex, one, zero, iunit, caddset, cmulset, lessrrel)
  and a ∈ real
  using MMIsar_valid by auto
then have
  (zero, a) ∈
  lessrrel ∩ real × real ∪ {(complex), complex} ∪
  real × {complex} ∪ {complex} ∪ real ←→
  (⎨(x ∈ complex . cmulset (caddset (one, one), x) = a}, a) ∈
  lessrrel ∩ real × real ∪
  {(complex), complex} ∪ real × {complex} ∪ {complex} × real
  by (rule MMIsar0.MMI_halfpos)
then show thesis by simp
qed

One more inequality.

lemma (in complex0) ledivp1t:
  assumes A1: a ∈ R b ∈ R and
  A2: 0 ≤ a 0 ≤ b
  shows (a/(b + 1))·b ≤ a

proof -
  let real = R
  let complex = C
  let one = 1
  let zero = 0
  let iunit = i
  let caddset = CplxAdd(R,A)
  let cmulset = CplxMul(R,A,M)
  let lessrrel = StrictVersion(CplxROrder(R,A,r))
  have MMIsar0
    (real, complex, one, zero, iunit, caddset, cmulset, lessrrel)
    using MMIsar_valid by simp
  then have
    (a ∈ real ∧ (a, zero) /∈
     lessrrel ∩ real × real ∪ {(complex), complex} ∪
     real × {complex} ∪ {complex} ∪ real) ∧
    b ∈ real ∧ (b, zero) /∈ lessrrel ∩ real × real ∪
    {(complex), complex} ∪ real × {complex} ∪
    {complex} × real ←→
    (a,cmulset(∪{x ∈ complex . cmulset(caddset(b, one), x) = a}, b) /∈
     lessrrel ∩ real × real ∪ {(complex), complex} ∪
     real × {complex} ∪ {complex} × real
     by (rule MMIsar0.MMI_ledivp1t)
  with A1 A2 show thesis by simp
qed
99.2 Natural real numbers

In standard mathematics, natural numbers are treated as a subset of real numbers. From the set theory point of view, however, they are quite different objects. In this section, we talk about "real natural" numbers, i.e., the counterpart of natural numbers that is a subset of the reals.

Two ways of saying that there are no natural numbers between \( n \) and \( n + 1 \).

**Lemma** (in complex0) no_nats_between:
- Assumes \( A1: n \in \mathbb{N} \quad k \in \mathbb{N} \)
- Shows
  - \( n \leq k \iff n < k + 1 \)
  - \( n < k \iff n + 1 \leq k \)
- Proof -
  - Let \( real = R \)
  - Let \( complex = C \)
  - Let \( one = 1 \)
  - Let \( iunit = i \)
  - Let \( caddset = \text{CplxAdd}(R,A) \)
  - Let \( cmulset = \text{CplxMul}(R,A,M) \)
  - Let \( lessrrel = \text{StrictVersion}(\text{CplxROrder}(R,A,r)) \)
  - Have \( I: \text{MMIsar0} \)
    - \( (\text{real, complex, one, zero, iunit, caddset, cmulset, lessrrel}) \)
    - Using \( \text{MMIsar_valid by simp} \)
  - Then have
    - \( n \in \bigcap \{ N \in \text{Pow}(\text{real}) . \text{one} \in N \quad (\forall n. n \in N \rightarrow \text{caddset} \langle n, \text{one} \rangle \in N) \} \land \)
    - \( k \in \bigcap \{ N \in \text{Pow}(\text{real}) . \text{one} \in N \quad (\forall n. n \in N \rightarrow \text{caddset} \langle n, \text{one} \rangle \in N) \} \rightarrow \)
    - \( \langle n, k \rangle \notin \text{lessrrel} \cap \text{real} \times \text{real} \cup \{ \langle \text{complex}, \text{complex} \rangle \} \cup \text{real} \times \{ \text{complex} \} \)
    - \( \cup \)
    - \( \{ \text{complex} \} \times \text{real} \leftarrow \)
    - \( \langle n, \text{caddset} \langle k, \text{one} \rangle \rangle \in \text{lessrrel} \cap \text{real} \times \text{real} \cup \{ \langle \text{complex}, \text{complex} \rangle \} \cup \text{real} \times \{ \text{complex} \} \)
    - \( \cup \)
    - \( \{ \text{complex} \} \times \text{real} \) by (rule \( \text{MMIsar0.MMI_nnleltpt} \))
  - Then have \( n \in \mathbb{N} \land k \in \mathbb{N} \rightarrow n \leq k \iff n < k + 1 \)
  - By simp
  - With \( A1 \) show \( n \leq k \iff n < k + 1 \) by simp
  - From \( I \) have
    - \( n \in \bigcap \{ N \in \text{Pow}(\text{real}) . \text{one} \in N \quad (\forall n. n \in N \rightarrow \text{caddset} \langle n, \text{one} \rangle \in N) \} \land \)
    - \( k \in \bigcap \{ N \in \text{Pow}(\text{real}) . \text{one} \in N \quad (\forall n. n \in N \rightarrow \text{caddset} \langle n, \text{one} \rangle \in N) \} \rightarrow \)
    - \( \langle n, k \rangle \in \text{lessrrel} \cap \text{real} \times \text{real} \cup \{ \langle \text{complex}, \text{complex} \rangle \} \cup \text{real} \times \{ \text{complex} \} \cup \)
\{(complex)\} \times \text{real} \leftrightarrow \langle k, \text{caddset} \langle n, \text{one} \rangle \rangle \notin \text{lessrrel} \cap \text{real} \times \text{real} \cup \{\{(complex), \text{complex}\}\} \cup \text{real} \times \{\text{complex}\} \cup \{(\text{complex})\} \times \text{real} \text{ by (rule MMIsar0.MMI_nnltp1let)}
then have \( n \in \mathbb{N} \land k \in \mathbb{N} \rightarrow n < k \leftrightarrow n + 1 \leq k \)
by simp
with A1 show \( n < k \leftrightarrow n + 1 \leq k \) by simp
qed

Metamath has some very complicated and general version of induction on (complex) natural numbers that I can’t even understand. As an exercise I derived a more standard version that is imported to the complex0 context below.

lemma (in complex0) cplx_nat_ind: assumes A1: \( \psi(1) \) and
A2: \( \forall k \in \mathbb{N}. \ \psi(k) \rightarrow \psi(k+1) \) and
A3: \( n \in \mathbb{N} \) shows \( \psi(n) \)
proof -
let real = \( R \)
let complex = \( C \)
let one = 1
let zero = 0
let iunit = i
let caddset = CplxAdd(R,A)
let cmulset = CplxMul(R,A,M)
let lessrrel = StrictVersion(CplxROrder(R,A,r))
have I: MMIsar0
(\text{real, complex, one, zero, iunit, caddset, cmulset, lessrrel})
using MMIsar_valid by simp
moreover from A1 A2 A3 have
\( \psi(\text{one}) \)
\( \forall k \in \bigcap \{ \mathbb{N} \in \text{Pow}(\text{real}). \ \text{one} \in \mathbb{N} \land (\forall n. n \in \mathbb{N} \rightarrow \text{caddset} \langle n, \text{one} \rangle \in \mathbb{N})\}. \)
\( \psi(\text{k}) \rightarrow \psi(\text{caddset} \langle \text{k}, \text{one} \rangle) \)
\( n \in \bigcap \{ \mathbb{N} \in \text{Pow}(\text{real}). \ \text{one} \in \mathbb{N} \land (\forall n. n \in \mathbb{N} \rightarrow \text{caddset} \langle n, \text{one} \rangle \in \mathbb{N})\} \)
by auto
ultimately show \( \psi(n) \) by (rule MMIsar0.nnind1)
qed

Some simple arithmetics.

lemma (in complex0) arith: shows
\( 2 + 2 = 4 \)
\( 2 \cdot 2 = 4 \)
\( 3 \cdot 2 = 6 \)
\( 3 \cdot 3 = 9 \)
proof -
let real = \( R \)
let complex = \( C \)
let one = 1
let zero = 0
let iunit = i
let caddset = CplxAdd(R,A)
let cmulset = CplxMul(R,A,M)
let lessrrel = StrictVersion(CplxROrder(R,A,r))

have I: MMIsar0
  (real, complex, one, zero, iunit, caddset, cmulset, lessrrel)
  using MMIsar_valid by simp

then have
  caddset ⟨caddset ⟨one, one⟩, caddset ⟨one, one⟩⟩ =
  caddset ⟨caddset ⟨one, one⟩, one⟩
  by (rule MMIsar0.MMI_2p2e4)
thus 2 + 2 = 4 by simp

from I have
  cmulset⟨caddset⟨caddset⟨one, one⟩, one⟩, caddset⟨one, one⟩⟩ =
  caddset ⟨caddset⟨caddset⟨caddset⟨one, one⟩, one⟩, one⟩, one⟩
  by (rule MMIsar0.MMI_2t2e4)
thus 2·2 = 4 by simp

from I have
  cmulset⟨caddset⟨caddset⟨caddset⟨one, one⟩, one⟩, one⟩, caddset⟨one, one⟩⟩ =
  caddset ⟨caddset⟨caddset⟨caddset⟨caddset⟨caddset⟨one, one⟩, one⟩, one⟩, one⟩, one⟩, one⟩
  by (rule MMIsar0.MMI_3t2e6)
thus 3·2 = 6 by simp

from I have cmulset
  ⟨caddset⟨caddset⟨one, one⟩, one⟩, caddset⟨caddset⟨one, one⟩, one⟩⟩ =
  caddset ⟨caddset⟨caddset⟨caddset⟨caddset⟨caddset⟨caddset⟨one, one⟩, one⟩, one⟩, one⟩, one⟩, one⟩, one⟩
  by (rule MMIsar0.MMI_3t3e9)
thus 3·3 = 9 by simp

qed

99.3 Infimum and supremum in real numbers

Real numbers form a complete ordered field. Here we import a couple of Metamath theorems about supremum and infimum.

If a set $S$ has a smallest element, then the infimum of $S$ belongs to it.

lemma (in complex0) lbinfmcl: assumes A1: $S \subseteq R$ and
  A2: $\exists x \in S. \forall y \in S. x \leq y$
shows $\Infim(S,R,\leq) \in S$

proof -
  let real = R
  let complex = ℂ
  let one = 1
  let zero = 0
let iunit = i
let caddset = CplxAdd(R,A)
let cmulset = CplxMul(R,A,M)
let lessrrel = StrictVersion(CplxROrder(R,A,r))

have I: MMIsar0
  (real, complex, one, zero, iunit, caddset, cmulset, lessrrel)
using MMIsar_valid by simp

then have
  S ⊆ real ∧ (∃x∈S. ∀y∈S. ⟨y, x⟩ /∈ lessrrel ∩ real × real ∪ 
  real × {complex} ∪ {complex} × real) → Sup(S, real, 
  converse(lessrrel ∩ real × real ∪ 
  {{complex}, complex}) ∪ real × {complex} ∪ 
  {{complex}} × real) ∈ S
by (rule MMIsar0.MMI_suprcl)

then have
  S ⊆ R ∧ (∃x∈S. ∀y∈S. x ≤ y) → Sup(S,R,converse(<)) ∈ S by simp
with A1 A2 show thesis using Infim_def by simp

qed

Supremum of any subset of reals that is bounded above is real.

lemma (in complex0) sup_is_real:
  assumes A ⊆ R and A ≠ 0 and ∃x∈R. ∀y∈A. y ≤ x
shows Sup(A, R,<) ∈ R

proof -
  let real = R
  let complex = C
  let one = 1
  let zero = 0
  let iunit = i
  let caddset = CplxAdd(R,A)
  let cmulset = CplxMul(R,A,M)
  let lessrrel = StrictVersion(CplxROrder(R,A,r))
  have MMIsar0
    (real, complex, one, zero, iunit, caddset, cmulset, lessrrel)
using MMIsar_valid by simp

then have
  A ⊆ real ∧ A ≠ 0 ∧ (∃x∈real. ∀y∈A. ⟨x, y⟩ /∈ 
  lessrrel ∩ real × real ∪ {{complex}, complex} ∪ 
  real × {complex} ∪ {complex} × real) → Sup(A, real, 
  lessrrel ∩ real × real ∪ {{complex}, complex} ∪ 
  real × {complex} ∪ {complex} × real) ∈ real
by (rule MMIsar0.MMI_suprcl)

with asms show thesis by simp

qed

If a real number is smaller that the supremum of A, then we can find an
element of $A$ greater than it.

**Lemma** (in `complex0`) suprlub:
- assumes $A \subseteq R$ and $A \neq 0$ and $\exists x \in R. \forall y \in A. y \leq x$
- and $B \in R$ and $B < \operatorname{Sup}(A, R, <)$
- shows $\exists z \in A. B < z$

**Proof** -
- let $real = R$
- let $complex = C$
- let $one = 1$
- let $zero = 0$
- let $iunit = i$
- let $caddset = \text{CplxAdd}(R, A)$
- let $cmulset = \text{CplxMul}(R, A, M)$
- let $lessrrel = \text{StrictVersion}(\text{CplxROrder}(R, A, r))$
- have $\text{MMIsar0}(\text{real, complex, one, zero, iunit, caddset, cmulset, lessrrel})$
  using $\text{MMIsar_valid}$ by simp
- then have $(A \subseteq real \wedge A \neq 0 \wedge (\exists x \in real. \forall y \in A. (x, y) \notin lessrrel \cap real \times real \cup \{(\text{complex}, \text{complex})\} \cup real \times \{\text{complex}\} \cup \{\text{complex}\} \times \{\text{complex}\} \cup \{\text{complex}\} \times \text{lessrrel} \cap real \times real \cup \{\text{complex}, \text{complex}\} \cup real \times \{\text{complex}\} \cup \{\text{complex}\} \times real \rightarrow (\exists z \in A. (B, z) \in lessrrel \cap real \times real \cup \{(\text{complex}, \text{complex})\} \cup real \times \{\text{complex}\} \cup \{\text{complex}\} \times real) \in lessrrel \cap real \times real \cup \{\text{complex}, \text{complex}\} \cup real \times \{\text{complex}\}$
  by (rule $\text{MMIsar0.MMI\_suprlub}$)
- with assms show thesis by simp
- qed

Something a bit more interesting: infimum of a set that is bounded below is real and equal to the minus supremum of the set flipped around zero.

**Lemma** (in `complex0`) infsup:
- assumes $A \subseteq R$ and $A \neq 0$ and $\exists x \in R. \forall y \in A. x \leq y$
- shows $\operatorname{Infim}(A, R, <) \in R$
- $\operatorname{Infim}(A, R, <) = (\neg \operatorname{Sup}(\{z \in R. (\neg z) \in A\}, R, <))$

**Proof** -
- let $real = R$
- let $complex = C$
- let $one = 1$
- let $zero = 0$
- let $iunit = i$
- let $caddset = \text{CplxAdd}(R, A)$
- let $cmulset = \text{CplxMul}(R, A, M)$
- let $lessrrel = \text{StrictVersion}(\text{CplxROrder}(R, A, r))$
have I: MMIsar0
(real, complex, one, zero, iunit, caddset, cmulset, lessrrel)
using MMIsar_valid by simp
then have
A ⊆ real ∧ A ≠ 0 ∧ (∃x∈real. ∀y∈A. ⟨y, x⟩ /∈
lessrrel ∩ real × real ∪ {{complex}, complex} ∪
real × {complex} ∪
{{complex}} × real) → Sup(A, real, converse
(lessrrel ∩ real × real ∪ {{complex}, complex}) ∪
real × {complex} ∪
{{complex}} × real)
∈ real
by (rule MMIsar0.MMI_infmsup)
then have A ⊆ R ∧ ¬(A = 0) ∧ (∃x∈R. ∀y∈A. x ≤ y) →
Sup(A,R,converse(<)) = (-Sup({z∈R. (-z) ∈ A },R,<))
by simp
with assms show
Infim(A,R,<) = (-Sup({z∈R. (-z) ∈ A },R,<))
using Infim_def by simp
from I have
A ⊆ real ∧ A ≠ 0 ∧ (∃x∈real. ∀y∈A. ⟨y, x⟩ /∈
lessrrel ∩ real × real ∪ {{complex}, complex} ∪
real × {complex} ∪
{{complex}} × real) → Sup(A, real, converse
(lessrrel ∩ real × real ∪ {{complex}, complex}) ∪
real × {complex} ∪ {{complex}} × real) ∈ real
by (rule MMIsar0.MMI_infmrc1)
with assms show Infim(A,R,<) ∈ R
using Infim_def by simp
qed
end

References

1570