Abstract

This is the proof document of the IsarMathLib project version 1.22.1. IsarMathLib is a library of formalized mathematics for Isabelle2021-1 (ZF logic).

Contents

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

2 First Order Logic
   2.1 Notions and lemmas in FOL

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

4 Natural numbers in IsarMathLib
   4.1 Induction
   4.2 Intervals

5 Order relations - introduction
   5.1 Definitions
   5.2 Intervals
   5.3 Bounded sets

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

7 Even more on order relations
   7.1 Maximum and minimum of a set
   7.2 Supremum and Infimum
   7.3 Strict versions of order relations
17.3 Saturated sets .............................................. 173

18 Finite sequences ........................................ 175
  18.1 Lists as finite sequences ............................. 175
  18.2 Lists and cartesian products ......................... 189

19 Inductive sequences .................................... 192
  19.1 Sequences defined by induction ..................... 192
  19.2 Images of inductive sequences ..................... 200
  19.3 Subsets generated by a binary operation .......... 200
  19.4 Inductive sequences with changing generating function 203

20 Enumerations ............................................. 207
  20.1 Enumerations: definition and notation .......... 207
  20.2 Properties of enumerations ....................... 208

21 Folding in ZF ............................................ 210
  21.1 Folding in ZF ........................................ 211

22 Partitions of sets ....................................... 215
  22.1 Bisections ........................................... 215
  22.2 Partitions ........................................... 218

23 Quasigroups .............................................. 219
  23.1 Definitions and notation ............................ 219

24 Loops ..................................................... 222
  24.1 Definitions and notation ............................ 222

25 Ordered loops ........................................... 225
  25.1 Definition and notation .............................. 225

26 Semigroups ............................................... 230
  26.1 Products of sequences of semigroup elements ... 230
  26.2 Products over sets of indices ..................... 234
  26.3 Commutative semigroups ............................. 237

27 Commutative Semigroups ................................. 249
  27.1 Sum of a function over a set ....................... 249

28 Monoids ................................................... 253
  28.1 Definition and basic properties .................... 253
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 \texttt{≡} symbol is the notion being defined, the right side says
how we define it. In Isabelle/ZF \texttt{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.

\begin{verbatim}
definition AreDisjoint (infix \texttt{is disjoint with} 90) where
A \texttt{is disjoint with} B \equiv A \cap B = 0
\end{verbatim}

We are ready to prove a theorem. Here we show that the relation of be-
ing 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 mean-
ingful names. After the "shows" keyword we give the statement to show.
The \texttt{←→} 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 sug-
gest 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. Impli-
cations are denoted with the \texttt{−→} 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 {is disjoint with} B ⟷ B {is disjoint with} A
proof -
  have A {is disjoint with} B ⟷ B {is disjoint with} A
  proof -
    { assume A {is disjoint with} B
      then have A ∩ B = 0 using AreDisjoint_def by simp
      hence B ∩ A = 0 by auto
      then have B {is disjoint with} A
        using AreDisjoint_def by simp
    } thus thesis by simp
  qed
moreover have B {is disjoint with} A ⟷ A {is disjoint with} B
proof -
  { assume B {is disjoint with} A
    then have B ∩ A = 0 using AreDisjoint_def by simp
    hence A ∩ B = 0 by auto
    then have A {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 \( a_0 \cdot a_1 \cdot \ldots \cdot a_n \), (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 Real_ZF_1.
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 hereditary 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 using 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: \( \forall x y z. \langle x, y \rangle \in r \land \langle y, z \rangle \in r \longrightarrow \langle x, z \rangle \in r \) shows trans(r)
proof -
  from A1 have
    \( \forall x y z. \langle x, y \rangle \in r \longrightarrow \langle y, z \rangle \in r \longrightarrow \langle x, z \rangle \in r \)
    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: \( \langle a, b \rangle \in r \land \langle b, c \rangle \in r \) shows \( \langle a, c \rangle \in r \)
proof -
  from A1 have \( \forall x y z. \langle x, y \rangle \in r \longrightarrow \langle y, z \rangle \in r \longrightarrow \langle x, z \rangle \in r \)
    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⟩ ∈ r ⟨b,a⟩ ∈ r
shows a=b
proof -
  from A1 have ∀ x y. ⟨x,y⟩ ∈ r ⟹ ⟨y,x⟩ ∈ r ⟹ 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) ≡
(p ∨ q ∨ r) ∧ (p ⟹ ¬q ∧ ¬r) ∧ (q ⟹ ¬p ∧ ¬r) ∧ (r ⟹ ¬p ∧ ¬q)

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

lemma Fol1_L5:
assumes p ∨ q ∨ r
and p ⟹ ¬q ∧ ¬r
and q ⟹ ¬p ∧ ¬r
and r ⟹ ¬p ∧ ¬q
shows Exactly_1_of_3_holds(p,q,r)
proof -
  from assms have (p ∨ q ∨ r) ∧ (p ⟹ ¬q ∧ ¬r) ∧ (q ⟹ ¬p ∧ ¬r) ∧ (r ⟹ ¬p ∧ ¬q)
    unfolding Exactly_1_of_3_holds_def by fast
  hence p ∨ q ∨ r by blast
  with A1 show q ∨ r by simp
qed

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

lemma Fol1_L6:
assumes A1: ¬p and A2: Exactly_1_of_3_holds(p,q,r)
shows q ∨ r
proof -
  from A2 have (p ∨ q ∨ r) ∧ (p ⟹ ¬q ∧ ¬r) ∧ (q ⟹ ¬p ∧ ¬r) ∧ (r ⟹ ¬p ∧ ¬q)
    unfolding Exactly_1_of_3_holds_def by fast
  hence p ∨ q ∨ r by blast
  with A1 show q ∨ 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 ¬r
proof -
  from A2 have
  
  (p ∨ q ∨ r) ∧ (p → ¬q ∧ ¬r) ∧ (q → ¬p ∧ ¬r) ∧ (r → ¬p ∧ ¬q)

  unfolding Exactly_1_of_3_holds_def by fast
with A1 show ¬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) ←→ (p ←→ q ←→ r) ∧ ¬(p ∧ q ∧ r)
proof
  assume Exactly_1_of_3_holds(p,q,r)
  then have
  
  (p ∨ q ∨ r) ∧ (p → ¬q ∧ ¬r) ∧ (q → ¬p ∧ ¬r) ∧ (r → ¬p ∧ ¬q)

  unfolding Exactly_1_of_3_holds_def by fast
thus (p ←→ q ←→ r) ∧ ¬(p ∧ q ∧ r) by blast
next assume (p ←→ q ←→ r) ∧ ¬(p ∧ q ∧ r)
hence
  
  (p ∨ q ∨ r) ∧ (p → ¬q ∧ ¬r) ∧ (q → ¬p ∧ ¬r) ∧ (r → ¬p ∧ ¬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 ←→ ¬(q ∨ r)
proof -
  from A1 have (p ∨ q ∨ r) ∧ (p → ¬q ∧ ¬r) ∧ (q → ¬p ∧ ¬r) ∧ (r → ¬p ∧ ¬q)

  unfolding Exactly_1_of_3_holds_def by fast
then show p ←→ ¬(q ∨ r) by blast
qed

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

definition Xor (infixl Xor 66) where
  p Xor q ≡ (p ∨ q) ∧ ¬(p ∧ 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
Equivalence relations are symmetric.

**Lemma equiv_is_sym**

**Assumptions**: \( A1: \text{equiv}(X, r) \) and \( A2: \langle x, y \rangle \in r \)

**Shows**: \( \langle y, x \rangle \in r \)

**Proof**
- From \( A1 \) have \( \text{sym}(r) \) using \( \text{equiv_def} \) by simp
- Then have \( \forall x \ y. \langle x, y \rangle \in r \implies \langle y, x \rangle \in r \)
  - Unfolding \( \text{sym_def} \) by fast
- With \( A2 \) show \( \langle y, x \rangle \in r \) 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.

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

**Lemma collection_contain**

**Assumptions**: \( A \subseteq B \) shows \( \bigcup A \subseteq \bigcup B \)

**Proof**
- Fix \( x \) assume \( x \in \bigcup A \)
- Then obtain \( X \) where \( x \in X \) and \( X \in A \) by auto
- With \( \text{assms} \) 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**

**Assumptions**: \( C \neq \emptyset \) and \( \forall y \in C. \ b(y) = A \)

**Shows**: \( \bigcup \{ b(y) \mid y \in C \} = A \) using \( \text{assms} \) by blast

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

**Lemma ZF1_1_L2**

**Assumptions**: \( A1: C \neq \emptyset \) and \( A2: \forall x \in C. \ b(x) \in A \)

**And**: \( A3: \forall x \ y. \ x \in C \land y \in C \implies b(x) = b(y) \)

**Shows**: \( \bigcup \{ b(x) \mid x \in C \} \in A \)

**Proof**
- From \( A1 \) obtain \( x \) where \( D1: x \in C \) by auto
If two meta-functions are the same on a cartesian product, then the subsets defined by them are the same. I am surprised Isabelle can not handle this automatically.

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\}
with A1 show z \in \{b(x, y). \langle x,y \rangle \in X \times Y\} by auto
qed

show \( \{b(x, y). \langle x,y \rangle \in X \times Y\} \subseteq \{a(x, y). \langle x,y \rangle \in X \times Y\} \)
proof
fix z assume z \in \{b(x, y). \langle x,y \rangle \in X \times Y\}
with A1 show z \in \{a(x, y). \langle x,y \rangle \in X \times Y\} by auto
qed

qed

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

lemma prod_subset: assumes U\times V \neq \emptyset U \times V \subseteq X \times Y shows U \subseteq X and V \subseteq Y
using assms by auto
A technical lemma about sections in cartesian products.

**Lemma section_proj:**

Assumes \( A \subseteq X \times Y \) and \( U \times V \subseteq A \) and \( x \in U \) \( y \in V \)

Shows \( U \subseteq \{ t \in X. \langle t, y \rangle \in A \} \) and \( V \subseteq \{ t \in Y. \langle x, t \rangle \in A \} \)

Using assms by auto

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

**Lemma ZF1_1_L4B:**

Assumes \( \forall x \in X. a(x) = b(x) \)

Shows \( \{ a(x). x \in X \} = \{ b(x). x \in X \} \)

Using assms by simp

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

**Lemma ZF1_1_L5:**

Assumes \( X \neq 0 \) and \( \forall x \in X. b(x) = c \)

Shows \( \{ b(x). x \in X \} = \{ c \} \) using assms by blast

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

**Lemma subset_with_property:**

Assumes \( Y = \{ x \in X. b(x) \} \)

Shows \( Y \subseteq X \)

Using assms by auto

We can choose an element from a nonempty set.

**Lemma nonempty_has_element:**

Assumes \( X \neq 0 \) shows \( \exists x. x \in X \)

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.

**Lemma inter_empty_empty:**

Shows \( \bigcap 0 = 0 \) 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_empty_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**

\( \text{ProductCollection}(T, S) \equiv \bigcup_{U \in T. \{U \times V. V \in S}\} \)

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

**Lemma ZF1_1_L6:**

Shows \( \bigcup \text{ProductCollection}(S, T) = \bigcup S \times \bigcup T \)

21
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 $<(\bigcup A) = P(x)>$ 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**

RestrictedTo (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 $\bigcup (\bigcup \{U \in C. \ \{A \in B. \ U = \bigcup A\}\} \subseteq \bigcup C$ by blast
  show $\bigcup C \subseteq \bigcup (\bigcup \{U \in C. \ \{A \in B. \ U = \bigcup A\}\}$
proof
  fix $x$ assume $x \in \bigcup C$
  show $x \in \bigcup (\bigcup \{U \in C. \ \{A \in B. \ U = \bigcup A\}\}$
proof -
    from $<x \in \bigcup C>$ 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 \( \langle \forall u \in C \land x \in \text{U} \rangle < \forall a \in B \land u = \bigcup A \rangle \)  show \( x \in \bigcup (\bigcup C \cup \{ A \in B \} \) by auto
  qed
  qed
  qed

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

lemma \texttt{consdef}: shows \( \text{cons}(a,A) = A \cup \{a\} \)
  using \texttt{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 \texttt{singl_diff_empty}: assumes \( A - \{x\} = 0 \)
  shows \( A = 0 \lor A = \{x\} \)
  using \texttt{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 \texttt{singl_diff_eq}: assumes \( A1: A - \{x\} = A \)
  shows \( x \not\in A \)
proof -
  have \( x \not\in A - \{x\} \) by auto
  with \( A1 \) show \( x \not\in A \) by simp
  qed

A basic property of sets defined by comprehension.

lemma \texttt{comprehension}: assumes \( a \in \{x \in X. p(x)\} \)
  shows \( a \in X \) and \( p(a) \) using \texttt{assms} by auto

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

lemma \texttt{image_rel_mono}: assumes \( r \subseteq s \)
  shows \( r(A) \subseteq s(A) \)
  using \texttt{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 \texttt{image_greater_rel}:
  assumes \( x \in U\{x\} \) and \( U\{x\} \subseteq C \)
  shows \( (U \cup C \times C)\{x\} = C \)
  using \texttt{assms} \texttt{image_Un_left} by blast

Reformulation of the definition of composition of two relations:

lemma \texttt{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 \texttt{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 $(y,z) = p$ by auto
  with $\langle p \in r \circ r \rangle$ obtain $x$ where $(y,x) \in r$ and $(x,z) \in r$
    using rel_compdef by auto
  from $\langle (y,x) \in r \rangle$ have $(x,y) \in \text{converse}(r)$ by simp
  with assms $\langle (x,z) \in r \rangle$ $\langle (y,z) = p \rangle$ 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 -
      { fix $p$ assume $p \in (r{x}) \times (r{x})$
        then obtain $y z$ where $(y,z) = p$ by auto
          from $\langle y \in r{x} \rangle$ have $(x,y) \in r$ by auto
          then have $(y,x) \in \text{converse}(r)$ by simp
            with assms $\langle z \in r{x} \rangle$ $\langle (y,z) = p \rangle$ have $p \in r \circ r$ by auto
        } thus thesis by auto
      qed
    } thus $(\bigcup\{(r{x}) \times (r{x}). x \in \text{domain}(r)\}) \subseteq r \circ r$
      by blast
  qed

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

**Lemma refl_union_singl_image**: assumes $A \subseteq X \times X$ and $\text{id}(X) \subseteq A$ shows $A \subseteq \bigcup\{A{x} \times A{x}. x \in X\}$

proof -
{ fix $p$ assume $p \in A$
    with assms(1) obtain $x y$ where $x \in X$ $y \in X$ and $p = \langle x,y \rangle$ by auto
    with assms(2) $\langle p \in A \rangle$ have $\exists x \in X. p \in A{x} \times A{x}$ by auto
  } thus thesis by auto
    qed

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

**Lemma set_mem_eq**: assumes $x \in A$ $A = B$ 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
Supersets$(X, \mathcal{A}) \equiv \{B \in \mathcal{P}(X) . \exists A \in \mathcal{A}. A \subseteq B\}$

The family itself is in its supersets.

lemma superset_gen: assumes $A \subseteq X$ $A \in \mathcal{A}$ shows $A \in \text{Supersets}(X, \mathcal{A})$
using assms Supersets_def by auto

end

4 Natural numbers in IsarMathLib

theory Nat_ZF_IML imports ZF.Arith

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 $n = \{0, 1, 2, \ldots, n - 1\}$. 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. Anyway, 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$\in$nat and A2: P(0) and A3: $\forall\, k \in \text{nat}.\, P(k) \Longrightarrow P(\text{succ}(k))$
shows P(n)
proof -
  note A1 A2
  moreover
  \{ fix x
    assume x$\in$nat P(x)
    with A3 have P(succ(x)) by simp \}
  ultimately show P(n) by (rule nat_induct)

26
A nonzero natural number has a predecessor.

**lemma** Nat_ZF_1_L3: assumes $A1: n \in \mathbb{N}$ and $A2: n \neq 0$

shows $\exists k \in \mathbb{N}. n = \text{succ}(k)$

**proof** -
from $A1$ have $n \in \{0\} \cup \{\text{succ}(k). k \in \mathbb{N}\}$
using nat_unfold by simp
with $A2$ show thesis by simp
qed

What is succ, anyway?

**lemma** succ_explained: shows $\text{succ}(n) = n \cup \{n\}$
using succ_iff by auto

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

**lemma** empty_in_every_succ: assumes $A1: n \in \mathbb{N}$
shows $0 \in \text{succ}(n)$

**proof** -
note $A1$
moreover have $0 \in \text{succ}(0)$ by simp
moreover
{ fix $k$ assume $k \in \mathbb{N}$ 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 $\forall k \in \mathbb{N}. 0 \in \text{succ}(k) \implies 0 \in \text{succ}(\text{succ}(k))$
by simp
ultimately show $0 \in \text{succ}(n)$ by (rule ind_on_nat)
qed

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

**lemma** succ_ineq: assumes $A1: n \in \mathbb{N}$
shows $\forall i \in n. \text{succ}(i) \in \text{succ}(n)$

**proof** -
note $A1$
moreover have $\forall k \in 0. \text{succ}(k) \in \text{succ}(0)$ by simp
moreover
{ fix $k$ assume $A2: \forall i \in k. \text{succ}(i) \in \text{succ}(k)$
{ fix $i$ assume $i \in \text{succ}(k)$
then have $i \in k \lor i = k$ by auto
moreover
{ assume $i \in k$
with $A2$ have $\text{succ}(i) \in \text{succ}(k)$ by simp
hence $\text{succ}(i) \in \text{succ}(\text{succ}(k))$ by auto }
moreover
{ assume $i = k$
then have $\text{succ}(i) \in \text{succ}(\text{succ}(k))$ by auto }

27
ultimately have $\text{succ}(i) \in \text{succ}(\text{succ}(k))$ by auto
} then have $\forall i \in \text{succ}(k). \text{succ}(i) \in \text{succ}(\text{succ}(k))$
by simp
} then have $\forall k \in \text{nat}.
( (\forall i \in k. \text{succ}(i) \in \text{succ}(k)) \rightarrow (\forall i \in \text{succ}(k). \text{succ}(i) \in \text{succ}(\text{succ}(k)))$
) by simp
ultimately show $\forall i \in \text{n}. \text{succ}(i) \in \text{succ}(\text{n})$ by (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: $k \subseteq n$
shows $\text{succ}(k) \subseteq \text{succ}(n)$
proof -
from A1 have T: $\text{Ord}(k)$ and $\text{Ord}(n)$
  using nat_into_Ord by auto
with A2 have $\text{succ}(k) \subseteq \text{succ}(n)$
  using subset_imp_le by simp
then show $\text{succ}(k) \subseteq \text{succ}(n)$ using le_imp_subset
by simp
qed

For any two natural numbers one of them is contained in the other.

lemma nat_incl_total: assumes A1: $i \in \text{nat}$  $j \in \text{nat}$
shows $i \subseteq j \lor j \subseteq i$
proof -
from A1 have T: $\text{Ord}(i)$  $\text{Ord}(j)$
  using nat_into_Ord by auto
then have $i \in j \lor i=j \lor j \in i$ using Ord_linear
by simp
moreover
{ assume $i \in j$
  with T have $i \leq j \lor j \leq i$
    using lt_def leI le_imp_subset by simp }
moreover
{ assume $i=j$
  then have $i \leq j \lor j \leq i$ by simp }
moreover
{ assume $j \in i$
  with T have $i \leq j \lor j \leq i$
    using lt_def leI le_imp_subset by simp }
ultimately show $i \leq j \lor j \leq i$ by auto
qed

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

lemma nat_union_succ: shows $\text{nat} = (\bigcup n \in \text{nat}. \text{succ}(n))$
proof
  show $\text{nat} \subseteq (\bigcup n \in \text{nat}. \text{succ}(n))$ by auto
next
{ fix k assume A2: k ∈ (⋃n ∈ nat. succ(n))
then obtain n where T: n ∈ nat and I: k ∈ succ(n)
by auto
then have k ≤ n using nat_into_Ord lt_def
by simp
with T have k ∈ nat using le_in_nat by simp
} then show (⋃n ∈ nat. succ(n)) ⊆ nat by auto
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 have succ(n) ⊆ (⋃n ∈ nat. succ(n)) by auto
then show succ(n) ⊆ nat using nat_union_succ by simp
qed

Element of a natural number is a natural number.

lemma elem_nat_is_nat: assumes A1: n ∈ nat and A2: k∈n
shows k < n k ∈ nat k ≤ n ⟨k,n⟩ ∈ Le
proof -
from A1 A2 show k < n using nat_into_Ord lt_def by simp
with A1 show k ∈ nat using lt_nat_in_nat by simp
from <k < n> show k ≤ n using leI by simp
with A1 <k ∈ nat> show ⟨k,n⟩ ∈ Le using Le_def
by simp
qed

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

lemma nat_union_nat: shows nat = ⋃ 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 ∈ nat shows n ⊆ nat
proof -
from A1 have n ⊆ ⋃ nat by auto
then show n ⊆ 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 ∈ nat and A2: k ∈ nat
shows
n ≤ n #+ k
n ⊆ n #+ k
n ⊆ k #+ n
proof -
from A1 A2 have n ≤ n 0 ≤ k n ∈ nat k ∈ nat
using nat_le_refl nat_0_le by auto
then have n #+ 0 ≤ n #+ k by (rule add_le_mono)
with A1 show \( n \leq n + k \) using add_0_right by simp
then show \( n \subseteq n + k \) using le_imp_subset by simp
then show \( n \subseteq k + n \) using add_commute by simp
qed

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

**Lemma: add_lt_mono**

**Assumes** \( k \in \text{nat} \) and \( j \in \text{nat} \)
**Shows**
\( (n + j) < (n + k) \)
\( (n + j) \in (n + k) \)

**Proof**

- From asssms have \( j < k \) using elem_nat_is_nat by blast
- Moreover note \( <k \in \text{nat}> \)
- Ultimately show \( (n + j) < (n + k) \) \( (n + j) \in (n + k) \)
  using add_lt_mono2 ltD by 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** \( A1: n \in \text{nat} \) and \( A2: m \in \text{nat} \)
**Shows** \( \forall i \in m + n. \ i \in m \lor (\exists j \in n. \ i = m + j) \)

**Proof**

- Note \( A1 \)
- Moreover from \( A2 \) have \( \forall i \in m + 0. \ i \in m \lor (\exists j \in 0. \ i = m + j) \)
  using add_0_right by simp
- Moreover have \( \forall k \in \text{nat}. \)
  \( (\forall i \in m + k. \ i \in m \lor (\exists j \in k. \ i = m + j)) \rightarrow \)
  \( (\forall i \in m + \text{succ}(k). \ i \in m \lor (\exists j \in \text{succ}(k). \ i = m + j)) \)

**Proof**

- \{ fix \( k \) assume \( A3: k \in \text{nat} \)
  \{ assume \( A4: \forall i \in m + k. \ i \in m \lor (\exists j \in k. \ i = m + j) \)
  \{ fix \( i \) assume \( i \in m + \text{succ}(k) \)
    then have \( i \in m + k \lor i = m + k \) using add_succ_right
    by auto
    moreover from \( A4 \) \( A3 \) have
    \( i \in m + k \rightarrow i \in m \lor (\exists j \in \text{succ}(k). \ i = m + j) \)
    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
ultimately show $\forall i \in m \#+ n. \ i \in m \lor (\exists j \in n. \ i = m \#+ j)$

by (rule ind_on_nat)

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 $k \in n \lor k=n$ by auto.
- with $A1$ have $k \in \text{nat}$ using $\text{elem_nat_is_nat}$ 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 $\text{nat_into_Ord \ Ord_succ_mem_iff}$ 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 $\text{assms}$ by auto.

### 4.2 Intervals

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

The interval is determined by two parameters: starting point and length. Recall that in standard Isabelle’s `Arith.thy` the symbol `#` is defined as the sum of natural numbers.

**definition**

$\text{NatInterval}(n,k) \equiv \{n \#+ j. \ j\in k\}$

Subtracting the beginning of 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 NatInterval_def by auto
from A1 II have j ∈ nat using elem_nat_is_nat by blast
moreover from I have i ≠ n = natify(j) using 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 length_start_decomp: assumes A1: n ∈ nat k ∈ nat
shows
n ∩ NatInterval(n,k) = 0
n ∪ NatInterval(n,k) = n #+ k
proof -
{ fix i assume A2: i ∈ n and i ∈ NatInterval(n,k)
  then obtain j where I: i = n #+ j and II: j ∈ k
  using NatInterval_def by auto
  from A1 have k ∈ nat using elem_nat_is_nat by blast
  with II have j ∈ nat using elem_nat_is_nat by blast
  with A1 I have n ≤ i using add_nat_le by simp
  moreover from A1 A2 have i < n using elem_nat_is_nat by blast
  ultimately have False using le_imp_not_lt by blast
} thus n ∩ NatInterval(n,k) = 0 by auto
from A1 have n ⊆ n #+ k using add_nat_le by simp
moreover
{ fix i assume i ∈ NatInterval(n,k)
  then obtain j where III: i = n #+ j and IV: j ∈ k
  using NatInterval_def by auto
  with A1 have j < k using elem_nat_is_nat by blast
  with A1 III have i ∈ n #+ k using add_lt_mono2 ltD
  by simp }
ultimately have n ∪ NatInterval(n,k) ⊆ n #+ k by auto
moreover from A1 have n #+ k ⊆ n ∪ NatInterval(n,k)
  using nat_sum_decomp NatInterval_def by auto
ultimately show n ∪ NatInterval(n,k) = n #+ k by auto
qed

Sme properties of three adjacent intervals.

lemma adjacent_intervals3: assumes n ∈ nat k ∈ nat m ∈ nat
shows
n #+ k #+ m = (n #+ k) ∪ NatInterval(n #+ k,m)
n #+ k #+ m = n ∪ NatInterval(n,k #+ m)
n #+ k #+ m = n ∪ NatInterval(n,k) ∪ NatInterval(n #+ k,m)
using assms add_assoc length_start_decomp by auto
end
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 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**

$\text{IsTotal } (\text{infixl } \{\text{is total on}\} 65) \text{ where}$

$r \{\text{is total on}\} X \equiv (\forall a \in X. \forall b \in X. (a, b) \in r \lor (b, a) \in r)$

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

**definition**

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

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**

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

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

**definition**

$\text{IsBoundedAbove}(A, r) \equiv (A = 0 \lor (\exists u. \forall x \in A. (x, u) \in r))$

We define sets bounded below analogously.

**definition**

$\text{IsBoundedBelow}(A, r) \equiv (A = 0 \lor (\exists l. \forall x \in A. (l, x) \in r))$
A set is bounded if it is bounded below and above.

**definition**

\[ \text{IsBounded}(A,r) \equiv (\text{IsBoundedAbove}(A,r) \land \text{IsBoundedBelow}(A,r)) \]

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**

\[ \text{Interval}(r,a,b) \equiv r\{a\} \cap r^{-}\{b\} \]

We also define the maximum (the greater of) two elements in the obvious way.

**definition**

\[ \text{GreaterOf}(r,a,b) \equiv (\text{if } \langle a,b \rangle \in r \text{ then } b \text{ else } a) \]

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

**definition**

\[ \text{SmallerOf}(r,a,b) \equiv (\text{if } \langle a,b \rangle \in r \text{ then } a \text{ else } b) \]

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

**definition**

\[ \text{HasAmaximum}(r,A) \equiv \exists M \in A. \forall x \in A. \langle x,M \rangle \in r \]

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

**definition**

\[ \text{HasAminimum}(r,A) \equiv \exists m \in A. \forall x \in A. \langle m,x \rangle \in r \]

Definition of the maximum of a set.

**definition**

\[ \text{Maximum}(r,A) \equiv \text{THE } M \in A \land (\forall x \in A. \langle x,M \rangle \in r) \]

Definition of a minimum of a set.

**definition**

\[ \text{Minimum}(r,A) \equiv \text{THE } m \in A \land (\forall x \in A. \langle m,x \rangle \in r) \]

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

**definition**

\[ \text{Supremum}(r,A) \equiv \text{Minimum}(r,\bigcap_{a \in A} r\{a\}) \]

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 \( A \subseteq X \) is \( \bigcap_{x \in A} \{ y \in X \mid \langle x, y \rangle \in r \} \), which is the same as \( \bigcap_{x \in A} r(\{ x \}) \), where \( r(\{ x \}) \) is the image of the singleton \( \{ x \} \) under relation \( r \).

deinition
\[
\text{HasASupremum}(r, A) \equiv \text{HasAMinimum}(r, \bigcap_{a \in A} r\{a\})
\]
The notion of ”having an infimum” is the same as the set of lower bounds having a maximum.

deinition
\[
\text{HasAnInfimum}(r, A) \equiv \text{HasAMaximum}(r, \bigcap_{a \in A} r-\{a\})
\]
Infimum is defined analogously.

deinition
\[
\text{Infimum}(r, A) \equiv \text{Maximum}(r, \bigcap_{a \in A} r-\{a\})
\]
We define a relation to be complete if every nonempty bounded above set has a supremum.

deinition
\[
\text{IsComplete} \quad \text{(is complete)}
\]
where
\[
\forall A. \text{IsBoundedAbove}(A, r) \land A \neq 0 \rightarrow \text{HasAMinimum}(r, \bigcap_{a \in A} r\{a\})
\]
The essential condition to show that a total relation is reflexive.

lemma \text{OrderZF1.L1}: assumes \( \text{IsTotal on } X \) and \( a \in X \)
shows \( \langle a, a \rangle \in r \)
using \text{assms \text{IsTotal_def by auto}}
A total relation is reflexive.

lemma \text{total_is_refl}:
assumes \( \text{IsTotal on } X \)
shows \( \text{refl}(X, r) \)
using \text{assms \text{OrderZF1.L1 refl_def by simp}}
A linear order is partial order.

lemma \text{OrderZF1.L2}: assumes \( \text{IsLinOrder}(X, r) \)
shows \( \text{IsPartOrder}(X, r) \)
using \text{assms \text{IsLinOrder_def IsPartOrder_def refl_def OrderZF1.L1 by auto}}
Partial order that is total is linear.

lemma \text{OrderZF1.L3}:
assumes \( \text{IsPartOrder}(X, r) \) and \( \text{IsTotal on } X \)
shows \( \text{IsLinOrder}(X, r) \)
using \text{assms \text{IsPartOrder_def IsLinOrder_def by simp}}
Relation that is total on a set is total on any subset.
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:**
asumes r {is total on} X and A ⊆ X and a ∈ X
shows A = {x ∈ A. ⟨x,a⟩ ∈ r} ∪ {x ∈ A. ⟨a,x⟩ ∈ r}
using assms IsTotal_def by auto

A technical fact about reflexive relations.

**lemma refl_add_point:**
asumes 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 \( x \in \text{Interval}(r,a,b) \iff \langle a,x \rangle \in r \land \langle x,b \rangle \in r \)
using \text{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 \( x \in \text{Interval}(r,a,b) \)
shows \( \langle a,x \rangle \in r \land \langle x,b \rangle \in r \)
using assms Order_ZF_2_L1 by auto

Order_ZF_2_L1, implication from right to left.

**lemma** Order_ZF_2_L1B: assumes \( \langle a,x \rangle \in r \land \langle x,b \rangle \in r \)
shows \( x \in \text{Interval}(r,a,b) \)
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 \( \text{refl}(X,r) \)
and \( a \in X \land b \in X \land \langle a,b \rangle \in r \)
shows \( a \in \text{Interval}(r,a,b) \land b \in \text{Interval}(r,a,b) \)
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 \( \text{refl}(X,r) \)
and \( a \in X \land b \in X \land \langle a,b \rangle \in r \)
shows \( \text{Interval}(r,a,b) \neq \emptyset \)
proof -
  from assms have \( a \in \text{Interval}(r,a,b) \)
  using Order_ZF_2_L2 by simp
  then show \( \text{Interval}(r,a,b) \neq \emptyset \) by auto
qed

If \( a,b,c,d \) are in this order, then \([b,c] \subseteq [a,d]\). We only need trascitivity for this to be true.

**lemma** Order_ZF_2_L3:
assumes \( \text{trans}(r) \) and \( A2: \langle a,b \rangle \in r \land \langle b,c \rangle \in r \land \langle c,d \rangle \in r \)
shows \( \text{Interval}(r,b,c) \subseteq \text{Interval}(r,a,d) \)
proof
  fix \( x \) assume \( A3: x \in \text{Interval}(r,b,c) \)
  note \( A1 \)
  moreover from \( A2 A3 \) have \( \langle a,b \rangle \in r \land \langle b,x \rangle \in r \) using Order_ZF_2_L1A by simp
  ultimately have \( T1: \langle a,x \rangle \in r \) by (rule Fol1_L3)
  note \( A1 \)
  moreover from \( A2 A3 \) have \( \langle x,c \rangle \in r \land \langle c,d \rangle \in r \) using Order_ZF_2_L1A by simp
  ultimately have \( \langle x,d \rangle \in r \) by (rule Fol1_L3)
with T1 show $x \in \text{Interval}(r,a,d)$ using Order_ZF_2_L1B
  by simp
qed

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: $a \in X$
  shows $\text{Interval}(r,a,a) = \{a\}$
proof
  from A1 A3 have $\langle a,a \rangle \in r$ using refl_def by simp
  with A1 A3 show $\{a\} \subseteq \text{Interval}(r,a,a)$ using Order_ZF_2_L2 by simp
  from A2 show $\text{Interval}(r,a,a) \subseteq \{a\}$ 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: $\langle a,b \rangle \notin r$
  shows $\text{Interval}(r,a,b) = 0$
proof -
  { assume $\text{Interval}(r,a,b) \neq 0$ then obtain $x$ where $x \in \text{Interval}(r,a,b)$
    using assms refl_def IsBoundedAbove_def IsBoundedBelow_def IsBounded_def by auto
  }
  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: $r \subseteq X \times X$
  shows $\text{Interval}(r,a,b) \subseteq X$
proof 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: $a \in X$
  shows IsBounded($\{a\}$,r)
proof 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 $r \subseteq X \times X$
  and IsBoundedAbove($A$,r)
  shows $A \subseteq X$ 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
moreover { 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

definition IsBoundedAbove": fixes A, r type: X shows r = X×X A" if A is bounded above by B, then A is bounded above by B.

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 -
\{ \text{assume } A=0 \lor B=0 \\
\text{with } A3 \text{ have } \text{IsBoundedAbove}(A \cup B, r) \text{ by auto } \}

moreover
\{ \text{assume } \neg (A = 0 \lor B = 0) \\
\text{then have } T1: A \neq 0 \land B \neq 0 \text{ by auto} \\
\text{let } U = \text{GreaterOf}(r, u, w) \\
\text{from } T1 \text{ A4 D1 have } u \in X \land w \in X \text{ by auto} \\
\text{using } \text{IsBoundedAbove_def} \text{ by auto} \\
\text{ultimately show thesis by auto} \}

qed

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

\textbf{lemma Order_ZF_3_L4:}
\begin{itemize}
\item assumes $A1: r \text{ is total on } X$ and $A2: \text{trans}(r)$
\item and $A3: \text{IsBoundedAbove}(A, r)$ and $A4: a \in X$ and $A5: r \subseteq X \times X$
\item shows $\text{IsBoundedAbove}(A \cup \{a\}, r)$
\end{itemize}
\textbf{proof -}
\begin{itemize}
\item from $A1$ have $\text{refl}(X, r)$
\item with $A3 \text{ and } A4 \text{ and } A5 \text{ and } A2 \text{ and } A5 \text{ and } A2 \text{ and } A5 \text{ and } A2 \text{ and } A5 \text{ and } A2 $ have $\langle k, x \rangle \in r$ using $\text{Order_ZF}_3\_L1 \text{ IsBounded_def } \text{Order_ZF}_3\_L3$ by simp
\end{itemize}
\textbf{qed}

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

\textbf{lemma Order_ZF_3_L5B:}
\begin{itemize}
\item assumes $A1: r \text{ is total on } X$ and $A2: \text{trans}(r)$
\item and $A3: l \in X$ and $m \in X$
\item and $A4: \forall x \in A. \langle l, x \rangle \in r \land \forall x \in B. \langle m, x \rangle \in r$
\item shows $\forall x \in A \cup B. \langle \text{SmallerOf}(r, l, m), x \rangle \in r$
\end{itemize}
\textbf{proof -}
\begin{itemize}
\item let $k = \text{SmallerOf}(r, l, m)$
\item from $A1 \text{ A3 have } T1: \langle k, l \rangle \in r$ and $T2: \langle k, m \rangle \in r$
\item using $\text{Order_ZF}_3\_L2$ by auto
\item fix $x$ assume $A5: x \in A \cup B$ show $\langle k, x \rangle \in r$
\item proof -
\begin{itemize}
\item assume $x \in A$
\item with $A4 \text{ T1 have } \langle k, l \rangle \in r$ and $\langle l, x \rangle \in r$ by simp
\item with $A2 \text{ have } \langle k, x \rangle \in r$ by (rule Fol1_L3)
\item moreover
\begin{itemize}
\item assume $x \notin A$
\item with $A5 \text{ A4 T2 have } \langle k, m \rangle \in r$ and $\langle m, x \rangle \in r$ by simp
\item with $A2 \text{ have } \langle k, x \rangle \in r$ by (rule Fol1_L3)
\end{itemize}
\end{itemize}
\end{itemize}
\textbf{40}
ultimately show thesis by auto

qed

qed

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 ⊆ X×X  
shows IsBoundedBelow(A∪B,r)

**proof** -

\{ assume A=0 ∨ B=0  
  with A3 have thesis by auto \}

moreover

\{ assume ~ (A = 0 ∨ B = 0)  
  then have T1: A≠0 B≠0 by auto  
  with A3 obtain l m where D1: ∀x∈A. ⟨l,x⟩ ∈ r ∀x∈B. ⟨m,x⟩ ∈ r  
    using IsBoundedBelow_def by auto  
  let L = SmallerOf(r,l,m)  
  from T1 A4 D1 have T1: l∈X m∈X by auto  
  with A1 A2 D1 have ∀x∈A∪B. (L,x) ∈ r  
    using Order_ZF_3_L5B by blast  
  then have IsBoundedBelow(A∪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 ∪ \{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∈X and A5: r ⊆ X×X  
shows IsBoundedBelow(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_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 ⊆ X×X  
shows IsBounded(A∪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 $r \text{ is total on } X$ 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 $x \in$ 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 $\exists u. \forall x \in$ Interval($r$, $a$, $b$). $\langle x, u \rangle \in r$
  $\exists l. \forall x \in$ Interval($r$, $a$, $b$). $\langle 1, x \rangle \in r$
    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

42
A subset of a set that is bounded below is bounded below.

**Lemma Order_ZF_3_L12:** assumes A1: IsBoundedBelow(A,r) and A2: B ⊆ A
shows IsBoundedBelow(B,r)

**Proof -**

\[
\begin{align*}
\{ & \text{ assume } A = 0 \\
& \text{ with asms have IsBoundedBelow(B,r) } \\
& \text{ using IsBoundedBelow_def by auto } \}
\]

moreover

\[
\begin{align*}
\{ & \text{ assume } A \neq 0 \\
& \text{ with A1 have } \exists \, \forall x \in A. \ (1,x) \in r \\
& \text{ using IsBoundedBelow_def by simp } \\
& \text{ with A2 have } \exists \, \forall x \in B. \ (1,x) \in r \text{ by auto } \\
& \text{ then have IsBoundedBelow(B,r) using IsBoundedBelow_def } \\
& \text{ by auto } \}
\]

ultimately show IsBoundedBelow(B,r) by auto

**QED**

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

**Lemma Order_ZF_3_L13:** assumes A1: IsBoundedAbove(A,r) and A2: B ⊆ A
shows IsBoundedAbove(B,r)

**Proof -**

\[
\begin{align*}
\{ & \text{ assume } A = 0 \\
& \text{ with asms have IsBoundedAbove(B,r) } \\
& \text{ using IsBoundedAbove_def by auto } \}
\]

moreover

\[
\begin{align*}
\{ & \text{ assume } A \neq 0 \\
& \text{ with A1 have } \exists \, \forall x \in A. \ (x,1) \in r \\
& \text{ using IsBoundedAbove_def by simp } \\
& \text{ with A2 have } \exists \, \forall x \in B. \ (x,1) \in r \text{ by auto } \\
& \text{ then have IsBoundedAbove(B,r) using IsBoundedAbove_def } \\
& \text{ by auto } \}
\]

ultimately show 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:**

\[
\begin{align*}
& \text{ assumes } A1: r \text{ is total on } X \\
& \text{ and } A2: \text{trans}(r) \text{ and } A3: \text{antisym}(r) \\
& \text{ and } A4: r \subseteq X \times X \text{ and } A5: X \neq 0 \\
& \text{ and } A6: \forall x \in X. \ \exists a \in A. \ x \neq a \land (x,a) \in r \\
& \text{ shows } \neg \text{IsBoundedAbove}(A,r) \\
\end{align*}
\]

**Proof -**

\[
\begin{align*}
& \text{ from A5 A6 have I: } A \neq 0 \text{ by auto }
\end{align*}
\]
moreover assume $\text{IsBoundedAbove}(A, r)$
ultimately obtain $u$ where II: $\forall x \in A. \langle x, u \rangle \in r$
using $\text{IsBounded}_{\text{def}}$ $\text{IsBoundedAbove}_{\text{def}}$ by auto
with A4 I have $u \in X$ by auto
with A6 obtain $b$ where b$\in A$ and III: $u \not= b$ and $\langle u, b \rangle \in r$
by auto
with II have $\langle b, u \rangle \in r$ $\langle u, b \rangle \in r$ by auto
with A3 have $b = u$ by (rule Fol1_L4)
with III have False by simp
} thus $\neg \text{IsBoundedAbove}(A, r)$ 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 $\text{IsBoundedAbove}({x \in A. \langle x, a \rangle \in r}, r)$
using $\text{IsBounded}_{\text{def}}$ 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: $\text{IsBoundedBelow}(A, r)$
shows $\text{IsBounded}({x \in A. \langle x, a \rangle \in r}, r)$
proof -
{ assume A=0
  then have $\text{IsBounded}({x \in A. \langle x, a \rangle \in r}, r)$
  using $\text{IsBoundedBelow}_{\text{def}}$ $\text{IsBoundedAbove}_{\text{def}}$ $\text{IsBounded}_{\text{def}}$
  by auto }
moreover
{ assume A$\neq$0
  with A1 obtain 1 where I: $\forall x \in A. \langle 1, x \rangle \in r$
  using $\text{IsBoundedBelow}_{\text{def}}$ by auto
  then have $\forall y \in {x \in A. \langle x, a \rangle \in r}. \langle 1, y \rangle \in r$ by simp
  then have $\text{IsBoundedBelow}({x \in A. \langle x, a \rangle \in r}, r)$
  by (rule Order_ZF_3_L9)
  then have $\text{IsBounded}({x \in A. \langle x, a \rangle \in r}, r)$
  using Order_ZF_3_L15 $\text{IsBounded}_{\text{def}}$ by simp }
ultimately show thesis by blast
qed

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 ($\leq$ 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 ≡ ∀ x y z. (x ∈ A ∧ y ∈ A ∧ z ∈ A) →
(⟨x,y⟩ ∈ R ←→ ¬(x=y ∨ ⟨y,x⟩ ∈ R)) ∧
(⟨x,y⟩ ∈ R ∧ ⟨y,z⟩ ∈ R → ⟨x,z⟩ ∈ R)

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

definition Sup(B,A,R) ≡ ⋃ {x ∈ A. (∀y ∈ B. (x,y) /∈ R) ∧
(∀y ∈ A. (y,x) ∈ R → (∃z ∈ B. ⟨y,z⟩ ∈ R))}

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

definition Infim(B,A,R) ≡ Sup(B,A,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)
part_ord(A,R)
linear(A,R)
tot_ord(A,R)
proof -
  from A1 have I:
    ∀ x y z. (x∈A ∧ y∈A ∧ z∈A) →
    (⟨x,y⟩ ∈ R ←→ ¬(x=y ∨ ⟨y,x⟩ ∈ R)) ∧
    (⟨x,y⟩ ∈ R ∧ ⟨y,z⟩ ∈ R → ⟨x,z⟩ ∈ R)
    unfolding StrictOrder_def by simp
  then have ∀x∈A. ⟨x,x⟩ /∈ R by blast
  then show irrefl(A,R) using irrefl_def by simp
moreover
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 ∩ A × A)
    using part Ord_Imp_asym by simp
  then have V: ∀ x y. (x,y) ∈ (R ∩ A × A) −→ ¬(y,x) ∈ (R ∩ A × A))
    unfolding asym_def by blast
  from I have VI: ∀ x y z. (x,y) ∈ R ∨ x=y ∨ (y,x) ∈ R
    unfolding linear_def by blast
  from III have VII:
       ∀ x y z. (x,y) ∈ R −→ (y,z) ∈ R −→ (x,z) ∈ R
    unfolding trans_on_def by blast
  { fix x y z
    assume T: x∈A y∈A z∈A
    have ⟨x,y⟩ ∈ R −→ ¬(x=y ∨ (y,x) ∈ R)
      proof
        assume A2: ⟨x,y⟩ ∈ R
        with V T have ¬((y,x) ∈ R) by blast
        moreover from II T A2 have x≠y using irrefl_def
        by auto
        ultimately show ¬(x=y ∨ (y,x) ∈ R) by simp
      next assume ¬(x=y ∨ (y,x) ∈ R)
        with VI T show ⟨x,y⟩ ∈ R by auto
      qed
    moreover from VII T have
\[(x,y) \in R \land (y,z) \in R \rightarrow (x,z) \in R\]

by blast

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 \(\forall x \ y \ z. \ (x \in A \land y \in A \land z \in A) \rightarrow\)

\((x,y) \in R \leftrightarrow \neg(x=y \lor (y,x) \in R)) \land\)

\((x,y) \in R \land (y,z) \in R \rightarrow (x,z) \in R)\)

by auto

then show \(R \text{ Orders } A\) 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 converse(converse(r)) = r one needs r to consist of ordered pairs, which does not follow from the StrictOrder definition above).

lemma cnvso: shows \(R \text{ Orders } A \leftrightarrow \text{ converse}(R) \text{ Orders } A\)

proof

let \(r = \text{ converse}(R)\)

assume \(R \text{ Orders } A\)

then have \(\text{ tot_od}(A,r)\) using orders_imp_tot_ord tot_ord_converse

by simp

then show \(r \text{ Orders } A\) using tot_ord_imp_orders

by simp

next

let \(r = \text{ converse}(R)\)

assume \(r \text{ Orders } A\)

then have \(A2: \forall x \ y \ z. \ (x \in A \land y \in A \land z \in A) \rightarrow\)

\((x,y) \in r \leftrightarrow \neg(x=y \lor (y,x) \in r)) \land\)

\((x,y) \in r \land (y,z) \in r \rightarrow (x,z) \in r)\)

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: \((y,x) \in r \leftrightarrow \neg(x=y \lor (x,y) \in r))\) and

II: \((y,x) \in r \land (z,y) \in r \rightarrow (z,x) \in r\)

by auto

from I have \((x,y) \in R \leftrightarrow \neg(x=y \lor (y,x) \in R)\)

by auto

moreover from II have \((x,y) \in R \land (y,z) \in R \rightarrow (x,z) \in R\)

by auto

47
ultimately have \((x, y) \in R \iff \neg(x = y \lor (y, x) \in R)) \land \\
(\langle x, y \rangle \in R \land (\langle y, z \rangle \in R \implies \langle x, z \rangle \in R))\) by simp

then have \(\forall x \ y \ z. \ (x \in A \land y \in A \land z \in A) \implies \)
\((\langle x, y \rangle \in R \iff \neg(x = y \lor (y, x) \in R)) \land \\
(\langle x, y \rangle \in R \land (\langle y, z \rangle \in R \implies \langle x, z \rangle \in R))\)

by auto

then show \(R\) Orders \(A\) using StrictOrder_def by simp

qed

Supremum is unique, if it exists.

\[\text{lemma supeu: assumes A1: R Orders A and A2: x \in A and A3: }\forall y \in B. \ (x, y) \not\in R \text{ and A4: } \forall y \in A. \ (y, x) \in R \implies (\exists z \in B. \ (y, z) \in R)\]

shows 
\(\exists ! x. \ x \in A \land (\forall y \in B. \ (x, y) \not\in R) \land (\forall y \in A. \ (y, x) \in R \implies (\exists z \in B. \ (y, z) \in R))\)

proof

from A2 A3 A4 show 
\(\exists x. \ x \in A \land (\forall y \in B. \ (x, y) \not\in R) \land (\forall y \in A. \ (y, x) \in R \implies (\exists z \in B. \ (y, z) \in R))\)

by auto

next fix \(x_1 \ x_2\)

assume A5:
\(x_1 \in A \land (\forall y \in B. \ (x_1, y) \not\in R) \land (\forall y \in A. \ (y, x_1) \in R \implies (\exists z \in B. \ (y, z) \in R))\)
\(x_2 \in A \land (\forall y \in B. \ (x_2, y) \not\in R) \land (\forall y \in A. \ (y, x_2) \in R \implies (\exists z \in B. \ (y, z) \in R))\)

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

then have \(\forall x \in A. \ \forall y \in A. \ (x, y) \in R \lor x = y \lor (y, x) \in R\)

unfolding linear_def by blast

with A5 have \(\langle x_1, x_2 \rangle \in R \lor x_1 = x_2 \lor (x_2, x_1) \in R\)

by blast

moreover 
{ assume \(\langle x_1, x_2 \rangle \in R\)

with A5 obtain \(z\) where \(z \in B\) and \(\langle x_1, z \rangle \in R\) by auto

with A5 have False by auto }

moreover 
{ assume \(\langle x_2, x_1 \rangle \in R\)

with A5 obtain \(z\) where \(z \in B\) and \(\langle x_2, z \rangle \in R\) by auto

with A5 have False by auto }

ultimately show \(x_1 = x_2\) by auto

qed

Supremum has expected properties if it exists.

\[\text{lemma sup_props: assumes A1: R Orders A and A2: } \exists x \in A. \ (\forall y \in B. \ (x, y) \not\in R) \land (\forall y \in A. \ (y, x) \in R \implies (\exists z \in B. \ (y, z) \in R))\]

shows 
\(\forall y \in B. \ (\langle \sup(B, A, R) \rangle, y \rangle \not\in R\)
∀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))}

from A2 obtain x where
x∈A and (∀y∈B. ⟨x,y⟩ /∈ R) and ∀y∈A. ⟨y,x⟩ ∈ R → (∃z∈B. ⟨y,z⟩ ∈ R)
by auto

with A1 have I:
∃x. x∈A ∧ (∀y∈B. ⟨x,y⟩ /∈ R) ∧ (∀y∈A. ⟨y,x⟩ ∈ R → (∃z∈B. ⟨y,z⟩ ∈ R))
using supeu by simp
then have (⋃S) ∈ A by (rule ZF1_1_L9)
then show Sup(B,A,R) ∈ A using Sup_def by simp
from I have II:
∀y∈B. ⟨y,⋃S⟩ /∈ R ∧ (∀y∈A. ⟨y,⋃S⟩ ∈ R → (∃z∈B. ⟨y,z⟩ ∈ R))
by (rule ZF1_1_L9)
hence ∀y∈B. ⟨y,⋃S⟩ /∈ R by blast
moreover have III: (⋃S) = Sup(B,A,R) using Sup_def by simp
ultimately show ∀y∈B. ⟨y,Sup(B,A,R)⟩ /∈ R by simp
from II have IV: ∀y∈A. ⟨y,⋃S⟩ ∈ R → (∃z∈B. ⟨y,z⟩ ∈ R)
by blast

{ fix y assume A3: y∈A and ⟨y,Sup(B,A,R)⟩ ∈ R
with III have ⟨y,⋃S⟩ ∈ R by simp
with IV A3 have ∃z∈B. ⟨y,z⟩ ∈ R by blast
} thus ∀y∈A. ⟨y,Sup(B,A,R)⟩ ∈ R → (∃z∈B. ⟨y,z⟩ ∈ 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:
∃x∈A. (∀y∈B. ⟨x,y⟩ /∈ R) ∧ (∀y∈A. ⟨y,x⟩ ∈ R → (∃z∈B. ⟨y,z⟩ ∈ R))
and A3: c ∈ A and A4: ∀z∈B. ⟨c,z⟩ /∈ R
shows ⟨c, Sup(B,A,R)⟩ /∈ R
proof -
from A1 A2 have
∀y∈A. ⟨y,Sup(B,A,R)⟩ ∈ R → (∃z∈B. ⟨y,z⟩ ∈ R)
by (rule sup_props)
with A3 A4 have ⟨c, Sup(B,A,R)⟩ /∈ R by auto

qed

7 Even more on order relations
This theory is a continuation of Order_ZF and talks about maximum 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

**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 a maximum implies that it is not empty.

**lemma** set_min_not_empty: assumes HasAminimum(r,A) shows A ≠ 0

**proof**
- from assms have HasAmaximum(r, ∩ a ∈ A. r{a}) unfolding HasAminimum_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

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 $\exists M. M \in A \land (\forall x \in A. \langle x, M \rangle \in r)$

proof

from A2 show $\exists M. M \in A \land (\forall x \in A. \langle x, M \rangle \in r)$

using HasAm maximum def by auto

fix $M_1, M_2$

assume

A2: $M_1 \in A \land (\forall x \in A. \langle x, M_1 \rangle \in r)$
M2 $\in A \land (\forall x \in A. \langle x, M_2 \rangle \in r)$

then have $\langle M_1, M_2 \rangle \in r$ $\langle M_2, M_1 \rangle \in r$ by auto

with A1 show $M_1 = M_2$ 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 HasAm minimum def by auto

fix $m_1, m_2$

assume

A2: $m_1 \in A \land (\forall x \in A. \langle m_1, x \rangle \in r)$
M2 $\in A \land (\forall x \in A. \langle m_2, x \rangle \in r)$

then have $\langle m_1, m_2 \rangle \in r$ $\langle m_2, m_1 \rangle \in r$ by auto

with A1 show $m_1 = m_2$ 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 = $\exists ! M. M \in A \land (\forall x \in A. \langle x, M \rangle \in r)$

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 = $\exists ! m. m \in A \land (\forall x \in A. \langle m, x \rangle \in r)$

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. ⟨ 1,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∈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⊆X and A4: a∈X and A5: HasAmaximum(r,A)
shows HasAmaximum(r,A∪{a})
proof -
from A3 A4 have A∪{a} ⊆ X by auto
with A1 have r {is total on} (A∪{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∪{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: A⊆X and A4: a∈X and A5: HasAminimum(r,A)
shows HasAminimum(r,A∪{a})
proof -
from A3 A4 have A∪{a} ⊆ X by auto
with A1 have r {is total on} (A∪{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∪{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: \( r \subseteq X \times X \) and
A4: \( \forall A. \text{IsBounded}(A, r) \land A \neq 0 \rightarrow \text{HasAminimum}(r, A) \) and
A5: \( B \neq 0 \) and A6: \( \text{IsBoundedBelow}(B, r) \)
shows \( \text{HasAminimum}(r, B) \)

**proof** -

from A5 obtain \( b \) where T: \( b \in B \) by auto
let \( L = \{ x \in B. \langle x, b \rangle \in r \} \)
from A3 A6 T have T1: \( b \in X \) using Order_ZF_3_L1B by blast
with A1 have T2: \( b \in L \)
  using total_is_refl refl_def by simp
then have \( L \neq 0 \) by auto
moreover have \( \text{IsBoundedBelow}(L, r) \)
proof -
  have \( L \subseteq B \) by auto
  with A6 have \( \text{IsBoundedBelow}(L, r) \)
    using Order_ZF_3_L12 by simp
  moreover have \( \text{IsBoundedAbove}(L, r) \)
    by (rule Order_ZF_3_L15)
  ultimately have \( \text{IsBoundedAbove}(L, r) \land \text{IsBoundedBelow}(L, r) \)
    by blast
  then show \( \text{IsBounded}(L, r) \)
    using IsBounded_def by simp
qed
ultimately have \( \text{IsBounded}(L, r) \land L \neq 0 \) by blast
with A4 have \( \text{HasAminimum}(r, L) \)
then obtain \( m \) where I: \( m \in L \) and II: \( \forall x \in L. \langle m, x \rangle \in r \)
using HasAminimum_def by auto
then have III: \( \langle m, b \rangle \in r \) by simp
from I have \( m \in B \) by simp
moreover have \( \forall x \in B. \langle m, x \rangle \in r \)
proof
  fix \( x \) assume A7: \( x \in B \)
  from A3 A6 have \( B \subseteq X \) using Order_ZF_3_L1B by blast
  with A1 A7 T1 have \( x \in L \cup \{ x \in B. \langle b, x \rangle \in r \} \)
    using Order_ZF_1_L5 by simp
  then have \( x \in L \lor \langle b, x \rangle \in r \) by auto
  moreover
  \{ assume \( x \in L \)
    with II have \( \langle m, x \rangle \in r \) by simp \}
  moreover
  \{ assume \( \langle b, x \rangle \in r \)
    with A2 III have trans(r) and \( \langle m, b \rangle \in r \land \langle b, x \rangle \in r \)
    by auto
    then have \( \langle m, x \rangle \in r \) by (rule Fol1_L3) \}
  ultimately show \( \langle m, x \rangle \in r \) by auto
qed
ultimately show \( \text{HasAminimum}(r, B) \)
by auto
qed

54
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 Order_ZF_4_L11A:**

Assumes

1. \( r \) is total on \( X \)
2. \( r \) is transitive
3. \( r \subseteq X \times X \)
4. For all \( A \), if \( A \) is bounded and \( A \neq \emptyset \) then \( A \) attains a maximum.
5. \( B \neq \emptyset \)
6. \( B \) is bounded above.

Shows \( B \) attains a maximum.

**Proof** -

1. From 5 obtain \( b \) where \( T: b \in B \) by auto.
2. From 3 and 6 have \( T_1: b \in X \) using Order_ZF_3_L1A by blast.
3. With 1 and 7 have \( T_2: b \in U \).
4. Using total_is_refl refl_def by simp.
5. Then have \( U \neq \emptyset \) by auto.
6. Moreover have \( U \) is bounded above.

**Proof** -

1. Have \( U \subseteq B \) by auto.
2. With 6 have \( U \) is bounded above.
4. Ultimately have \( U \) is bounded above and \( U \) is bounded below.

**QED**

Ultimately have \( U \) is bounded above and \( U \neq \emptyset \) by blast.

With 4 have HasMaximum\((r, U)\) by simp.

Then obtain \( m \) where \( I: m \in U \) and \( II: \forall x \in U. \langle x, m \rangle \in r \)

Using HasMaximum_def by auto.

Then have \( III: \langle b, m \rangle \in r \) by simp.

From 1 have \( m \in B \) by simp.

Moreover have \( \forall x \in B. \langle x, m \rangle \in r \)

**Proof** -

1. Fix \( x \) assume \( A7: x \in B \)
2. From 3 and 6 have \( B \subseteq X \) using Order_ZF_3_L1A by blast.
3. With 1 and 7 have \( x \in \{ x \in B. \langle x, b \rangle \in r \} \cup U \)
4. Using Order_ZF_1_L5 by simp.

Then have \( x \in U \lor \langle x, b \rangle \in r \) by auto.

Moreover

1. Assume \( x \in U \)
2. With \( II \) have \( \langle x, m \rangle \in r \) by simp.

Moreover

1. Assume \( \langle x, b \rangle \in r \)
2. With \( A2 \) and \( III \) have trans\((r)\) and \( \langle x, b \rangle \in r \land \langle b, m \rangle \in r \)

By auto.
then have $(x,m) \in r$ by (rule Fol1_L3) 
ultimately show $(x,m) \in r$ by auto
qed
ultimately show HasAmaximum$(r,B)$ using HasAmaximum_def by auto
qed

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 $\forall a \in A. \ (L,a) \in r$
shows $(L,\text{Minimum}(r,A)) \in r$
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 $\forall a \in A. \ (a,M) \in r$
shows $(\text{Maximum}(r,A),M) \in r$
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 $A1: \text{antisym}(r)$ and $A2: M \in A$ and $A3: \forall a \in A. \ (a,M) \in r$
shows $\text{Maximum}(r,A) = M$
proof -
from $A2$ $A3$ have $I: \text{HasAmaximum}(r,A)$ using HasAmaximum_def by auto
with $A1$ have $\exists! M. \ M \in A \land (\forall x \in A. \ (x,M) \in r)$
using Order_ZF_4_L1 by simp
moreover from $A2$ $A3$ have $M \in A \land (\forall x \in A. \ (x,M) \in r)$ by simp
moreover from $A1$ $I$ have $\text{Maximum}(r,A) \in A \land (\forall x \in A. \ (x,\text{Maximum}(r,A)) \in r)$
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 $A1: \text{antisym}(r)$ and $A2: m \in A$ and $A3: \forall a \in A. \ (m,a) \in r$
shows $\text{Minimum}(r,A) = m$
proof -
from $A2$ $A3$ have $I: \text{HasAminimum}(r,A)$ 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 Minimum(r,A), x \rangle \in r)
  using Order_ZF_4_L4 by simp
ultimately show 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: antisym(r) and A2: r \{is total on\} X and
  A3: A \subseteq X and
  A4: \neg 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
  }
  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 assms 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 assms by auto

57
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 \texttt{sup_leq_up_bnd} instead.

\textbf{lemma} \texttt{Order_ZF\_5\_L3:} \textbf{assumes} $\text{A1: antisym(r)}$ \textbf{and} $\text{A2: A} \neq 0$ \textbf{and} $\text{A3: HasAminimum(r,} \bigcap a \in A. r\{a\})$ \textbf{and} $\text{A4: } \forall a \in A. \langle a, u \rangle \in r$
\textbf{shows} $\langle \text{Supremum(r,A)}, u \rangle \in r$

\textbf{proof -} \\
\textbf{let} $U = \bigcap a \in A. r\{a\}$
\textbf{from} A4 \textbf{have} $\forall a \in A. u \in r\{a\}$ \textbf{using} image_singleton_iff \textbf{by simp}
\textbf{with} A2 \textbf{have} $u \in U$ \textbf{by auto}
\textbf{with} A1 A3 \textbf{show} $\langle \text{Supremum(r,A)}, u \rangle \in r$
\textbf{using} Order\_ZF\_4\_L4 \texttt{Supremum_def} \textbf{by simp}

\textbf{qed}

\textbf{Supremum is less or equal than any upper bound.}

\textbf{lemma} \texttt{sup_leq_up_bnd:} \textbf{assumes} $\text{antisym(r)}$ $\text{HasAsupremum(r,A)}$ $\forall a \in A. \langle u, a \rangle \in r$
\textbf{shows} $\langle \text{Supremum(r,A)}, u \rangle \in r$

\textbf{proof -} \\
\textbf{let} $U = \bigcap a \in A. r\{a\}$
\textbf{from} assms(3) \textbf{have} $\forall a \in A. u \in r\{a\}$ \textbf{using} image_singleton_iff \textbf{by simp}
\textbf{with} assms(2) \textbf{have} $u \in U$ \textbf{using} set_sup_not_empty \textbf{by auto}
\textbf{with} assms(1,2) \textbf{show} $\langle \text{Supremum(r,A)}, u \rangle \in r$
\textbf{unfolding} HasAsupremum_def \texttt{Supremum_def} \textbf{using} Order\_ZF\_4\_L4 \textbf{by simp}

\textbf{qed}

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

\textbf{lemma} \texttt{Order\_ZF\_5\_L4:} \textbf{assumes} $\text{A1: antisym(r)}$ \textbf{and} $\text{A2: A} \neq 0$ \textbf{and} $\text{A3: HasAmaximum(r,} \bigcap a \in A. r\{-a\})$ \textbf{and} $\text{A4: } \forall a \in A. \langle l, a \rangle \in r$
\textbf{shows} $\langle l, \text{Infimum(r,A)} \rangle \in r$

\textbf{proof -} \\
\textbf{let} $L = \bigcap a \in A. r\{-a\}$
\textbf{from} A4 \textbf{have} $\forall a \in A. l \in r\{-a\}$ \textbf{using} image_singleton_iff \textbf{by simp}
\textbf{with} A2 \textbf{have} $l \in L$ \textbf{by auto}
\textbf{with} A1 A3 \textbf{show} $\langle l, \text{Infimum(r,A)} \rangle \in r$
\textbf{using} Order\_ZF\_4\_L3 \texttt{Infimum_def} \textbf{by simp}

\textbf{qed}

\textbf{Infimum is greater or equal than any upper bound.}

\textbf{lemma} \texttt{inf_geq_lo_bnd:} \textbf{assumes} $\text{antisym(r)}$ $\text{HasAnInfimum(r,A)}$ $\forall a \in A. \langle u, a \rangle \in r$

\textbf{58}
shows \((u, \text{Infimum}(r, A)) \in r\)

**proof**
- let \(U = \bigcap_{a \in A.} r\{a\}\)
  - from assms(3) have \(\forall a \in A. \ u \in r\{a\}\) using \(\text{vimage} \_ \text{singleton}_\text{iff}\) by simp
  - with assms(2) have \(u \in U\) using \(\text{set}_\text{inf}_\text{not}\_\text{empty}\) by auto
  - with assms(1,2) show \(\langle u, \text{Infimum}(r, A) \rangle \in r\)
    unfolding \(\text{HasAnInfimum}_\text{def}\) \(\text{Infimum}_\text{def}\) using \(\text{Order}_\text{ZF} \_\text{4}_\text{L3}\) 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** \(\text{Order}_\text{ZF} \_\text{5}_\text{L5}\): assumes \(A1: \text{antisym}(r)\) and \(A2: A \neq 0\) and
\(A3: \forall x \in A. \Laurent marker{\langle x,z \rangle} \in r\) \(\text{and}\)
\(A4: \forall y. (\forall x \in A. \Laurent marker{\langle x,y \rangle} \in r) \rightarrow \Laurent marker{\langle z,y \rangle} \in r\)
shows
\(\text{HasAminimum}(r, \bigcap_{a \in A.} r\{a\})\)
\(z = \text{Supremum}(r,A)\)

**proof**
- let \(B = \bigcap_{a \in A.} r\{a\}\)
  - from assms(2,3,4) have \(I: z \in B \ \ \forall y \in B. \Laurent marker{\langle z,y \rangle} \in r\)
    by auto
  - then show \(\text{HasAminimum}(r, \bigcap_{a \in A.} r\{a\})\)
    using \(\text{HasAminimum}_\text{def}\) by auto
  - from \(A1\) \(I\) show \(z = \text{Supremum}(r,A)\)
    unfolding \(\text{Supremum}_\text{def}\) by simp

qed

The dual theorem to \(\text{Order}_\text{ZF} \_\text{5}_\text{L5}\): 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** \(\text{inf}_\text{glb}\):
- assumes \(\text{antisym}(r)\) \(A \neq 0\) \(\forall x \in A. \Laurent marker{\langle z,x \rangle} \in r\)
  \(\forall y. (\forall x \in A. \Laurent marker{\langle y,x \rangle} \in r) \rightarrow \Laurent marker{\langle y,z \rangle} \in r\)
  \(\in r\)
  shows
  \(\text{HasAmaximum}(r, \bigcap_{a \in A.} r\{a\})\)
\(z = \text{Infimum}(r,A)\)

**proof**
- let \(B = \bigcap_{a \in A.} r\{a\}\)
  - from assms(2,3,4) have \(I: z \in B \ \ \forall y \in B. \Laurent marker{\langle y,z \rangle} \in r\)
    by auto
  - then show \(\text{HasAmaximum}(r, \bigcap_{a \in A.} r\{a\})\)
    unfolding \(\text{HasAmaximum}_\text{def}\) by auto
  - from assms(1) \(I\) show \(z = \text{Infimum}(r,A)\)
    using \(\text{Order}_\text{ZF} \_\text{4}_\text{L14}\) \(\text{Infimum}_\text{def}\) by simp

qed

Supremum and infimum of a singleton is the element.

**lemma** \(\text{sup}_\text{inf}_\text{singl}\): assumes \(\text{antisym}(r)\) \(\text{refl}(X,r)\) \(z \in X\)
shows
  \( \text{HasAsupremum}(r,\{z\}) \quad \text{Supremum}(r,\{z\}) = z \) and
  \( \text{HasAnInfimum}(r,\{z\}) \quad \text{Infimum}(r,\{z\}) = z \)

proof -
  from assms show \( \text{Supremum}(r,\{z\}) = z \) and \( \text{Infimum}(r,\{z\}) = z \)
    using inf_glb Order_ZF_5_L5 unfolding refl_def by auto
  from assms show \( \text{HasAsupremum}(r,\{z\}) \)
    using Order_ZF_5_L5 unfolding HasAsupremum_def refl_def by blast
  from assms show \( \text{HasAnInfimum}(r,\{z\}) \)
    using inf_glb unfolding HasAnInfimum_def refl_def by blast
qed

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

**lemma** Order_ZF_5_L6:
  assumes A1: antisym\( (r) \) and A2: \( A \neq 0 \) and
  A3: HasAmaximum\( (r,A) \)
  shows\( \text{Maximum}(r,A) = \text{Supremum}(r,A) \)
proof -
  let \( M = \text{Maximum}(r,A) \)
  from assms(1,3) have \( M \in A \) and I: \( \forall x \in A. \langle x,M \rangle \in r \)
    using Order_ZF_4_L3 by auto
  with assms(1,2) have II: \( \forall y. (\forall x \in A. \langle x,y \rangle \in r) \longrightarrow \langle M,y \rangle \in r \)
    by simp
  with A1 A2 II show \( \text{HasAminimum}(r,\nabla a \in A. r\{a\}) \)
    by (rule Order_ZF_5_L5)
  then show \( \text{HasAsupremum}(r,A) \)
    unfolding HasAsupremum_def by simp
  from assms(1,2) \( M \in A \) I show \( M = \text{Supremum}(r,A) \)
    using Order_ZF_5_L5(2) by blast
qed

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.

**lemma** max_is_sup: assumes antisym\( (r) \) \( A \neq 0 \) HasAmaximum\( (r,A) \)
  shows HasAsupremum\( (r,A) \) and \( \text{Maximum}(r,A) = \text{Supremum}(r,A) \)
proof -
  let \( M = \text{Maximum}(r,A) \)
  from assms(1,3) have \( M \in A \) and I: \( \forall x \in A. \langle x,M \rangle \in r \) using Order_ZF_4_L3
    by auto
  with assms(1,2) have \( \text{HasAminimum}(r,\nabla a \in A. r\{a\}) \)
    using Order_ZF_5_L5(1) by blast
  then show \( \text{HasAsupremum}(r,A) \)
    unfolding HasAsupremum_def by simp
  from assms(1,2) \( M \in A \) I show \( M = \text{Supremum}(r,A) \)
    using Order_ZF_5_L5(2) by blast
qed
Minimum is the infimum if it exists.

**lemma min_is_inf**: assumes \( \text{antisym}(r) \) \( A \neq 0 \) \( \text{HasAminimum}(r,A) \)

shows \( \text{HasAnInfimum}(r,A) \) and \( \text{Minimum}(r,A) = \text{Infimum}(r,A) \)

**proof** -

let \( M = \text{Minimum}(r,A) \)

from \( \text{assms}(1,3) \) have \( M \in A \) and \( I: \forall x \in A. \langle M,x \rangle \in r \) using Order_ZF_4_L4

by auto

with \( \text{assms}(1,2) \) have \( \text{HasAmaximum}(r,\bigcap a \in A. r\{a\}) \) using inf_glb(1) by blast

then show \( \text{HasAnInfimum}(r,A) \) unfolding \( \text{HasAnInfimum}_\text{def} \) by simp

from \( \text{assms}(1,2) \) \( \langle M \in A \rangle \) I show \( M = \text{Infimum}(r,A) \) using inf_glb(2) by blast

**qed**

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

**lemma min_max_two_el**: assumes \( r \{\text{is total on}\} X \ x \in X \ y \in X \)

shows \( \text{HasAminimum}(r,\{x,y\}) \) and \( \text{HasAmaximum}(r,\{x,y\}) \)

using \( \text{assms} \) unfolding \( \text{IsTotal}_\text{def} \) \( \text{HasAminimum}_\text{def} \) \( \text{HasAmaximum}_\text{def} \) by auto

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

**lemma inf_sup_two_el**: assumes \( \text{antisym}(r) \) \( r \{\text{is total on}\} X \ x \in X \ y \in X \)

shows \( \text{HasAnInfimum}(r,\{x,y\}) \)

\( \text{Minimum}(r,\{x,y\}) = \text{Infimum}(r,\{x,y\}) \)

\( \text{HasAsupremum}(r,\{x,y\}) \)

\( \text{Maximum}(r,\{x,y\}) = \text{Supremum}(r,\{x,y\}) \)

using \( \text{assms} \) \( \text{min_max_two_el} \) \( \text{max_is_sup} \) \( \text{min_is_inf} \) by auto

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

**lemma sup_in_space**: assumes \( r \subseteq X \times X \) \( \text{antisym}(r) \) \( \text{HasAminimum}(r,\bigcap a \in A. r\{a\}) \)

shows \( \text{Supremum}(r,A) \in X \) and \( \forall x \in A. \langle x,\text{Supremum}(r,A) \rangle \in r \)

**proof** -

from \( \text{assms}(3) \) have \( A \neq 0 \) using set_sup_not_empty unfolding \( \text{HasAsupremum}_\text{def} \) by simp

then obtain a where \( a \in A \) by auto

with \( \text{assms}(1,2,3) \) show \( \text{Supremum}(r,A) \in X \) unfolding \( \text{Supremum}_\text{def} \)

using Order_ZF_4_L4 Order_ZF_5_L1 by blast

from \( \text{assms}(2,3) \) show \( \forall x \in A. \langle x,\text{Supremum}(r,A) \rangle \in r \) unfolding \( \text{Supremum}_\text{def} \)

using Order_ZF_4_L4 by blast

**qed**

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

**lemma inf_in_space**:
assumes \( r \subseteq X \times X \) antisym(r) HasAmaximum(r, \( \bigcap a \in A. \ r\{a\} \))

shows Infimum(r,A) \in X and \( \forall x \in A. \ (\text{Infimum}(r,A),x) \in r \)

proof -
from assms(3) have \( A \neq \emptyset \) 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 \( \forall x \in A. \ (\text{Infimum}(r,A),x) \in r \) 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: antisym(r) and
A3: \( r \) is complete and
A4: \( A \neq \emptyset \) and A5: \( \exists x \in X. \ \forall y \in A. \ (y,x) \in r \)
shows Supremum(r,A) \in X and \( \forall x \in A. \ (x,\text{Supremum}(r,A)) \in r \)

proof -
from A3 A4 A5 have 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 \( \forall x \in A. \ (x,\text{Supremum}(r,A)) \in r \)
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)
\( \forall T \in T. \ \text{HasAnInfimum}(r,T) \)
\( \text{HasAnInfimum}(r,\{\text{Infimum}(r,T).T \in T\}) \)
shows \( \text{HasAnInfimum}(r, \bigcup T) \) and \( \text{Infimum}(r,\{\text{Infimum}(r,T).T \in T\}) = \text{Infimum}(r, \bigcup T) \)

proof -
let i = \( \text{Infimum}(r,\{\text{Infimum}(r,T).T \in T\}) \)
note assms(2)
moreover from assms(4,5) have \( \bigcup T \neq \emptyset \) using set_inf_not_empty by blast
moreover have \( \forall T \in T. \forall t \in T. \ (i,t) \in r \)
proof -
\{ fix T t assume T \in T t \in T
with assms(1,2,4) have \( (\text{Infimum}(r,T),t) \in r \)
unfolding HasAnInfimum_def using inf_in_space(2) by blast
moreover from assms(1,2,5) \( \langle T \in T \rangle \) have \( \langle i,\text{Infimum}(r,T) \rangle \in r \)
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

62
qed

hence I: \( \forall t \in \bigcup T. \, \langle i, t \rangle \in r \) by auto

moreover have J: \( \forall y. \, \left( \forall x \in \bigcup T. \, \langle y, x \rangle \in r \right) \rightarrow \langle y, i \rangle \in r \)

proof -
{ fix y x assume A: \( \forall x \in \bigcup T. \, \langle y, x \rangle \in r \)
  with assms(2,4) have \( \forall a \in \{ \text{Infimum}(r, T). \, T \in T \} \). \( \langle y, a \rangle \in r \) using inf_geq_lo_bnd
  by simp
  with assms(2,5) have \( \langle y, i \rangle \in r \) 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) \( \langle \bigcup T \rangle \neq 0 \) 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)
\( \forall T \in T. \, \text{HasAsupremum}(r, T) \)
\( \text{HasAsupremum}(r, \{ \text{Supremum}(r, T). \, T \in T \}) \)
shows \( \text{HasAsupremum}(r, \bigcup T) \) and \( \text{Supremum}(r, \{ \text{Supremum}(r, T). \, T \in T \}) = \text{Supremum}(r, \bigcup T) \)

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 \( \forall T \in T. \forall t \in T. \, \langle t, s \rangle \in r \)
proof -
{ fix T t assume T \in T \, t \in T
  with assms(1,2,4) have \( \langle t, \text{Supremum}(r, T) \rangle \in r \)
  unfolding HasAsupremum_def using sup_in_space(2) by blast
  moreover from assms(1,2,5) \( \langle T \in T \rangle \) have \( \langle \text{Supremum}(r, T), s \rangle \in r \)
  unfolding HasAsupremum_def using sup_in_space(2) by blast
  moreover note assms(3)
  ultimately have \( \langle t, s \rangle \in r \) unfolding trans_def by blast
}
thus thesis by simp

qed

hence I: \( \forall t \in \bigcup T. \, \langle t, s \rangle \in r \) by auto

moreover have J: \( \forall y. \, \left( \forall x \in \bigcup T. \, \langle x, y \rangle \in r \right) \rightarrow \langle s, y \rangle \in r \)

proof -
{ fix y x assume A: \( \forall x \in \bigcup T. \, \langle x, y \rangle \in r \)
  with assms(2,4) have \( \forall a \in \{ \text{Supremum}(r, T). \, T \in T \}. \, \langle a, y \rangle \in r \) using sup_leq_up_bnd
  by simp
  with assms(2,5) have \( \langle s, y \rangle \in r \) by (rule sup_leq_up_bnd)
}
thus thesis by simp

63
ultimately have HasAminimum(r, ∩ a∈∪T. r(a)) by (rule Order_ZF_5_L5)
then show HasAsupremum(r, ∪T) unfolding HasAsupremum_def by simp
from assms(2) ⟨∪T ≠ 0⟩ I J show s = Supremum(r, ∪T) by (rule Order_ZF_5_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** Order_ZF_5_L8:
assumes A1: \( r \subseteq X \times X \) and A2: IsLinOrder(X,r) and A3: \( r \) {is complete} and A4: \( A \subseteq X \neq 0 \) and A5: \( \exists x \in X. \forall y \in A. \langle y, x \rangle \in r \) and A6: \( \langle y, \text{Supremum}(r,A) \rangle \in r \quad y \neq \text{Supremum}(r,A) \)
shows \( \exists z \in A. \langle y, z \rangle \in r \land y \neq z \)

**proof** -
from A2 have I: antisym(r) and II: trans(r) and III: \( r \) {is total on} \( X \) using IsLinOrder_def by auto
from A1 A6 have T1: \( y \in X \) by auto
{ assume A7: \( \forall z \in A. \langle y, z \rangle / \in r \lor y = z \)
  from A4 I have antisym(r) and \( A \neq 0 \) by auto
  moreover have \( \forall x \in A. \langle x, y \rangle \in r \) proof
    fix x assume A8: \( x \in A \)
    with A4 have T2: \( x \in X \) by auto
    from A7 A8 have \( \langle y, x \rangle \notin r \lor y = x \) by simp
    with III T1 T2 show \( \langle x, y \rangle \in r \)
      using IsTotal_def total_is_refl refl_def by auto
  qed
moreover have \( \forall u. \langle \forall x \in A. \langle x, u \rangle \in r \rangle \rightarrow \langle y, u \rangle \in r \)
proof-
  { fix u assume A9: \( \forall x \in A. \langle x, u \rangle \in r \)
    from A4 A5 have IsBoundedAbove(A,r) and \( A \neq 0 \)
      using IsBoundedAbove_def by auto
    with A3 A4 A6 I A9 have \( \langle y, \text{Supremum}(r,A) \rangle \in r \land \langle \text{Supremum}(r,A), u \rangle \in r \)
      using IsComplete_def Order_ZF_5_L3 by simp
    with II have \( \langle y, u \rangle \in r \) by (rule Fol1_L3)
  } then show \( \forall u. \langle \forall x \in A. \langle x, u \rangle \in r \rangle \rightarrow \langle y, u \rangle \in r \)
    by simp
qed
ultimately have \( y = \text{Supremum}(r,A) \)
  by (rule Order_ZF_5_L5)
with A6 have False by simp
} then show \( \exists z \in A. \langle y, z \rangle \in r \land y \neq z \) by auto
qed
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
StrictVersion(r) ≡ r - \{\langle x,x \rangle. x \in \text{domain}(r)\}

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

lemma def_of_strict_ver: shows\(\langle x,y \rangle \in \text{StrictVersion}(r) \leftrightarrow \langle x,y \rangle \in r \land x \neq y\)
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 and \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 = \text{StrictVersion}(r)
s 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 ∨ q ∨ r
  using strict_of_tot by auto
moreover from A1 A4 have p → ¬ q ∧ ¬ r
  using def_of_strict_ver strict_of_antisym by simp
moreover from A4 have q → ¬ p ∧ ¬ r
  using def_of_strict_ver by simp
moreover from A1 A4 have r → ¬ p ∧ ¬ 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 ∈ X b ∈ 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

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

\text{ultimately have } \langle a,c \rangle \in \text{StrictVersion}(r) \\
\text{using def_of_strict_ver by simp}

{ \text{with } A3 \text{ show thesis by simp} }

\text{qed}

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

\text{lemma strict_of_transB:}
\text{assumes } A1: \text{trans}(r) \text{ and } A2: \text{antisym}(r)
\text{shows } \text{trans}(\text{StrictVersion}(r))

\text{proof -}
\text{let } s = \text{StrictVersion}(r)
\text{from } A1 A2 \text{ have }
\forall x y z. \langle x,y \rangle \in s \land \langle y,z \rangle \in s \longrightarrow \langle x,z \rangle \in s
\text{using strict_of_transA by blast}
\text{then show } \text{trans}(\text{StrictVersion}(r)) \text{ by (rule Fol1_L2)}
\text{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.

\text{lemma strict_of_compl:}
\text{assumes } A1: r \subseteq X\times X \text{ and } A2: \text{IsLinOrder}(X,r) \text{ and } \text{IsLinOrder_def}
\text{A3: } r \text{ {is complete} } \text{ and } \text{IsLinOrder_def}
\text{A4: } A \subseteq X \text{ and } A5: s = \text{StrictVersion}(r) \text{ and } \text{def_of_strict_ver}
\text{A6: } \exists u \in X. \forall y \in A. \langle y,u \rangle \in s
\text{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 \longrightarrow ( \exists z \in A. \langle y,z \rangle \in s ) )

\text{proof -}
\text{let } x = \text{Supremum}(r,A)
\text{from } A2 \text{ have } I: \text{antisym}(r) \text{ using IsLinOrder_def by simp}
\text{moreover from } A5 A6 \text{ have } \exists u \in X. \forall y \in A. \langle y,u \rangle \in r \\
\text{using def_of_strict_ver by auto}
\text{moreover note } A1 A3 A4
\text{ultimately have II: } x \in X \land \forall y \in A. \langle x,y \rangle \in r
\text{using Order_ZF_5_L7 by auto}
\text{then have III: } \exists x \in X. \forall y \in A. \langle y,x \rangle \in r \text{ by auto}
\text{from } A5 I II \text{ have } x \in X \land \forall y \in A. \langle x,y \rangle \notin s
\text{using geq_impl_not_less by auto}
\text{moreover from } A1 A2 A3 A4 A5 III \text{ have }
\forall y \in X. \langle y,x \rangle \in s \longrightarrow ( \exists z \in A. \langle y,z \rangle \in s )
\text{using def_of_strict_ver Order_ZF_5_L8 by simp}
\text{ultimately show}

67
\[ \exists x \in X. (\forall y \in A. \langle x, y \rangle \notin \mathcal{S}) \land (\forall y \in X. \langle y, x \rangle \in \mathcal{S} \rightarrow (\exists z \in A. \langle y, z \rangle \in \mathcal{S})) \]

by auto
qed

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

**Lemma strict_ver_rel:**

**assumes** A1: \( r \subseteq A \times A \)

**shows** StrictVersion(r) \( \subseteq A \times A \)

**using** assms StrictVersion_def by auto

end

8 Functions - introduction

theory func1 imports ZF.func Fol1 ZF1

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 \( X \rightarrow Y \). It just happens that the colon ":" is a synonym of the set membership symbol in Isabelle/ZF so we can write \( f: X \rightarrow Y \) instead of \( f \in X \rightarrow Y \). 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 \( f: X \rightarrow Y \) then \( f \subseteq X \times Y \). 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) \( \equiv \forall A \in T. f-(A) \in S \)

A definition that allows to get the first factor of the domain of a binary function \( f: X \times Y \rightarrow Z \).

**Definition**

fstdom(f) \( \equiv \text{domain(domain(f))} \)
If a function maps $A$ into another set, then $A$ is the domain of the function.

```isar
define func1_1_L1: assumes f:A→C shows 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`.

```isar
define fun_is_fun: assumes f:X→Y shows function(f)
  using assms fun_is_function by simp
```

A lemma explains what `fstdom` is for.

```isar
define fstdomdef: assumes A1: f:X×Y→Z and A2: Y≠0
  shows fstdom(f) = X
proof -
  from A1 have domain(f) = X×Y using func1_1_L1
    by simp
  with A2 show fstdom(f) = X unfolding fstdom_def by auto
qed
```

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

```isar
define func1_1_L1A: assumes A1: f:X→Y and A2: ∀ x∈X. f(x) ∈ Z
  shows f:X→Z
proof -
  { fix x assume x∈X
    with A2 have f(x) ∈ Z by simp }
  with A1 show f:X→Z by (rule Pi_type)
qed
```

A variant of `func1_1_L1A`.

```isar
define func1_1_L1B: assumes A1: f:X→Y and A2: Y⊆Z
  shows f:X→Z
proof -
  from A1 A2 have ∀ x∈X. f(x) ∈ Z
    using apply_funtype by auto
  with A1 show f:X→Z using func1_1_L1A by blast
qed
```

There is a value for each argument.

```isar
define func1_1_L2: assumes A1: f:X→Y and x∈X
  shows ∃ y∈Y. (x,y) ∈ f
proof-
  from A1 have f(x) ∈ Y using apply_type by simp
  moreover from A1 have (x,f(x))∈ 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 have \( \forall x. (x \in f^{-1}(D)) \rightarrow (x \in X) \) 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 \( f : X \rightarrow Y \) 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 \( \langle p \in f \rangle \) have \( x \in X \land f(x) \in Y \)
      using func1_1_L5 by blast
  qed
with I show \( p \in X \times Y \) by auto
qed

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

**lemma** func1_1_L5A:
assumes A1: \( f:X \to Y \text{ } x \in X \text{ } y = f(x) \)
shows \( (x,y) \in f \text{ } y \in \text{range}(f) \)
proof -
from A1 show \( (x,y) \in f \) using apply_Pair by simp
then show \( y \in \text{range}(f) \) 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: \( f:X \to Y \)
shows \( f = \{ \langle x, f(x) \rangle . x \in X \} \)
proof
from A1 show \( \{ \langle x, f(x) \rangle . x \in X \} \subseteq f \) using func1_1_L5A
by auto
next
\{ fix \( p \) assume \( p \in f \)
with A1 have \( p \in X \times Y \) using fun_subset_prod
by auto
with A1 \( \langle p \rangle \in f \) have \( p \in \{ \langle x, f(x) \rangle . x \in X \} \)
using apply_equality by auto
\} thus \( f \subseteq \{ \langle x, f(x) \rangle . x \in X \} \) by auto
qed

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

**lemma** func1_1_L5B:
assumes A1: \( f:X \to Y \)
shows \( \text{range}(f) \subseteq Y \)
proof
fix y assume \( y \in \text{range}(f) \)
then obtain \( x \) where \( (x,y) \in f \)
using range_def converse_def domain_def by auto
with A1 show \( y \in Y \) using func1_1_L5B by blast
qed

The image of any set is contained in the range.

**lemma** func1_1_L6: assumes A1: \( f:X \to Y \)
shows \( f(B) \subseteq \text{range}(f) \) and \( f(B) \subseteq Y \)
proof -
show \( f(B) \subseteq \text{range}(f) \) using image_iff rangeI by auto
with A1 show \( f(B) \subseteq Y \) using func1_1_L5B by blast
qed

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

**lemma** func1_1_L6A: assumes A1: \( f:X \to Y \)
shows \( f^{-1}(A) \subseteq X \)
proof
fix x
assume A2: \( x \in f^{-1}(A) \) then obtain y where \( (x, y) \in f \)
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: \( \exists ! y. \ (y \in Y \wedge (x, y) \in f) \)
shows \( \exists ! y. \ (x, y) \in f \)
proof
from A2 show \( \exists y. \ (x, y) \in f \) by auto
fix y n assume \( (x, y) \in f \) and \( (x, n) \in f \)
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 \( \forall x \in X. \ \exists ! y. \ y \in Y \wedge (x, y) \in f \)
s 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 func.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 \( \{ \langle x, y \rangle \in X \times Y. \ b(x) = y \} : X \to Y \) 
proof - 
  let \( f = \{ \langle x, y \rangle \in X \times Y. \ b(x) = y \} \) 
  have \( f \subseteq X \times Y \) by auto 
moreover have \( \forall x \in X. \ \exists ! y. \ y \in Y \land \langle x, y \rangle \in f \) 
proof 
    fix \( x \) 
    assume A2: \( x \in X \) 
    have \exists ! y. \ y \in Y \land \langle x, y \rangle \in f 
    proof 
      from A1 A2 
      show \exists y. \ y \in Y \land \langle x, y \rangle \in f 
      by simp 
    qed 
  qed 
ultimately show \( \{ \langle x, y \rangle \in X \times Y. \ b(x) = y \} : X \to Y \) 
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 \( \{ \langle x, b(x) \rangle. \ x \in X \} : X \to Y \) 
proof - 
  let \( f = \{ \langle x, b(x) \rangle . \ x \in X \} \) 
  \{ fix \( x \) assume A2: \( x \in X \) 
  have \exists ! y. \ y \in Y \land \langle x, y \rangle \in f \} 
  proof 
    from A1 A2 
    show \exists y. \ y \in Y \land \langle x, y \rangle \in f 
    by simp 
  qed 
  qed 
ultimately show \( \{ \langle x, y \rangle \in X \times Y. \ b(x) = y \} : X \to Y \) 
using func1_1_L11 by simp 
qed 

The value of a function defined by a meta-function is this meta-function.

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

proof -
from A1 have \( \langle x, f(x) \rangle \in f \) 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 A1: \( f: X \rightarrow Y \) \( x \in X \)
and A2: \( f = \{ \langle x, b(x) \rangle . x \in X \} \)
shows \( f(x) = b(x) \)
proof -
from A1 have \( \langle x, f(x) \rangle \in f \) using apply_iff by simp
with A2 show thesis 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 \( f = \{ \langle x, b(x) \rangle . x \in X \} \)
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 \( f = \{ \langle x, g(x) \rangle . x \in X \} \)
shows \( f: X \rightarrow \text{range}(f) \)
proof -
let \( f = \{ \langle x, b(x) \rangle . x \in X \} \)
have \( \forall x \in X. g(x) \in \text{range}(\{ \langle x, g(x) \rangle . x \in X \}) \) unfolding range_def
by auto
then have \( \{ \langle x, g(x) \rangle . x \in X \} : X \rightarrow \text{range}(\{ \langle x, g(x) \rangle . x \in X \}) \) 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 \( \{ \langle x, b(x) \rangle . x \in X \}(x) = b(x) \)
proof -
let \( f = \{ \langle x, b(x) \rangle . x \in X \} \)
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

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

**lemma func1_1_L11C:**
assumes A1: \( f: X \rightarrow Y \) and A2: \( \forall x \in A. b(x) \in B \)
and A3: \( X \cap A = 0 \) and Dg: \( g = f \cup \{ \langle x, b(x) \rangle . x \in A \} \)
shows \( g : X \cup A \rightarrow Y \cup B \)
\( \forall x \in X. g(x) = f(x) \)
∀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 = 0
  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→Y and A2: a∉X
  and Dg: g = f ∪ {(a,b)}
  shows
  g : X∪{a} → Y∪{b}
  ∀x∈X. g(x) = f(x)
  g(a) = b
proof -
  let h = {(a,b)}
  from A1 A2 Dg have I:
    f:X→Y ∀x∈{a}. b∈{b} X∩{a} = 0  g = f ∪ {(x,b). x∈{a}}
    by auto
  then show g : X∪{a} → Y∪{b}
    by (rule func1_1_L11C)
from I show ∀x∈X. g(x) = f(x)
  by (rule func1_1_L11C)
from I have ∀x∈{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: \( X \cap A = 0 \) and A4: \( a \notin X \cup A \)

and Dg: \( g = f \cup \{(x,b(x))\} \cup \{(a,c)\} \)

shows
- \( g: X \cup A \rightarrow Y \cup B \) 
- \( \forall x \in X. \ g(x) = f(x) \)
- \( \forall x \in A. \ g(x) = b(x) \)
- \( g(a) = c \)

**proof** -

- let \( h = f \cup \{(x,b(x))\}. x \in A \}

from assms show \( g: X \cup A \rightarrow Y \cup B \) 

using func1_1_L11C func1_1_L11D by simp

from A1 A2 A3 have I:
- \( f: X \rightarrow Y \) \( \forall x \in A. \ b(x) \in B \) \( X \cap A = 0 \) \( 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: \( \forall x \in X \cup A. \ g(x) = h(x) \) 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 \( \forall x \in X. \ h(x) = b(x) \) by (rule func1_1_L11C)

with III show \( \forall x \in A. \ g(x) = b(x) \) by simp

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 \( \forall x \in A \cap B. \ h(x) \in Y \) \( \forall x \in A-B. \ f(x) \in Y \) \( \forall x \in B-A. \ g(x) \in Y \)

shows \( \{(x,\text{if } x \in A-B \text{ then } f(x) \text{ else if } x \in B-A \text{ then } g(x) \text{ else } h(x)\} \). x \in A \cup B \)

**proof** -

- let \( F = \{(x,\text{if } x \in A-B \text{ then } f(x) \text{ else if } x \in B-A \text{ then } g(x) \text{ else } h(x)\}. x \in A \cup B \}

from assms have \( \forall x \in A \cup B. \ (\text{if } x \in A-B \text{ then } f(x) \text{ else if } x \in B-A \text{ then } g(x) \text{ else } h(x)) \in Y \)

by auto

then show thesis by (rule ZF_fun_from_total)

qed

Inverse image of intersection is the intersection of inverse images.

**lemma** invim_inter_inter_invim: assumes \( f: X \rightarrow Y \)
The inverse image of an intersection of a nonempty collection of sets is the intersection of the inverse images. This generalizes invim_inter_inter_invim which is proven for the case of two sets.

**lemma** func1_1_L12:
assumes A1: B ⊆ Pow(Y) and A2: B ≠ 0 and A3: f:X→Y
shows f-(∩B) = (∩U∈B. f-(U))
**proof**
from A2 show f-(∩B) ⊆ (∩U∈B. f-(U)) by blast
show (∩U∈B. f-(U)) ⊆ f-(∩B)
**proof**
fix x assume A4: x ∈ (∩U∈B. f-(U))
from A3 have ∀U∈B. f-(U) ⊆ X using func1_1_L6A by simp
with A4 have ∀U∈B. x∈X by auto
with A2 have x∈X by auto
with A3 have ∃!y. (x,y) ∈ f using Pi_iff_old by simp
with A2 A4 show x ∈ f-(∩B) using vimage_iff by blast
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→Y
shows f-(A ∩ f(X)) = f-(A)
using assms invim_inter_inter_invim inv_im_dom func1_1_L6A
by blast

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) ≠ 0 shows A≠0
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)≠0 shows A≠0
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 $\forall x \in X. f(x) = g(x)$

shows $f = g$ using assms fun_extension_iff by simp

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 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 $f^{-1}(A) = \{x \in X. f(x) \in A\}$
proof -
  have $f^{-1}(A) = (\bigcup y \in A . f^{-1}(y))$
    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 $y \in \{f(x). x \in A\}$
    qed
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 \in A \rangle \) show \( y \in f(A) \) using image_iff by auto

qed

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

**Lemma image_id_same**: assumes \( A \subseteq X \) shows \( \text{id}(X)(A) = A \)

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 \( A \subseteq X \) shows \( \text{id}(X)^{-}(A) = A \)

using assms id_type id_conv by auto

What is the image of a singleton?

**Lemma singleton_image**: assumes \( f: X \rightarrow Y \) and \( x \in X \) 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 \( f: X \rightarrow Y \) \( x \in A \subseteq X \) shows \( f(x) \in f(A) \)

using assms func_imagedef by auto

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

**Lemma range_image_domain**: assumes \( A1: f: X \rightarrow Y \) shows \( f(X) = \text{range}(f) \)

proof

show \( f(X) \subseteq \text{range}(f) \) using image_def by auto

\{ fix \( y \) assume \( y \in \text{range}(f) \)
then obtain \( x \) where \( \langle y, x \rangle \in \text{converse}(f) \) by auto
with \( A1 \) have \( x \in X \) using func1_1_L5 by blast
with \( A1 \) have \( f(x) \in f(X) \) using func_imagedef by auto
with \( A1 \) \langle \in \text{converse}(f) \rangle \) have \( y \in f(X) \)
using apply_equality by auto
\}
then show \( \text{range}(f) \subseteq f(X) \) by auto

qed

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

**Lemma diff_image_diff**: assumes \( A1: f: X \rightarrow Y \) and \( A2: A \subseteq X \) shows \( f(X) - f(A) \subseteq f(X-A) \)
proof
  fix y assume y ∈ f(X) - f(A)
  hence y ∈ f(X) and I: y ∉ f(A) by auto
  with A1 obtain x where x∈X and II: y = f(x)
      using func_imagedef by auto
  with A1 A2 I have x∉A
    using func1_1_L15D by auto
  with <x∈X> have x ∈ X-A X-A ⊆ X by auto
  with A1 II show y ∈ f(X-A)
    using func1_1_L15D by simp
qed

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

lemma image_of_Inter: assumes A1: f:X→Y and
  A2: I≠0 and A3: ∀i∈I. P(i) ⊆ X
shows f(⋂i∈I. P(i)) ⊆ ( ⋂i∈I. f(P(i)) )
proof
  fix y assume A4: y ∈ f(⋂i∈I. P(i))
  from A1 A2 A3 have f(⋂i∈I. P(i)) = {f(x). x ∈ ( ⋂i∈I. P(i) )}
    using ZF1_1_L7 func_imagedef by simp
  with A4 obtain x where x ∈ ( ⋂i∈I. P(i) ) and y = f(x)
    by auto
  with A1 A2 A3 show y ∈ ( ⋂i∈I. f(P(i)) ) using func_imagedef
    by auto
qed

The image of union is the union of images.

lemma image_of_Union: assumes A1: f:X→Y and
  A2: ∀A∈M. A ⊆ X
shows f(⋃M) = ⋃{f(A). A∈M}
proof
  from A2 have ⋃M ⊆ X by auto
  { fix y assume y ∈ f(⋃M)
    with A1 ⋃M ⊆ X> obtain x where x∈⋃M and I: y = f(x)
      using func_imagedef by auto
    then obtain A where A∈M and x∈A by auto
    with assms I have y ∈ ⋃{f(A). A∈M} using func_imagedef by auto
  } thus f(⋃M) ⊆ ⋃{f(A). A∈M} by auto
  { fix y assume y ∈ ⋃{f(A). A∈M}
    then obtain A where A∈M and y ∈ f(A) by auto
    with assms ⋃M ⊆ X> have y ∈ f(⋃M) using func_imagedef by auto
  } thus ⋃{f(A). A∈M} ⊆ f(⋃M) by auto
qed

The image of a nonempty subset of domain is nonempty.

lemma func1_1_L15A:
  assumes A1: f: X→Y and A2: A⊆X and A3: A≠0
shows f(A) ≠ 0
proof -
  from A3 obtain x where x∈A by auto

80
with \( A_1, A_2 \) have \( f(x) \in f(A) \)
    using func_imagedef by auto
then show \( f(A) \neq 0 \) 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 \( A_1 : f : X \to Y \) and \( A_2 : g : Y \to Z \)
and \( A_3 : A \subseteq X \)
shows \( g(f(A)) = \{ g(f(x)). \ x \in A \} \)
\( g(f(A)) = (g \circ f)(A) \)
proof -
from \( A_1, A_3 \) have \( \{ f(x). \ x \in A \} \subseteq Y \)
    using apply_funtype by auto
with \( A_2 \) have \( g(f(x)). \ x \in A \} \)
    using func_imagedef by auto
with \( A_1, A_3 \) show I: \( g(f(A)) = \{ g(f(x)). \ x \in A \} \)
    using func_imagedef by simp
from \( A_1, A_3 \) 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 \( A_1, A_2, A_3 \) 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 \( A_1 : f \in X \to Y \) and \( A_2 : \forall x \in A. \ b(x) \in X \)
shows \( f(\{ b(x). \ x \in A \}) = \{ f(b(x)). \ x \in A \} \)
proof -
from \( A_2 \) have \( \{ b(x). \ x \in A \} \subseteq X \) by auto
with \( A_1 \) show thesis using func_imagedef by auto
qed

What are the values of composition of three functions?

**lemma func1_1_L18:** assumes \( A_1 : f : A \to B \) g:B\to C h:C\to D \)
and \( A_2 : x \in A \)
shows
\[(h \circ g \circ f)(x) \in D\]
\[(h \circ g \circ f)(x) = h(g(f(x)))\]

**proof**
- from A1 have \((h \circ g \circ f) : A \to D\)
  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.

**lemma** `comp_fun_subset`
- assumes A1: \(g : A \to B\) and A2: \(f : C \to D\) and A3: \(B \subseteq C\)
- shows \(f \circ g : A \to D\)

**proof**
- from A1 A3 have \(g : A \to C\) by (rule `func1_1_L1B`)
- with A2 show \(f \circ g : A \to D\) 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: \(g : B \to A\) and A2: \(f : A \to C\)
- shows \((\forall y \in B. f(g(y)) = y) \iff f \circ g = id(B)\)

**proof**
- from assms have \(f \circ g : B \to C\) and \(id(B) : B \to B\)
  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 \(\forall y \in B. (f \circ g)(y) = f(g(y))\) and \(\forall y \in B. id(B)(y) = y\)
  by auto
- ultimately show \((\forall y \in B. f(g(y)) = y) \iff f \circ g = id(B)\) by simp

**qed**

A lemma about a value of a function that is a union of some collection of functions.

**lemma** `fun_Union_apply`
- assumes A1: \(\bigcup F : X \to Y\) and
  A2: \(f \in F\) and A3: \(f : A \to B\) 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 \((x, f(x)) \in F\) by auto
- with A1 show \((\bigcup F)(x) = f(x)\) using `apply_equality`
8.2 Functions restricted to a set

Standard Isabelle/ZF defines the notion $\text{restrict}(f,A)$ 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 $\text{restrict}(f,A)$ 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 $A1: f:X \to Y$ and $A2: B \subseteq X$ shows $\text{restrict}(f,B)-\{(A) = f-\{(A) \cap B$ proof - let $g = \text{restrict}(f,B)$ from $A1$ $A2$ have $g:B \to Y$ 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 $A1: f:X \to Y$ and $A2: g: A \to Z$ 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 $A1: f:X \to Y$ and $A2: g:A \to Z$ and $A3: A \subseteq X$ and $A4: f \cap A \times Z = g$ shows $g = \text{restrict}(f,A)$ proof from $A4$ show $g \subseteq \text{restrict}(f,A)$ using restrict_iff by auto show $\text{restrict}(f,A) \subseteq g$ proof fix $z$ assume $A5: z \in \text{restrict}(f,A)$ then obtain $x \ y$ where $D1: z \in f \land x \in A \land z = \langle x, y \rangle$ using restrict_iff by auto with $A1$ have $y = f(x)$ using apply_iff by auto with $A1$ $A2$ $A3$ $A4$ $D1$ have $y = g(x)$ using func1_1_L15 restrict_if by auto

83
with A2 D1 show \( z \in g \) using apply_Pair by simp
qed

Which function space a restricted function belongs to?

**lemma func1_2_L4:**
assumes A1: \( f : X \to Y \) and A2: \( A \subseteq X \) and A3: \( \forall x \in A. \ f(x) \in Z \)
sows restrict\( (f,A) : A \to Z \)
proof -
let \( g = \text{restrict}(f,A) \)
from A1 A2 have \( g : A \to Y \)
using restrict_type2 by simp
moreover {
fix \( x \) assume \( x \in A \)
with A1 A3 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 A1: \( f : X \to Y \) and A2: \( A \subseteq X \)
sows restrict\( (f,A) : A \to Y \)
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 composition of two functions is the same as composition with a restriction.

**lemma comp_restrict:**
assumes A1: \( f : A \to B \) and A2: \( g : X \to C \) and A3: \( B \subseteq X \)
sows \( g \circ f = \text{restrict}(g,B) \circ f \)
proof -
from assms have \( g \circ f : A \to C \) using comp_fun_subset
by simp
moreover from assms have \( \text{restrict}(g,B) \circ f : A \to C \)
using restrict_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 comp_fun_apply apply_funtype restrict
by simp
ultimately show \( g \circ f = \text{restrict}(g,B) \circ f \)
by (rule 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 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 constant (= c) functions on a set X in a natural way as ConstantFunction(X, c).

definition
ConstantFunction(X, c) ≡ X×{c}

Constant function belongs to the function space.

lemma func1_3_L1:
  assumes A1: c∈Y
  shows ConstantFunction(X, c) : X→Y
proof -
  from A1 have X×{c} = {(x,y) ∈ X×Y. c = y}
    by auto
  with A1 show thesis using func1_1_L11A
    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 (x,c) ∈ ConstantFunction(X, c)
    using ConstantFunction_def by simp
  ultimately show thesis using apply_iff 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 a difference of two sets is the difference of images

lemma inj_image_dif:
  assumes A1: f ∈ inj(A,B) and A2: C ⊆ A
  shows f(A-C) = f(A) - f(C)
proof
  show f(A - C) ⊆ f(A) - f(C)
    proof
      fix y assume A3: y ∈ f(A - C)
      from A1 have f:A→B using inj_def by simp
      moreover have A-C ⊆ A by auto
      ultimately have f(A-C) = {f(x). x ∈ A-C}
        using func_imagedef by simp
      with A3 obtain x where I: f(x) = y and x ∈ A-C

85
by auto
hence \( x \in A \) by auto
with \( \langle f: A \to B \rangle \) I have \( y \in f(A) \)
using \texttt{func_imagedef} by auto
moreover have \( y \notin f(C) \)
proof -
\{ assume \( y \in f(C) \)
with \( A2 <f:A \to B> \) obtain \( x_0 \)
where \( II: f(x_0) = y \) and \( x_0 \in C \)
using \texttt{func_imagedef} by auto
with \( A1 A2 I <x \in A> \) have
\( f \in \text{inj}(A,B) \) \( f(x) = f(x_0) \) \( x \in A \) \( x_0 \in A \)
by auto
then have \( x = x_0 \) by (rule \texttt{inj_apply_equality})
with \( <x \in A-C, x_0 \in C> \) have \( \text{False} \) by simp
\} thus thesis by auto
qed
ultimately show \( y \in f(A) - f(C) \)
by simp
qed
from \( A1 A2 \) show \( f(A) - f(C) \subseteq f(A - C) \)
using \texttt{inj_def} \texttt{diff_image_diff} by auto
qed
For injections the image of intersection is the intersection of images.

\begin{lemma}
\texttt{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
show \( f(A \cap B) \subseteq f(A) \cap f(B) \) using \texttt{image_Int_subset} by simp
\{ from \( A1 \) have \( f:X \to Y \) using \texttt{inj_def} by simp
fix \( y \) assume \( y \in f(A) \cap f(B) \)
then have \( y \in f(A) \) and \( y \in f(B) \) by auto
with \( A2 <f:X \to Y> \) 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 \texttt{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 \texttt{inj_def} by auto
with \( <x_A \in A, x_B \in B> \) have \( f(x_A) \in \{ f(x) \mid x \in A \cap B \} \) by auto
moreover from \( A2 <f:X \to Y> \) have \( f(A \cap B) = \{ f(x) \mid x \in A \cap B \} \)
using \texttt{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
\end{lemma}

For surjection from \( A \) to \( B \) the image of the domain is \( B \).

\begin{lemma}
\texttt{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

For injections the inverse image of an image is the same set.

lemma inj_vimage_image: assumes f ∈ inj(X,Y) and A ⊆ X
  shows f-(f(A)) = A
proof -
  have f-(f(A)) = (converse(f) O 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 ∈ surj(X,Y) and A2: A ⊆ Y
  shows f(f-(A)) = A
proof -
  have f(f-(A)) = (f O 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 ∈ surj(X,Y) and A2: B ⊆ Pow(Y)
  shows { f(U). U ∈ {f-(V). V ∈ B} } = B
proof
  { fix W assume W ∈ { f(U). U ∈ {f-(V). V ∈ B} }  
    then obtain U where I: U ∈ {f-(V). V ∈ B} and
                       II: W = f(U) by auto
    then obtain V where V ∈ B and U = f-(V) by auto
    with II have W = f(f-(V)) by simp
    moreover from assms <V ∈ B> have f ∈ surj(X,Y) and V ∈ Y by auto
    ultimately have W = V using surj_image_vimage by simp
    with <V ∈ B> have W ∈ B by simp
  } thus { f(U). U ∈ {f-(V). V ∈ B} } ⊆ B by auto
  { fix W assume W ∈ B
    let U = f-(W)
    from <W ∈ B> have U ∈ {f-(V). V ∈ B} by auto
    moreover from A1 A2 <W ∈ B> have W = f(U) using surj_image_vimage
      by auto
    ultimately have W ∈ { f(U). U ∈ {f-(V). V ∈ B} } by auto
  } thus B ⊆ { f(U). U ∈ {f-(V). V ∈ 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 $\text{restrict}(f, A-\{a\}) \in \text{bij}(A-\{a\}, B-\{f(a)\})$
proof -
  let $C = A-\{a\}$
  from $A1$ have $f \in \text{inj}(A, B)$ $C \subseteq A$
    using bij_def by auto
  then have $\text{restrict}(f, C) \in \text{bij}(C, f(C))$
    using restrict_bij by simp
  moreover have $f(C) = B-\{f(a)\}$
    proof -
      from $A2$ have $f : A \rightarrow B$
        using bij_is_fun by simp
      moreover from $A1$ have $f(A) = B$
        using bij_is_fun surj_range_image_domain by auto
      moreover from $A1$ $A2$ have $f(\{a\}) = \{f(a)\}$
        using bij_is_fun singleton_image by blast
      ultimately show $f(C) = B-\{f(a)\}$ by simp
      qed
  qed
ultimately show 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 bij_is_fun by simp
  then show $\text{domain}(f) = X$ using 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 $I : f : A \rightarrow B$ using inj_is_fun by simp
  with $A2$ $A3$ obtain $x$ where $II: x \in C$ $y = f(x)$
    using func_imagedef by auto
  with $A1$ $A2$ show $\text{converse}(f)(y) \in C$ using left_inverse by auto
  from $A1$ $A2$ $II$ show $f(\text{converse}(f)(y)) = y$
    using func1_1_L5A right_inverse by auto
qed

For injections if a value at a point belongs to the image of a set, then the point belongs to the set.

lemma 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 inj_inv_back_in_set by simp
moreover from A1 A3 have \( \text{converse}(f)(f(x)) = x \)
  using left_inverse_eq by simp
ultimately show \( x \in C \) by simp
qed

For injections the image of intersection is the intersection of images.

**lemma inj_image_of_Inter:**
assumes A1: \( f \in \text{inj}(A,B) \) and A2: \( I \neq 0 \) 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 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 \( x = \text{converse}(f)(y) \)
    from A2 obtain \( i_0 \) where \( i_0 \in I \) by auto
    with A1 A4 have II: \( y \in \text{range}(f) \) using inj_is_fun func1_1_L6 by auto
    with A1 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: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 \( f \in \text{inj}(A,B) \)
shows \( f \in \text{inj}(A,\text{range}(f)) \)
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 $f \in \text{inj}(A,B)$
  shows $f \in \text{bij}(A,\text{range}(f))$
proof -
  from assms have $f \in \text{surj}(A,\text{range}(f))$ 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 $A1: f \in \text{surj}(X,Y)$ and $A2: a \not\in X$ and $A3: g = f \cup \{(a,b)\}$
  shows $g \in \text{surj}(X \cup \{a\}, Y \cup \{b\})$
proof -
  from $A1$ $A2$ $A3$ have $g : X \cup \{a\} \rightarrow Y \cup \{b\}$
    using surj_def func1_1_L11D by simp
  moreover have $\forall y \in Y \cup \{b\}. \exists x \in X \cup \{a\}. y = g(x)$
     proof
       fix $y$ assume $y \in Y \cup \{b\}$
       then have $y \in Y \cup y = b$ 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
               then have $\exists x \in X \cup \{a\}. y = g(x)$ by auto }
         } ultimately show $\exists x \in X \cup \{a\}. y = g(x)$ by auto
     qed
  qed 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.

lemma inj_extend_point: assumes $f \in \text{inj}(X,Y)$ a\not\in X b\not\in Y
  shows $(f \cup \{(a,b)\}) \in \text{inj}(X \cup \{a\}, Y \cup \{b\})$
proof -
  from assms have $\text{cons}((a,b),f) \in \text{inj}(\text{cons}(a, X), \text{cons}(b, Y))$
    using assms inj_extend by simp
  moreover have $\text{cons}((a,b),f) = f \cup \{(a,b)\}$ and
    $\text{cons}(a, X) = X \cup \{a\}$ and $\text{cons}(b, Y) = Y \cup \{b\}$
    by auto

90
ultimately show thesis by simp
qed

A lemma about extending a bijection by one point.

lemma bij_extend_point: assumes f ∈ bij(X,Y) a ∉ X b ∉ Y
shows (f ∪ {(a,b)}) ∈ bij(X∪{a},Y∪{b})
using assms surj_extend_point inj_extend_point bij_def
by simp

A quite general form of the $a^{-1}b = 1$ implies $a = b$ law.

lemma comp_inv_id_eq:
assumes A1: converse(b) 0 a = id(A) and
A2: a ⊆ A×B b ∈ surj(A,B)
shows a = b
proof -
  from A1 have (b 0 converse(b)) 0 a = b 0 id(A)
    using comp_assoc by simp
  with A2 have id(B) 0 a = b 0 id(A)
    using right_comp_inverse by simp
  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 a = b by (rule comp_inv_id_eq)
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: \( a \in \text{bij}(A,B) \) \( b \in \text{bij}(B,A) \) and A2: \( b \circ a = \text{id}(A) \)

**shows** \( a = \text{converse}(b) \) and \( b = \text{converse}(a) \)

**proof** -

from A1 have \( a \in \text{bij}(A,B) \) and \( \text{converse}(b) \in \text{bij}(A,B) \) using bij_converse_bij

by auto

moreover from assms have \( \text{converse}(\text{converse}(b)) \circ a = \text{id}(A) \)

using bij_converse_converse by simp

ultimately show \( a = \text{converse}(b) \) by (rule comp_inv_id_eq_bij)

with assms show \( b = \text{converse}(a) \) 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: \( b : B \rightarrow A \) and A3: \( \forall x \in A. \ b(a(x)) = x \)

**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 \( \forall x \in A \) have \( y \in B \) 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 \( \forall w \in B. \forall y \in B. \ b(w) = b(y) \rightarrow w = y \)

**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 \( \forall w \in B \) 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 \( \forall a \in \text{surj}(A,B) \) \( \forall y \in B \) obtain \( x_y \) where

\( x_y \in A \) and III: \( a(x_y) = y \)

using surj_def by auto

moreover from A3 \( \forall x_w \in A \) \( \forall x_y \in A \) have \( b(a(x_w)) = x_w \) and

\( b(a(x_y)) = x_y \)

92
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 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 of 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 \( f_1:X_1 \rightarrow Y_1 \) \( f_2:X_2 \rightarrow Y_2 \) and

\( g = \{(p,\langle f_1(\text{fst}(p)),f_2(\text{snd}(p))\rangle). \ p \in X_1 \times X_2\} \)

shows \( g:X_1 \times X_2 \rightarrow Y_1 \times Y_2 \) 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 \( f:X_1 \rightarrow X_2 \) \( g:X_3 \rightarrow X_4 \)

shows \( \{(x,y),(fx,gy)\). \ (x,y)\in X_1 \times X_3 \rightarrow X_1 \times X_2 \times X_4 \)

proof -
have \{⟨⟨x,y⟩,⟨fx,gy⟩⟩. ⟨x,y⟩∈X_1×X_3\} = \{p,⟨f(fst(p)),g(snd(p))⟩⟩. p ∈ X_1×X_3\}
by auto
with asms show thesis using cart_prod_fun by simp
qed

Product of two surjections is a surjection.

theorem prod_functions_surj:
assumes f∈surj(A,B) g∈surj(C,D)
sows \{⟨⟨a1,a2⟩,⟨fa1,ga2⟩⟩. ⟨a1,a2⟩∈A×C\} ∈ surj(A×C,B×D)
proof -
let h = \{⟨⟨x, y⟩. f(x), g(y)⟩⟩. ⟨x,y⟩ ∈ A×C\}
from asms have fun: f:A→Bg:C→D unfolding surj_def by auto
then have pfun: h : A×C→B×D using prod_fun by auto
{fix b assume b∈B×D
then obtain b1 b2 where b=(b1,b2) b1∈B b2∈D by auto
with asms obtain a1 a2 where f(a1)=b1 g(a2)=b2 a1∈A a2∈C
unfolding surj_def by blast
hence ⟨⟨a1,a2⟩,⟨b1,b2⟩⟩ ∈ h by auto
with pfun have h(a1,a2)=(b1,b2) using apply_equality by auto
with <b=(b1,b2)> <a1∈A> <a2∈C> have ∃a∈A×C. h(a)=b
by auto
} hence ∀b∈B×D. ∃a∈A×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 f1:X_1→Y_1  f2:X_2→Y_2 and
g = \{⟨p,⟨f1(fst(p)),f2(snd(p))⟩⟩. p ∈ X_1×X_2\}
sows g−(A_1×A_2) = f_1−(A_1) × f_2−(A_2)
proof -
from asms have g: X_1×X_2 → Y_1×Y_2 using cart_prod_fun
by simp
then have g−(A_1×A_2) = \{p ∈ X_1×X_2. g(p) ∈ A_1×A_2\} using func1_1_L15
by simp
with asms <g: X_1×X_2 → Y_1×Y_2> show g−(A_1×A_2) = f_1−(A_1) × f_2−(A_2)
using ZF_fun_from_tot_val func1_1_L15 by auto
qed

For a function of two variables defined on X×Y, if we fix an x ∈ X we obtain a function on Y. Note that if domain(f) is X×Y, range(domain(f)) extracts Y from X×Y.

definition Fix1stVar(f,x) ≡ \{(y,f(x,y)). y ∈ range(domain(f))\}
For every $y \in Y$ we can fix the second variable in a binary function $f : X \times Y \to Z$ to get a function on $X$.

definition
$\text{Fix2ndVar}(f,y) \equiv \{ (x,f(x,y)) : x \in \text{domain}(\text{domain}(f)) \}$

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 $\text{fix_var_fun_domain}$: assumes $A1: f : X \times Y \to Z$ shows $x \in X \to \text{Fix1stVar}(f,x) = \{ (y,f(x,y)) : y \in Y \}$ $y \in Y \to \text{Fix2ndVar}(f,y) = \{ (x,f(x,y)) : x \in X \}$ proof -
from $A1$ have $I: \text{domain}(f) = X \times Y$ using $\text{func1_1_L1}$ by simp 
{ assume $x \in X$
  with $I$ have $\text{range}(\text{domain}(f)) = Y$ by auto 
  then have $\text{Fix1stVar}(f,x) = \{ (y,f(x,y)) : y \in Y \}$ using $\text{Fix1stVar_def}$ by simp 
} then show $x \in X \to \text{Fix1stVar}(f,x) = \{ (y,f(x,y)) : y \in Y \}$ 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 $y \in Y \to \text{Fix2ndVar}(f,y) = \{ (x,f(x,y)) : x \in X \}$ by simp qed

If we fix the first variable, we get a function of the second variable.

lemma $\text{fix_1st_var_fun}$: assumes $A1: f : X \times Y \to Z$ and $A2: x \in X$ shows $\text{Fix1stVar}(f,x) : Y \to Z$ proof -
from $A1$ $A2$ have $\forall y \in Y. f(x,y) \in Z$ using $\text{apply_funtype}$ by simp 
then have $\{ (y,f(x,y)) : y \in Y \} : Y \to Z$ using $\text{ZF_fun_from_total}$ by simp 
with $A1$ $A2$ show $\text{Fix1stVar}(f,x) : Y \to Z$ 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 $A1: f : X \times Y \to Z$ and $A2: y \in Y$ shows $\text{Fix2ndVar}(f,y) : X \to Z$ proof -
from $A1$ $A2$ have $\forall x \in X. f(x,y) \in Z$ using $\text{apply_funtype}$ by simp 
then have $\{ (x,f(x,y)) : x \in X \} : X \to Z$ using $\text{ZF_fun_from_total}$ by simp 
with $A1$ $A2$ show $\text{Fix2ndVar}(f,y) : X \to Z$
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 fix_var_val:**

assumes \( A1: f : X \times Y \rightarrow Z \) and \( A2: x \in X \ y \in Y \)

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 fix_var_fun_domain by auto
  moreover from \( A1 \ A2 \) have
    \[
    \text{Fix1stVar}(f, x) : Y \rightarrow Z
    \]
    \[
    \text{Fix2ndVar}(f, y) : X \rightarrow Z
    \]
    using fix_1st_var_fun fix_2nd_var_fun by auto
  ultimately have \( f_1 : Y \rightarrow Z \) and \( f_2 : X \rightarrow Z \)
  by auto
  with \( A2 \) have \( f_1(y) = f(x, y) \) and \( f_2(x) = f(x, y) \)
  using ZF_fun_from_tot_val by auto
  with I show
    \[
    \text{Fix1stVar}(f, x)(y) = f(x, y)
    \]
    \[
    \text{Fix2ndVar}(f, y)(x) = f(x, y)
    \]
  by auto

qed

Fixing the second variable commutes with restrictig the domain.

**lemma fix_2nd_var_restr_comm:**

assumes \( A1: f : X \times Y \rightarrow Z \) and \( A2: y \in Y \) and \( A3: X_1 \subseteq X \)

shows \( \text{Fix2ndVar}(\text{restrict}(f, X_1 \times Y), y) = \text{restrict}(\text{Fix2ndVar}(f, y), X_1) \)

**proof**

- let \( g = \text{Fix2ndVar}(\text{restrict}(f, X_1 \times Y), y) \)
  let \( h = \text{restrict}(\text{Fix2ndVar}(f, y), X_1) \)
  from \( A3 \) have I: \( X_1 \times Y \subseteq X \times Y \) by auto
  with \( A1 \) have II: \( \text{restrict}(f, X_1 \times Y) : X_1 \times Y \rightarrow Z \)
    using restrict_type2 by simp
  with \( A2 \) have \( g : X_1 \rightarrow Z \)
    using fix_2nd_var_fun by simp
  moreover
  from \( A1 \ A2 \) have III: \( \text{Fix2ndVar}(f, y) : X \rightarrow Z \)
    using fix_2nd_var_fun by simp
  with \( A3 \) have \( h : X_1 \rightarrow Z \)
    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 assms have Fix1stVar(f,x)-(A) = {y∈Y. Fix1stVar(f,x)(y) ∈ A}
    using fix_1st_var_fun func1_1_L15 by blast
  with assms show thesis using fix_var_val func1_1_L15 by auto
qed

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:
assumes A1: f : X×Y → Z and A2: y∈Y
shows Fix2ndVar(f,y)-(A) = {x∈X. ⟨x,y⟩ ∈ f-(A)}
proof -
  from assms have I: Fix2ndVar(f,y)-(A) = {x∈X. Fix2ndVar(f,y)(x) ∈ A}
    using fix_2nd_var_fun func1_1_L15 by blast
  with assms show thesis using fix_var_val func1_1_L15 by auto
qed

end

9 Semilattices and Lattices
theory Lattice_ZF imports Order_ZF_1a func1
begin
Lattices can be introduced in algebraic way as commutative idempotent
(x · 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 \textit{Order.ZF} as the predicate \textit{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).

\textbf{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).

\textbf{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).

\textbf{definition}
\( \text{IsAlattice} \) \( \text{(infixl \{is a lattice on\} 90)} \) where
\[ r \text{ \{is a lattice on\} } L \equiv \text{IsJoinSemilattice}(L, r) \land \text{IsMeetSemilattice}(L, r) \]

Join is a binary operation whose value on a pair \( \langle x, y \rangle \) is defined as the supremum of the set \( \{x, y\} \).

\textbf{definition}
\[ \text{Join}(L, r) \equiv \{ \langle p, \text{Supremum}(r, \{\text{fst}(p), \text{snd}(p)\}) \rangle \mid p \in L \times L \} \]

Meet is a binary operation whose value on a pair \( \langle x, y \rangle \) is defined as the infimum of the set \( \{x, y\} \).

\textbf{definition}
\[ \text{Meet}(L, r) \equiv \{ \langle p, \text{Infimum}(r, \{\text{fst}(p), \text{snd}(p)\}) \rangle \mid p \in L \times L \} \]

Linear order is a lattice.

\textbf{lemma} \text{lin_is_latt: assumes} r \subseteq L \times L \text{ and IsLinOrder}(L, r) \text{ shows} r \text{ \{is a lattice on\} } L \]
\textbf{proof -}
from \text{assms}(2) have \text{IsPartOrder}(L, r) using \text{Order.ZF}._1.L2 by simp
with \text{assms} have \text{IsMeetSemilattice}(L, r) unfolding \text{IsLinOrder_def} \text{IsMeetSemilattice_def}
using \text{inf_sup_two_el(1)} by auto
moreover from \text{assms} \langle \text{IsPartOrder}(L, r) \rangle \text{ have IsJoinSemilattice}(L, r)
unfolding \text{IsLinOrder_def} \text{IsJoinSemilattice_def} using \text{inf_sup_two_el(3)}
by auto
ultimately show \text{thesis} unfolding \text{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 have ∀p ∈ L×L. Supremum(r,{fst(p),snd(p)}) ∈ L
    unfolding IsJoinSemilattice_def IsPartOrder_def HasAsupremum_def using sup_in_space
    by auto
  then show thesis unfolding Join_def using ZF_fun_from_total by simp
qed

The value of Join(L,r) on a pair ⟨x,y⟩ is the supremum of the set {x,y}, hence it is greater or equal than both.

lemma join_val:
  assumes IsJoinSemilattice(L,r) x ∈ L y ∈ L
  defines j ≡ Join(L,r)⟨x,y⟩
  shows j ∈ L j = Supremum(r,{x,y}) ⟨x,j⟩ ∈ r ⟨y,j⟩ ∈ r
proof -
  from assms(1)
  have Join(L,r) : L×L → L
    using join_is_binop by simp
  with assms(2,3,4)
  show j = Supremum(r,{x,y}) unfolding Join_def
    using ZF_fun_from_tot_val
    by auto
  from assms(2,3,4)
  <Join(L,r) : L×L → L> show j∈L using apply_funtype
  by auto
  from assms(1,2,3)
  have r ⊆ L×L antisym(r) HasAminimum(r,⋂z∈{x,y}. r{z})
    unfolding IsJoinSemilattice_def IsPartOrder_def HasAnInfimum_def by auto
  with <j = Supremum(r,{x,y})> show ⟨x,j⟩ ∈ r and ⟨y,j⟩ ∈ 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×L → L
proof -
  from assms have ∀p ∈ L×L. Infimum(r,{fst(p),snd(p)}) ∈ 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 Meet(L,r) on a pair ⟨x,y⟩ is the infimum of the set {x,y}, hence it is less or equal than both.

lemma meet_val:
  assumes IsMeetSemilattice(L,r) x ∈ L y ∈ L
  defines m ≡ Meet(L,r)⟨x,y⟩
  shows m ∈ L m = Infimum(r,{x,y}) ⟨m,x⟩ ∈ r ⟨m,y⟩ ∈ r
proof -
  from assms(1) have Meet(L,r) : L×L → L using meet_is_binop by simp
  with assms(2,3,4) show m = Infimum(r,{x,y}) unfolding Meet_def using
  ZF_fun_from_tot_val
  by auto
  from assms(2,3,4) (Meet(L,r) : L×L → L> show m∈L using apply_funtype
  by simp
  with assms(1,2,3) have r ⊆ L×L antisym(r) HasAmaximum(r,\(\bigcap z \in \{x,y\}.
  r-{z}\))
  unfolding IsMeetSemilattice_def IsPartOrder_def HasAnInfimum_def
  by auto
  with <m = Infimum(r,{x,y})> show (m,x) ∈ r and (m,y) ∈ r
  using inf_in_space(2) by auto
qed

The next locale defines a a notation for join-semilattice. We will use the \(\sqcup\)
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 \(\sqcup\) 71)
  defines join_def [simp]: x \(\sqcup\) y ≡ Join(L,r)⟨x,y⟩
  fixes sup (sup _ )
  defines sup_def [simp]: sup A ≡ Supremum(r,A)

Join of the elements of the lattice is in the lattice.

lemma (in join_semilatt) join_props: assumes x∈L y∈L
  shows x\(\sqcup\)y ∈ L and x\(\sqcup\)y = sup {x,y}
proof -
  from joinLatt assms have Join(L,r)⟨x,y⟩ ∈ L using join_is_binop apply_funtype
  by blast
  thus x\(\sqcup\)y ∈ L by simp
  from joinLatt assms have Join(L,r)⟨x,y⟩ = Supremum(r,{x,y}) using join_val(2)
  by simp
  thus x\(\sqcup\)y = sup {x,y} by simp
qed

Join is associative.

lemma (in join_semilatt) join_assoc: assumes x∈L y∈L z∈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 ... = sup {sup {x}, sup {y,z}}
  unfolding IsJoinSemilattice_def IsPartOrder_def using join_props sup_inf_singl(2)
  by simp
by auto
also have ... = 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 \( \forall T \in T. \) HasAsupremum(r,T)
    unfolding IsJoinSemilattice_def IsPartOrder_def HasAsupremum_def
    using sup_inf_singl(1)
    by blast
  moreover from joinLatt assms have HasAsupremum(r,\{Supremum(r,T).T \in T\})
    unfolding IsJoinSemilattice_def IsPartOrder_def HasAsupremum_def
    using sup_in_space(1) sup_inf_singl(2) by auto
  ultimately have Supremum(r,\{Supremum(r,T).T \in T\}) = Supremum(r,\bigcup T)
    by (rule sup_sup)
  moreover have \( \{Supremum(r,T).T \in T\} = \{sup \{x\}, sup \{y,z\}\} \) and \( \bigcup T = \{x,y,z\} \)
    by auto
  ultimately show \( (sup \{sup \{x\}, sup \{y,z\}\}) = sup \{x,y,z\} \) by simp
  qed
also have ... = sup \{sup \{x\}, sup \{y,z\}\} by simp
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 \( \forall T \in T. \) HasAsupremum(r,T)
    unfolding IsJoinSemilattice_def IsPartOrder_def HasAsupremum_def
    using sup_inf_singl(1)
    by blast
  moreover from joinLatt assms have HasAsupremum(r,\{Supremum(r,T).T \in T\})
    unfolding IsJoinSemilattice_def IsPartOrder_def HasAsupremum_def
    using sup_in_space(1) sup_inf_singl(2) by auto
  ultimately have Supremum(r,\{Supremum(r,T).T \in T\}) = Supremum(r,\bigcup T)
    by (rule sup_sup)
  moreover have \( \{Supremum(r,T).T \in T\} = \{sup \{x,y\}, sup \{z\}\} \) and \( \bigcup T = \{x,y,z\} \)
    by auto
  ultimately show \( (sup \{sup \{x,y\}, sup \{z\}\}) = sup \{x,y,z\} \) by auto
  qed
also from assms joinLatt have ... = sup \{x\} \sqcup z by auto
    unfolding IsJoinSemilattice_def IsPartOrder_def using join_props
  also from joinLatt assms(1,2) have ... = x \sqcup y \sqcup z by simp
    finally show x \sqcup (y \sqcup z) = x \sqcup y \sqcup z by simp
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 \( \sqcap \))
defines join_def [simp]: \( x \sqcap y \equiv \text{Meet}(L,r)(x,y) \)
fixes sup (inf _)
defines sup_def [simp]: inf \( A \) \( \equiv \text{Infimum}(r,A) \)

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 \( x \sqcap y = \text{inf \{x,y\}} \)

proof -
from meetLatt assms have \( x \sqcap (y \sqcap z) = x \sqcup (\text{inf \{y,z\}}) \)
by simp
thus \( x \sqcap y = \text{inf \{x,y\}} \) 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 have \( x \sqcap (y \sqcap z) = x \sqcap (\text{inf \{y,z\}}) \)
by simp
also from assms meetLatt have \( x \sqcap (y \sqcap z) = \text{inf \{y,z\}} \)
unfolding IsMeetSemilattice_def IsPartOrder_def using meet_props sup_inf_singl(4)
by auto
also have \( ... = \text{inf \{x,y,z\}} \)
proof -
let \( \mathcal{T} = \{x\}, \{y,z\} \)
from meetLatt have \( r \subseteq L \times L \) antisym(r) trans(r)

unfolding IsMeetSemilattice_def IsPartOrder_def by auto
moreover from meetLatt assms have \( \forall T \in \mathcal{T}. \) HasAnInfimum\( (r, T) \)
unfolding IsMeetSemilattice_def IsPartOrder_def using sup_inf_singl(3)
by blast
moreover from meetLatt assms have HasAnInfimum\( (r, \{\text{Infimum}(r, T). T \in \mathcal{T}\}) \)
unfolding IsMeetSemilattice_def IsPartOrder_def HasAnInfimum_def
using inf_in_space(1) sup_inf_singl(4) by auto
ultimately have \( \text{Infimum}(r, \{\text{Infimum}(r, T). T \in \mathcal{T}\}) = \text{Infimum}(r, \bigcup \mathcal{T}) \) by (rule inf_inf)
moreover have \( \{\text{Infimum}(r, T). T \in \mathcal{T}\} = \{\inf \{x\}, \inf \{y, z\}\} \) and \( \bigcup \mathcal{T} = \{x, y, z\} \)
by auto
ultimately show \( \inf (\inf \{x\}, \inf \{y, z\}) = \inf \{x, y, z\} \) by simp
also have \( \ldots = \inf \{x, y, z\} \) by simp
qed
proof
- let \( \mathcal{T} = \{\{x, y\}, \{z\}\} \)
from meetLatt have \( r \subseteq L \times L \) antisym\( (r) \) trans\( (r) \)
unfolding IsMeetSemilattice_def IsPartOrder_def by auto
moreover from meetLatt assms have \( \forall T \in \mathcal{T}. \) HasAnInfimum\( (r, T) \)
unfolding IsMeetSemilattice_def IsPartOrder_def HasAnInfimum_def
using inf_in_space(1) sup_inf_singl(4) by auto
ultimately have \( \text{Infimum}(r, \{\text{Infimum}(r, T). T \in \mathcal{T}\}) = \text{Infimum}(r, \bigcup \mathcal{T}) \) by (rule inf_inf)
moreover have \( \{\text{Infimum}(r, T). T \in \mathcal{T}\} = \{\inf \{x, y\}, \inf \{z\}\} \) and \( \bigcup \mathcal{T} = \{x, y, z\} \)
by auto
ultimately show \( \inf (\inf \{x, y\}, \inf \{z\}) = \inf \{x, y, z\} \) by auto
also from assms meetLatt have \( \ldots = \inf \{x, y, z\} \) by auto
qed
by auto
also from assms meetLatt have \( \ldots = (\inf \{x, y, z\}) \cap z \)
unfolding IsMeetSemilattice_def IsPartOrder_def using meet_props sup_inf_singl(4)
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 \( x \in L \) shows \( x \cap x = x \)
using meetLatt assms meet_val IsMeetSemilattice_def IsPartOrder_def
10 Order on natural numbers

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:
\begin{align*}
\text{assumes} & \quad a \in \mathbb{N} \land b \in \mathbb{N} \\
\text{shows} & \quad a \leq b \lor b \leq a
\end{align*}

**proof** -
\begin{align*}
& \text{from } \text{assms have } I: \text{Ord}(a) \land \text{Ord}(b) \\
& \quad \text{using nat_into_Ord by auto} \\
& \quad \text{then have } a \in b \lor a = b \lor b \in a \\
& \quad \text{using Ord_linear by simp} \\
& \quad \text{with } I \text{ have } a < b \lor a = b \lor b < a \\
& \quad \text{using ltI by auto} \\
& \quad \text{with } I \text{ show } a \leq b \lor b \leq a \\
& \quad \text{using le_iff by auto} \\
\text{qed}
\end{align*}

≤ is antisymmetric, transitive, total, and linear. Proofs by rewrite using definitions.

**lemma** NatOrder_ZF_1_L2:
\begin{align*}
\text{shows} & \quad \text{antisym}(\text{Le}) \\
& \quad \text{trans}(\text{Le}) \\
& \quad \text{Le \{is total on\} } \mathbb{N} \\
& \quad \text{IsLinOrder}(\mathbb{N}, \text{Le})
\end{align*}

**proof** -
\begin{align*}
& \text{show } \text{antisym}(\text{Le}) \\
& \quad \text{using antisym_def Le_def le_anti_sym by auto} \\
& \quad \text{moreover show } \text{trans}(\text{Le}) \\
& \quad \text{using trans_def Le_def le_trans by blast}
\end{align*}
moreover show \( \text{Le} \) \{is total on\} \( \text{n} \)
using \( \text{IsTotal_def Le_def NatOrder_ZF_1_L1} \) by simp
ultimately show \( \text{IsLinOrder(nat,Le)} \)
using \( \text{IsLinOrder_def} \) 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 \( A1: n \in \text{n} \)\n  shows \( \text{IsLinOrder(n,Le)} \)
proof -
  from \( A1 \) have \( n \subseteq \text{n} \) using \( \text{nat_subset_nat} \)
  by simp
  then show thesis using \( \text{NatOrder_ZF_1_L2 ord_linear_subset} \)
  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 \( f, g : X \to R \) 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 \( \langle a, b \rangle \) 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\rightarrow\text{range}(f))\times(X\rightarrow\text{range}(f))\}

The result of the lift belongs to the function space.

lemma func_ZF_1_L1:
\text{assumes A1: } f : Y\times Y\rightarrow Y \text{ and } A2: p \in (X\rightarrow\text{range}(f))\times(X\rightarrow\text{range}(f)) \text{ shows }\{(x,f(fst(p)(x)),snd(p)(x))\}. x \in X : X\rightarrow\text{range}(f) \text{ proof - } \text{have } \forall x\in X. f(fst(p)(x),snd(p)(x)) \in \text{range}(f) \text{ proof - } \text{fix x assume } x\in X \text{ let } p = (fst(p)(x),snd(p)(x)) \text{ from } A2 \text{ have } \text{fst(p)(x)} \in \text{range}(f) \text{ snd(p)(x)} \in \text{range}(f) \text{ using apply_type by auto } \text{ with A1 have } p \in Y\times Y \text{ using func1_1_L5B by blast } \text{ with A1 have } (p, f(p)) \in f \text{ using apply_Pair by simp } \text{ with A1 show } f(p) \in \text{range}(f) \text{ using rangeI by simp } \text{qed } \text{ then show thesis using ZF_fun_from_total by simp } \text{qed }

The values of the lift are defined by the value of the liftee in a natural way.

lemma func_ZF_1_L2:
\text{assumes A1: } f : Y\times Y\rightarrow Y \text{ and } A2: p \in (X\rightarrow\text{range}(f))\times(X\rightarrow\text{range}(f)) \text{ and } A3: x\in X \text{ and } A4: P = \{(x,f(fst(p)(x)),snd(p)(x))\}. x \in X \text{ shows } P(x) = f(fst(p)(x),snd(p)(x)) \text{ proof - } \text{from A1 A2 have } \{(x,f(fst(p)(x)),snd(p)(x))\}. x \in X \rightarrow \text{range}(f) \text{ using func_ZF_1_L1 by simp } \text{ with A4 have } P : X \rightarrow \text{range}(f) \text{ by simp } \text{ with } A3 A4 \text{ show } P(x) = f(fst(p)(x),snd(p)(x)) \text{ using ZF_fun_from_tot_val by simp } \text{qed }

Function lifted to a function space results in function space operator.

theorem func_ZF_1_L3:
\text{assumes } f : Y\times Y\rightarrow Y
and \( F = f \) \{lifted to function space over\} \( X \)
shows \( F : (X \rightarrow \text{range}(f)) \times (X \rightarrow \text{range}(f)) \rightarrow (X \rightarrow \text{range}(f)) \)
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 \( A1: f : Y \times Y \rightarrow Y \)
and \( A2: F = f \) \{lifted to function space over\} \( X \)
and \( A3: s:X \rightarrow \text{range}(f) \) \( r:X \rightarrow \text{range}(f) \)
and \( A4: x \in X \)
shows \( (F \langle s,r \rangle)(x) = f \langle s(x),r(x) \rangle \)

**Proof**

- let \( p = \langle s,r \rangle \)
- let \( P = \{(x,f \langle \text{fst}(p)(x), \text{snd}(p)(x) \rangle) . x \in X \} \)
  from \( A1 \) \( A3 \) \( A4 \) have
    \( f : Y \times Y \rightarrow Y \) \( p \in (X \rightarrow \text{range}(f)) \times (X \rightarrow \text{range}(f)) \)
    \( x \in X \) \( P = \{(x,f \langle \text{fst}(p)(x), \text{snd}(p)(x) \rangle) . x \in X \} \)
    by auto
  then have \( P(x) = f \langle \text{fst}(p)(x), \text{snd}(p)(x) \rangle \)
    by \( \text{rule func_ZF_1_L2} \)
  hence \( P(x) = f \langle s(x),r(x) \rangle \) by auto
  moreover have \( P = F \langle s,r \rangle \)
    proof
      from \( A1 \) \( A2 \) have \( F : (X \rightarrow \text{range}(f)) \times (X \rightarrow \text{range}(f)) \rightarrow (X \rightarrow \text{range}(f)) \)
        using func_ZF_1_L3 by simp
      moreover from \( A3 \) have \( p \in (X \rightarrow \text{range}(f)) \times (X \rightarrow \text{range}(f)) \)
        by auto
      moreover from \( A2 \) have
        \( F = \{ \langle p,\{(x,f \langle \text{fst}(p)(x), \text{snd}(p)(x) \rangle) . x \in X \} \rangle . x \in X \} \).
        \( p \in (X \rightarrow \text{range}(f)) \times (X \rightarrow \text{range}(f)) \)}
        using Lift2FcnSpce_def by simp
      ultimately show thesis using ZF_fun_from_tot_val by simp
    qed
  ultimately show \( (F \langle s,r \rangle)(x) = f \langle s(x),r(x) \rangle \) by auto
qeda

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 \(\cdot\) 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**

IsAssociative (infix (is associative on) 65) where

$P \{\text{is associative on} \} G \equiv P : G \times G \to G \land \forall x \in G. \forall y \in G. \forall z \in G. (P(P(x, y), z) = P(x, P(y, z)))$

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**

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: $F = f \{\text{lifted to function space over} \} X$

and A3: $s : X \to \text{range}(f)$ $r : X \to \text{range}(f)$

and A4: $f \{\text{is commutative on} \} G$

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 $F(s, r) : X \to \text{range}(f)$ $F(r, s) : X \to \text{range}(f)$

  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 $\text{range}(f) \subseteq G$

    using func1_1_L5B by simp

    with A3 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

  qed
The lift of a commutative function is commutative on the function space.

**lemma** `func_ZF_2_L2`:

assumes `f : G × G → G`
and `f {is commutative on} G`
and `F = f {lifted to function space over} X`
shows `F {is commutative on} (X → 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 → range(f) r : X → range(f) q : X → range(f)`
and `A4: f {is associative on} G`
sows `F ⟨F ⟨s,r⟩,q⟩ = F ⟨s,F ⟨r,q⟩⟩`
proof -
from `A4 A2` have `F : (X → range(f)) × (X → range(f)) → (X → range(f))`
using `IsAssociative_def func_ZF_1_L3` by auto
with `A3` have `I:
F ⟨s,r⟩ : X → range(f)
F ⟨r,q⟩ : X → range(f)
F ⟨F ⟨s,r⟩,q⟩ : X → range(f)
F ⟨s,F ⟨r,q⟩⟩ : X → range(f)`
using `apply_type` by auto
moreover have `∀ x ∈ X. (F ⟨F ⟨s,r⟩,q⟩)(x) = (F ⟨s,F ⟨r,q⟩⟩)(x)`
proof
fix `x` assume `x ∈ X`
from `A4 A2` have `f : G × G → G`
using `IsAssociative_def` by simp
then have `range(f) ⊆ G`
using `func1_1_L5B` by simp
with `A3` have `s(x) ∈ G r(x) ∈ G q(x) ∈ G`
using `apply_type` by auto
with `A2 I A3 A4` show `⟨f:G×G→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`
sows `F {is associative on} (X → 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
  \( \forall s \in X \to \text{range}(f). \forall r \in X \to \text{range}(f). \forall q \in X \to \text{range}(f). \)
  \( F(F(s,r),q) = F(F(s,F(r,q))) \)
  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 \( x \in A \land y \in A \) 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 (infix {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: \( g = \text{restrict}(f, A \times A) \)
sows \( g(g(x,y),z) = g(x,g(y,z)) \)

**Proof** -

from A4 A2 have I: \( x \in X \) \( y \in X \) \( z \in X \)
  by auto
from A3 A4 A5 have
  \( g(g(x,y),z) = f(f(x,y),z) \)

110
\[ g(x, g(y, z)) = f(x, f(y, z)) \]

\[ \text{using IsOpClosed_def restrict_if by auto} \]

moreover from A1 I have
\[ f(f(x, y), z) = f(x, f(y, z)) \]

\[ \text{using IsAssociative_def by simp} \]

ultimately show thesis by simp

qed

An associative operation restricted to a set that is closed with resp. to this operation is associative on the set.

\textbf{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 \( \text{restrict}(f,A \times A) \) \{is associative on\} \( A \)

\textbf{proof -}
- let \( g = \text{restrict}(f,A \times A) \)
from A1 have \( f:X \times X \rightarrow X \)
  \[ \text{using IsAssociative_def by simp} \]
moreover from A2 have \( A \times A \subseteq X \times X \) by auto
moreover from A3 have \( \forall p \in A \times A. \ g(p) \in A \)
  \[ \text{using IsOpClosed_def restrict_if by auto} \]
ultimately have \( g : A \times A \rightarrow A \)
  \[ \text{using func1_2_L4 by simp} \]
moreover from A1 A2 A3 have
\( \forall x \in A. \ \forall y \in A. \ \forall z \in A. \ g(g(x, y), z) = g(x, g(y, z)) \)
  \[ \text{using func_ZF_4_L2 by simp} \]
ultimately show thesis
  \[ \text{using IsAssociative_def by simp} \]
qeq

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 \( A \).

\textbf{lemma func_ZF_4_L4:} assumes \( A \) \{is closed under\} \( f \)
and A2: \( A \subseteq B \) and xA yA and g = restrict(f,B \times B)
shows \( g(x, y) \in A \)
  \[ \text{using assms IsOpClosed_def restrict by auto} \]

If a set \( A \) is closed under an operation, then it is closed under this operation restricted to any superset of \( A \).

\textbf{lemma func_ZF_4_L5:}
assumes A1: \( A \) \{is closed under\} \( f \)
and A2: \( A \subseteq B \)
shows \( A \) \{is closed under\} \( \text{restrict}(f,B \times B) \)

\textbf{proof -}
- let \( g = \text{restrict}(f,B \times B) \)
from A1 A2 have \( \forall x \in A. \ \forall y \in A. \ g(x, y) \in A \)
  \[ \text{using func_ZF_4_L4 by simp} \]
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.

**lemma** func_ZF_4_L6:
assumes A {is closed under} f
and B {is closed under} f
and x ∈ A∩B y∈ A∩B
shows f⟨x,y⟩ ∈ A∩B using assms IsOpClosed_def by auto

Intersection of sets that are closed with respect to an operation is closed under the operation.

**lemma** func_ZF_4_L7:
assumes A {is closed under} f
B {is closed under} f
shows A∩B {is closed under} f
using assms IsOpClosed_def by simp

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.

definition Composition(X) ≡  
{(p,fst(p) O snd(p)). p ∈ (X→X)×(X→X)}

Composition operation is a function that maps $(X \to X) \times (X \to X)$ into $X \to X$.

**lemma** func_ZF_5_L1: shows Composition(X) : (X→X)×(X→X)→(X→X)
using comp_fun Composition_def ZF_fun_from_total by simp

The value of the composition operation is the composition of arguments.

**lemma** func_ZF_5_L2: assumes f:X→X and g:X→X
shows Composition(X)(f,g) = f O g
proof -
from assms have 
Composition(X) : (X→X)×(X→X)→(X→X)
⟨f,g⟩ ∈ (X→X)×(X→X)
Composition(X) = \{(p,fst(p) O snd(p)). p \in (X \rightarrow X) \times (X \rightarrow X)\}
using func_ZF_5_L1 Composition_def by auto
then show Composition(X)(f,g) = f O g
using ZF_fun_from_tot_val by auto
qed

What is the value of a composition on an argument?

lemma func_ZF_5_L3: assumes f:X \rightarrow X and g:X \rightarrow X and x\in X
shows (Composition(X)(f,g))(x) = f(g(x))
using assms func_ZF_5_L2 comp_fun_apply by simp

The essential condition to show that composition is associative.

lemma func_ZF_5_L4: assumes A1: f:X \rightarrow X g:X \rightarrow X h:X \rightarrow X
and A2: C = Composition(X)
shows C(C(f,g),h) = C( f,C(g,h))
proof -
from A2 have C : ((X \rightarrow X) \times (X \rightarrow X)) \rightarrow (X \rightarrow X)
using func_ZF_5_L1 by simp
with A1 have I:
C(f,g) : X \rightarrow X
C(g,h) : X \rightarrow X
C(C(f,g),h) : X \rightarrow X
C( f,C(g,h)) : X \rightarrow X
using apply_funtype by auto
moreover have
\forall x \in X. C(C(f,g),h)(x) = C(f,C(g,h))(x)
proof
fix x assume x\in X
with A1 A2 I have
C(C(f,g),h) (x) = f(g(h(x)))
C( f,C(g,h))(x) = f(g(h(x)))
using func_ZF_5_L3 apply_funtype by auto
then show C(C(f,g),h)(x) = C( f,C(g,h))(x)
by simp
qed
ultimately show thesis using fun_extension_iff by simp
qed

Composition is an associative operation on X \rightarrow X (the space of functions that map X into itself).

lemma func_ZF_5_L5: shows Composition(X) {is associative on} (X \rightarrow X)
proof -
let C = Composition(X)
have \forall f:X \rightarrow X. \forall g:X \rightarrow X. \forall h:X \rightarrow X.
C(C(f,g),h) = C(f,C(g,h))
using func_ZF_5_L4 by simp
then show thesis using func_ZF_5_L1 IsAssociative_def by simp
qed
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: \( f: X \rightarrow Y \) and A2: \( \forall x \in X. \ f(x) = x \)

shows \( f = \text{id}(X) \)

**proof**

- from assms have \( f: X \rightarrow Y \) and \( \text{id}(X): X \rightarrow X \) and \( \forall x \in X. \ f(x) = \text{id}(X)(x) \)

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: \( f: X \rightarrow X \)

shows \( \text{Composition}(X)(\text{id}(X), f) = f \)

Composition(X)(f, id(X)) = f

**proof**

- have \( \text{Composition}(X) : (X \rightarrow X) \times (X \rightarrow X) \rightarrow (X \rightarrow X) \)

using func_ZF_5_L1 by simp

with A1 have \( \text{Composition}(X)(\text{id}(X), f) : X \rightarrow X \)

Composition(X)(f, id(X)) : X \rightarrow X

using id_type apply_funtype by auto

moreover note A1

moreover from A1 have

\( \forall x \in X. \ \text{(Composition}(X)(\text{id}(X), f))(x) = f(x) \)

\( \forall x \in X. \ \text{(Composition}(X)(f, id(X)))(x) = f(x) \)

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 \( \{x\} \rightarrow \{x\} = \{\text{id}(\{x\})\} \)

**proof**

- show \( \{\text{id}(\{x\})\} \subseteq (\{x\} \rightarrow \{x\}) \)

using id_def by simp

{ let \( g = \text{id}(\{x\}) \)

fix \( f \) assume \( f: \{x\} \rightarrow \{x\} \)

then have \( f: \{x\} \rightarrow \{x\} \) and \( g: \{x\} \rightarrow \{x\} \)

using id_def by auto

moreover from \( \langle f : \{x\} \rightarrow \{x\}\rangle \) have \( \forall x \in \{x\}. \ f(x) = g(x) \)

using apply_funtype id_def by auto
ultimately have \( f = g \) by (rule func_eq)

\[
\{x\} \rightarrow \{x\} \subseteq \{\text{id}(\{x\})\}
\]
by auto

qed

Another trivial fact: identity is the only bijection of a singleton with itself.

\begin{verbatim}
lemma single_bij_id: shows \( \text{bij}(\{x\},\{x\}) = \{\text{id}(\{x\})\} \)
proof
  show \( \{\text{id}(\{x\})\} \subseteq \text{bij}(\{x\},\{x\}) \) using \( \text{id}_\text{bij} \)
    by simp
  { fix \( f \) assume \( f \in \text{bij}(\{x\},\{x\}) \)
    then have \( f : \{x\} \rightarrow \{x\} \) using \( \text{bij}_\text{is}_\text{fun} \)
      by simp
    then have \( f \in \{\text{id}(\{x\})\} \) using \( \text{singleton}_\text{fun}_\text{id} \)
      by simp
  } then show \( \text{bij}(\{x\},\{x\}) \subseteq \{\text{id}(\{x\})\} \) by auto
qed
\end{verbatim}

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.

\begin{verbatim}
lemma id_fixpoint_rem: assumes A1: \( f : X \rightarrow X \)
  and A2: \( p \in X \) and A3: \( f(p) = p \) and
  A4: 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 \( \text{id}_\text{def} \) 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 \( \text{restrict} \) by simp
        with A4 \( \langle x \in X - \{p\} \rangle \) have \( f(x) = x \)
        using \( \text{id}_\text{def} \) 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 \( \text{id}_\text{def} \) by simp
  ultimately show \( f = \text{id}(X) \) by (rule func_eq)
qed
\end{verbatim}

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
\( f \), such as associativity, commutativity and existence of the neutral element.
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 \( p = (A, B) \), 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 \( \text{fst}(p) \) and \( \text{snd}(p) \), resp. Recall that in Isabelle/ZF \( \text{fst}(p) \) and \( \text{snd}(p) \) denote the first and second components of an ordered pair \( p \). See the lemma \text{lift_subsets_explained} for a more intuitive notation.

definition
Lift2Subsets (infix \{lifted to subsets of\} 65) where
\[ f \{\text{lifted to subsets of} \} X \equiv \{ (p, f(\text{fst}(p) \times \text{snd}(p))). \ p \in Pow(X) \times Pow(X) \} \]

The lift to subsets defines a binary operation on the subsets.

lemma lift_subsets_binop: assumes A1: \( f : X \times X \to Y \) 
shows \( (f \{\text{lifted to subsets of} \} X) : Pow(X) \times Pow(X) \to Pow(Y) \)
proof -
let \( F = \{ (p, f(\text{fst}(p) \times \text{snd}(p))). \ p \in Pow(X) \times Pow(X) \} \)
from A1 have \( \forall p \in Pow(X) \times Pow(X). f(\text{fst}(p) \times \text{snd}(p)) \in Pow(Y) \)
using func1_1_L6 by simp
then have \( F : Pow(X) \times Pow(X) \to Pow(Y) \)
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: \( f : X \times X \to Y \) and A2: \( A \subseteq X \) B \( \subseteq X \) and A3: \( F = f \{\text{lifted to subsets of} \} X \)
shows \( F(A, B) \subseteq Y \) and \( F(A, B) = f(A \times B) \)
proof -
let \( p = (A, B) \)
from assms have 
I: \( F : Pow(X) \times Pow(X) \to Pow(Y) \) and \( p \in Pow(X) \times Pow(X) \)
using lift_subsets_binop by auto
moreover from A3 have \( F = \{ (p, f(\text{fst}(p) \times \text{snd}(p))). \ p \in Pow(X) \times Pow(X) \} \)
unfolding Lift2Subsets_def by simp
ultimately show \( F(A, B) = f(A \times B) \)
using ZF_fun_from_tot_val by auto
also

116
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\} \( X \)
says \( f(x,y) \in F(A,B) \)

**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: \( z \in F(A,B) \)
says \( \exists x \ y. x \in A \land y \in B \land z = f(x,y) \)

**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\} \( X \) and
A4: \( F \) \{is commutative on\} \( Pow(X) \)

**proof**
- have \( \forall A \in Pow(X) . \forall B \in Pow(X) . F(A,B) = F(B,A) \)

**proof**
- \{ fix A assume A \in Pow(X) 
fix B assume B \in Pow(X) 
have \( F(A,B) = F(B,A) \) 

117
proof -

have \( \forall z \in F(A,B). \; z \in F(B,A) \)
proof
  fix z assume I: z \( \in F(A,B) \)
  with A1 A3 \( \langle A \in \text{Pow}(X) \rangle < B \in \text{Pow}(X) \rangle \) have
  \( \exists x \; y. \; x \in A \land y \in B \land z = f(x,y) \)
  using lift_subset_nec by simp
  then obtain x y where x\( \in A \) and y\( \in B \) and z = f(x,y)
  by auto
  with A2 \( \langle A \in \text{Pow}(X) \rangle < B \in \text{Pow}(X) \rangle \) have \( z = f(y,x) \)
  using IsCommutative_def by auto
  with A1 A3 I \( \langle A \in \text{Pow}(X) \rangle < B \in \text{Pow}(X) \rangle < x \in A \rangle < y \in B \rangle \) show \( z \in F(B,A) \) using lift_subset_suff by simp
  qed

moreover have \( \forall z \in F(B,A). \; z \in F(A,B) \)
proof
  fix z assume I: z \( \in F(B,A) \)
  with A1 A3 \( \langle A \in \text{Pow}(X) \rangle < B \in \text{Pow}(X) \rangle \) have
  \( \exists x \; y. \; x \in B \land y \in A \land z = f(x,y) \)
  using lift_subset_nec by simp
  then obtain x y where x\( \in B \) and y\( \in A \) and z = f(x,y)
  by auto
  with A2 \( \langle A \in \text{Pow}(X) \rangle < B \in \text{Pow}(X) \rangle \) have \( z = f(y,x) \)
  using IsCommutative_def by auto
  with A1 A3 I \( \langle A \in \text{Pow}(X) \rangle < B \in \text{Pow}(X) \rangle < x \in B \rangle < y \in A \rangle \) show \( z \in F(A,B) \) using lift_subset_suff by simp
  qed

ultimately show \( F(A,B) = F(B,A) \) by auto
qed

thus thesis by auto
qed

then show \( F \{ \text{is commutative on} \} \; \text{Pow}(X) \)
  unfolding IsCommutative_def by auto
qed

Lifting to subsets inherits associativity. To show that \( F\langle\langle A, B \rangle, C \rangle = F\langle \langle A, F\langle B, C \rangle \rangle \rangle \) 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: \( f \{ \text{is associative on} \} \; X \) and A2: \( F = f \{ \text{lifted to subsets of} \} \; X \)
shows \( F \{ \text{is associative on} \} \; \text{Pow}(X) \)
proof -
from A1 have f : X\( \times X \rightarrow X \) unfolding IsAssociative_def by simp
with A2 have F : \( \text{Pow}(X) \!\times\! \text{Pow}(X) \rightarrow \text{Pow}(X) \)
  using lift_subsets_binop by simp
moreover have \( \forall A \in \text{Pow}(X). \; \forall B \in \text{Pow}(X). \; \forall C \in \text{Pow}(X). \) \( F\langle F(A,B), C \rangle = F(A,F(B,C)) \)
proof -
{ fix A B C

118
assume A ∈ Pow(X)  B ∈ Pow(X)  C ∈ Pow(X)

have F(F(A,B),C) ⊆ F(A,F(B,C))

proof
fix z assume I: z ∈ F(F(A,B),C)

from <f:X×X → X> A2  <A ∈ Pow(X)>  <B ∈ Pow(X)>

have F(A,B) ∈ Pow(X)

using lift_subsets_binop apply_functype by blast

with <f:X×X → X> A2  <C ∈ Pow(X)>  I have

∃ 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 <f:X×X → X> A2  <A ∈ Pow(X)>  <B ∈ Pow(X)>  II have

∃ 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  <A ∈ Pow(X)>  <B ∈ Pow(X)>  <C ∈ Pow(X)>  III

<s∈A>  <t∈B>  <y∈C>  have IV: z = f(s,f(t,y))

using IsAssociative_def by blast

from <f:X×X → X> A2  <B ∈ Pow(X)>  <C ∈ Pow(X)>  <t∈B>  <y∈C>

have f(t,y) ∈ F(B,C) using lift_subset_suff by simp

moreover from <f:X×X → X> A2  <B ∈ Pow(X)>  <C ∈ Pow(X)>

have F(B,C) ⊆ X using lift_subsets_binop apply_functype

by blast

moreover note <f:X×X → X> A2  <A ∈ Pow(X)>  <s∈A>  IV

ultimately show z ∈ F(A,F(B,C))

using lift_subset_suff by simp

qed

moreover have F(A,F(B,C)) ⊆ F(F(A,B),C)

proof
fix z assume I: z ∈ F(A,F(B,C))

from <f:X×X → X> A2  <B ∈ Pow(X)>  <C ∈ Pow(X)>

have F(B,C) ∈ Pow(X)

using lift_subsets_binop apply_functype by blast

with <f:X×X → X> A2  <A ∈ Pow(X)>  I have

∃ 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 <f:X×X → X> A2  <B ∈ Pow(X)>  <C ∈ Pow(X)>  II have

∃ 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)

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)>

119
ultimately have IV: \( z = f(\langle x, f(s, t) \rangle) \) using IsAssociative_def by blast

moreover note \( f : X \times X \to X \) \( A \in \text{Pow}(X) \) \( B \in \text{Pow}(X) \) IV
ultimately show thesis unfolding IsAssociative_def by auto

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 \( a \cdot (b + c) = a \cdot b + a \cdot c \) and \( (b + c) \cdot a = b \cdot a + c \cdot a \). 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
\[ \text{IsDistributive}(X, A, M) \equiv (\forall a \in X. \forall b \in X. \forall c \in X. \]
\[ M(a, A(b, c)) = A(M(a, b), M(a, c)) \land A\langle b, M(c, a) \rangle = A\langle M(b, a), M(c, a) \rangle) \]

The essential condition to show that distributivity is preserved by restrictions to sets that are closed with respect to both operations.

lemma func_ZF_7_L1:
assumes A1: IsDistributive(X, A, M)
and A2: \( Y \subseteq X \)
and A3: Y \{is closed under\} A Y \{is closed under\} M
and A4: \( A_r = \text{restrict}(A, Y \times Y) \) M_r = \text{restrict}(M, Y \times Y)
and A5: \( a \in Y \) \( b \in Y \) c \in Y
shows \[ M_r(\langle a, A_r(b, c) \rangle) = A_r(\langle M_r(a, b), M_r(a, c) \rangle) \land M_r(\langle A_r(b, c), a \rangle) = A_r(\langle M_r(b, a), M_r(c, a) \rangle) \]
proof -
from A3 A5 have A\langle b, c \rangle \in Y M(a, b) \in Y M(a, c) \in Y
M(b,a) ∈ Y  M(c,a) ∈ Y using IsOpClosed_def by auto
with A5 A4 have
  A_r(b,c) ∈ Y  M_r(a,b) ∈ Y  M_r(a,c) ∈ Y
  M_r(b,a) ∈ Y  M_r(c,a) ∈ 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 ⊆ X
  and Y {is closed under} A
  Y {is closed under} M
  and A_r = restrict(A,Y × Y)  M_r = restrict(M,Y × Y)
  shows IsDistributive(Y,A_r,M_r)
proof -
  from assms have ∀a∈Y.∀b∈Y.∀c∈Y.
    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))
    using func_ZF_7_L1 by simp
  then show thesis using IsDistributive_def by simp
qed

end

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→Y and A⊆X and ∀x∈A. ⟨L,f(x)⟩ ∈ 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 func_ZF_8_L2:
  assumes f:X\rightarrow Y and A\subseteq X and \( \forall x \in A. \ (f(x),U) \in r \)
  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 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 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
only show that \( R \cap (B \times B) \) is).

lemma max_image_ord_iso:
  assumes A1: antisym\((r)\) and A2: antisym\((R)\) and
  A3: f \in ord_iso\((A,r,B,R)\) and
  A4: HasAmaximum\((r,A)\)
  shows HasAmaximum\((R,B)\) 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
    let x = converse\((f)\)(y)
    from A3 have converse\((f)\) \in ord_iso\((B,R,A,r)\)
  }
using ord_iso_sym by simp
then have converse(f): B → A
  using ord_iso_def bij_is_fun by simp
with ⟨y ∈ B⟩ have x ∈ A
    by simp
with A1 A3 A4 ⟨x ∈ A, M ∈ A⟩ have ⟨f(x), f(M)⟩ ∈ R
  using Order_ZF_4_L3 ord_iso_apply by simp
with A3 ⟨y ∈ B⟩ have ⟨y, f(M)⟩ ∈ R
  using right_inverse_bij ord_iso_def by auto
} then have II: ∀y ∈ B. ⟨y, f(M)⟩ ∈ R by simp
with A2 I show Maximum(R,B) = f(M)
  by (rule Order_ZF_4_L14)
from I II show 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 ∈ ord_iso(A,r,A,r)
  and HasAmaximum(r,A)
  shows Maximum(r,A) = f(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 ∈ ord_iso(A,r,B,R) and A2: a ∈ A
  shows restrict(f,A-{a}) ∈ ord_iso(A-{a},r,B-{f(a)},R)
proof -
  let f₀ = restrict(f,A-{a})
  have A-{a} ⊆ A by auto
  with A1 have f₀ ∈ ord_iso(A-{a},r,f(A-{a}),R)
    using ord_iso_restrict_image by simp
  moreover
  from A1 have f ∈ 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→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
qed
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: HasAmaximum(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⟩ ∈ r ∀ b∈B. ⟨b, M⟩ ∈ R and
A4: antisym(r) antisym(R) and
A5: ⟨M, M⟩ ∈ r ←→ ⟨M, M⟩ ∈ R
shows f ∪ {⟨M, M⟩} ∈ ord_iso(A∪{M} ,r,B∪{M} ,R)
proof -
  let g = f ∪ {⟨M, M⟩}
  from A1 A2 have
    g : A∪{M} → B∪{M} and
    I: ∀ x∈A. g(x) = f(x) and II: g(M) = M
    using ord_iso_def bij_def inj_def func1_1_L11D
    by auto
  from A1 A2 have g ∈ bij(A∪{M},B∪{M})
    using ord_iso_def bij_extend_point by simp
  moreover have ∀ x ∈ A∪{M}. ∀ y ∈ A∪{M}.
    ⟨x,y⟩ ∈ r ←→ ⟨g(x), g(y)⟩ ∈ R
  proof -
    { fix x y
      assume x ∈ A∪{M} and y ∈ A∪{M}
      then have x∈A ∧ y ∈ A ∨ x∈A ∧ y = M ∨ x = M ∧ y ∈ 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
        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 ∧ y ∈ A
        with A2 A3 A4 have ⟨x,y⟩ /∈ r
          using antisym_def by auto
      moreover
      { assume A6: ⟨g(x), g(y)⟩ ∈ R
        from A1 I II -x = M ∧ y ∈ A have
A kind of converse to \texttt{ord_iso_rem_max}: if two linearly ordered sets sets are order isomorphic after removing the maxima, then they are order isomorphic.

\textbf{lemma \texttt{rem_max_ord_iso}}:

\begin{itemize}
  \item \texttt{assumes A1: IsLinOrder(X,r) IsLinOrder(Y,R) and}
  \item \texttt{A2: HasAmaximum(r,X) HasAmaximum(R,Y)}
  \item \texttt{ord_iso(X - \{Maximum(r,X)\},r,Y - \{Maximum(R,Y)\},R) \neq 0}
  \end{itemize}

\texttt{shows ord_iso(X,r,Y,R) \neq 0}

\texttt{proof -}

\begin{itemize}
  \item \texttt{let M\_A = Maximum(r,X)}
  \item \texttt{let A = X - \{M\_A\}}
  \item \texttt{let M\_B = Maximum(R,Y)}
  \item \texttt{let B = Y - \{M\_B\}}
  \item \texttt{from A2 obtain f where f \in ord_iso(A,r,B,R)}
    \item \texttt{by auto}
  \end{itemize}

\texttt{moreover have M\_A \notin A and M\_B \notin B}

\item \texttt{by auto}

\item \texttt{moreover from A1 A2 have}
  \item \texttt{\( \forall a \in A. \langle a, M\_A \rangle \in r \) and \( \forall b \in B. \langle b, M\_B \rangle \in R \)}
  \item \texttt{using IsLinOrder_def Order_ZF_4_L3 by auto}
  \item \texttt{moreover from A1 have antisym(r) and antisym(R)}
    \item \texttt{using IsLinOrder_def by auto}
  \item \texttt{moreover from A1 A2 have \( \langle M\_A, M\_A \rangle \in r \leftrightarrow \langle M\_B, M\_B \rangle \in R \)}
    \item \texttt{using IsLinOrder_def Order_ZF_4_L3 IsLinOrder_def}
      \item \texttt{total_is_refl refl_def by auto}
  \item \texttt{ultimately have}
    \item \texttt{f \cup \{\langle M\_A, M\_B \rangle\} \in ord_iso(AU\{M\_A\} ,r,BU\{M\_B\} ,R)}
\end{itemize}
by (rule ord_iso_extend)
moreover from A1 A2 have
A∪{M_A} = X and B∪{M_B} = Y
using IsLinOrder_def Order_ZF_4_L3 by auto
ultimately show ord_iso(X,r,Y,R) ≠ 0
using ord_iso_extend 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 × {y} (a “slice”) and Y. We will call this the SliceProjection(Y×{y}). This is really the ZF equivalent of the meta-function fst(x).

definition
SliceProjection(X) ≡ \{⟨p,fst(p)⟩. p ∈ X \}

A slice projection is a bijection between X × {y} and X.

lemma slice_proj_bij: shows
SliceProjection(X×{y}): X×{y} → X
domain(SliceProjection(X×{y})) = X×{y}
∀ p∈X×{y}. SliceProjection(X×{y})(p) = fst(p)
SliceProjection(X×{y}) ∈ bij(X×{y},X)
proof -
let P = SliceProjection(X×{y})
have ∀ p ∈ X×{y}. fst(p) ∈ X by simp
moreover from this have
\{⟨p,fst(p)⟩. p ∈ X×{y} \} : X×{y} → X
by (rule ZF_fun_from_total)
ultimately show I: P: X×{y} → X and II: ∀ p∈X×{y}. P(p) = fst(p)
using ZF_fun_from_tot_val SliceProjection_def by auto
hence
∀ a ∈ X×{y}. ∀ b ∈ X×{y}. P(a) = P(b) −→ a=b
by auto
with I have P ∈ inj(X×{y},X) using inj_def
by simp
moreover from II have ∀ x∈X. ∃ p∈X×{y}. P(p) = x
by simp
with I have P ∈ surj(X×{y},X) using surj_def
by simp
ultimately show P ∈ bij(X×{y},X)
using bij_def by simp
from I show domain(SliceProjection(X×{y})) = X×{y}
using func1_1_L1 by simp
qed

Given 2 functions f : A → B and g : C → D, we can consider a function
h : A × C → B × D such that h(x, y) = ⟨f(x), g(y)⟩

**definition**

ProdFunction where

ProdFunction(f, g) ≡ \{⟨z, (f(fst(z)), g(snd(z)))⟩ . z ∈ domain(f) × domain(g)\}

For given functions f : A → B and g : C → D the function ProdFunction(f, g) maps A × C to B × D.

**lemma prodFunction:**

assumes f:A→B g:C→D
shows ProdFunction(f,g):(A×C)→(B×D)

**proof** -
from assms have ∀z ∈ domain(f)×domain(g). (f(fst(z)), g(snd(z))) ∈ B×D
using func1_1_L1 apply_type by auto
with assms show thesis unfolding ProdFunction_def using func1_1_L1
ZF_fun_from_total
by simp
qed

For given functions f : A → B and g : C → D and points x ∈ A, y ∈ C the value of the function ProdFunction(f, g) on ⟨x, y⟩ is ⟨f(x), g(y)⟩.

**lemma prodFunctionApp:**

assumes f:A→B g:C→D x∈A y∈C
shows ProdFunction(f,g)⟨x,y⟩= ⟨f(x),g(y)⟩

**proof** -
let z = ⟨x, y⟩
from assms have z ∈ A×C and ProdFunction(f,g):(A×C)→(B×D)
using prodFunction by auto
moreover from assms(1,2) have ProdFunction(f,g) = \{⟨z, (f(fst(z)), g(snd(z)))⟩ . z ∈ A×C\}
unfolding ProdFunction_def using func1_i_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 x∈X f:X→Y
shows ⟨x,t⟩ ∈ ProdFunction(f,f)-(V) ←→ t∈X ∧ ⟨fx,ft⟩ ∈ V

**proof** -
from assms(2) have T:ProdFunction(f,f)-(V) = \{z ∈ X×X. ProdFunction(f,f)(z) ∈ V\}
using prodFunction func1_i_L15 by blast
with assms show thesis using prodFunctionApp by auto
qed

### 12.3 Induced relations and order isomorphisms

When we have two sets X, Y, function f : X → 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 \( f : X \to Y \) the \texttt{InducedRelation}(f,R).

**Definition**

\[
\text{InducedRelation}(f,R) \equiv \{ p \in \text{domain}(f) \times \text{domain}(f). \langle f(fst(p)),f(snd(p)) \rangle \in R \}
\]

A reformulation of the definition of the relation induced by a function.

**Lemma** \texttt{def_of_ind_relA}:

\texttt{assumes} \( \langle x,y \rangle \in \text{InducedRelation}(f,R) \)
\texttt{shows} \( \langle f(x),f(y) \rangle \in R \)
\texttt{using} \texttt{assms} \texttt{InducedRelation_def} \texttt{by simp}

A reformulation of the definition of the relation induced by a function, kind of converse of \texttt{def_of_ind_relA}.

**Lemma** \texttt{def_of_ind_relB}:

\texttt{assumes} \( f:A \to B \) \texttt{and} \( x \in A \) \texttt{and} \( y \in A \) \texttt{and} \( \langle f(x),f(y) \rangle \in R \)
\texttt{shows} \( \langle x,y \rangle \in \text{InducedRelation}(f,R) \)
\texttt{using} \texttt{assms} \texttt{func1_1_L1 InducedRelation_def by simp}

A property of order isomorphisms that is missing from standard Isabelle’s Order.thy.

**Lemma** \texttt{ord_iso_apply_conv}:

\texttt{assumes} \( f \in \text{ord_iso}(A,r,B,R) \) \texttt{and} \( \langle f(x),f(y) \rangle \in R \) \texttt{and} \( x \in A \) \texttt{and} \( y \in A \)
\texttt{shows} \( \langle x,y \rangle \in r \)
\texttt{using} \texttt{assms} \texttt{ord_iso_def by simp}

The next lemma tells us where the induced relation is defined.

**Lemma** \texttt{ind_rel_domain}:

\texttt{assumes} \( R \subseteq B \times B \) \texttt{and} \( f:A \to B \)
\texttt{shows} \( \text{InducedRelation}(f,R) \subseteq A \times A \)
\texttt{using} \texttt{assms} \texttt{func1_1_L1 InducedRelation_def by auto}

A bijection is an order homomorphisms between a relation and the induced one.

**Lemma** \texttt{bij_is_ord_iso}:

\texttt{assumes} \( A1: f \in \text{bij}(A,B) \)
\texttt{shows} \( f \in \text{ord_iso}(A,\text{InducedRelation}(f,R),B,R) \)
\texttt{proof} –
\texttt{let} \( r = \text{InducedRelation}(f,R) \)

128
{ 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
  assume A4: ⟨x,y⟩ ∈ r ⟨y,x⟩ ∈ 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(x)⟩ ∈ 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
    using ord_iso_apply by auto
  with A3 have ⟨f(x),f(z)⟩ ∈ R by (rule Fol1_L3)
  moreover from A2 A4 have x∈A z∈A by auto
  ultimately have ⟨x, z⟩ ∈ r using ord_iso_apply_conv


by simp
} then have \( \forall x \ y \ z. (x, y) \in r \land (y, z) \in r \rightarrow (x, z) \in r \)
by blast
then show \( \text{trans}(r) \) by (rule Fol1_L2)
qed

Order isomorphisms preserve totality.

lemma ord_iso_pres_tot: assumes \( A1: f \in \text{ord}_\text{iso}(A,r,B,R) \) and
\( A2: r \subseteq A \times A \) and \( A3: R \ \{\text{is total on} \} B \)
shows \( r \ \{\text{is total on} \} A \)
proof -
{ fix \( x \ y \)
assume \( x \in A \ y \in A \) \( (x,y) \notin r \)
with \( A1 \) have \( \langle f(x),f(y) \rangle \notin R \) using ord_iso_apply_conv
by auto
moreover
from \( A1 \) have \( f:A \rightarrow B \)
using ord_iso_is_bij bij_is_fun
by simp
with \( A3 \) \( \langle x \in A \rangle \langle y \in A \rangle \) have
\( \langle f(x),f(y) \rangle \in R \lor \langle f(y),f(x) \rangle \in R \)
using apply_funtype IsTotal_def
by simp
ultimately have \( \langle f(y),f(x) \rangle \in R \) by simp
with \( A1 \) \( \langle x \in A \rangle \langle y \in A \rangle \) have \( \langle y,x \rangle \in r \)
using ord_iso_apply_conv
by simp
} then have \( \forall x \in A. \forall y \in A. \langle x,y \rangle \in r \lor \langle y,x \rangle \in r \)
by blast
then show \( r \ \{\text{is total on} \} A \) using IsTotal_def
by simp
qed

Order isomorphisms preserve linearity.

lemma ord_iso_pres_lin: assumes \( f \in \text{ord}_\text{iso}(A,r,B,R) \) and
\( r \subseteq A \times A \) and \( \text{IsLinOrder}(B,R) \)
shows \( \text{IsLinOrder}(A,r) \)
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.

lemma ind_rel_pres_lin:
assumes \( A1: f \in \text{bij}(A,B) \) and \( A2: \text{IsLinOrder}(B,R) \)
shows \( \text{IsLinOrder}(A,\text{InducedRelation}(f,R)) \)
proof -
let \( r = \text{InducedRelation}(f,R) \)
from \( A1 \) have \( f \in \text{ord}_\text{iso}(A,r,B,R) \) and \( r \subseteq A \times A \)
using bij_is_ord_iso domain_of_bij InducedRelation_def
by auto
with \( A2 \) show \( \text{IsLinOrder}(A,r) \) using ord_iso_pres_lin
by simp

130
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: \( \text{IsBoundedAbove}(C,r) \) \( C \neq \emptyset \)

shows \( \text{IsBoundedAbove}(f(C),R) \) \( f(C) \neq \emptyset \)

**Proof** -

from A3 obtain \( u \) where I: \( \forall x \in C. \ (x,u) \in r \)  
using \( \text{IsBoundedAbove}_\text{def} \) by auto

from A1 have \( f:A \rightarrow B \) using \( \text{ord_iso_is_bij} \) \( \text{bij_is_fun} \)  
by simp

from A2 A3 have \( C \subseteq A \) using \( \text{Order_ZF}_3.\text{L1A} \)  
by blast

from A3 obtain \( x \) where \( x \in C \) by auto

with A2 I have \( u \in A \) by auto  
{ fix \( y \) assume \( y \in f(C) \)  
with \( f:A \rightarrow B \) \( C \subseteq A \) obtain \( x \) where \( x \in C \) and \( y = f(x) \)  
using \( \text{func_imagedef} \) by auto

with A1 I \( C \subseteq A \) \( u \in A \) have \( \langle y,f(u) \rangle \in R \)  
using \( \text{ord_iso_apply} \) by auto

} then have \( \forall y \in f(C). \ (y,f(u)) \in R \) by simp

then show \( \text{IsBoundedAbove}(f(C),R) \) by \( \text{rule Order_ZF}_3.\text{L10} \)

from A3 \( f:A \rightarrow B \) \( C \subseteq A \) 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** -

from A4 obtain \( m \) where  
I: \( m \in f(C) \) and II: \( \forall y \in f(C). \ (m,y) \in R \)  
using \( \text{HasAminimum}_\text{def} \) by auto

let \( k = \text{converse}(f)(m) \)

from A1 have \( f:A \rightarrow B \) using \( \text{ord_iso_is_bij} \) \( \text{bij_is_fun} \)  
by simp

from A1 have \( f \in \text{inj}(A,B) \) using \( \text{ord_iso_is_bij} \) \( \text{bij_is_inj} \)  
by simp

with A3 I have \( k \in C \) and III: \( f(k) = m \)  
using \( \text{inj}_\text{inv}_\text{back}_\text{in}_\text{set} \) by auto

moreover  
{ fix \( x \) assume A5: \( x \in C \)  
with A3 II \( f:A \rightarrow B \) \( k \in C \) III have \( k \in A \) \( x \in A \) \( (f(k),f(x)) \in R \)  
using \( \text{func_imagedef} \) by auto

with A1 have \( \langle k,x \rangle \in r \) using \( \text{ord_iso_apply}_\text{conv} \)  
131
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: \( f \in \text{ord_iso}(A, r, B, R) \)
- A2: \( r \subseteq A \times A \) \( R \subseteq B \times B \) 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: \( f(r\{a\}) = \{f(x) . x \in r\{a\}\} \)

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 = converse(f)(y)

from A2 A5 have \( \langle f(a), y \rangle \in R \) \( f(a) \in B \) 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 converse(f)(f(a)) = a

using ord_iso_is_bij left_inverse_bij by blast

ultimately have \( f(x) \in \{f(x) . x \in r\{a\}\} \)

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 \( f(r\{a\}) = \{f(x) . x \in r\{a\}\} \)

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 isomorhisms 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_up_bounds:

- **assumes** A1: \( f \in \text{ord_iso}(A, r, B, R) \)
- A2: \( r \subseteq A \times A \) \( R \subseteq B \times B \) 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: \( f(r\{a\}) = \{f(x) . x \in r\{a\}\} \)

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 = converse(f)(y)

from A2 A5 have \( \langle f(a), y \rangle \in R \) \( f(a) \in B \) 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 converse(f)(f(a)) = a

using ord_iso_is_bij left_inverse_bij by blast

ultimately have \( f(x) \in \{f(x) . x \in r\{a\}\} \)

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 \( f(r\{a\}) = \{f(x) . x \in r\{a\}\} \)

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
A3: $C \subseteq A$
shows $\{f(r(a)). a \in C\} = \{R(b). b \in f(C)\}$
proof
from A1 have $f:A \rightarrow B$
  using ord_iso_is_bij bij_is_fun by simp
{ fix $Y$ assume $Y \in \{f(r(a)). a \in C\}$
  then obtain $a$ where $a \in C$ and $I: Y = f(r(a))$
  by auto
  from A3 $<a \in C>$ have $a \in A$ by auto
  with A1 A2 have $f(r(a)) = R\{f(a)\}$
  using ord_iso_pres_rel_image by simp
  moreover from A3 $<f:A \rightarrow B, <a \in C>$ have $f(a) \in f(C)$
  using func_imagedef by auto
  ultimately have $f(r(a)) \in \{R(b). b \in f(C)\}$
  by auto
  with I have $Y \in \{R(b). b \in f(C)\}$ by simp
} then show $\{f(r(a)). a \in C\} \subseteq \{R(b). b \in f(C)\}$
by blast
{ fix $Y$ assume $Y \in \{R(b). b \in f(C)\}$
  then obtain $b$ where $b \in f(C)$ and $II: Y = R\{b\}$
  by auto
  with A3 $<f:A \rightarrow B>$ obtain $a$ where $a \in C$ and $b = f(a)$
  using func_imagedef by auto
  with A3 II have $a \in A$ and $Y = R\{f(a)\}$ by auto
  with A1 A2 have $Y = f(r(a))$
  using ord_iso_pres_rel_image by simp
  with $<a \in C>$ have $Y \in \{f(r(a)). a \in C\}$ by auto
} then show $\{R(b). b \in f(C)\} \subseteq \{f(r(a)). a \in C\}$
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: $r \subseteq A \times A$ $R \subseteq B \times B$ and
A3: $C \subseteq A$ and A4: $C \neq 0$
shows $f(\bigcap a \in C. r(a)) = (\bigcap b \in f(C). R\{b\})$
proof -
from A1 have $f \in \text{inj}(A,B)$
  using ord_iso_is_bij bij_is_inj by simp
moreover note A4
moreover from A2 A3 have $\forall a \in C. r(a) \subseteq A$ by auto
ultimately have $f(\bigcap a \in C. r(a)) = (\bigcap a \in C. f(r(a)))$
  using inj_image_of_Inter by simp
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 $f(\bigcap a \in C. r(a)) = (\bigcap b \in f(C). R\{b\})$
Order isomorphisms preserve completeness.

**Lemma** ord_iso_pres_compl:
- assumes A1: \( f \in \text{ord}_\text{iso}(A,r,B,R) \) and
- A2: \( r \subseteq A \times A \) \( R \subseteq B \times B \) and
- A3: \( R \) is complete
- shows \( r \) is complete

**Proof** -

\[
\begin{align*}
\text{fix } C \\
\text{assume } A4: \text{IsBoundedAbove}(C,r) \land C \neq 0 \\
\text{with } A1 A2 A3 \text{ have } \\
\quad \text{HasAminimum}(R, \bigcap b \in f(C). R\{b\}) \\
\quad \text{using ord_iso_pres_bound_above IsComplete_def by simp} \\
\text{moreover from } A2 <\text{IsBoundedAbove}(C,r)> \text{ have } I: C \subseteq A \text{ using Order_ZF_3_L1A by blast} \\
\text{with } A1 A2 <C \neq 0> \text{ have } f(\bigcap a \in C. r\{a\}) = (\bigcap b \in f(C). R\{b\}) \\
\quad \text{using ord_iso_pres_min_up_bounds by simp} \\
\text{ultimately have HasAminimum}(R,f(\bigcap a \in C. r\{a\})) \\
\quad \text{by simp} \\
\text{moreover from } A2 \text{ have } \forall a \in C. r\{a\} \subseteq A \\
\quad \text{by auto} \\
\text{with } <C \neq 0> \text{ have } (\bigcap a \in C. r\{a\}) \subseteq A \text{ using ZF1_1_L7 by simp} \\
\text{moreover note } A1 A2 \\
\text{ultimately have HasAminimum}(r, \bigcap a \in C. r\{a\}) \\
\quad \text{using ord_iso_pres_has_min by simp} \\
\text{then show } r \text{ is complete using IsComplete_def by simp} \\
\end{align*}
\]

**Qed**

If the original relation is complete, then the induced one is complete.

**Lemma** ind_rel_pres_compl:
- assumes A1: \( f \in \text{bij}(A,B) \) and
- A2: \( R \subseteq B \times B \) and
- A3: \( R \) is complete
- shows \( \text{InducedRelation}(f,R) \) is complete

**Proof** -

\[
\begin{align*}
\text{let } r = \text{InducedRelation}(f,R) \\
\text{from } A1 \text{ have } f \in \text{ord}_\text{iso}(A,r,B,R) \\
\quad \text{using bij_is_ord_iso by simp} \\
\text{moreover from } A1 A2 \text{ have } r \subseteq A \times A \\
\quad \text{using bij_is_fun ind_rel_domain by simp} \\
\text{moreover note } A2 A3 \\
\text{ultimately show } r \text{ is complete} \\
\quad \text{using ord_iso_pres_compl by simp} \\
\end{align*}
\]

**Qed**
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 \( \text{FinPow}(X) \) as those elements of the powerset that are finite.

\[ \text{definition} \]
\[ \text{FinPow}(X) \equiv \{ A \in \text{Pow}(X). \text{Finite}(A) \} \]

The cardinality of an element of finite powerset is a natural number.

\[ \text{lemma card_fin_is_nat: assumes } A \in \text{FinPow}(X) \]
\[ \text{shows } |A| \in \text{nat} \text{ and } A \approx |A| \]
\[ \text{using } \text{assms FinPow_def Finite_def cardinal_cong nat_into_Card Card_cardinal_eq} \text{ by auto} \]

A reformulation of \( \text{card_fin_is_nat} \): for a finit set \( A \) there is a bijection between \( |A| \) and \( A \).

\[ \text{lemma fin_bij_card: assumes } A1: A \in \text{FinPow}(X) \]
\[ \text{shows } \exists b. b \in \text{bij}(|A|, A) \]
\[ \text{proof -} \]
\[ \text{from } A1 \text{ have } |A| \approx A \text{ using card_fin_is_nat eqpoll_sym by blast} \]
\[ \text{then show thesis using eqpoll_def by auto} \]

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 that has \( n \) elements.

**Lemma card_card**: assumes \( A \approx n \) and \( n \in \text{nat} \)
shows \( |A| = n \)
using assms 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
\[ |A \cup \{a\}| = \text{succ}( |A| ) \text{ and } |A \cup \{a\}| = |A| \cup \{|A|\} \]
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 \( \text{FinPow}(X) = (\bigcup n \in \text{nat}. \{A \in \text{Pow}(X). A \preceq n\}) \)
proof -
have \( \text{FinPow}(X) \subseteq (\bigcup n \in \text{nat}. \{A \in \text{Pow}(X). A \preceq n\}) \)
using finpow_decomp FinPow_def eqpoll_imp_lepoll
by auto
moreover have
\( (\bigcup n \in \text{nat}. \{A \in \text{Pow}(X). A \preceq n\}) \subseteq \text{FinPow}(X) \)
using lepoll_nat_imp_Finite FinPow_def by auto
ultimately show thesis by auto
qed
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)
sows 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)
      qed
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
\}

} thus thesis by blast

qed
ultimately have ∀ B ∈ Pow(X). (B ≲ n → P(B))
by (rule ind_on_nat)
\}

then have ∀ n ∈ nat. ∀ B ∈ Pow(X). (B ≲ n → P(B))
by auto
with A3 show P(B) using finpow_union_card_nat
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 finites 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))
    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
      ultimately show ∀A ∈ FinPow(X). A ≤ n → P(A)
        by (rule ind_on_nat)
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 
\[ A2: \forall A \in \text{FinPow}(X). A \neq 0 \implies (\exists a \in A. P(A\{a\}) \implies P(A)) \]
and \( A3: B \in \text{FinPow}(X) \)
shows \( P(B) \)
**proof -**

\begin{align*}
\text{note } A1 \\
\text{moreover have } \forall k \in \text{nat}. \\
(\forall B \in \text{FinPow}(X). B \leq k \implies P(B)) \implies \\
(\forall C \in \text{FinPow}(X). C \leq \text{succ}(k) \implies P(C)) \implies \\
\text{proof -} \\
\{ \text{fix } k \text{ assume } k \in \text{nat} \\
\text{assume } A4: \forall B \in \text{FinPow}(X). B \leq k \implies P(B) \\
\text{have } \forall C \in \text{FinPow}(X). C \leq \text{succ}(k) \implies P(C) \}
\end{align*}

\begin{align*}
\text{proof -} \\
\{ \text{fix } C \text{ assume } C \in \text{FinPow}(X) \\
\text{assume } C \leq \text{succ}(k) \\
\text{note } A1 \\
\text{moreover} \\
\{ \text{assume } C \neq 0 \\
\text{with } A2 <C \in \text{FinPow}(X) > \text{ obtain } a \text{ where} \\
a \in C \text{ and } P(C\{a\}) \implies P(C) \\
\text{by auto} \\
\text{with } A4 <C \in \text{FinPow}(X) > <C \leq \text{succ}(k)> \\
\text{have } P(C) \text{ using Diff_sing_lepoll fin_rem_point_fin} \\
\text{by simp } \} \\
\text{ultimately have } P(C) \text{ by auto} \\
\} \text{ thus thesis by simp} \\
\text{thus thesis by blast} \\
\text{qed} \\
\text{moreover note } A3 \\
\text{ultimately show } P(B) \text{ by (rule FinPow_card_ind)} \\
\text{qed}
\end{align*}

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 \( A1: \forall A \in \text{FinPow}(X). \)
\[ A = 0 \lor (\exists a \in A. A = \{a\} \lor P(A\{a\}) \implies P(A)) \]
and \( A2: A \in \text{FinPow}(X) \) and \( A3: A \neq 0 \)
shows \( P(A) \)
**proof -**
have $0 = 0 \lor P(0)$ by simp
moreover have
\[ \forall k \in \text{nat}. \]
\[ (\forall B \in \text{FinPow}(X). B \subseteq k \rightarrow (B = 0 \lor P(B))) \rightarrow \]
\[ (\forall A \in \text{FinPow}(X). A \subseteq \text{succ}(k) \rightarrow (A = 0 \lor P(A))) \]
proof -
{ fix k assume k ∈ nat
  assume A4: \( \forall B \in \text{FinPow}(X). B \subseteq k \rightarrow (B = 0 \lor P(B)) \)
  have \( \forall A \in \text{FinPow}(X). A \subseteq \text{succ}(k) \rightarrow (A = 0 \lor P(A)) \)
proof -
{ fix A assume A ∈ \text{FinPow}(X)
  assume A ⊆ succ(k) A ≠ 0
  from A1 <A ∈ \text{FinPow}(X)> <A≠0> obtain a
  where a ∈ A and A = \{a\} \lor P(A-{a}) → P(A)
  by auto
  let B = A-{a}
  from A4 <A ∈ \text{FinPow}(X)> <A ⊆ succ(k)> <a∈A>
  have B = 0 \lor P(B)
    using Diff_sing_lepoll fin_rem_point_fin
    by simp
  with <a∈A> <A = \{a\} \lor P(A-{a}) → P(A)>
  have P(A) by auto
} thus thesis by auto
qed
moreover note A2
ultimately have A = 0 \lor 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: \( \forall V \in T. \forall W \in T. V \cap W \in T \) and
  A2: N ≠ 0 and A3: N ∈ \text{FinPow}(T)
  shows \( (\bigcap N \in T) \)
proof -
  have \( 0 = 0 \lor (\bigcap N \in T) \) by simp
moreover have \( \forall M \in \text{FinPow}(T). (M = 0 \lor \bigcap M \in T) \rightarrow \)
\( \forall W \in T. M \cup \{W\} = 0 \lor \bigcap (M \cup \{W\}) \in T) \)
proof -
{ fix M assume M ∈ \text{FinPow}(T)
  assume A4: M = 0 \lor \bigcap M \in T
  { assume M = 0
  hence \( \forall W \in T. M \cup \{W\} = 0 \lor \bigcap (M \cup \{W\}) \in T \)
  by auto }
  moreover
}
\{ \text{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 \}
hence \( \forall W \in T. M \cup \{W\} = 0 \lor \bigcap (M \cup \{W\}) \in T \)
by blast \}
thus \text{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.

\text{lemma union_two_union_fin:}
\text{assumes A1: } 0 \in C \text{ and A2: } \forall A \in C. \forall B \in C. A \cup B \in C \text{ and}
A3: \( N \in \text{FinPow}(C) \)
shows \( \bigcup N \in C \)
\text{proof -}
\{ \text{fix } M \text{ assume } M \in \text{FinPow}(C) \}
\text{assume } \bigcup M \in C
\text{fix } A \text{ assume } A \in C
\text{have } \bigcup (M \cup \{A\}) = (\bigcup M) \cup A \text{ 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 \text{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.

\text{lemma empty_in_finpow: shows } 0 \in \text{FinPow}(X)
using \text{FinPow_def by simp}

Singleton is in the finite powerset.

\text{lemma singleton_in_finpow: assumes } x \in X
shows \( \{x\} \in \text{FinPow}(X) \) using assms \text{FinPow_def} by simp

Union of two finite subsets is a finite subset.

**lemma union_finpow:** assumes \( A \in \text{FinPow}(X) \) and \( B \in \text{FinPow}(X) \)
shows \( A \cup B \in \text{FinPow}(X) \)
using assms \text{FinPow_def} by auto

Union of finite number of finite sets is finite.

**lemma fin_union_finpow:** assumes \( M \in \text{FinPow}(\text{FinPow}(X)) \)
shows \( \bigcup M \in \text{FinPow}(X) \)
using assms \text{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 \( A2: A - \{x\} \in \text{FinPow}(X) \)
shows \( A \in \text{FinPow}(X) \)
proof -
from assms 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 \text{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 \( N \in \text{FinPow}(B) \)
shows \( \{K(V). V \in N\} \in \text{FinPow}(C) \)
proof -
have \( \{K(V). V \in N\} \in \text{FinPow}(C) \) using \text{FinPow_def} by auto
moreover have \( \forall A \in \text{FinPow}(B) . \)
\( \{K(V). V \in A\} \in \text{FinPow}(C) \implies (\forall a \in B. \{K(V). V \in (A \cup \{a\})\} \in \text{FinPow}(C)) \)
proof -
\{ fix A assume A \in FinPow(B) \\
assume \( \{K(V). V \in A\} \in \text{FinPow}(C) \)
fix a assume a \in B \\
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 \( \forall V \in B. K(V) \in C \) and \( a \in B \) have \( \{K(a)\} \in \text{FinPow}(C) \)
using singleton_in_finpow by simp
ultimately show thesis using union_finpow by simp
qed
\} thus thesis by simp
qed
moreover note \( N \in \text{FinPow}(B) \)
ultimately show \( \{K(V). \ V \in N\} \in \text{FinPow}(C) \)
by (rule FinPow_induct)

qed

Union of a finite indexed family of finite sets is finite.

lemma union_fin_list_fin:
assumes A1: \( n \in \text{nat} \) and A2: \( \forall k \in n. \ N(k) \in \text{FinPow}(X) \)
shows \( \{N(k). \ k \in n\} \in \text{FinPow}(\text{FinPow}(X)) \) and \( (\bigcup k \in n. \ N(k)) \in \text{FinPow}(X) \)
proof -
from A1 have \( n \in \text{FinPow}(n) \)
using nat_finpow_nat fin_finpow_self by auto
with A2 show \( \{N(k). \ k \in n\} \in \text{FinPow}(\text{FinPow}(X)) \)
by (rule fin_image_fin)
then show \( (\bigcup k \in n. \ N(k)) \in \text{FinPow}(X) \)
using fin_union_finpow by simp

end

14 Finite sets

theory Finite1 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 suprisingly 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
Prfin(T,A,M) ≡ ( (M = 0) | (∃N ∈ Fin(T). ∀V ∈ M. ∃ U ∈ N. (V = U ∩ A)))

Now we show the main induction step in a separate lemma. This will make the proof of the theorem FinRestr below look short and nice. The premises of the ind_step lemma are those needed by the main induction step in lemma Fin_induct (see standard Isabelle’s Finite.thy).

lemma ind_step: assumes A: ∀ V ∈ TA. ∃ U ∈ T. V = U ∩ A
and A1: W ∈ TA and A2: M ∈ Fin(TA)
and A3: W /∈ M and A4: Prfin(T,A,M)
shows Prfin(T,A,cons(W,M))
proof -
{ assume A7: M=0 have Prfin(T, A, cons(W, M))
proof-
from A1 A obtain U where A5: U ∈ T and A6: W = U ∩ A by fast
let N = {U}
from A5 have T1: N ∈ Fin(T) by simp
from A7 A6 have T2: ∀ V ∈ cons(W,M). ∃ U ∈ N. V = U ∩ A by simp
from A7 T1 T2 show Prfin(T, A, cons(W, M)) using Prfin_def by auto
qed }
moreover
{ assume A8: M≠0 have Prfin(T, A, cons(W, M))
proof-
from A1 A obtain U where A5: U ∈ T and A6: W = U ∩ A by fast
from A8 A4 obtain N0
where A9: N0 ∈ Fin(T) and A10: ∀ V ∈ M. ∃ U0 ∈ N0. (V = U0 ∩ A)
using Prfin_def by auto
let N = cons(U,N0)
from A5 A9 have N ∈ Fin(T) by simp
moreover from A10 A6 have ∀ V ∈ cons(W,M). ∃ U ∈ N. V = U ∩ A by simp
ultimately have ∃ N ∈ Fin(T). ∀ V ∈ cons(W,M). ∃ U ∈ N. V = U ∩ A by auto
with A8 show Prfin(T, A, cons(W, M)) using Prfin_def by simp
qed }
ultimately have Prfin(T,A,M) by (rule Fin_induct)
qed

Now we are ready to prove the statement we need.

theorem FinRestr0: assumes A: ∀ V ∈ TA. ∃ U ∈ T. V = U ∩ A
shows ∀ M ∈ Fin(TA). Prfin(T,A,M)
proof -
{ fix M
assume M ∈ Fin(TA)
moreover have Prfin(T,A,0) using Prfin_def by simp
moreover
{ fix W M assume W ∈ TA M ∈ Fin(TA) W /∈ M Prfin(T,A,M)
with A have Prfin(T,A,cons(W,M)) by (rule ind_step) }
ultimately have Prfin(T,A,M) by (rule Fin_induct)
This is a different form of the above theorem:

**Theorem ZF1FinRestr:**

- **Assumes:** \( A1: M \in \text{Fin}(TA) \) and \( A2: M \neq 0 \)
- **and:** \( A3: \forall V \in TA. \exists U \in T. V = U \cap A \)
- **Shows:** \( \exists N \in \text{Fin}(T). (\forall V \in M. \exists U \in N. (V = U \cap A)) \land N \neq 0 \)

**Proof:**

- From \( A3 \) \( A1 \) have \( \text{Prfin}(T,A,M) \) using \( \text{FinRestr0} \) by \( \text{blast} \)
- Then have \( \exists N \in \text{Fin}(T). (\forall V \in M. \exists U \in N. (V = U \cap A)) \) using \( A2 \text{ Prfin_def} \) by \( \text{simp} \)
- Then obtain \( N \) where \( D1: N \in \text{Fin}(T) \land (\forall V \in M. \exists U \in N. (V = U \cap A)) \) by \( \text{auto} \)
- With \( A2 \) have \( N \neq 0 \) by \( \text{auto} \)
- With \( D1 \) show thesis by \( \text{auto} \)

**QED**

Purely technical lemma used in **Topology_ZF_1** to show that if a topology is \( T_2 \), then it is \( T_1 \).

**Lemma Finite1_L2:**

- **Assumes:** \( \exists U V. (U \in T \land V \in T \land x \in U \land y \in V \land U \cap V = 0) \)
- **Shows:** \( \exists U \in T. (x \in U \land y \notin U) \)

**Proof:**

- From \( A \) obtain \( U V \) where \( D1: U \in T \land V \in T \land x \in U \land y \in V \land U \cap V = 0 \) by \( \text{auto} \)
- With \( D1 \) show thesis by \( \text{auto} \)

**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.

**Lemma Finite1_L3_IndStep:**

- **Assumes:** \( \forall A B. ((A \in C \land B \in C) \rightarrow A \cup B \in C) \)
- **and:** \( A2: A \in C \) and \( A3: N \in \text{Fin}(C) \) and \( A4: A \notin N \) and \( A5: \cup N \in C \)
- **Shows:** \( \forall \cup \text{cons}(A,N) \in C \)

**Proof:**

- Have \( \cup \text{cons}(A,N) = A \cup \cup N \) by \( \text{blast} \)
- With \( A1 A2 A5 \) show thesis by \( \text{simp} \)

**QED**

The lemma: a collection closed with respect to taking a union of two sets is closed under taking finite unions.

**Lemma Finite1_L3:**

- **Assumes:** \( A1: 0 \in C \) and \( A2: \forall A B. ((A \in C \land B \in C) \rightarrow A \cup B \in C) \) and \( A3: N \in \text{Fin}(C) \)
- **Shows:** \( \cup N \in C \)

**Proof:**

- **Note A3**
moreover from A1 have $\bigcup 0 \in C$ by simp
moreover
{ fix A N
  assume A \in C N \in \text{Fin}(C) \ A \notin N \union N \in C
  with A2 have $\bigcup \text{cons}(A, N) \in C$ by (rule Finite1_L3_IndStep) }
ultimately show $\bigcup \{ N \in C \}$ by (rule Fin_induct)
qed

A collection closed with respect to taking a 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: N \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 IntPr_def by simp }
moredover
{ assume A7: N \neq 0 have \text{IntPr}(T, \text{cons}(A, N))
  proof -
    from A7 A5 A2 A1 have $\bigcap N \cap A \in T$ using 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 IntPr_def by simp
  qed }
moreditionally 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 IntPr_def by simp
moreover
{ fix A N
  assume A \in T N \in \text{Fin}(T) A \notin N \text{IntPr}(T, N)
  with A1 have \text{IntPr}(T, \text{cons}(A, N)) by (rule Finite1_L4_IndStep) }

147
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 ∩ 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 and \{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
  using apply_type by simp
  moreover note A2
  ultimately have \{f(x). x ∈ N\} ∈ Fin(Y)
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 \( \{b(y). y \in N\} \in \text{Fin}(Z) \) by (rule fin_image_fin)
moreover 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: \( \forall a \in A. \ \forall b \in B. \ \{\langle a,b\rangle\} \in \text{Fin}(A \times B) \) by simp
have \( \forall a \in A. \ \{\langle a,b\rangle. b \in B\} \in \text{Fin}(\text{Fin}(A \times B)) \)

**Proof** -
fix a assume A3: \( a \in A \)
with T1 have \( \forall b \in B. \ \{\langle a,b\rangle\} \in \text{Fin}(A \times B) \) by simp
moreover note A2
ultimately show \( \{\langle a,b\rangle. b \in B\} \in \text{Fin}(\text{Fin}(A \times B)) \) by (rule fin_image_fin)
qed

then have \( \forall a \in A. \ \text{Fin}(\text{Fin}(A \times B)) \) by simp
moreover have \( \forall a \in A. \ \text{Fin}(\text{Fin}(A \times B)) \) by blast
ultimately have \( \forall a \in A. \ \{a\} \times B \in \text{Fin}(A \times B) \) by simp
moreover note A1
ultimately have \( \{ \{a\} \times B. \ a \in A\} \in \text{Fin}(\text{Fin}(A \times B)) \)
by (rule fin_image_fin)
then have \( \bigcup \{\{a\} \times B. \ a \in A\} \in \text{Fin}(A \times B) \)
using Fin_UnionI by simp
moreover have \( \bigcup \{\{a\} \times B. \ a \in A\} = A \times B \) 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**
Characteristc(A,default,x) ≡ (if x\( \in \) A then x else default)

A finite subset is a finite subset of itself.

**lemma** Finite1_L13:
assumes A1: A \( \in \) Fin(X)
shows A \( \in \) Fin(A)
proof -
\{ assume A=0 hence A \( \in \) Fin(A) by simp \}
moreover
\{ assume A2: A \neq 0 then obtain c where D1:c\( \in \) A
by auto
then have \( \forall x \in X. \) Characteristic(A,c,x) \( \in \) A
using Characteristic_def by simp
moreover note A1
ultimately have \( \{ \text{Characteristic}(A,c,x). \ x \in A\} \in \text{Fin}(A) \) by (rule fin_image_fin)
moreover from D1 have \( \{ \text{Characteristic}(A,c,x). \ x \in A\} = A \) using Characteristic_def by simp
ultimately have A \( \in \) 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 \( \in \) Fin(X) \( B \in \) Fin(Y)
shows A \( \times \) B \( \in \) Fin(X \( \times \) Y)
proof -
from A1 have A\( \times \) B \( \subseteq \) X \( \times \) Y using FinD by auto
then have Fin(A\( \times \) B) \( \subseteq \) Fin(X \( \times \) Y) using Fin_mono by simp
moreover from A1 have A\( \times \) B \( \in \) Fin(A\( \times \) 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: \( \{b(x). \ x \in A\} \in \text{Fin}(B) \) \( \{c(x). \ x \in A\} \in \text{Fin}(C) \)
and A2: f : B \( \times \) C \( \rightarrow \) E
shows \{{f(b(x),c(x))}. x \in A} \in \text{Fin}(E)

proof -
from \text{A1} have \{b(x). x \in A\} \times \{c(x). x \in A\} \in \text{Fin}(B \times C)
  using \text{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 \text{Fin_subset_lemma})
with \text{A2} have T1: {f{(b(x),c(x))}. x \in A} \in \text{Fin}(E)
  using \text{Finite1_L6A} by auto
from T0 have \forall x \in A. \{b(x),c(x)\} \in B \times C
  using \text{FinD} by auto
with \text{A2} have
  f{(b(x),c(x)). x \in A} = \{f{(b(x),c(x)). x \in A}
  using \text{func1_1_L17} by simp
with T1 show thesis by simp
qed

Singletons are in the finite powerset.

lemma \text{Finite1_L16}: \text{assumes} x \in X \text{ shows} \{x\} \in \text{Fin}(X)
  using \text{assms emptyI consI} by simp

A special case of \text{Finite1_L15} where the second set is a singleton. In \text{Group_ZF_3} theory this corresponds to the situation where we multiply by a constant.

lemma \text{Finite1_L16AA}: \text{assumes} \{b(x). x \in A\} \in \text{Fin}(B)
  and \ c \in C \text{ and} \ f : B \times C \rightarrow E
  shows \{f{(b(x),c)}. x \in A\} \in \text{Fin}(E)
proof -
from \text{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 \text{Finite1_L6C})
qed

First order version of the induction for the finite powerset.

lemma \text{Finite1_L16B}: \text{assumes} \text{A1: P(0)} \text{ and} \ \text{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\})
  \text{ shows} \ P(B)
proof -
  note \ <B \in \text{Fin}(X)> \text{ and} \ <P(0)>
moreover
  \{ \text{fix A x}
    \text{ assume} \ x \in X \ A \in \text{Fin}(X) \ x \notin A \ P(A)
    \text{ moreover have} \ \text{cons}(x,A) = A \cup \{x\} \text{ by auto}
    \text{ moreover note} \ A3

\text{151}
ultimately have \( P(\text{cons}(x,A)) \) by simp 
ultimately show \( P(B) \) by (rule Fin_induct)

**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 a important role in the construction of real numbers in the \textit{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** \textit{Finite1_L17}: assumes \( c \in Y \) and \( X \neq 0 \)

shows \( \text{ConstantFunction}(X,c) \in \text{FinRangeFunctions}(X,Y) \)

using assms \textit{func1_3_L1} \textit{func_imagedef} \textit{func1_3_L2} \textit{Finite1_L16} 
\textit{FinRangeFunctions_def} by simp

Finite range functions have finite range.

**lemma** \textit{Finite1_L18}: assumes \( f \in \text{FinRangeFunctions}(X,Y) \)

shows \( \{ f(x). x \in X \} \in \text{Fin}(Y) \)

using assms \textit{FinRangeFunctions_def} \textit{func_imagedef} by simp

An alternative form of the definition of finite range functions.

**lemma** \textit{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 \textit{func_imagedef} \textit{FinRangeFunctions_def} by simp

A composition of a finite range function with another function is a finite range function.

**lemma** \textit{Finite1_L20}: assumes \( A1: f \in \text{FinRangeFunctions}(X,Y) \)

and \( A2: g : Y \rightarrow Z \)

shows \( g \circ f \in \text{FinRangeFunctions}(X,Z) \)

proof -

from \( A1 \ A2 \) have \( g\{f(x). x \in X\} \in \text{Fin}(Z) \)

using \textit{Finite1_L18} \textit{Finite1_L6A} by simp

with \( A1 \ A2 \) have \( \{(g \circ f)(x). x \in X\} \in \text{Fin}(Z) \)

using \textit{FinRangeFunctions_def} \textit{apply_funtype} \textit{func1_1_L17} \textit{comp_fun_apply} by auto

with \( A1 \ A2 \) show thesis using 
\textit{FinRangeFunctions_def} \textit{comp_fun} \textit{Finite1_L19} by auto

qed
Image of any subset of the domain of a finite range function is finite.

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
end

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
qed

For total and transitive relations finite set has a maximum.
Theorem Finite_ZF_1_1_T1A:
assumes A1: \text{r is total on } X \text{ and } A2: \text{trans(r)}
and A3: B \in \text{Fin}(X)
shows B = 0 \lor \text{HasAmaximum(r,B)}
proof -
  have 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.
  \quad x \notin A \land (A = 0 \lor \text{HasAmaximum(r,A)}) \longrightarrow (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:
assumes A1: \text{r is total on } X \text{ and } A2: \text{trans(r)}
and A3: A \in \text{Fin}(X) \text{ and } A4: x \in X \text{ and } 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 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:
assumes A1: \text{r is total on } X \text{ and } A2: \text{trans(r)}
and A3: B \in \text{Fin}(X)
shows B = 0 \lor \text{HasAminimum(r,B)}
proof -
  have 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.
  \quad x \notin A \land (A = 0 \lor \text{HasAminimum(r,A)}) \longrightarrow (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_1_T1:
assumes A1: \text{r is total on } X \text{ and } A2: \text{trans(r)}
and A3: B \in \text{Fin}(X)
shows IsBounded(B,r)
proof -
from A1 A2 A3 have B=0 ∨ HasAmaximum(r,B) B=0 ∨ HasAminimum(r,B)
using Finite_ZF_1_1_T1A Finite_ZF_1_1_T1B by auto
then have
  B = 0 ∨ IsBoundedBelow(B,r) B = 0 ∨ 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 ∈ Fin(X) and A3: A≠0
shows
  Maximum(r,A) ∈ A
  Minimum(r,A) ∈ A
  ∀ x∈A. ⟨x,Maximum(r,A)⟩ ∈ r
  ∀ x∈A. ⟨Minimum(r,A),x⟩ ∈ 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) ∈ A
  Minimum(r,A) ∈ A
  ∀ x∈A. ⟨x,Maximum(r,A)⟩ ∈ r ∀ x∈A. ⟨Minimum(r,A),x⟩ ∈ 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∈X b∈X c∈X
shows
  Maximum(r,{a,b,c}) ∈ {a,b,c}
  Minimum(r,{a,b,c}) ∈ {a,b,c}
  Maximum(r,{a,b,c}) ∈ X
  Minimum(r,{a,b,c}) ∈ X
  ⟨a,Maximum(r,{a,b,c})⟩ ∈ r
  ⟨b,Maximum(r,{a,b,c})⟩ ∈ r
  ⟨c,Maximum(r,{a,b,c})⟩ ∈ r
proof -
from A2 have I: {a,b,c} ∈ Fin(X) {a,b,c} ≠ 0
by auto
with A1 show II: Maximum(r,\{a,b,c\}) ∈ \{a,b,c\}
  by (rule Finite_ZF_1_T2)
moreover from A1 I show III: Minimum(r,\{a,b,c\}) ∈ \{a,b,c\}
  by (rule Finite_ZF_1_T2)
moreover from A2 have \{a,b,c\} ⊆ X
  by auto
ultimately show
  Maximum(r,\{a,b,c\}) ∈ X
  Minimum(r,\{a,b,c\}) ∈ X
  by auto
from A1 I have ∀x∈\{a,b,c\}. ⟨x,Maximum(r,\{a,b,c\})⟩ ∈ r
  by (rule Finite_ZF_1_T2)
then show
  ⟨a,Maximum(r,\{a,b,c\})⟩ ∈ r
  ⟨b,Maximum(r,\{a,b,c\})⟩ ∈ r
  ⟨c,Maximum(r,\{a,b,c\})⟩ ∈ 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 finite. Works for relations that are total, transitive and antisym-
metric.

lemma Finite_ZF_1_1_L3:
  assumes A1: r {is total on} X
  and A2: trans(r) and A3: antisym(r)
  and A4: r ⊆ X×X and A5: X≠0
  and A6: ∀x∈X. ∃a∈A. x≠a ∧ ⟨x,a⟩ ∈ r
  shows A /∈ Fin(X)
proof -
  from assms have ¬IsBounded(A,r)
    using Order_ZF_3_L14 IsBounded_def
    by simp
  with A1 A2 show A /∈ Fin(X)
    using Finite_ZF_1_T1 by auto
qed

end

16 Finite sets and order relations

theory FinOrd_ZF imports Finite_ZF func_ZF_1

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 devel-
opment is based on the notion of finite powerset defined in Finite_ZF rather
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:**

- **Assumptions:**
  - \( r \) (is total on) \( X \) and \( A2: \text{trans}(r) \)
  - \( A3: B \in \text{FinPow}(X) \) and \( A4: B \neq 0 \)

- **Shows:** \( \text{HasAmaximum}(r,B) \)

**Proof:***

- **Have:**
  \( 0 = 0 \lor \text{HasAmaximum}(r,0) \) by simp

- **Moreover have:**
  \( \forall A \in \text{FinPow}(X). A = 0 \lor \text{HasAmaximum}(r,A) \rightarrow (\forall x \in A. (A \cup \{x\}) = 0 \lor \text{HasAmaximum}(r,A \cup \{x\})) \)

**Proof:***

- **Fix** \( A \)
  - **Assume:** \( A \in \text{FinPow}(X). A = 0 \lor \text{HasAmaximum}(r,A) \)
  - **Have:** \( \forall x \in A. (A \cup \{x\}) = 0 \lor \text{HasAmaximum}(r,A \cup \{x\}) \)

**Proof:***

- **Fix** \( x \)
  - **Assume:** \( x \in X \)
  - **Note:** \( A = 0 \lor \text{HasAmaximum}(r,A) \)

**Moreover:***

- **Have:** \( 0 = 0 \lor \text{HasAmaximum}(r,0) \) by simp
- **Use:** \( \text{total_is_refl} \)
  - **Have:** \( A \in \text{FinPow}(X) \)
  - **Have:** \( x \in X \)
  - **Use:** \( \text{FinPow_def} \)
  - **Use:** \( \text{Order_ZF}4\_L9 \)

**Proof:***

- **Fix** \( A \)
  - **Assume:** \( A \in \text{FinPow}(X) \)
  - **Assume:** \( A = 0 \lor \text{HasAmaximum}(r,A) \)

**Moreover:***

- **Have:** \( A = 0 \lor \text{HasAmaximum}(r,A) \)
- **Use:** \( \text{total_is_refl} \)
  - **Have:** \( A \in \text{FinPow}(X) \)
  - **Have:** \( x \in X \)
  - **Use:** \( \text{FinPow_def} \)
  - **Use:** \( \text{Order_ZF}4\_L9 \)

**Ultimate:***

- **Have:** \( A = 0 \lor \text{HasAmaximum}(r,A \cup \{x\}) \)
- **Use:** \( \text{auto} \)

**Thus:** \( \forall x \in X. (A \cup \{x\}) = 0 \lor \text{HasAmaximum}(r,A \cup \{x\}) \)

**Proof:***

- **Thus:** \( \forall x \in X. (A \cup \{x\}) = 0 \lor \text{HasAmaximum}(r,A \cup \{x\}) \)
- **End.***
For linearly ordered nonempty finite sets the maximum is in the set and indeed it is the greatest element of the set.

**lemma** linord_maxProps: 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

### 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** empty_ordiso: shows ord_iso(0,r,0,R) ≠ 0
**proof** -
have 0 ≈ 0 using eqpoll_refl by simp
then obtain f where f ∈ bij(0,0)
using eqpoll_def by blast
then show thesis using ord_iso_def by auto
qed

Even weirder than empty_ordiso The order automorphism of the empty set is unique.

**lemma** empty_ordiso_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)
  qed

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 -
  { fix A assume A ∈ FinPow(X) and A ≠ 0
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
} then have ∀A ∈ FinPow(X).
  A = 0 ∨ (∃a∈A. A = {a} ∨ P(A-{a}) → P(A))
  by auto
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 -
  note A1 A2
  moreover have
  ∀C ∈ FinPow(X). C ≠ 0 → (P(C - {Maximum(r,C)}) → P(C))
  proof -
  { fix C assume C ∈ FinPow(X) and C ≠ 0
    let x = Maximum(r,C)
    let A = C - {x}
    assume P(A)
    moreover from C ∈ FinPow(X) have A ∈ FinPow(X)
      using fin_rem_point_fin by simp
    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
    moreover note A3
    ultimately have P(A ∪ {x}) by auto
    moreover from x ∈ C have A ∪ {x} = C
      by auto
    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
\{ \text{fix } A \text{ assume } A \in \text{FinPow}(X) \land A \neq 0 \}
\begin{align*}
  & \text{let } M = \text{Maximum}(r, A) \\
  & \text{let } A_0 = A - \{M\} \\
  & \text{assume } A = \{M\} \lor \text{ord}_\text{iso}(A_0, r, A_0, r) = \{\text{id}(A_0)\}
\end{align*}

moreover
\begin{align*}
  & \text{assume } A = \{M\} \\
  & \text{have } \text{ord}_\text{iso}(\{M\}, r, \{M\}, r) = \{\text{id}(\{M\})\}
\end{align*}

using \text{id}_\text{ord}_\text{auto}_\text{singleton} by simp

\begin{align*}
  & \text{with } \langle A = \{M\}\rangle \text{ have } \text{ord}_\text{iso}(A, r, A, r) = \{\text{id}(A)\}
\end{align*}

by simp

moreover
\begin{align*}
  & \text{assume } \text{ord}_\text{iso}(A_0, r, A_0, r) = \{\text{id}(A_0)\} \\
  & \text{have } \text{ord}_\text{iso}(A, r, A, r) = \{\text{id}(A)\}
\end{align*}

proof

show \{\text{id}(A)\} \subseteq \text{ord}_\text{iso}(A, r, A, r)

using \text{id}_\text{ord}_\text{iso} by simp

\begin{align*}
  & \text{fix } f \text{ assume } f \in \text{ord}_\text{iso}(A, r, A, r) \\
  & \text{with } A_1 \langle A \in \text{FinPow}(X) \rangle \land A \neq 0 \text{ have} \\
  & \quad \text{restrict}(f, A_0) \in \text{ord}_\text{iso}(A_0, r, A - \{f(M)\}, r) \\
  & \quad \text{using } \text{IsLinOrder}_\text{def} \ \text{fin}_\text{has}_\text{max} \ \text{ord}_\text{iso}_\text{rem}_\text{max} \ 	ext{by auto}
\end{align*}

\begin{align*}
  & \text{with } A_1 \langle A \in \text{FinPow}(X) \rangle \land A \neq 0 \land f \in \text{ord}_\text{iso}(A, r, A, r) \\
  & \quad \langle \text{ord}_\text{iso}(A_0, r, A_0, r) = \{\text{id}(A_0)\}\rangle \\
  & \quad \text{have } \text{restrict}(f, A_0) = \text{id}(A_0) \\
  & \quad \text{using } \text{IsLinOrder}_\text{def} \ \text{fin}_\text{has}_\text{max} \ \text{max}_\text{auto}_\text{fixpoint} \ 	ext{by auto}
\end{align*}

moreover from A1 \langle f \in \text{ord}_\text{iso}(A, r, A, r) \rangle
\begin{align*}
  & < A \in \text{FinPow}(X) > < A \neq 0 > \text{ have} \\
  & f : A \rightarrow A \text{ and } M \in A \text{ and } f(M) = M \\
  & \text{using } \text{ord}_\text{iso}_\text{def} \ \text{bij}_\text{is}_\text{fun} \ \text{IsLinOrder}_\text{def} \\
  & \quad \text{fin}_\text{has}_\text{max} \ \text{Order}_\text{ZF}_4\_L3 \ \text{max}_\text{auto}_\text{fixpoint} \ 	ext{by auto}
\end{align*}

ultimately have \( f = \text{id}(A) \) using \text{id}_\text{fixpoint}_\text{rem} by simp

\begin{align*}
  & \text{then show } \text{ord}_\text{iso}(A, r, A, r) \subseteq \{\text{id}(A)\}
\end{align*}

by auto

qed

ultimately have \text{ord}_\text{iso}(A, r, A, r) = \{\text{id}(A)\}

by auto

} then have \( \forall A \in \text{FinPow}(X) \). A = 0 \lor
\begin{align*}
  & (A = \{\text{Maximum}(r, A)\} \lor \\
  & \text{ord}_\text{iso}(A - \{\text{Maximum}(r, A)\}, r, A - \{\text{Maximum}(r, A)\}, r) = \\
  & \{\text{id}(A - \{\text{Maximum}(r, A)\})\} \rightarrow \text{ord}_\text{iso}(A, r, A, r) = \{\text{id}(A)\})
\end{align*}

by auto

moreover note A2 A3

161
ultimately show ord_iso(B,r,B,r) = {id(B)}
    by (rule fin_ord_ind)
qed

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.

lemma fin_order_iso:
    assumes A1: IsLinOrder(X,r) IsLinOrder(Y,R) and
    A2: n ∈ nat
    shows ∀ A ∈ FinPow(X). ∀ B ∈ FinPow(Y).
        A ≈ n ∧ B ≈ n −→ ord_iso(A,r,B,R) ≠ 0
proof -
    note A2 moreover have ∀ A ∈ FinPow(X). ∀ B ∈ FinPow(Y).
            A ≈ 0 ∧ B ≈ 0 −→ ord_iso(A,r,B,R) ≠ 0
        using eqpoll_0_is_0 empty_ord_iso by blast
    moreover have ∀ k ∈ nat.
            (∀ A ∈ FinPow(X). ∀ B ∈ FinPow(Y).
                A ≈ k ∧ B ≈ k −→ ord_iso(A,r,B,R) ≠ 0) −→
            (∀ C ∈ FinPow(X). ∀ D ∈ FinPow(Y).
                C ≈ succ(k) ∧ D ≈ succ(k) −→ ord_iso(C,r,D,R) ≠ 0)
proof -
    { fix k assume k ∈ nat
    assume A3: ∀ A ∈ FinPow(X). ∀ B ∈ FinPow(Y).
        A ≈ k ∧ B ≈ k −→ ord_iso(A,r,B,R) ≠ 0
    have ∀ C ∈ FinPow(X). ∀ D ∈ FinPow(Y).
        C ≈ succ(k) ∧ D ≈ succ(k) −→ ord_iso(C,r,D,R) ≠ 0
    proof -
    { fix C assume C ∈ FinPow(X)
    fix D assume D ∈ FinPow(Y)
    assume C ≈ succ(k) D ≈ succ(k)
    then have C ≠ 0 and D≠ 0
        using eqpoll_succ_imp_not_empty by auto
    let M_C = Maximum(r,C)
    let M_D = Maximum(R,D)
    let C_0 = C - {M_C}
    let D_0 = D - {M_D}
    from <C ∈ FinPow(X)> have C ⊆ X
        using FinPow_def by simp
    with A1 have IsLinOrder(C,r)
        using ord_linear_subset by blast
    from <D ∈ FinPow(Y)> have D ⊆ Y
        using FinPow_def by simp
    with A1 have IsLinOrder(D,R)
        using ord_linear_subset by blast
    from A1 <C ∈ FinPow(X)> <D ∈ FinPow(Y)>
        <C ≠ 0> <D≠ 0> have
            HasAmaximum(r,C) and HasAmaximum(R,D)

Every two finite linearly ordered sets are order isomorphic.

**lemma fin_ord_iso_ex:**
assumes \( A1: \text{IsLinOrder}(X,r) \) \( \text{IsLinOrder}(Y,R) \) and
\( A2: A \in \text{FinPow}(X) \) \( B \in \text{FinPow}(Y) \) and \( A3: B \approx A \)
says \( \exists ! f. f \in \text{ord_iso}(A,r,B,R) \)
proof
- from \( A2 \) obtain \( n \) where \( n \in \text{nat} \) and \( A \approx n \)
  using \( \text{finpow_decomp} \) by auto
- from \( A3 <A \approx n> \) have \( B \approx n \) by (rule \( \text{eqpoll_trans} \))
  with \( A1 A2 <A \approx n> <n \in \text{nat}> \) show \( \text{ord_iso}(A,r,B,R) \neq 0 \)
    using \( \text{fin_order_iso} \) 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 \( A1: \text{IsLinOrder}(X,r) \) \( \text{IsLinOrder}(Y,R) \) and
\( A2: A \in \text{FinPow}(X) \) \( B \in \text{FinPow}(Y) \) and \( A3: B \approx A \)
says \( \exists f. f \in \text{ord_iso}(A,r,B,R) \)
proof
- from \( \text{assms} \) show \( \exists f. f \in \text{ord_iso}(A,r,B,R) \)
  using \( \text{fin_ord_iso_ex} \) by blast
- fix \( f \ g \)
  assume \( A4: f \in \text{ord_iso}(A,r,B,R) \) \( g \in \text{ord_iso}(A,r,B,R) \)
  then have \( \text{converse}(g) \in \text{ord_iso}(B,R,A,r) \)
    using \( \text{ord_iso_sym} \) by simp
  with \( <f \in \text{ord_iso}(A,r,B,R)> \) 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

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 ⊆ X × X and a function f : X → 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

164
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) ≡ (∀x y. ⟨x,y⟩ ∈ r → ⟨f(x),f(y)⟩ ∈ 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) ≡ {⟨c,⋃x∈c. r{f(x)}⟩. c ∈ (A//r)}

Elements of equivalence classes belong to the set.

lemma EquivClass_1_L1:
assumes A1: equiv(A,r) and A2: C ∈ A//r and A3: x∈C
shows x∈A
proof -
from A2 have C ⊆ ∪ (A//r) by auto
with A1 A3 show x∈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 ⊆ X shows {r{x}. x∈A} ⊆ 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

165
Elements that belong to the same equivalence class are equivalent.

**Lemma** `EquivClass_1_L2A`:

Assumes `equiv(A,r)` and `C ∈ A//r`. If `x ∈ C` and `y ∈ C`, then `⟨x,y⟩ ∈ r`.

Proof:

1. From `A2`, `r{y} ∈ A//r` using `quotientI` by `simp`.
2. With `A1` and `A3`, use `EquivClass_1_L2` and `equiv_class_eq_iff` by `simp`.

**QED**

Every `x` is in the class of `y`, then they are equivalent.

**Lemma** `EquivClass_1_L2B`:

Assumes `equiv(A,r)`, `y ∈ A`, and `x ∈ r{y}`.

Proof:

1. From `A1` and `A3`, use `EquivClass_1_L2A` by `simp`.
2. With `A2`, use `Congruent_def` by `simp`.
3. With `A1`, use `equiv_class_eq` by `simp`.

**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 `equiv(A,r)`, `C ∈ A//r`, `x ∈ C`, and `Congruent(r,f)`.

Proof:

1. From `A1` and `A2`, use `equiv_class_eq` by `simp`.
2. With `A3`, use `equiv_class_eq` by `simp`.

**QED**

The values of congruent functions are in the space.

**Lemma** `EquivClass_1_L4`:

Assumes `equiv(A,r)`, `C ∈ A//r`, `x ∈ C`, and `Congruent(r,f)`.

Proof:

1. From `A1` and `A2`, use `equiv_def` by `simp`.
2. With `A1`, use `equiv_class_eq` by `simp`.

**QED**
with A1 show thesis using equiv_type by auto

qed

Equivalence classes are not empty.

lemma EquivClass_1_L5:
  assumes A1: refl(A,r) and A2: C ∈ A//r
  shows C ≠ 0
proof -
  from A2 obtain x where I: C = r{x} and x ∈ A
    using quotient_def by auto
  from A1 ⟨x ∈ A⟩ have x ∈ 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: equiv(A,r) and A2: Congruent(r,f)
  and A3: C ∈ 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 ≠ 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 \( \forall x y. x \in C \land y \in C \rightarrow r(f(x)) = r(f(y)) \)
    using EquivClass_1_L3 by blast
  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 functions). The predicate Congruent2 corresponds to congruent2 in Isabelle’s standard EquivClass theory, but uses ZF-functions rather than meta-functions.

definition
  Congruent2(r,f) ≡
  \( \langle \forall x_1 x_2 y_1 y_2. \langle x_1,x_2 \rangle \in r \land \langle y_1,y_2 \rangle \in r \rightarrow \langle f(x_1,y_1), f(x_2,y_2) \rangle \in r \rangle \)

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: C_1 ∈ A//r C_2 ∈ A//r
and A4: z_1 ∈ C_1×C_2 z_2 ∈ C_1×C_2
shows r{f(z_1)} = r{f(z_2)}

proof -
from A4 obtain x_1 y_1 x_2 y_2 where
  x_1∈C_1 and y_1∈C_2 and z_1 = ⟨x_1,y_1⟩ and
  x_2∈C_1 and y_2∈C_2 and z_2 = ⟨x_2,y_2⟩
by auto
with A1 A3 have ⟨x_1,x_2⟩ ∈ r and ⟨y_1,y_2⟩ ∈ r
using EquivClass_1_L2A by auto
with A2 have ⟨f(x_1,y_1),f(x_2,y_2)⟩ ∈ r
using Congruent2_def by simp
with A1 ⟨z_1 = ⟨x_1,y_1⟩⟩ ⟨z_2 = ⟨x_2,y_2⟩⟩ 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: C_1 ∈ A//r and A3: C_2 ∈ A//r
and A4: z ∈ C_1×C_2 and A5: Congruent2(r,f)
shows f(z) ∈ A

proof -
from A4 obtain x y where x∈C_1 and y∈C_2 and z = ⟨x,y⟩
by auto
with A1 A2 A3 have x∈A and y∈A
  using EquivClass_1_L1 by auto
with A1 A4 have ⟨x,x⟩ ∈ r and ⟨y,y⟩ ∈ r
  using equiv_def refl_def by auto
with A5 have ⟨f(x),f(y)⟩ ∈ r
  using Congruent2_def by simp
with A1 ⟨z = ⟨x,y⟩⟩ 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∈A y∈A
and A3: Congruent2(r,f)
shows \( f(x, y) \in A \)

proof -

from A1 A2 have \( r(x) \in A//r \) \( r(y) \in A//r \)

\( \langle x, y \rangle \in r(x) \times r(y) \)

using equiv_class_self quotientI by auto

with A1 A3 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 \( \bigcup z \in \text{fst}(p) \times \text{snd}(p). r(f(z)) \in A//r \)

proof -

from A3 have \( \text{fst}(p) \in A//r \) and \( \text{snd}(p) \in A//r \)

by auto

with A1 A2 have

I: \( \forall z \in \text{fst}(p) \times \text{snd}(p). f(z) \in A \)

using EquivClass_1_L8 by simp

from A3 A1 have \( \text{fst}(p) \times \text{snd}(p) \neq 0 \)

using equiv_def EquivClass_1_L5 Sigma_empty_iff

by auto

moreover from A1 I have

\( \forall z \in \text{fst}(p) \times \text{snd}(p). r(f(z)) \in A//r \)

using quotientI by simp

moreover from A1 A2 <\( \text{fst}(p) \in A//r \) <\( \text{snd}(p) \in A//r \) have

\( \forall z_1, z_2. z_1 \in \text{fst}(p) \times \text{snd}(p) \land z_2 \in \text{fst}(p) \times \text{snd}(p) \rightarrow r(f(z_1)) = r(f(z_2)) \)

using EquivClass_1_L7 by blast

ultimately show thesis by (rule ZF1_1_L2)

qed

Congruent functions of two variables can be projected.

theorem EquivClass_1_T1:

assumes equiv(A,r) Congruent2(r,f)

shows 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: equiv(A,r) and A2: Congruent2(r,f)

and A3: \( x \in A \) \( y \in A \)

shows ProjFun2(A,r,f)(r\{x\},r\{y\}) = 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

169
moreover have
\[ \forall z \in r\{x\} \times r\{y\}. \quad r\{f(z)\} = r\{f(x,y)\} \]

proof
fix \(z\) assume A4: \(z \in r\{x\} \times r\{y\}\)
from A1 A3 have
\[ r\{x\} \in A/r \quad r\{y\} \in A/r \]
\[ \langle x,y \rangle \in r\{x\} \times r\{y\} \]
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
\[ (\bigcup \{z \in r\{x\} \times r\{y\}. \quad r\{f(z)\}\}) = r\{f(x,y)\} \]
by (rule ZF1_1_L1)
moreover have
\[ \text{ProjFun2}(A,r,f)(r\{x\},r\{y\}) = (\bigcup \{z \in r\{x\} \times r\{y\}. \quad r\{f(z)\}\}) \]
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
A1: \(\text{equiv}(A,r)\) and A2: \(\text{Congruent2}(r,f)\)
and A3: \(f\) {is commutative on} \(A\)
and A4: \(c1 \in A/r\) \(c2 \in A/r\)
shows \(\text{ProjFun2}(A,r,f)(c1,c2) = \text{ProjFun2}(A,r,f)(c2,c1)\)
proof -
from A4 obtain \(x\) \(y\) where D1:
\[ c1 = r\{x\} \quad c2 = r\{y\} \]
\[ x \in A \quad y \in A \]
using quotient_def by auto
with A1 A2 have \(\text{ProjFun2}(A,r,f)(c1,c2) = r\{f(x,y)\}\)
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
\( r\{f(y,x)\} = \text{ProjFun2}(A,r,f)\langle c2,c1 \rangle \)
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)
and 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 A1: equiv(A,r) and A2: Congruent2(r,f)
and A3: f \{is associative on\} A
and A4: c1 ∈ A//r c2 ∈ A//r c3 ∈ A//r
and A5: g = ProjFun2(A,r,f)
shows g\langle g\langle c1,c2 \rangle,c3 \rangle = g\langle c1,g\langle c2,c3 \rangle \rangle
proof -
from A4 obtain x y z where D1:
\( c1 = r\{x\} \quad c2 = r\{y\} \quad c3 = r\{z\} \)
x∈A y∈A z∈A
using quotient_def by auto
with A3 have T1:f\{x,y\} ∈ A f\{y,z\} ∈ A
using IsAssociative_def apply_type by auto
with A1 A2 D1 A5 have
\( g\langle g\langle c1,c2 \rangle,c3 \rangle = r\{f\langle f\{x,y\},z \rangle \} \)
using EquivClass_1_L10 by simp
also from D1 A3 have
\( \ldots = r\{f\langle x,f\langle y,z \rangle \rangle \} \)
using IsAssoiciative_def by simp
also from T1 A1 A2 D1 A5 have
\( \ldots = g\langle c1,g\langle c2,c3 \rangle \rangle \)
using EquivClass_1_L10 by simp
finally show thesis by simp
qed

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

171
\( g \in (A/r) \times (A/r) \rightarrow A/r \)

using \texttt{EquivClass\_1\_T1} by \texttt{simp}

moreover from A1 A2 A3 have

\[ \forall c1 \in A/r. \forall c2 \in A/r. \forall c3 \in A/r. \\
g(g(c1,c2),c3) = g(c1,g(c2,c3)) \]

using \texttt{EquivClass\_2\_L2} by \texttt{simp}

ultimately show thesis

using \texttt{IsAssociative\_def} by \texttt{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.

{\textbf{lemma} \texttt{EquivClass\_2\_L3:}}

\begin{enumerate}
\item \texttt{assumes A1: IsDistributive(X,A,M)}
\item \texttt{and A2: equiv(X,r)}
\item \texttt{and A3: Congruent2(r,A) Congruent2(r,M)}
\item \texttt{and A4: a \in X/r \ b \in X/r \ c \in X/r}
\item \texttt{and A5: A_p = ProjFun2(X,r,A) M_p = ProjFun2(X,r,M)}
\item \texttt{shows M_p(a,A_p(b,c)) = A_p( M_p(a,b), M_p(a,c)) \land}
\item \texttt{M_p(A_p(b,c),a) = A_p( M_p(b,a), M_p(c,a))}
\end{enumerate}

\texttt{proof}

\begin{enumerate}
\item \texttt{from A4 obtain x y z where x \in X \ y \in X \ z \in X}
\item \texttt{a = r(x) \ b = r(y) \ c = r(z)}
\item \texttt{using quotient\_def by \texttt{auto}}
\item \texttt{with A1 A2 A3 A5 show}
\item \texttt{M_p(a,A_p(b,c)) = A_p( M_p(a,b), M_p(a,c)) \land}
\item \texttt{M_p(A_p(b,c),a) = A_p( M_p(b,a), M_p(c,a))}
\item \texttt{using \texttt{EquivClass\_1\_L8A} \texttt{EquivClass\_1\_L10} \texttt{IsDistributive\_def}}
\item \texttt{by \texttt{auto}}
\end{enumerate}

\texttt{qed}

Distributivity is preserved by projections to quotient spaces, provided both operations are congruent with respect to the equivalence relation.

{\textbf{lemma} \texttt{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))}

\texttt{proof-}

\begin{enumerate}
\item \texttt{let A_p = ProjFun2(X,r,A)}
\item \texttt{let M_p = ProjFun2(X,r,M)}
\item \texttt{from A1 A2 A3 have}
\item \texttt{\forall a \in X/r. \forall b \in X/r. \forall c \in X/r.}
\item \texttt{M_p(a,A_p(b,c)) = A_p( M_p(a,b), M_p(a,c)) \land}
\item \texttt{M_p(A_p(b,c),a) = A_p( M_p(b,a), M_p(c,a))}
\item \texttt{using \texttt{EquivClass\_2\_L3} by \texttt{simp}}
\item \texttt{then show thesis using \texttt{IsDistributive\_def} by \texttt{simp}}
\end{enumerate}

\texttt{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) O r)(A)
  using IsSaturated_def vimage_def image_comp
  by simp
also from A1 have ... = r(A)
  using equiv_comp_eq by simp
finally show A = r(A) by simp
next assume A = r(A)
  with A1 have A = (converse(r) O 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 A1: equiv(X,r) and A2: A⊆X
shows A ⊆ r(A)
proof
  fix a assume a∈A
  with A1 A2 have a ∈ r{a}
    using equiv_class_self by auto
  with <a∈A> show a ∈ r(A) by auto
qed
The next lemma shows that if "∼" 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 \( A \subseteq X \) and \( r \subseteq X \times X \)

\( a \in A \) and \( a \sim b \) implies \( b \in A \)

shows IsSaturated \((r, A)\)

**proof** -

from A2 A4 have \( r(A) \subseteq A \)

using image_iff by blast

moreover from A1 A3 have \( A \subseteq r(A) \)

using `EquivClass_3_L2` by simp

ultimately have \( A = r(A) \) by auto

with A1 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 \( A \subseteq X \) and \( r \subseteq X \times X \)

\( (x, y) \in r \)

shows \( x \in A \)

**proof** -

from A2 A5 have \( x \in r(x) \)

using `equiv_class_self` by simp

with A1 A3 A4 A5 have \( x \in r(A) \)

using `equiv_class_eq` `equiv_class_self` by auto

with A1 A2 show \( x \in A \)

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 \( A \subseteq X \) and \( r \subseteq X \times X \)

\( (x, y) \in r \)

shows \( x \in A \) \( \longleftrightarrow \) \( y \in A \)

**proof**

assume \( y \in A \)

with assms show \( x \in A \) using `EquivClass_3_L4` by simp

next assume \( x \in A \)

from A5 have \( (y, x) \in r \)

using `equiv_is_sym` by blast

with A1 A2 A3 A4 \( \langle x \in A \rangle \) show \( y \in A \)

174
using EquivClass_3_L4 by simp

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 $A3$: $A \subseteq X$ and $A4$: $x \in X$ and $A5$: $B = \{r\{x\}. \ x \in A\}$

shows $x \in A \longleftrightarrow r\{x\} \in B$

**proof**

assume $x \in A$

with $A5$ show $r\{x\} \in B$ by auto

next assume $r\{x\} \in B$

with $A5$ obtain $y$ where $y \in A$ and $r\{x\} = r\{y\}$

by auto

with $A1 A3$ have $(x, y) \in r$

using eq_equiv_class by auto

with $A1 A2 A3 A4$ <y in A> show $x \in A$

using EquivClass_3_L4 by simp

qed

A technical lemma involving a projection of a saturated set and a logical epression with exclusive or. Note that we don’t really care what $Xor$ is here, this is true for any predicate.

**lemma** EquivClass_3_L7: assumes equiv($X$, $r$) and IsSaturated($r$, $A$) and $A \subseteq X$

and $B = \{r\{x\}. \ x \in A\}$

and $(x \in A) \ Xor (y \in A)$

shows $(r\{x\} \in B) \ Xor (r\{y\} \in B)$

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 $n \rightarrow X$, 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**

\[
\text{ShiftedSeq}(b, n) \equiv \{(j, b(j \#- n)) \mid j \in \text{NatInterval}(n, \text{domain}(b))\}
\]

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.

**Definition**

\[
\text{Tail}(a) \equiv \{(k, a(\text{succ}(k))) \mid k \in \text{pred}(\text{domain}(a))\}
\]

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 \(\text{Init}\).

**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 \(\text{Append}\).

**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 \(\{0, 1, \ldots, n-1\} = n\)) 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.
definition
Last(a) ≡ a(prep(domain(a)))

Shifted sequence is a function on a the interval of natural numbers.

lemma shifted_seq_props:
assumes A1: n ∈ nat k ∈ nat and A2: b:k→X
shows
ShiftedSeq(b,n): NatInterval(n,k) → X
∀ i ∈ NatInterval(n,k). ShiftedSeq(b,n)(i) = b(i # n)
∀ j∈k. ShiftedSeq(b,n)(n +# j) = b(j)
proof -
let I = NatInterval(n,domain(b))
from A2 have Fact: I = NatInterval(n,k) using func1_1_L1 by simp
with A1 A2 have ∀ j∈ I. b(j # n) ∈ X
using inter_diff_in_len apply_funtype by simp
then have
{⟨j, b(j # n)⟩. j ∈ I} : I → X by (rule ZF_fun_from_total)
with Fact show thesis_1: ShiftedSeq(b,n): NatInterval(n,k) → X
using ShiftedSeq_def by simp
{ fix i from Fact thesis_1 have ShiftedSeq(b,n): I → X by simp
moreover
assume i ∈ NatInterval(n,k)
with Fact have i ∈ I by simp
moreover from Fact have
ShiftedSeq(b,n) = {⟨i, b(i # n)⟩. i ∈ I}
using ShiftedSeq_def by simp
ultimately have ShiftedSeq(b,n)(i) = b(i # n)
by (rule ZF_fun_from_tot_val)
} then show thesis1:
∀ i ∈ NatInterval(n,k). ShiftedSeq(b,n)(i) = b(i # n)
by simp
{ fix j
let i = n +# j
assume A3: j∈k
with A1 have j ∈ nat using elem_nat_is_nat by blast
then have i # n = j using diff_add_inverse by simp
with A3 thesis1 have ShiftedSeq(b,n)(i) = b(j)
using NatInterval_def by auto
} then show ∀ j∈k. ShiftedSeq(b,n)(n +# j) = b(j)
by simp
qed

Basis properties of the concatenation of two finite sequences.

theorem concat_props:
assumes A1: n ∈ nat k ∈ nat and A2: a:n→X b:k→X
shows
Concat(a,b): n #+ k → X
∀ i∈n. Concat(a,b)(i) = a(i)
∀ i ∈ NatInterval(n,k). Concat(a,b)(i) = b(i #- n)
∀ j ∈ k. Concat(a,b)(n #+ j) = b(j)

proof -
from A1 A2 have
  a:n→X and I: ShiftedSeq(b,n): NatInterval(n,k) → X
  and n ∩ NatInterval(n,k) = 0
  using shifted_seq_props length_start_decomp by auto
then have
  a ∪ ShiftedSeq(b,n): n ∪ NatInterval(n,k) → X ∪ X
  by (rule fun_disjoint_Un)
with A1 A2 show Concat(a,b): n #+ k → X
  using func1_1_L1 Concat_def length_start_decomp by auto
{ fix i assume i ∈ n
  with A1 I have i /∈ domain(ShiftedSeq(b,n))
  using length_start_decomp func1_1_L1 by auto
  with A2 have Concat(a,b)(i) = a(i)
  using func1_1_L1 fun_disjoint_apply1 Concat_def by simp
} thus ∀i∈n. Concat(a,b)(i) = a(i) by simp
{ fix i assume A3: i ∈ NatInterval(n,k)
  with A1 A2 have i /∈ domain(a)
  using length_start_decomp func1_1_L1 by auto
  with A1 A2 A3 have Concat(a,b)(i) = b(i #- n)
  using func1_1_L1 fun_disjoint_apply2 Concat_def shifted_seq_props
  by simp
} thus II: ∀i ∈ NatInterval(n,k). Concat(a,b)(i) = b(i #- n)
  by simp
{ fix j
  let i = n #+ j
  assume A3: j∈k
  with A1 have j ∈ nat using elem_nat_is_nat by blast
  then have i #- n = j using diff_add_inverse by simp
  with A3 II have Concat(a,b)(i) = b(j)
  using NatInterval_def by auto
} thus ∀j ∈ k. Concat(a,b)(n #+ j) = b(j)
  by simp
qed

Properties of concatenating three lists.

lemma concat_concat_list:
  assumes A1: n ∈ nat k ∈ nat m ∈ nat and
  A2: a:n→X b:k→X c:m→X and
  A3: d = Concat(Concat(a,b),c)
  shows
  d : n #+k #+ m → X
  ∀ j ∈ n. d(j) = a(j)
  ∀ j ∈ k. d(n #+ j) = b(j)
  ∀ j ∈ m. d(n #+ k #+ j) = c(j)

proof -
from A1 A2 have I:
Properties of concatenating a list with a concatenation of two other lists.

lemma concat_list_concat:
  assumes A1: n ∈ nat k ∈ nat m ∈ nat and
  A2: a:n→X b:k→X c:m→X and
  A3: e = Concat(a, Concat(b,c))
  shows
e : n #+k #+ m → X
∀ j ∈ n. e(j) = a(j)
∀ j ∈ k. e(n #+ j) = b(j)
∀ j ∈ m. e(n #+ k #+ j) = c(j)
proof -
  from A1 A2 have I:
n ∈ nat k #+ m ∈ nat
  a:n→X Concat(b,c): k #+ m → X
  using concat_props by auto
  with A3 show e : n #+k #+ m → X
    using concat_props add_assoc by simp
  from I have ∀ j ∈ n. Concat(a, Concat(b,c))(j) = a(j)
    by (rule concat_props)
  with A3 show ∀ j ∈ n. e(j) = a(j) by simp
  from I have II:

\[ \forall j \in k \#+ m. \text{Concat}(a, \text{Concat}(b,c))(n \#+ j) = \text{Concat}(b,c)(j) \]
by (rule \texttt{concat_props})

\[
\{ \text{fix } j \text{ assume } A4: j \in k \\
\text{moreover from } A1 \text{ have } k \subseteq k \#+ m \text{ using add_nat_le by simp} \\
\text{ultimately have } j \in k \#+ m \text{ by auto} \\
\text{with } A3 \text{ II have } e(n \#+ j) = \text{Concat}(b,c)(j) \text{ by simp} \\
\text{also from } A1 A2 A4 \text{ have } ... = b(j) \\
\text{using \texttt{concat_props} by simp} \\
\text{finally have } e(n \#+ j) = b(j) \text{ by simp} \\
\} \text{ thus } \forall j \in k. e(n \#+ j) = b(j) \text{ by simp} \\
\{ \text{fix } j \text{ assume } A5: j \in m \\
\text{with } A1 II A3 \text{ have } e(n \#+ k \#+ j) = \text{Concat}(b,c)(k \#+ j) \\
\text{using add_lt_mono add_assoc by simp} \\
\text{also from } A1 A2 A5 \text{ have } ... = c(j) \\
\text{using \texttt{concat_props} by simp} \\
\text{finally have } e(n \#+ k \#+ j) = c(j) \text{ by simp} \\
\} \text{ then show } \forall j \in m. e(n \#+ k \#+ j) = c(j) \\
\text{by simp} \\
\}
\]

\texttt{qed}

Concatenation is associative.

\textbf{theorem} \texttt{concat_assoc}:
\begin{enumerate}
\item \texttt{assumes } A1: n \in \texttt{nat} k \in \texttt{nat} m \in \texttt{nat} and
A2: \texttt{a:n\rightarrow X b:k\rightarrow X c:m\rightarrow X} \\
\item \texttt{shows} \texttt{Concat(Concat(a,b),c) = Concat(a,Concat(b,c))} \\
\item \texttt{proof} - \\
\texttt{let } d = \texttt{Concat(Concat(a,b),c)} \\
\texttt{let } e = \texttt{Concat(a,Concat(b,c))} \\
\texttt{from } A1 A2 \texttt{ have} \\
\texttt{d : n \#+k \#+ m \rightarrow X \text{ and } e : n \#+k \#+ m \rightarrow X} \\
\texttt{using concat_concat_list concat_list_concat by auto} \\
\texttt{moreover have } \forall i \in n \#+k \#+ m. d(i) = e(i) \\
\item \texttt{proof} - \\
\texttt{\{ fix } i \texttt{ assume } i \in n \#+k \#+ m \\
\texttt{moreover from } A1 \texttt{ have} \\
n \#+k \#+ m = n \cup \texttt{NatInterval(n,k)} \cup \texttt{NatInterval(n \#+ k,m)} \\
\texttt{using adjacent_intervals3 by simp} \\
\texttt{ultimately have} \\
i \in n \lor i \in \texttt{NatInterval(n,k)} \lor i \in \texttt{NatInterval(n \#+ k,m)} \\
\texttt{by simp} \\
\texttt{moreover} \\
\texttt{\{ assume } i \in n \\
\texttt{with } A1 A2 \texttt{ have } d(i) = e(i) \\
\texttt{using concat_concat_list concat_list_concat by simp } \\
\texttt{moreover} \\
\texttt{\{ assume } i \in \texttt{NatInterval(n,k)} \\
\texttt{then obtain } j \texttt{ where } j \in k \texttt{ and } i = n \#+ j \\
\texttt{using NatInterval_def by auto} \\
\texttt{with } A1 A2 \texttt{ have } d(i) = e(i) \\
\}
\end{enumerate}
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

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 → X and A2: x∈X and A3: b = Append(a,x)
shows b : succ(n) → X
∀k∈n. b(k) = a(k)
b(n) = x
proof -
note A1
moreover have I: n ∉ n using mem_not_refl by simp
moreover from A1 A3 have II: b = a ∪ {⟨n,x⟩}
using func1_1_L1 Append_def by simp
ultimately have b : n ∪ {n} → X ∪ {x}
by (rule func1_1_L11D)
with A2 show b : succ(n) → X
using succ_explained set_elem_add by simp
from A1 I II show ∀k∈n. b(k) = a(k) and b(n) = x
  using func1_1_L11D by auto
qed

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 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 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 }
      } thus thesis by simp
  qed
ultimately show \( b = c \) by (rule func_eq)

qed

Properties of \( \text{Init} \).

**theorem init_props:**
assumes \( A1: n \in \text{nat} \) and \( A2: a: \text{succ}(n) \to X \)
shows
\[ \text{Init}(a): n \to X \]
\[ \forall k \in n. \text{Init}(a)(k) = a(k) \]
\[ a = \text{Append}(\text{Init}(a), a(n)) \]

**proof**
- have \( n \subseteq \text{succ}(n) \) by auto
with \( A2 \) have \( \text{restrict}(a,n): n \to X \)
  using 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 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:
\[ \text{Init}(a): n \to X \quad a(n) \in X \]
\[ b = \text{Append}(\text{Init}(a), a(n)) \]
using 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 thesis2 \( A3 \) 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

If we take \( \text{init} \) of the result of append, we get back the same list.

**lemma init_append:**
assumes \( A1: n \in \text{nat} \) and \( A2: a: n \to X \) and \( A3: x \in X \)
shows \( \text{Init}(\text{Append}(a,x)) = a \)
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 \( \forall k\in n. \) Init(Append(a,x))(k) = Append(a,x)(k)
    using init_props by auto
  with A2 A3 have \( \forall k\in n. \) Init(Append(a,x))(k) = a(k) using append_props
    by simp
  with \( \langle \text{Init(Append(a,x)):n→X} \rangle \) A2 show thesis by (rule func_eq)
qed

A reformulation of definition of Init.

lemma init_def: assumes n ∈ nat and x:succ(n)→X
  shows Init(x) = restrict(x,n)
  using assms func1_1_L1 Init_def by simp

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) \}
  shows b: succ(n)→X
    \( \forall k\in n. \) b(k) = a(k)
    b(n) = y
  using assms Append_def func1_1_L1 append_props by auto

The next lemma is a bit displaced as it is mainly about finite sets. It is
proven here because it uses the notion of 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 \( A ∪ \{ x \} \) by appending x.
This is just a specialised version of lemma bij_extend_point from func1.thy.

lemma bij_append_point:
  assumes A1: n ∈ nat and A2: b ∈ bij(n,X) and A3: x /∈ X
  shows Append(b,x) ∈ bij(succ(n), X ∪ \{ x \})
proof -
  from A2 A3 have b ∪ \{ (n,x) \} ∈ bij(n ∪ \{ n \}, X ∪ \{ x \})
    using mem_not_refl bij_extend_point by simp
  moreover have Append(b,x) = b ∪ \{ (n,x) \}
  proof -
    from A2 have b:n→X
      using bij_def surj_def by simp
    then have b : n → X ∪ \{ x \} using func1_1_L1B
      by blast
    then show Append(b,x) = b ∪ \{ (n,x) \}
      using Append_def func1_1_L1 by simp
  qed
  ultimately show thesis using succ_explained by auto
qed
The next lemma rephrases the definition of \( \text{Last} \). Recall that in ZF we have \( \{0, 1, 2, \ldots, n\} = n + 1 = \text{succ}(n) \).

**lemma last_seq_elem:** assumes \( a : \text{succ}(n) \to X \) shows \( \text{Last}(a) = a(n) \)

using assms func1_1_L1 pred_succ_eq Last_def by simp

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 \( A1: n \in \text{nat} \) and \( A2: a : \text{succ}(n) \to X \) \( b : \text{succ}(n) \to X \) and \( A3: \text{restrict}(a, n) = \text{restrict}(b, n) \) and \( A4: a(n) = b(n) \)

shows \( a = b \)

proof -

\[
\begin{align*}
\{ & \text{fix } k \text{ assume } k \in \text{succ}(n) \\
& \text{then have } k \in n \lor k = n \text{ by auto} \\
& \text{moreover} \\
& \{ & \text{assume } k \in n \\
& \text{then have } \text{restrict}(a, n)(k) = a(k) \text{ and } \text{restrict}(b, n)(k) = b(k) \text{ by auto} \\
& \text{using restrict by auto} \\
& \{ & \text{assumes } A3 \text{ have } a(k) = b(k) \text{ by simp} \\
& \text{moreover} \\
& \{ & \text{assume } k = n \\
& \text{with } A4 \text{ have } a(k) = b(k) \text{ by simp} \\
& \text{ultimately have } a(k) = b(k) \text{ by auto} \\
& \} \text{ then have } \forall k \in \text{succ}(n). a(k) = b(k) \text{ by simp} \\
& \text{with } A2 \text{ show } a = b \text{ by (rule func_eq)}
\end{align*}
\]

qed

Concatenating a list of length 1 is the same as appending its first (and only) element. Recall that in ZF set theory \( 1 = \{0\} \).

**lemma append_1elem:** assumes \( A1: n \in \text{nat} \) and \( A2: a : n \to X \) \( A3: b : 1 \to X \)

shows \( \text{Concat}(a, b) = \text{Append}(a, b(0)) \)

proof -

let \( C = \text{Concat}(a, b) \)

let \( A = \text{Append}(a, b(0)) \)

from \( A1 \) \( A2 \) \( A3 \) have \( I: \\
\begin{align*}
& n \in \text{nat} \quad 1 \in \text{nat} \\
& a : n \to X \quad b : 1 \to X \text{ by auto} \\
& \text{have } C : \text{succ}(n) \to X \\
\end{align*}
\)

proof -

from \( I \) have \( C \circ 1 = X \)

by (rule concat_props)

with \( A1 \) show \( C : \text{succ}(n) \to X \) by simp

qed

moreover from \( A2 \) \( A3 \) have \( A : \text{succ}(n) \to X \)
using apply_funtype append Props by simp
moreover have \( \forall k \in \text{succ}(n). C(k) = A(k) \)
proof
fix \( k \) assume \( k \in \text{succ}(n) \)
moreover
{ assume \( k \in n \)
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 \( C(k) = A(k) \) by simp
}
moreover have \( C(n) = A(n) \)
proof
from \( I \) have \( \forall j \in 1. C(n #+ j) = b(j) \)
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

A simple lemma about lists of length 1.

lemma list_len1_singleton: assumes \( A1: x \in X \)
shows \( \{\langle 0,x \rangle\} : 1 \to X \)
proof
from \( A1 \) have \( \{\langle 0,x \rangle\} : \{0\} \to X \) using pair_func_singleton
by simp
moreover have \( \{0\} = 1 \) by auto
ultimately show thesis by simp
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
Appending an element is the same as concatenating with certain pair.

**Lemma:** append_concat_pair:
- Assumes \( n \in \mathbb{N} \) and \( a : n \rightarrow X \) and \( x \in X \)
- Shows \( \text{Append}(a,x) = \text{Concat}(a,\{0,x\}) \)
- 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: \( n \in \mathbb{N} \) \( k \in \mathbb{N} \) and
  A2: \( a : n \rightarrow X \) \( b : k \rightarrow X \) and
  A3: \( x \in X \)
- Shows \( \text{Append}(\text{Concat}(a,b),x) = \text{Concat}(a, \text{Append}(b,x)) \)
- Proof -
  - From A1 A2 A3 have
    - \( n \# + k \in \mathbb{N} \) \( \text{Concat}(a,b) : n \# + k \rightarrow X \) \( x \in X \)
    - Using concat_props by auto
  - Then have
    - \( \text{Append}(\text{Concat}(a,b),x) = \text{Concat}(\text{Concat}(a,b),\{0,x\}) \)
    - By (rule append_concat_pair)
  - Moreover
    - From A1 A2 A3 have
      - \( n \in \mathbb{N} \) \( k \in \mathbb{N} \) \( 1 \in \mathbb{N} \)
      - \( a : n \rightarrow X \) \( b : k \rightarrow X \) \( \{0,x\} : 1 \rightarrow X \)
      - Using list_len1_singleton by auto
    - Then have
      - \( \text{Concat}(\text{Concat}(a,b),\{0,x\}) = \text{Concat}(a, \text{Concat}(b,\{0,x\})) \)
      - By (rule concat_assoc)
    - Moreover from A1 A2 A3 have \( \text{Concat}(b,\{0,x\}) = \text{Append}(b,x) \)
      - Using list_len1_singleton append_1elem pair_val by simp
    - Ultimately show \( \text{Append}(\text{Concat}(a,b),x) = \text{Concat}(a, \text{Append}(b,x)) \)
      - By simp
- QED

An identity involving concatenating with init and appending the last element.

**Lemma:** concat_init_last_elem:
- Assumes \( n \in \mathbb{N} \) \( k \in \mathbb{N} \) and
  a: \( n \rightarrow X \) and \( b : \text{succ}(k) \rightarrow X \)
- Shows \( \text{Append}(\text{Concat}(a,\text{Init}(b)),b(k)) = \text{Concat}(a,b) \)
- 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 \( A_1: n \in \text{nat} \) and \( A_2: a : n \to X \) and\( A_3: x \in X \) and \( A_4: c : X \to Y \)
shows
\[
c \circ \text{Append}(a,x) : \text{succ}(n) \to Y
\]
\[
c \circ \text{Append}(a,x) = \text{Append}(c \circ a, c(x))
\]
proof -
- let \( b = \text{Append}(a,x) \)
- let \( d = \text{Append}(c \circ a, c(x)) \)
- from \( A_2 \) \( A_4 \) have \( c \circ a : n \to Y \)
  using \( \text{comp_fun} \) by \( \text{simp} \)
- from \( A_2 \) \( A_3 \) have \( b : \text{succ}(n) \to X \)
  using \( \text{append_props} \) by \( \text{simp} \)
- with \( A_4 \) show \( c \circ b : \text{succ}(n) \to Y \)
  using \( \text{comp_fun} \) by \( \text{simp} \)
moreover from \( A_3 \) \( A_4 \) \(<c \circ a : n \to Y>\) have \( d : \text{succ}(n) \to Y \)
  using \( \text{apply_funtype} \) \( \text{append_props} \) by \( \text{simp} \)
moreover have \( \forall k \in \text{succ}(n). (c \circ b) (k) = d(k) \)
proof -
- \{ fix \( k \) assume \( k \in \text{succ}(n) \)
  with \( <b : \text{succ}(n) \to X> \) have \( (c \circ b) (k) = c(b(k)) \)
  using \( \text{comp_fun_apply} \) by \( \text{simp} \)
  with \( A_2 \) \( A_3 \) \( A_4 \) \(<c \circ a : n \to Y> \) \(<c \circ a : n \to Y> \) \(<k \in \text{succ}(n)> \)
  have \( (c \circ b) (k) = d(k) \)
  using \( \text{append_props} \) \( \text{comp_fun_apply} \) \( \text{apply_funtype} \)
  by \( \text{auto} \)
  \} thus thesis by \( \text{simp} \)
qed
ultimately show \( c \circ b = d \) by \( \text{rule func_eq} \)
qed

A lemma about appending an element to a list defined by set comprehension.

**lemma** \text{set_list_append}:
assumes \( A_1: \forall i \in \text{succ}(k). b(i) \in X \) and
\( A_2: a = \{ \langle i,b(i) \rangle . i \in \text{succ}(k) \} \)
shows
\( a : \text{succ}(k) \to X \)
\( \{ \langle i,b(i) \rangle . i \in k \} : k \to X \)
\( a = \text{Append}(\{ \langle i,b(i) \rangle . i \in k \},b(k)) \)
proof -
- from \( A_1 \) have \( \{ \langle i,b(i) \rangle . i \in \text{succ}(k) \} : \text{succ}(k) \to X \)
  by \( \text{rule ZF_fun_from_total} \)
- with \( A_2 \) show \( a : \text{succ}(k) \to X \) by \( \text{simp} \)
- from \( A_1 \) have \( \forall i \in k. b(i) \in X \)
  by \( \text{simp} \)
then show \( \{ \langle i,b(i) \rangle . i \in k \} : k \to X \)
  by \( \text{rule ZF_fun_from_total} \)
- with \( A_2 \) show \( a = \text{Append}(\{ \langle i,b(i) \rangle . i \in k \},b(k)) \)

188
An induction theorem for lists.

**lemma list_induct:** assumes A1: \( \forall b \in 1 \rightarrow X. \ P(b) \) and
\[ A2: \forall b \in \text{NLists}(X). \ P(b) \rightarrow (\forall x \in X. \ P(\text{Append}(b, x))) \] and
A3: \( d \in \text{NLists}(X) \)
says \( P(d) \)
proof -
\[ \{ \text{fix } n \}
\]
moreover from A1 have \( \forall b \in \text{succ}(0) \rightarrow X. \ P(b) \) by simp

moreover have \( \forall k \in \text{nat}. \ ((\forall b \in \text{succ}(k) \rightarrow X. \ P(b)) \rightarrow (\forall c \in \text{succ}(\text{succ}(k)) \rightarrow X. \ P(c))) \)
proof -
\[ \{ \text{fix } k \text{ assume } k \in \text{nat assume } \forall b \in \text{succ}(k) \rightarrow X. \ P(b) \}
\]
have \( \forall c \in \text{succ}(\text{succ}(k)) \rightarrow X. \ P(c) \)
proof
\[ \{ \text{fix } c \text{ assume } c : \text{succ}(\text{succ}(k)) \rightarrow X \}
\]
let \( b = \text{Init}(c) \)
let \( x = c(\text{succ}(k)) \)
from \( <k \in \text{nat}> <c : \text{succ}(\text{succ}(k)) \rightarrow X> \text{ have } b : \text{succ}(k) \rightarrow X \)
using init_props by simp
with A2 \( <k \in \text{nat}> <\forall b \in \text{succ}(k) \rightarrow X. \ P(b) > \text{ have } \forall x \in X. \ P(\text{Append}(b, x)) \)
using NLists_def by auto
with \( <c : \text{succ}(\text{succ}(k)) \rightarrow X> \text{ have } P(\text{Append}(b, x)) \)
using apply_funtype
by simp

with \( <k \in \text{nat}> <c : \text{succ}(\text{succ}(k)) \rightarrow X> \text{ show } P(c) \)
using init_props by simp
qed

thus thesis 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 NLists_def by auto
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 \( (n + 1) \rightarrow X \) of lists of length \( n + 1 \) of elements of \( X \) and the cartesian product \( (n \rightarrow X) \times X \).

**lemma lists_cart_prod:** assumes \( n \in \text{nat} \)
says \( \{\langle x, \text{Init}(x), x(n)\rangle. \ x \in \text{succ}(n) \rightarrow X \} \in \text{bij}(\text{succ}(n) \rightarrow X, (n \rightarrow X) \times X) \)
proof -
\[ \{ \text{let } f = \{\langle x, \text{Init}(x), x(n)\rangle. \ x \in \text{succ}(n) \rightarrow X\} \}
\]
from assms have \( \forall x \in \text{succ}(n) \rightarrow X. \ (\text{Init}(x), x(n)) \in (n \rightarrow X) \times X \)
using init_props succ_iff apply_funtype by simp
then have I: \( f: (\text{succ}(n) \rightarrow X) \rightarrow ((\text{succ}(n) \rightarrow X) \times X) \) by (rule ZF_fun_from_total) moreover from \( I \) have \( \forall x \in \text{succ}(n) \rightarrow X. \forall y \in \text{succ}(n) \rightarrow X. f(x) = f(y) \) \( \rightarrow x = y \)

using ZF_fun_from_tot_val init_def finseq_restr_eq by auto moreover have \( \forall p \in (n \rightarrow X) \times X. \exists x : \text{succ}(n) \rightarrow X. f(x) = p \) proof 
- fix p assume p \( \in (n \rightarrow X) \times X \)
  let \( x = \text{Append}(\text{fst}(p), \text{snd}(p)) \)
  from \( I \) have \( x : \text{succ}(n) \rightarrow X \) using append_props by simp 
with \( I \) have \( f(x) = (\text{Init}(x), x(n)) \) using succ_iff ZF_fun_from_tot_val by simp 
moreover from \( I \) have \( \forall x \in (n \rightarrow X) \rightarrow X. f(x) = p \)
by auto 
ultimately have \( f(x) = (\text{fst}(p), \text{snd}(p)) \) by auto 
thus \( \exists x : \text{succ}(n) \rightarrow X. f(x) = p \) by auto 
qed 

moreover have \( \forall y \in X. \exists x : 1 \rightarrow X. f(x) = y \) proof 
- fix y assume y \( \in X \)
  let \( x = \{ (0, y) \} \)
  from \( I \) have \( x : 1 \rightarrow X \) and \( f(x) = y \) using list_singleton_pair by auto 
ultimately have \( x = y \) by simp 
thus \( \exists x : 1 \rightarrow X. f(x) = y \) by auto 
qed 

ultimately show thesis using inj_def surj_def bij_def by simp qed 

We can identify a set \( X \) with lists of length one of elements of \( X \).

**lemma singleton_list_bij:** shows \( \{ (x, x(0)). x \in 1 \rightarrow X \} \in \text{bij}(1 \rightarrow X, X) \)

proof - 
- let \( f = \{ (x, x(0)). x \in 1 \rightarrow X \} \)
  have \( \forall x \in 1 \rightarrow X. x(0) \in X \) using apply_funtype by simp 
then have I: \( f: (1 \rightarrow X) \rightarrow X \) by (rule ZF_fun_from_total) moreover have \( \forall x \in 1 \rightarrow X. \forall y \in 1 \rightarrow X. f(x) = f(y) \) \( \rightarrow x = y \)

proof - 
  \{ fix x y 
    assume x : 1 \rightarrow X y : 1 \rightarrow X and f(x) = f(y) 
    with I have x(0) = y(0) using ZF_fun_from_tot_val by auto 
    moreover from \( x : 1 \rightarrow X \) \( y : 1 \rightarrow X \) 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 \( \forall y \in X. \exists x \in 1 \rightarrow X. f(x) = y \) proof 
  fix y assume y \( \in X \)
  let \( x = \{ (0, y) \} \)
  from I have \( x : 1 \rightarrow X \) and \( f(x) = y \) using list_len1_singleton ZF_fun_from_tot_val pair_val by auto 
  thus \( \exists x \in 1 \rightarrow X. f(x) = y \) by auto 
qed 

ultimately show thesis using inj_def surj_def bij_def by simp qed 

190
We can identify a set of \( X \)-valued lists of length with \( X \).

**Lemma list_singleton_bij:** shows
\[
\{x,\{(0,x)\}.x\in X\} \in \text{bij}(X,1\rightarrow X) \text{ and } \\
\{\langle y,y(0)\>. y\in 1\rightarrow X\} = \text{converse}(\{x,\{(0,x)\}.x\in X\}) \text{ and } \\
\{x,\{(0,x)\}.x\in X\} = \text{converse}(\{\langle y,y(0)\>. y\in 1\rightarrow X\})
\]

**Proof** -
- let \( f = \{(y,y(0)). y\in 1\rightarrow X\} \)
- let \( g = \{\langle x,\{(0,x)\}\>.x\in X\} \)
- have \( 1 = \{0\} \) by auto
- then have \( f \in \text{bij}(1\rightarrow X, X) \) and \( g:X\rightarrow (1\rightarrow X) \)
  using singleton_list_bij pair_func_singleton ZF_fun_from_total by auto
  moreover have \( \forall y:1\rightarrow X.g(f(y)) = y \)
  proof
  - fix \( y \) assume \( y:1\rightarrow X \)
  - have \( f:(1\rightarrow X)\rightarrow X \) using singleton_list_bij bij_def inj_def by simp
  - with \( <1 = \{0\}> <y:1\rightarrow X> <g:X\rightarrow (1\rightarrow X)> \) show \( g(f(y)) = y \)
    using ZF_fun_from_tot_val apply_funtype func_singleton_pair by simp
  qed
- ultimately show \( g \in \text{bij}(X,1\rightarrow X) \) and \( f = \text{converse}(g) \) and \( g = \text{converse}(f) \)
  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 \( \{x\in 1\rightarrow X. x(0) \in U\} = \{ \{(0,y)\}. y\in U\} \)

**Proof**
- have \( 1 = \{0\} \) by auto
- \{ fix \( x \) assume \( x \in \{x\in 1\rightarrow X. x(0) \in U\} \)
  - with \( <1 = \{0\}> \) have \( x = \{(0, x(0))\} \) using func_singleton_pair by auto 
  \} thus \( \{x\in 1\rightarrow X. x(0) \in U\} \subseteq \{ \{(0,y)\}. y\in U\} \) 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 \( <1 = \{0\}> \) assms have \( x:1\rightarrow X \) using pair_func_singleton by auto 
  \} thus \( \{ \{(0,y)\}. y\in U\} \subseteq \{x\in 1\rightarrow X. x(0) \in U\} \) by auto
  qed

A technical lemma about extending a list by values from a set.

**Lemma list_append_from:** assumes \( A1: n \in \text{nat} \) and \( A2: U \subseteq n\rightarrow X \) and \( A3: V \subseteq X \)
shows \( \{x \in \text{succ}(n)\rightarrow X. \text{Init}(x) \in U \land x(n) \in V\} = (\bigcup y\in V.\{\text{Append}(x,y).x\in U\}) \)

**Proof** -
- \{ fix \( x \) assume \( x \in \{x \in \text{succ}(n)\rightarrow X. \text{Init}(x) \in U \land x(n) \in V\} \)
  - then have \( x \in \text{succ}(n)\rightarrow X \) and \( \text{Init}(x) \in U \) and \( I: x(n) \in V \)
by auto
let y = x(n)
from A1 and <x ∈ succ(n)→X> have x = Append(Init(x),y)
using init_props by simp
with I and <Init(x) ∈ U> have x ∈ (∪{y∈V. Append(a,y).a∈U}) by auto
}
moreover
{ fix x assume x ∈ (∪{y∈V. Append(a,y).a∈U})
then obtain a y where y∈V and a∈U and x = Append(a,y) by auto
with A2 A3 have x: succ(n)→X using append_props by blast
from A2 A3 <y∈V> <a∈U> have a:n→X and y∈X by auto
with A1 <a∈U> <y∈V> <x = Append(a,y)> have Init(x) ∈ U and x(n) ∈ V
using append_props init_append by auto
with <x: succ(n)→X> have x ∈ {x ∈ succ(n)→X. Init(x) ∈ U ∧ x(n) ∈ V} by auto
}
ultimately show thesis by blast
qed
end

19 Inductive sequences

theory InductiveSeq_ZF imports Nat_ZF_IML FiniteSeq_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.

19.1 Sequences defined by induction

One way of defining a sequence (that is a function \( a : \mathbb{N} \rightarrow X \)) 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.

First we define a helper notion of the sequence defined inductively up to a given natural number \( n \).

definition
InductiveSequenceN(x,f,n) ≡
THE a. a: succ(n) → domain(f) ∧ a(0) = x ∧ (∀k∈n. a(succ(k)) = f(a(k)))

From that we define the inductive sequence on the whole set of natural numbers for all \( n \).
numbers. Recall that in Isabelle/ZF the set of natural numbers is denoted \( \text{nat} \).

definition
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 \( n = \{0,1,..,n-1\} \) and \( \text{succ}(n) = \{0,1,..,n\} \).

lemma indseq_exun0: assumes \( A1: f: X\rightarrow X \) and \( A2: x\in X \) shows
\( \exists! a. a: \text{succ}(0)\rightarrow X \land a(0) = x \land (\forall k\in0. a(\text{succ}(k)) = f(a(k))) \)

proof
fix \( a \) \( b \) assume \( A3: a: \text{succ}(0)\rightarrow X \land a(0) = x \land (\forall k\in0. a(\text{succ}(k)) = f(a(k))) \)
moreover have \( \text{succ}(0) = \{0\} \) by auto
ultimately have \( a = \{0\} \rightarrow X \land b: \{0\} \rightarrow X \) by auto
then have \( a = \{(0, a(0))\} \land b = \{(0, b(0))\} \) 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 restrict_type2 by auto
moreover have \( \{0\} = \text{succ}(0) \) by auto
ultimately have \( a : \text{succ}(0) \rightarrow X \) by simp
with \( A1 \) show
\( \exists a. a: \text{succ}(0) \rightarrow X \land a(0) = x \land (\forall k\in0. a(\text{succ}(k)) = f(a(k))) \)
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 indseq_restrict:
assumes \( A1: f: X\rightarrow X \) and \( A2: x\in X \) and \( A3: n \in \text{nat} \) and
\( A4: a: \text{succ}(/\text{succ}(n))\rightarrow X \land a(0) = x \land (\forall k\in\text{succ}(n). a(\text{succ}(k)) = f(a(k))) \)
and \( A5: a_\tau = \text{restrict}(a,\text{succ}(n)) \)
shows
\( a_\tau: \text{succ}(n) \rightarrow X \land a_\tau(0) = x \land (\forall k\in n. a_\tau(\text{succ}(k)) = f(a_\tau(k))) \)

proof
from \( A3 \) have \( \text{succ}(n) \subseteq \text{succ}(\text{succ}(n)) \) by auto
with \( A4 \) \( A5 \) have \( a_\tau: \text{succ}(n) \rightarrow X \) using restrict_type2 by auto
moreover
from \( A3 \) have \( 0 \in \text{succ}(n) \) using empty_in_every_succ by simp

193
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 → X and A2: x ∈ X and A3: n ∈ nat and A4: ∃! a. a: succ(n) → X ∧ a(0) = x ∧ (∀k ∈ n. a(succ(k)) = f(a(k)))
shows
∃! a. a: succ(succ(n)) → X ∧ a(0) = x ∧ (∀k ∈ succ(n). a(succ(k)) = f(a(k)))
proof
fix a b assume A5: a: succ(succ(n)) → X ∧ a(0) = x ∧ (∀k ∈ succ(n). a(succ(k)) = f(a(k))) and A6: b: succ(succ(n)) → X ∧ b(0) = x ∧ (∀k ∈ succ(n). b(succ(k)) = f(b(k)))
show a = b
proof -
let a_r = restrict(a,succ(n))
let b_r = restrict(b,succ(n))
moreover have a_r = restrict(a,succ(n)) by simp
ultimately have I:
a_r: succ(n) → X ∧ a_r(0) = x ∧ (∀k ∈ n. a_r(succ(k)) = f(a_r(k)))
  by (rule indseq_restrict)
moreover have b_r = restrict(b,succ(n)) by simp
ultimately have b_r: succ(n) → X ∧ b_r(0) = x ∧ (∀k ∈ n. b_r(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) ∈ nat by simp
moreover from A5 A6 have a: succ(succ(n)) → X and b: succ(succ(n)) → 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 ∧
\[
( \forall k \in \text{succ}(n). \ a(\text{succ}(k)) = f(a(k)) )
\]

proof -
from A4 obtain a where III: a: succ(n) \to X and IV: a(0) = x and V: \forall k \in n. a(\text{succ}(k)) = f(a(k)) by auto
let b = a \cup \{(\text{succ}(n), f(a(n)))\}
from A1 III have
VI: b: \text{succ}(\text{succ}(n)) \to X and
VII: \forall k \in \text{succ}(n). b(k) = a(k) and
VIII: b(\text{succ}(n)) = f(a(n))
using apply_funtype finseq_extend by auto
from A3 have 0 \in \text{succ}(n) using empty_in_every_succ by simp
with IV VII have IX: b(0) = x by auto
\{ fix k assume k \in \text{succ}(n)
then have k\in n \lor k = n by auto
moreover
\{ assume A7: k \in n
with A3 VII have b(\text{succ}(k)) = a(\text{succ}(k))
using succ_ineq by auto
also from A7 V VII have a(\text{succ}(k)) = f(b(k)) by simp
finally have b(\text{succ}(k)) = f(b(k)) by simp \}
moreover
\{ assume A8: k = n
with VIII have b(\text{succ}(k)) = f(a(k)) by simp
with A8 VII VIII have b(\text{succ}(k)) = f(b(k)) by simp \}
ultimately have b(\text{succ}(k)) = f(b(k)) by auto
\} then have \forall k \in \text{succ}(n). b(\text{succ}(k)) = f(b(k)) by simp
with VI IX show thesis by auto
qed
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\to X and A2: x\in X and A3: n \in nat
shows
\exists! a. a: \text{succ}(n) \to X \land a(0) = x \land ( \forall k \in n. a(\text{succ}(k)) = f(a(k)) )
proof -
note A3
moreover from A1 A2 have
\exists! a. a: \text{succ}(0) \to X \land a(0) = x \land ( \forall k \in 0. a(\text{succ}(k)) = f(a(k)) )
using indseq_exun0 by simp
moreover from A1 A2 have \forall k \in nat.
\{ \exists! a. a: \text{succ}(k) \to X \land a(0) = x \land
( \forall i \in k. a(\text{succ}(i)) = f(a(i))) \}
using indseq_exun_ind by simp
ultimately show
\exists! a. a: \text{succ}(n) \to X \land a(0) = x \land ( \forall k \in n. a(\text{succ}(k)) = f(a(k)) )

195
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:**

1. 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**

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
  \( a: \text{succ}(j) \to X \) \( a(0) = x \) \( \forall k \in j. \ a(\text{succ}(k)) = f(a(k)) \)
  using \( \text{fin_indseq_props} \) by \( \text{auto} \)
with \( A3, A4 \) have
  \( b: \text{succ}(i) \to X \) \( b(0) = x \) \( \forall k \in i. \ b(\text{succ}(k)) = f(b(k)) \)
  using \( \text{succ_subset} \) \( \text{restrict_type2} \) \( \text{empty_in_every_succ} \) \( \text{restrict} \) \( \text{succ_ineq} \)
  by \( \text{auto} \)
moreover from \( A1, A2, A3 \) have
  \( c: \text{succ}(i) \to X \) \( c(0) = x \) \( \forall k \in i. \ c(\text{succ}(k)) = f(c(k)) \)
  using \( \text{fin_indseq_props} \) by \( \text{simp} \)
moreover from \( A1, A2, A3 \) have
  \( \exists! \ a. \ a: \text{succ}(i) \to X \) \( a(0) = x \) \( \forall k \in i. \ a(\text{succ}(k)) = f(a(k)) \)
  using \( \text{indseq_exun} \) by \( \text{simp} \)
ultimately show \( b = c \) by \( \text{blast} \)
\qed

For any two natural numbers one of the corresponding inductive sequences is contained in the other.

lemma \( \text{indseq_subsets} \): assumes \( A1: f: X \to X \) and \( A2: x \in X \) and \( A3: i \in \text{nat} \) \( j \in \text{nat} \) and \( A4: a = InductiveSequenceN(x,f,i) \) \( b = InductiveSequenceN(x,f,j) \)
shows \( a \subseteq b \lor b \subseteq a \)
proof -
  from \( A3 \) have \( i \subseteq j \lor j \subseteq i \) using \( \text{nat_incl_total} \) by \( \text{simp} \)
moreover
  \{ assume \( i \subseteq j \)
    with \( A1, A2, A3, A4 \) 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 \( A1, A2, A3, A4 \) 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 show \( a \subseteq b \lor b \subseteq a \) by \( \text{auto} \)
\qed

The first theorem about properties of infinite inductive sequences: inductive sequence is a indeed a sequence (i.e. a function on the set of natural numbers.

theorem \( \text{indseq_seq} \): assumes \( A1: f: X \to X \) and \( A2: x \in X \)
shows \( \text{InductiveSequence}(x,f) : \text{nat} \to X \)
proof -
  let \( S = \{ \text{InductiveSequenceN}(x,f,n) . n \in \text{nat} \} \)
  \{ 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 \( a : \text{succ}(n) \to X \) using fin_indseq_props by simp
then have \( \exists A \ B. \ a : A \to B \) by auto
\}
then have \( \forall a \in S. \ \exists A \ B. \ a : A \to B \) 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 indseq_subsets by simp
\}
then have \( \forall a \in S. \ a \subseteq b \lor b \subseteq a \) by auto
ultimately have \( \bigcup S : \text{domain}(\bigcup S) \to \text{range}(\bigcup S) \) using fun_Union by simp
with \( A1 \ A2 \) have \( I : \bigcup S : \text{nat} \to \text{range}(\bigcup S) \) using domain_UN fin_indseq_domain 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 \( \langle k,y \rangle \in (\bigcup S) \) by (rule func1_1_L5A)
then obtain \( n \) where \( n \in \text{nat} \) and \( \text{II:} \ \langle k,y \rangle \in \text{InductiveSequenceN}(x,f,n) \) by auto
with \( A1 \ A2 \) have \( \text{InductiveSequenceN}(x,f,n) : \text{succ}(n) \to X \) using fin_indseq_props by simp
with \( \text{II} \) have \( y \in X \) using 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 func1_1_L1A by blast
then show \( \text{InductiveSequence}(x,f) : \text{nat} \to X \) using 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 indseq_seq succnat_subset_nat by auto
then have \( \text{restrict}(a,\text{succ}(n)) : \text{succ}(n) \to X \)
using restrict_type2 by simp
moreover from A1 A2 A3 have b : succ(n) → X
using fin_indseq_props by simp
moreover
{ fix k assume A4: k ∈ succ(n)
  from A1 A2 A3 I have
  \( \bigcup S : \text{nat} → X \ b ∈ S \ b : \text{succ(n)} → X \)
  using InductiveSequence_def fin_indseq_props by auto
  with A4 have restrict(a,succ(n))(k) = b(k)
  using fun_Union_apply InductiveSequence_def restrict_if
  by simp
} then have ∀k ∈ succ(n). restrict(a,succ(n))(k) = b(k)
by simp
ultimately show thesis by (rule func_eq)
qed

The first element of the inductive sequence starting at \( x \) and generated by \( f \) is indeed \( x \).

\textbf{theorem indseq_valat0:} assumes A1: \( f: X \rightarrow X \) and A2: \( x \in X \)
shows InductiveSequence(x,f)(0) = x
\begin{proof}
  let a = InductiveSequence(x,f)
  let b = InductiveSequenceN(x,f,0)
  have T: 0 ∈ \text{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
\end{proof}

\textbf{An infinite inductive sequence satisfies the inductive relation that defines it.}

\textbf{theorem indseq_vals:}
assumes A1: \( f: X \rightarrow X \) and A2: \( x \in X \) and A3: \( n \in \text{nat} \)
shows InductiveSequence(x,f)(succ(n)) = f(InductiveSequence(x,f)(n))
\begin{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)) ∈ \text{nat}
  n ∈ succ(succ(n))
  by auto
  then have a(succ(n)) = restrict(a,succ(succ(n)))(succ(n))
  using restrict_if by simp
\end{proof}

199
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

19.2 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 \( \{ f^{(n)}(x) : n \in N \} \) for some \( x \) in the domain of \( f \), where \( f^{(n)} \) denotes the \( n \)'th iteration of the function \( f \). For a function \( f : X \to X \) and a point \( x \in X \) 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: \( f : X \to X \) and A2: \( x \in X \) and 
A3: \( A = \text{InductiveSequence}(x,f)(\text{nat}) \)

shows \( x \in A \) and \( \forall y \in A. f(y) \in A \)

**proof -**

let a = \( \text{InductiveSequence}(x,f) \)
from A1 A2 have a : nat \to X using indseq_seq
by simp
with A3 have I: \( A = \{ a(n). n \in \text{nat} \} \) using func_imagedef
by auto hence a(0) \in A by auto
with A1 A2 show x\in A using indseq_valat0 by simp
{ fix y assume y\in A
  with I obtain n where II: n \in 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)) \in A by auto
  ultimately have f(y) \in A by simp
} then show \( \forall y \in A. f(y) \in A \) by simp
qed

19.3 Subsets generated by a binary operation

In algebra we often talk about sets "generated" by an element, that is sets of the form \( \{ a^n | n \in Z \} \). This is related to a general notion of "power" (as in \( a^n = a \cdot a \cdot \ldots \cdot a \)) or multiplicity \( n \cdot a = a + a + \ldots + a \). 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 $A1$: $f : X \times Y \to X$ and $A2$: $x \in X \ y \in Y$ and

$A3$: $a = \text{InductiveSequence}(x, \text{Fix2ndVar}(f, y))$

shows

$a : \text{nat} \to X$

$a(\text{nat}) \in \text{Pow}(X)$

$x \in a(\text{nat})$

$\forall z \in a(\text{nat}). \text{Fix2ndVar}(f, y)(z) \in a(\text{nat})$

proof -

let $g = \text{Fix2ndVar}(f, y)$

from $A1 \ A2$ have $I: g : X \to X$

using $\text{fix}_2\text{nd}_\text{var}_\text{fun}$ by simp

with $A2 \ A3$ show $a : \text{nat} \to X$

using $\text{indseq}_\text{seq}$ by simp

then show $a(\text{nat}) \in \text{Pow}(X)$ using $\text{func1}_1\_\text{L6}$ by simp

from $A2 \ A3$ I show $x \in a(\text{nat})$ using $\text{ind_seq}_\text{image}$ by blast

from $A2 \ A3$ I have

$g : X \to X \ x \in X \ a(\text{nat}) = \text{InductiveSequence}(x, g)(\text{nat})$

by auto

then show $\forall z \in a(\text{nat}). \text{Fix2ndVar}(f, y)(z) \in a(\text{nat})$

by (rule $\text{ind}_\text{seq}_\text{image}$)

dq

A simple corollary to the theorem $\text{binop}_\text{gen}_\text{set}$: a set that contains all iterations of the application of a binary operation exists.

lemma $\text{binop}_\text{gen}_\text{set}_\text{ex}$: assumes $A1$: $f : X \times Y \to X$ 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 $\text{binop}_\text{gen}_\text{set}$

by auto

moreover

$\{ \text{fix } z \text{ assume } T: z \in A$

with $A1 \ A2$ have $\text{Fix2ndVar}(f, y)(z) \in A$

using $\text{binop}_\text{gen}_\text{set}$ 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 $\text{fix}_\text{var}_\text{val}$ 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

dq

A more general version of $\text{binop}_\text{gen}_\text{set}$ where the generating binary operation acts on a larger set.

theorem $\text{binop}_\text{gen}_\text{set1}$: assumes $A1$: $f : X \times Y \to X$ and

$A2$: $X_1 \subseteq X$ and $A3$: $x \in X_1 \ y \in Y$ 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
\[
a : \text{nat} \rightarrow X_1 \\
a(\text{natur}) \in \text{Pow}(X_1) \\
x \in a(\text{natur}) \\
\forall z \in a(\text{natur}). \text{Fix2ndVar}(f, y)(z) \in a(\text{natur}) \\
\forall z \in a(\text{natur}). f(z, y) \in a(\text{natur})
\]

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_val by simp
with II have III: \( g : X_1 \rightarrow X_1 \)
    using func1_1_L1A by blast
with A3 A5 III show \( x \in a(\text{natur}) \) using ind_seq_image by blast
then have IV: \( a(\text{natur}) \in \text{Pow}(X_1) \) using func1_1_L6 by simp
from A3 A5 III have \( g : X_1 \rightarrow X_1 \)
    \( x \in X_1 \)
    \( \text{a(natur)} = \text{InductiveSequence}(x, g)(\text{natur}) \)
    by auto
then have \( \forall z \in a(\text{natur}). \text{Fix2ndVar}(h, y)(z) \in a(\text{natur}) \)
    by (rule ind_seq_image)
moreover
\[ \{ \text{fix } z \text{ assume } z \in a(\text{natur}) \}
    \text{with A2 IV have } z \in X_1 \text{ by auto}
    \text{with A1 A2 A3 have } g(z) = \text{Fix2ndVar}(f, y)(z)
    \text{using fix_2nd_var_restr_comm restrict by simp}
\] then have \( \forall z \in a(\text{natur}). g(z) = \text{Fix2ndVar}(f, y)(z) \) by simp
ultimately show \( \forall z \in a(\text{natur}). \text{Fix2ndVar}(f, y)(z) \in a(\text{natur}) \) by simp
moreover
\[ \{ \text{fix } z \text{ assume } z \in a(\text{natur}) \}
    \text{with A2 IV have } z \in X \text{ by auto}
    \text{with A1 A3 have } \text{Fix2ndVar}(f, y)(z) = f(z, y)
    \text{using fix_var_val by simp}
\] then have \( \forall z \in a(\text{natur}). \text{Fix2ndVar}(f, y)(z) = f(z, y) \)
    by simp
ultimately show \( \forall z \in a(\text{natur}). f(z, y) \in a(\text{natur}) \)
    by simp
qed

A generalization of \text{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: \( f : X \times Y \rightarrow X \) and

202
A2: \( X_1 \subseteq X \) and \( A3: x \in X_1 \ y \in Y \) and
A4: \( \forall t \in X_1. \ f(t, y) \in X \)
shows \( \{ A \in \text{Pow}(X_1). \ x \in A \wedge (\forall z \in A. \ f(z, y) \in A) \} \neq 0 \)

proof -
let \( a = \text{InductiveSequence}(x, \text{Fix2ndVar} \ (\text{restrict}(f, X_1 \times Y), y)) \)
let \( A = a(\text{nat}) \)
from A1 A2 A3 A4 have
\( A \in \text{Pow}(X_1) \ x \in A \ \forall z \in A. \ f(z, y) \in A \)
using \( \text{binop_gen_set1} \) by auto
thus thesis by auto
qed

19.4 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 appropriately. Namely, we replace the generating function in the definitions of \( \text{InductiveSequenceN} \) 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
\( \text{StateTransfFunNMeta}(F, n, s) \equiv \)
\( \text{if} \ (\text{snd}(s) \in n) \ \text{then} \ \langle F(\text{snd}(s))(\text{fst}(s)), \ \text{succ}(\text{snd}(s)) \rangle \ \text{else} \ s \)

Then we define the actual generating function on sets of pairs from \( X \times \{0,1,..,n\} \).

definition
\( \text{StateTransfFunN}(X, F, n) \equiv \{ s, \ \text{StateTransfFunNMeta}(F, n, s) \}. \ 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
\( \text{StatesSeq}(x, X, F, n) \equiv \)
\( \text{InductiveSequenceN}(\langle x, 0 \rangle, \ \text{StateTransfFunN}(X, F, n), n) \)

Finally we can define the sequence given by a initial point \( x \) and a sequence \( F \) of \( n \) functions.
definition

\[ \text{InductiveSeqVarFN}(x,X,F,n) \equiv \{ (k, \text{fst}(\text{StatesSeq}(x,X,F,n)(k))) \mid k \in \text{succ}(n) \} \]

The state transformation function (\text{StateTransfFunN} is a function that transforms \( X \times n \) into itself.

**lemma** state_trans_fun: assumes A1: \( n \in \mathbb{N} \) 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: \( x \in X \) \( k \in \text{succ}(n) \) and \( (x,k) = s \) 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 using apply_funtype StateTransfFunNMeta_def by auto }
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 \( \{ (s, \text{StateTransfFunNMeta}(F,n,s)) \mid s \in X \times \text{succ}(n) \} : X \times \text{succ}(n) \rightarrow X \times \text{succ}(n) \) by (rule ZF_fun_from_total)
then show \( \text{StateTransfFunN}(X,F,n): X \times \text{succ}(n) \rightarrow X \times \text{succ}(n) \) using 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** states_seq_props:

assumes A1: \( n \in \mathbb{N} \) and A2: \( F: n \rightarrow (X\rightarrow X) \) and A3: \( x\in X \) and A4: \( b = \text{StatesSeq}(x,X,F,n) \)
shows \( b: \text{succ}(n) \rightarrow X \times \text{succ}(n) \)
\( b(0) = \langle x,0 \rangle \)
\( \forall k \in \text{succ}(n). \text{snd}(b(k)) = k \)
\( \forall k\in n. b(\text{succ}(k)) = \langle F(k)(\text{fst}(b(k))), \text{succ}(k) \rangle \)

**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: \( b : \text{succ}(n) \rightarrow X \times \text{succ}(n) \)

204
Basic properties of sequences defined by equation \( x_{n+1} = f(n, x_n) \).

**Theorem fin_indseq_var_f_props:**

assumes
- A1: \( n \in \text{nat} \)
- A2: \( x \in X \)
- A3: \( F: n \rightarrow (X \rightarrow X) \)
- A4: \( a = \text{InductiveSeqVarFN}(x, X, F, n) \)

shows
- \( a: \text{succ}(n) \rightarrow X \)
- \( a(0) = x \)
- \( \forall k \in \text{nat}. a(\text{succ}(k)) = F(k)(a(k)) \)

**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

qed
by simp
moreover from A4 have III:
  a = {⟨k,fst(StatesSeq(x,X,F,n)(k))⟩. k ∈ succ(n)}
  using InductiveSeqVarFN_def by simp
ultimately have a(0) = 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 ∈ n
  with A1 have T1: succ(k) ∈ succ(n) and T2: k ∈ succ(n)
    using succ_ineq by auto
  from II T1 III have a(succ(k)) = fst(b(succ(k)))
    by (rule ZF_fun_from_tot_val)
  with A1 A2 A3 A5 have a(succ(k)) = F(k)(fst(b(k)))
    using states_seq_props by simp
  moreover from II T2 III have a(k) = fst(b(k))
    by (rule ZF_fun_from_tot_val)
  ultimately have a(succ(k)) = F(k)(a(k))
    by simp
} then show ∀k∈n. a(succ(k)) = F(k)(a(k))
  by simp
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)
    using fin_indseq_var_f_props by simp
  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

end
20 Enumerations

theory Enumeration_ZF imports NatOrder_ZF FiniteSeq_ZF FinOrd_ZF

begin

Suppose $r$ is a linear order on a set $A$ that has $n$ elements, where $n \in \mathbb{N}$. In the 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 Enumeration of $A$.

20.1 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. $\leq$ is the (natural) order on natural numbers, defined is Nat_ZF theory in the standard Isabelle library.

definition
  $\text{Enumeration}(A, r) \equiv \bigcup \text{ord_iso}(|A|, \leq, A, r)$

To set up the notation we define a locale $\text{enums}$. 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 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 enums =

fixes $X$ $r$
assumes linord: IsLinOrder($X$, $r$)

fixes ler (infix $\leq$ 70)
defines ler_def[simp]: $x \leq y \equiv \langle x, y \rangle \in r$

fixes $\sigma$

207
defines $\sigma$ [simp]: $\sigma(A) \equiv \text{Enumeration}(A,r)$

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]: $a \sqcup b \equiv \text{Concat}(a,b)$

20.2 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 isomorphism, 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 $\sigma(A) \in \text{ord_iso}(|A|,\text{Le},A,r)$
and $A \subseteq X$
using enum_props FinPow_def by auto
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 (rule func1_1_L1B)
qed

208
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)
    - By simp
  - 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 linord NatOrder_ZF_1_L2 IsLinOrder_def by auto
moreover from A2 \(|A| \in \text{nat}\) have
  \(|A|,|A|) \in \text{Le} and \(b, b) \in r\)
using linord NatOrder_ZF_1_L2 IsLinOrder_def
total_is_refl refl_def by auto
hence \(|A|,|A|) \in \text{Le} \iff (b, b) \in r\) by simp
ultimately have \(f \in \text{ord_iso}(\text{card}(A) \cup \{b\}, \text{Le}, A \cup \{b\}, r)\)
by (rule ord_iso_extend)
with A1 A2 show \(f \in \text{ord_iso}(\text{card}(A) \cup \{b\}, \text{Le}, A \cup \{b\}, r)\)
using card_fin_add_one by simp
qed
ultimately have \(f = \sigma(A \cup \{b\})\)
using 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 Append_def by simp
  moreover from A1 have \(\text{domain}(\sigma(A)) = |A|\)
    using enum_props func1_1_L1 by blast
  ultimately show \(\sigma(A) \leftarrow b = f\) by simp
qed
ultimately show \(\sigma(A) \cup \{b\} = \sigma(A) \leftarrow b\) by simp
qed

What is the enumeration of a singleton?

lemma (in enums) enum_singleton:
  assumes A1: \(x \in X\) shows \(\sigma(\{x\}): 1 \rightarrow X \text{ and } \sigma(\{x\})(0) = x\)
proof -
  from A1 have
    \(0 \in \text{FinPow}(X)\) and \(x \in (X - 0)\) and \(\forall a \in 0. \ a \leq x\)
    using empty_in_finpow by auto
  then have \(\sigma(0 \cup \{x\}) = \sigma(0) \leftarrow x\) by (rule enum_append)
  with A1 show \(\sigma(\{x\}): 1 \rightarrow X \text{ and } \sigma(\{x\})(0) = x\)
    using empty_enum empty_append1 by auto
qed

end

21 Folding in ZF

theory Fold_ZF imports InductiveSeq_ZF

begin

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 \(\{x_k\}_{k \in 0..n}\)
of elements of \(X\) and consider the product \(x_0 \cdot x_1 \cdots x_n\). To do the same thing
in formalized mathematics we have to define precisely what is meant by that "...". The definition 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 fold1). 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 $Y$.

### 21.1 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 tranformations to define the sequence of partial folds (called FoldSeq) by means of 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 $f : X \times Y \rightarrow X$.

**definition**

\[
\text{Seq2TrSeq}(f, a) \equiv \{(k, \text{Fix2ndVar}(f, a(k))) \mid k \in \text{domain}(a)\}
\]

Definition of a sequence of transformations.

**definition**

\[
\text{FoldSeq}(f, x, a) \equiv \\
\text{InductiveSeqVarFN}(x, \text{fstdom}(f), \text{Seq2TrSeq}(f, a), \text{domain}(a))
\]

Definition of a sequence of partial folds.

**definition**

\[
\text{Fold}(f, x, a) \equiv \text{Last}(\text{FoldSeq}(f, x, a))
\]

Definition of a fold.
If $X$ is a set with a binary operation $f : X \times Y \to X$ then $\text{Seq2TrSeqN}(f, a)$ 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**

Let $T = \text{Seq2TrSeq}(f, a)$

From $A1 A3$ have $D: \text{domain}(a) = n$ using $\text{func1_1_L1}$ by simp

With $A2 A3$ show $T : n \to (X \to X)$

Using $\text{apply_functype} \text{fix_2nd_var_fun} \text{ZF_fun_from_total} \text{Seq2TrSeq_def}$ by simp

With $A4 D$ have $I: \forall k \in n. T(k) = \text{Fix2ndVar}(f, a(k))$

Using $\text{Seq2TrSeq_def} \text{ZF_fun_from_tot_val0}$ by simp

{ fix $k$ fix $x$
  assume $A5: k \in n. x \in X$
  with $A1 A3$ have $a(k) \in Y$ using $\text{apply_functype}$
  by auto
  with $A2 A5 I$ have $(T(k))(x) = f(x, a(k))$
  using $\text{fix_var_val}$ 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 $T = \text{Seq2TrSeq}(f, a)$

From $A1 A3$ have $D: \text{domain}(y) = n$

Using $\text{func1_1_L1}$ by simp

From $\langle f : X \times Y \to X \rangle \langle Y \neq 0 \rangle$ have $I: \text{fstdom}(f) = X$

Using $\text{fstdomdef}$ by simp

With $A1 A2 A3 A4 A6 D$ show

$I1: F: \text{succ}(n) \to X$ and $F(0) = x$

Using $\text{seq2trans_seq Props} \text{FoldSeq_def} \text{fin_indseq_var_f Props}$ by auto

From $A1 A2 A3 A4 A6 I$ have $\forall k \in n. F(\text{succ}(k)) = T(k)(F(k))$

Using $\text{seq2trans_seq Props} \text{FoldSeq_def} \text{fin_indseq_var_f Props}$ by simp

Moreover

{ fix $k$ assume $A5: k \in n$ hence $k \in \text{succ}(n)$ by auto

|

212
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 \(\forall k \in n. F(succ(k)) = f(F(k), y(k))\)
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 fin_indseq_var_f_restrict but a 
proof using fold_seq_props and induction turns out to be shorter.

lemma foldseq_restrict: assumes 
  \(n \in \text{nat}\) \(k \in \text{succ}(n)\) and 
  \(i \in \text{nat}\) \(f : X \times Y \to X\) \(a : n \to Y\) \(b : i \to Y\) and 
  \(n \subseteq i\) \(\forall j \in n. b(j) = a(j)\) \(x \in X\) \(Y \neq 0\)
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 
    \(n \in \text{nat}\) \(k \in \text{succ}(n)\)
    \(Q(0) = P(0)\) and 
    \(\forall j \in n. Q(j) = P(j)\) \(\longrightarrow\) \(Q(succ(j)) = P(succ(j))\)
    using fold_seq_props by auto
  then show \(Q(k) = P(k)\) by (rule fin_nat_ind)
qed

A special case of foldseq_restrict when the longer sequence is created from 
the shorter one by appending one element.

corollary fold_seq_append:
  assumes \(n \in \text{nat}\) \(f : X \times Y \to X\) \(a : n \to Y\) and 
  \(x \in X\) \(k \in \text{succ}(n)\) \(y \in Y\)
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

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.

theorem fold_props:
  assumes A1: \(n \in \text{nat}\) and 
  A2: \(f : X \times Y \to X\) \(a : n \to Y\) \(x \in X\) \(Y \neq 0\)
shows 
  \(\text{Fold}(f,x,a) = \text{FoldSeq}(f,x,a)(n)\) and 
  \(\text{Fold}(f,x,a) \in X\)
proof -
  from assms have FoldSeq(f,x,a) : succ(n) → X
  using fold_seq_props by simp
  with A1 show
    Fold(f,x,a) = FoldSeq(f,x,a)(n) and Fold(f,x,a) ∈ X
  using last_seq_elem apply_funtype Fold_def by auto
qed

A corner case: what happens when we fold an empty list?

theorem fold_empty: assumes A1: f : X×Y → X and
  A2: a:0→Y x∈X Y≠0
shows Fold(f,x,a) = x
proof -
  let F = FoldSeq(f,x,a)
  from assms have I:
    0 ∈ nat f : X×Y → 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 ∈ nat f : X×Y → X a:0→Y x∈X Y≠0 and
    F = FoldSeq(f,x,a)
  by auto
  then have F(0) = x by (rule fold_seq_props)
  ultimately show 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.

theorem fold_append:
  assumes A1: n ∈ nat and A2: f : X×Y → X and
  A3: a:n→Y and A4: x∈X and A5: y∈Y
shows
  FoldSeq(f,x,Append(a,y))(n) = Fold(f,x,a) and
  Fold(f,x,Append(a,y)) = f⟨Fold(f,x,a), y⟩
proof -
  let b = Append(a,y)
  let P = FoldSeq(f,x,b)
  from A5 have I: Y ≠ 0 by auto
  with assms show thesis1: P(n) = Fold(f,x,a)
  using fold_seq_append fold_props by simp
  from assms I have II:
    succ(n) ∈ nat f : X×Y → X
    b : succ(n) → Y x∈X Y ≠ 0
    P = FoldSeq(f,x,b)
  using append_props by auto
  then have
    ∀k ∈ succ(n). P(succ(k)) = f(P(k), b(k))
  by (rule fold_seq_props)
  with A3 A5 thesis1 have P(succ(n)) = f⟨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

22 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.

22.1 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
$\text{Bisections}(X) = \{p \in \text{Pow}(X) \times \text{Pow}(X).\}$
$fst(p) \neq 0 \land snd(p) \neq 0 \land fst(p) \cap snd(p) = 0 \land fst(p) \cup snd(p) = X\}$

Properties of bisections.

lemma bisec_props: assumes $\langle A, B \rangle \in \text{Bisections}(X)$ shows
$A \neq 0 \land B \neq 0 \land A \subseteq X \land B \subseteq X \land A \cap B = 0 \land A \cup B = X \land X \neq 0$
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 $A \neq 0 \land B \neq 0 \land A \cap B = 0$
shows $\langle A, B \rangle \in \text{Bisections}(A \cup B)$ using assms Bisections_def
by auto

215
Bisection of $X$ is a pair of subsets of $X$.

**lemma** bisec_is_pair: assumes $Q \in \text{Bisections}(X)$
shows $Q = \langle \text{fst}(Q), \text{snd}(Q) \rangle$
using assms Bisections_def by auto

The set of bisections of the empty set is empty.

**lemma** bisec_empty: shows $\text{Bisections}(0) = 0$
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: $x \not\in X$ and A2: $\langle A, B \rangle \in \text{Bisections}(X \cup \{x\})$
shows $\langle A = \{x\} \vee B = \{x\} \rangle \vee (\langle A - \{x\}, B - \{x\} \rangle \in \text{Bisections}(X))$
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 $\langle A - \{x\}, B - \{x\} \rangle \in \text{Bisections}(X)$
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: $\langle A, B \rangle \in \text{Bisections}(X \cup \{x\})$
and A2: $\langle A - \{x\}, B - \{x\} \rangle \in \text{Bisections}(X)$
shows $\langle \langle A, B - \{x\} \rangle \in \text{Bisections}(X) \land x \in B \rangle \lor$
$\langle \langle A - \{x\}, B \rangle \in \text{Bisections}(X) \land x \in A \rangle$
proof -
from A1 have $x \in A \cup B$
using Bisections_def by auto
hence $x \in A \lor x \in B$ by simp
from A1 have $A - \{x\} = A \lor B - \{x\} = B$
using Bisections_def by auto

216
moreover
{ assume A - {x} = A
  with A2: \( x \in A \cup B \) have
  \( \langle A, B - \{x\} \rangle \in \text{Bisections}(X) \land x \in B \)
  using singl_diff_eq by simp }
moreover
{ assume B - {x} = B
  with A2: \( x \in A \cup B \) have
  \( \langle A - \{x\}, B \rangle \in \text{Bisections}(X) \land x \in A \)
  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: \( x \not\in X \) and A2: \( \langle \{x\}, A \rangle \in \text{Bisections}(X \cup \{x\}) \)
  shows \( A = X \) and \( X \neq 0 \)
proof -
  from A2 have A \( \subseteq X \) using Bisections_def by auto
moreover
  { fix a assume a \( \in X \)
    with A2 have a \( \in \{x\} \cup A \) using Bisections_def by simp
    with A1: \( a \in X \) have a \( \in A \) by auto }
ultimately show A \( = X \) by auto
with A2 show X \( \neq 0 \) 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: \( x \not\in X \) and A2: \( \langle A, \{x\} \rangle \in \text{Bisections}(X \cup \{x\}) \)
  shows \( A = X \) and \( X \neq 0 \)
proof -
  from A2 have A \( \subseteq X \) using Bisections_def by auto
moreover
  { fix a assume a \( \in X \)
    with A2 have a \( \in A \cup \{x\} \) using Bisections_def by simp
    with A1: \( a \in X \) have a \( \in A \) by auto }
ultimately show A \( = X \) by auto
with A2 show X \( \neq 0 \) 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: \( A \in \text{FinPow}(X) \) and A2: \( Q \in \text{Bisections}(A) \)
  shows \( \text{fst}(Q) \in \text{FinPow}(X) \) and \( \text{snd}(Q) \in \text{FinPow}(X) \)
proof -
  from A2 have \( \langle \text{fst}(Q), \text{snd}(Q) \rangle \in \text{Bisections}(A) \)
    using bisec_is_pair by simp
then have \( \text{fst}(Q) \subseteq A \) and \( \text{snd}(Q) \subseteq A \)
using bisec_props by auto
with A1 show \( \text{fst}(Q) \in \text{FinPow}(X) \) and \( \text{snd}(Q) \in \text{FinPow}(X) \)
using FinPow_def subset_Finite by auto
qed

### 22.2 Partitions

This sections 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**

\[
\text{Partition} (\_, \{ \text{is partition} \}) [90] [91] \text{ where }
P \{ \text{is partition} \} \equiv \forall x \in \text{domain}(P).
\text{P}(x) \neq 0 \land (\forall y \in \text{domain}(P). x \neq y \rightarrow \text{P}(x) \cap \text{P}(y) = 0)
\]

A fact about lists of mutually disjoint sets.

**lemma** list_partition: assumes A1: \( n \in \text{nat} \) and
A2: \( a : \text{succ}(n) \rightarrow X \ a \{ \text{is partition} \} \)
shows \( \bigcup_{i \in n. a(i)} \cap a(n) = 0 \)
proof -
{ assume \( (\bigcup_{i \in n. a(i)}) \cap a(n) \neq 0 \)
then have \( \exists x. x \in (\bigcup_{i \in n. a(i)}) \cap a(n) \)
by (rule nonempty_has_element)
then obtain x where \( x \in (\bigcup_{i \in n. a(i)}) \) and I: \( x \in a(n) \)
by auto
then obtain i where \( i \in n \) and \( x \in a(i) \) 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 \( \forall x \in X. \{\langle x, \{b(x)\}\rangle. x \in X\}(x) = \{b(x)\} \) and
\( \{\langle x, \{b(x)\}\rangle. x \in X\} \{ \text{is partition} \} \)
proof -
let p = \( \{\langle x, \{b(x)\}\rangle. x \in X\} \)
\{ fix x assume x \in X \}

218
from A1 have b : X → Y using inj_def
  by simp
  with <x ∈ X> have {b(x)} ∈ Pow(Y)
  using apply_funtype by simp
} hence ∀x ∈ X. {b(x)} ∈ Pow(Y) by simp
then have p : X → Pow(Y) using ZF_fun_from_total
  by simp
then have domain(p) = X using func1_1_L1
  by simp
from <p : X → Pow(Y)> show I: ∀x ∈ X. p(x) = {b(x)}
  using ZF_fun_from_tot_val0 by simp
{ fix x assume x ∈ X
  with I have p(x) = {b(x)} by simp
  hence p(x) ≠ 0 by simp
  moreover
  { fix t assume t ∈ X and x ≠ t
    with A1 <x ∈ X> have b(x) ≠ b(t) using inj_def
    by auto
    with I <x ∈ X> <t ∈ X> have p(x) ∩ p(t) = 0
    by auto }
  ultimately have
  p(x) ≠ 0 ∧ (∀t ∈ X. x≠t → p(x) ∩ p(t) = 0)
  by simp
} with <domain(p) = X> show {(x, {b(x)}). x ∈ X} {is partition}
  using Partition_def by simp
qed

end

23 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.

23.1 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) where

\( A \) \{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**

\( \text{IsAquasigroup}(G,A) \equiv A:G \times G \rightarrow G \land A \) \{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** \( lrdiv_binop \): assumes \( \text{IsAquasigroup}(G,A) \) shows

\( \text{LeftDiv}(G,A):G \times G \rightarrow G \land \text{RightDiv}(G,A):G \times G \rightarrow G \)

**proof** -

\[ \{ \text{fix } p \text{ assume } p \in G \times G \]

\[ \text{with } \text{assms have } \]

\[ \bigcup \{x \in G. A(fst(p),x) = snd(p))\} \in G \text{ and } \bigcup \{x \in G. A(x,fst(p)) = snd(p))\} \in G \]

\[ \text{unfolding } \text{IsAquasigroup_def HasLatinSquareProp_def HasLeftDiv_def HasRightDiv_def} \]

\[ \text{using ZF1_1_L9(2) by auto} \]

\[ \text{then show } \text{LeftDiv}(G,A):G \times G \rightarrow G \land \text{RightDiv}(G,A):G \times G \rightarrow G \]

\[ \text{unfolding LeftDiv_def RightDiv_def using ZF_fun_from_total by auto} \]

\[ \text{qed} \]

220
We will use multiplicative notation for the quasigroup operation. The right and left division will be denoted \( a/b \) and \( a\backslash b \), resp.

```plaintext
locale quasigroup0 =  
  fixes G A  
  assumes qgroupassum: IsAquasigroup(G,A)

fixes qgroper (infixl · 70)  
defines qgroper_def[simp]: x·y ≡ A⟨x,y⟩

fixes leftdiv (infixl \ 70)  
defines leftdiv_def[simp]: x\y ≡ LeftDiv(G,A)⟨x,y⟩

fixes rightdiv (infixl / 70)  
defines rightdiv_def[simp]:x/y ≡ RightDiv(G,A)⟨y,x⟩
```

The quasigroup operation is closed on \( G \).

```plaintext
lemma (in quasigroup0) qg_op_closed: assumes x ∈ G y ∈ G  
shows x·y ∈ G  
using qgroupassum assms IsAquasigroup_def apply_funtype by auto
```

A couple of properties of right and left division:

```plaintext
lemma (in quasigroup0) lrdiv_props: assumes x ∈ G y ∈ G  
shows ∃!z. z ∈ G ∧ z·x = y y/x ∈ G (y/x)·x = y  
  and  
  ∃!z. z ∈ G ∧ x z = y x\(x\backslash y) = y
proof -  
let z_r = \bigcup\{z ∈ G. z·x = y\}  
from qgroupassum have I: RightDiv(G,A):G×G→G using lrdiv_binop(2)  
by simp  
with assms have RightDiv(G,A)⟨x,y⟩ = z_r  
  unfolding RightDiv_def using ZF_fun_from_tot_val by auto  
moreover  
from qgroupassum assms show ∃!z. z ∈ G ∧ z·x = y  
  unfolding IsAquasigroup_def HasLatinSquareProp_def HasRightDiv_def  
by simp  
then have z_r·x = y by (rule ZF1_1_L9)  
ultimately show (y/x)·x = y by simp  
let z_l = \bigcup\{z ∈ G. x z = y\}  
from qgroupassum have II: LeftDiv(G,A):G×G→G using lrdiv_binop(1)  
by simp  
with assms have LeftDiv(G,A)⟨x,y⟩ = z_l  
  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 z_l = y by (rule ZF1_1_L9)  
ultimately show x·(x\backslash y) = y by simp
```
```
We can cancel the left element on both sides of an equation.

**lemma** (in quasigroup0) qg-cancel_left:

assumes $x \in G, y \in G$ and $xy = xz$

shows $y = z$

using qggroupassum 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 $x \in G, y \in G$ and $yx = zx$

shows $y = z$

using qggroupassum assms qg_op_closed lrdiv_props(1) by blast

Two additional identities for right and left division:

**lemma** (in quasigroup0) lrdiv_ident: assumes $x \in G, y \in G$

shows $(y \cdot x)/x = y$ and $x \backslash (x \cdot y) = y$

proof -

from assms have $(y \cdot x)/x \in G$ and $((y \cdot x)/x) \cdot x = y \cdot x$

using qg_op_closed lrdiv_props(2,3) by auto

with assms show $(y \cdot x)/x = y$ using qg_cancel_right by simp

from assms have $x \backslash (x \cdot y) \in G$ and $x \cdot (x \backslash (x \cdot y)) = x \cdot y$

using qg_op_closed lrdiv_props(5,6) by auto

with assms show $x \backslash (x \cdot y) = y$ using qg_cancel_left by simp

qed

end

## 24 Loops

theory Loop_ZF imports Quasigroup_ZF

begin

This theory specifies the definition and proves basic properites of loops.

Loops are very similar to groups, the only property that is missing is associativity of the operation.

### 24.1 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($G, A$) ∧ (∃$e \in G$. ∀$x \in G$. $A(e, x) = x$

∧ $A(x, e) = x$)
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**

\[ \text{TheNeutralElement}(G,A) \equiv \left( \text{THE } e. \ e \in G \land (\forall g \in G. f(e,g) = g \land f(g,e) = g) \right) \]

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**

\[ \exists e \in G. \forall x \in G. \ e \cdot x = x \land x \cdot e = x \]

**fixes**

\[ e \]

**defines**

\[ \text{neut}[\text{simp}]: \ 1 \equiv \text{TheNeutralElement}(G,A) \]

In the loop context the pair \((G,A)\) forms a loop.

**lemma** (in loop0) is_loop: shows \( \text{IsAloop}(G,A) \)

**unfolding**

\( \text{IsAloop_def} \) using \( \text{ex_ident qgroupassum} \) 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 \( \text{IsAloop}(G,A) \) shows \( \text{loop0}(G,A) \)

**using** \( \text{assms unfolding} \) \( \text{IsAloop_def loop0_axioms_def quasigroup0_def loop0_def} \) by auto

The neutral element is unique in the loop.

**lemma** (in loop0) neut_uniq_loop: shows

\[ \exists! e. \ e \in G \land (\forall x \in G. e \cdot x = x \land x \cdot e = x) \]

**proof**

from \( \text{ex_ident show} \) \( \exists e. \ e \in G \land (\forall x \in G. e \cdot x = x \land x \cdot e = x) \) by auto

next

fix \( e \ y \)

assume \( e \in G \land (\forall x \in G. e \cdot x = x \land x \cdot e = x) \)

\( y \in G \land (\forall x \in G. y \cdot x = x \land x \cdot y = x) \)

then have \( e \cdot y = y \) and \( e \cdot y = e \) by auto

thus \( e = y \) by simp

qed

The neutral element as defined in the 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 \( 1 \equiv \text{TheNeutralElement}(G,A) \) by simp
then have $1 = n$ unfolding \texttt{TheNeutralElement_def} by simp
moreover have $n \in G \land (\forall x \in G. \ n \cdot x = x \land x \cdot n = x)$ using \texttt{neut_uniq_loop}
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 $G$.

\textbf{definition}
\[
\text{LeftInv}(G,A) \equiv \{ (x, \bigcup \{ y \in G. \ A(y,x) = \text{TheNeutralElement}(G,A) \}) \mid x \in G \}
\]

Definition of the right inverse as a function on $G$:

\textbf{definition}
\[
\text{RightInv}(G,A) \equiv \{ (x, \bigcup \{ y \in G. \ A(x,y) = \text{TheNeutralElement}(G,A) \}) \mid x \in G \}
\]

In a loop $G$ right and left inverses are functions on $G$.

\textbf{lemma (in loop0) lr_inv_fun:} shows $\text{LeftInv}(G,A):G \to G \text{ RightInv}(G,A):G \to G$
unfolding $\text{LeftInv}_\text{def}$ $\text{RightInv}_\text{def}$
using $\text{neut_props_loop} \text{ lrdiv_props(1,4)} \text{ ZF1_1_L9} \text{ ZF_fun_from_total}$
by auto

Right and left inverses have desired properties.

\textbf{lemma (in loop0) lr_inv_props:} assumes $x \in G$
shows $\text{LeftInv}(G,A)(x) \in G \ (\text{LeftInv}(G,A)(x)) \cdot x = 1$
$\text{RightInv}(G,A)(x) \in G \ x \cdot (\text{RightInv}(G,A)(x)) = 1$
proof 
from \text{assms} show $\text{LeftInv}(G,A)(x) \in G$ and $\text{RightInv}(G,A)(x) \in G$
using \text{lr_inv_fun apply_funtype} by auto
from \text{assms} have $\exists ! y. \ y \in G \land y \cdot x = 1$
using $\text{neut_props_loop(1)} \text{ lrdiv_props(1)}$ by simp
then have $(\bigcup \{ y \in G. \ y \cdot x = 1 \}) \cdot x = 1$
by (rule ZF1_1_L9)
with \text{assms} show $(\text{LeftInv}(G,A)(x)) \cdot x = 1$
using $\text{lr_inv_fun(1)} \text{ ZF_fun_from_tot_val}$ unfolding $\text{LeftInv}_\text{def}$ by simp
from \text{assms} have $\exists ! y. \ y \in G \land x \cdot y = 1$
using $\text{neut_props_loop(1)} \text{ lrdiv_props(4)}$ by simp
then have $x \cdot (\bigcup \{ y \in G. \ x \cdot y = 1 \}) = 1$
by (rule ZF1_1_L9)
with \text{assms} show $x \cdot (\text{RightInv}(G,A)(x)) = 1$
using $\text{lr_inv_fun(2)} \text{ ZF_fun_from_tot_val}$ unfolding $\text{RightInv}_\text{def}$ by simp
qed

end
25 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].

25.1 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 \((x, y) \in r\) is equivalent to both \((A(x, z), A(x, z)) \in r\) and \((A(z, x), A(z, x)) \in r\). 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 \((x, y) \in r\) 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
\[
\text{IsAnOrdLoop}(L, A, r) \equiv \\
\text{IsAloop}(L, A) \land r \subseteq L \times L \land \text{IsPartOrder}(L, r) \land (\forall x \in L. \forall y \in L. \forall z \in L. ((x, y) \in r \leftrightarrow (A(x, z), A(y, z)) \in r) \land ((x, y) \in r \leftrightarrow (A(z, x), A(z, y)) \in r))
\]

We define the set of nonnegative elements in the obvious way as \(L^+ = \{x \in L : 0 \leq x\}\).

definition
\[
\text{Nonnegative}(L, A, r) \equiv \{x \in L. (\text{TheNeutralElement}(L, A), x) \in r\}
\]

The \text{PositiveSet}(L, A, r) is a set similar to \text{Nonnegative}(L, A, r), but without the neutral element.

definition
\[
\text{PositiveSet}(L, A, r) \equiv \{x \in L. (\text{TheNeutralElement}(L, A), x) \in r \land \text{TheNeutralElement}(L, A) \neq x\}
\]

We will use the additive notation for ordered loops.

locale loop1 =
fixes L and A and r
assumes ordLoopAssum: IsAnOrdLoop(L, A, r)
defines neut_def[simp]: 0 ≡ TheNeutralElement(L,A)

defines looper_def[simp]: x + y ≡ A(x,y)

defines lesseq_def[simp]: x ≤ y ≡ (x,y) ∈ r

defines sless_def[simp]: x < y ≡ x≤y ∧ x≠y

defines nonnegative_def[simp]: L⁺ ≡ Nonnegative(L,A,r)

defines positive_def[simp]: L⁺ ≡ PositiveSet(L,A,r)

defines leftdiv_def[simp]: -x+y ≡ LeftDiv(L,A)(x,y)

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 z∈L
  shows x+z ≤ y+z and z+x ≤ z+y
  proof -
    from ordLoopAssum assms have
      (⟨x,y⟩ ∈ r ⇔ ⟨A(x,z),A(y,z)⟩ ∈ r) ∧ (⟨x,y⟩ ∈ r ⇔ ⟨A(z,x),A(z,y)⟩ ∈ r) 
    using lsq_members unfolding IsAnOrdLoop_def by blast
    with assms(1) show x+z ≤ y+z and z+x ≤ 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∈L
  shows x+z < y+z and z+x < z+y
proof -
  from assms have x+z ≤ y+z and z+x ≤ z+y
  using ord_trans_inv by auto
moreover have x+z ≠ y+z and z+x ≠ 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 ≠ 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 ≠ 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∈L y∈L z∈L and x+z ≤ y+z
  shows x≤y
proof -
  from ordLoopAssum assms(1,2,3) have ⟨x,y⟩ ∈ r ←→ ⟨A⟨x,z⟩,A⟨y,z⟩⟩ ∈ r
  unfolding IsAnOrdLoop_def by blast
  with assms(4) show x≤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∈L y∈L z∈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∈L y∈L z∈L and z+x ≤ z+y
  shows x≤y
proof -
  from ordLoopAssum assms(1,2,3) have ⟨x,y⟩ ∈ r ←→ ⟨A⟨z,x⟩,A⟨z,y⟩⟩ ∈ r
  unfolding IsAnOrdLoop_def by blast
  with assms(4) show x≤y by simp
qed
We can cancel an element from both sides of a strict inequality on the left side.

```plaintext
lemma (in loop1) strict_ineq_cancel_left: assumes x∈L y∈L z∈L and z+x < z+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:

```plaintext
lemma (in loop1) nonneg_definition: shows x ∈ L⁺ ←→ 0 ≤ x using ordLoopAssum IsAnOrdLoop_def Nonnegative_def by auto
```

The nonnegative set is contained in the loop.

```plaintext
lemma (in loop1) nonneg_subset: shows L⁺ ⊆ L using Nonnegative_def by auto
```

The positive set is contained in the loop.

```plaintext
lemma (in loop1) positive_subset: shows L⁺ ⊆ L using PositiveSet_def by auto
```

The definition of the positive set in the notation used in the loop1 locale:

```plaintext
lemma (in loop1) posset_definition: shows x ∈ L⁺ ←→ (0 ≤ x ∧ x≠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:

```plaintext
lemma (in loop1) posset_definition1: shows x ∈ L⁺ ←→ 0<x using ordLoopAssum IsAnOrdLoop_def PositiveSet_def by auto
```

The order in an ordered loop is antisymmeric.

```plaintext
lemma (in loop1) loop_ord_antisym: assumes x≤y and y≤x
  shows x=y
  proof -
    from ordLoopAssum assms have antisym(r) (x,y) ∈ r (y,x) ∈ r unfolding IsAnOrdLoop_def IsPartOrder_def by auto
    then show x=y by (rule Fol1_L4)
  qed
```

The loop order is transitive.

```plaintext
lemma (in loop1) loop_ord_trans: assumes x≤y and y≤z shows x≤z
  proof -
    from ordLoopAssum assms have trans(r) (x,y) ∈ r ∧ (y,z) ∈ r unfolding IsAnOrdLoop_def IsPartOrder_def by auto
    then have (x,z) ∈ r by (rule Fol1_L3)
    thus thesis by simp
```

228
A form of mixed transitivity for the strict order:

**lemma (in loop1) loop_strict_ord_trans**: assumes \( x \leq y \) and \( y < z \) shows \( x < z \)

**proof**

- from assms have \( x \leq y \) and \( y \leq z \) by auto
- then have \( x \leq z \) 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 \( x \leq z \) 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 \( 0 \leq -x + y \) \((-x + y) \in L^+ \)

**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 \( 0 + x \leq x + (-x + y) \) by simp
- with \( \langle x \in L \rangle \) \( \langle 0 \in L \rangle \) \(( -x + y) \in L \) show \( 0 \leq -x + y \) using ineq_cancel_left by simp
- then show \((-x + y) \in L^+ \) using nonneg_definition by simp
- from \( \langle x \in L \rangle \) \( \langle y \in L \rangle \) 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 \( \langle x \in L \rangle \) \( \langle 0 \in L \rangle \) \(( y-x) \in L \) 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 \( 0 < -x + y \) \((-x + y) \in L^+ \)

**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

229
by auto
with assms have \( x+0 < x+(-x+y) \) by simp
with \( x \in L \cdot (0 \in L \cdot (-x+y) \in L) \) show \( 0 < -x+y \) using strict_ineq_cancel_left
by simp
then show \((x+y) \in L_+\) using posset_definition1 by simp
from \( x \in L \cdot y \in L \) 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 \( x \in L \cdot 0 \in L \cdot (y-x) \in L \) show \( 0 < y-x \) using strict_ineq_cancel_right
by simp
then show \((y-x) \in L_+\) using posset_definition1 by simp
qed

end

26 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.

26.1 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 $a \leftarrow x$ 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 · 72)

defines $\prod$ def [simp]: $x \cdot y \equiv f(x,y)$

fixes $\sqcup$ (infixl ⊔ 69)

defines $\sqcup$ def [simp]: $a \sqcup b \equiv \text{Concat}(a,b)$

locale semigr0 =

fixes $G\ f$

assumes assoc_assum: $f$ {is associative on} $G$

fixes $\prod$ (infixl · 72)

defines $\prod$ def [simp]: $x \cdot y \equiv f(x,y)$

fixes $\sqcup$ (infixl ⊔ 69)

defines $\sqcup$ 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 in the semigr0 context.

lemma (in 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 semigr0) semigr_binop: shows $f : G \times G \to G$
using assoc_assum IsAssociative_def by simp

Semigroup operation is closed.

lemma (in 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 semigr0 context.

lemma (in semigr0) append_1elem_nice:
assumes $n \in \text{nat}\ a : n \to X\ b : 1 \to X$
shows $a \sqcup b = a \leftarrow b(0)$
Lemma \texttt{concat\_init\_last\_elem} rewritten in the notation used in the \texttt{semigr0} context.

\begin{verbatim}
lemma (in semigr0) concat\_init\_last:  
  assumes n \in \texttt{nat} \quad k \in \texttt{nat} and  
  a: n \to X \quad \text{and} \quad b : \text{succ}(k) \to X  
  shows \((a \uplus \text{Init}(b)) \leftarrow b(k) = a \uplus b\)  
  using \texttt{assms} \texttt{concat\_init\_last\_elem} by simp
\end{verbatim}

The product of semigroup (actually, magma – we don’t need associativity for this) elements is in the semigroup.

\begin{verbatim}
lemma (in semigr0) prod\_type:  
  assumes n \in \texttt{nat} \quad \text{and} \quad a : \text{succ}(n) \to \texttt{G}  
  shows \((\prod a) \in \texttt{G}\)  
  proof -  
  from \texttt{assms} have  
  \begin{align*}
  \text{succ}(n) \in \texttt{nat} \quad f : \texttt{G} \times \texttt{G} \to \texttt{G} \quad \text{Tail}(a) : n \to \texttt{G} \\
  \text{using} \texttt{semigr\_binop tail\_props} \quad \text{by auto} \\
  \text{moreover from} \texttt{assms} \text{have} \quad a(0) \in \texttt{G} \quad \text{and} \quad \texttt{G} \neq \texttt{0}  \\
  \text{using} \texttt{empty\_in\_every\_succ apply\_funtype} \quad \text{by auto} \\
  \text{ultimately show} \quad (\prod a) \in \texttt{G} \quad \text{using} \texttt{Fold1\_def fold\_props} \quad \text{by simp}
  \end{align*}
  qed
\end{verbatim}

What is the product of one element list?

\begin{verbatim}
lemma (in semigr0) prod\_of\_1elem: assumes A1: a: 1 \to \texttt{G}  
  shows \((\prod a) = a(0)\)  
  proof -  
  have f : \texttt{G} \times \texttt{G} \to \texttt{G} \quad \text{using} \texttt{semigr\_binop by simp}  
  moreover from A1 have Tail(a) : 0 \to \texttt{G} \quad \text{using tail\_props} \quad \text{by blast}  
  moreover from A1 have a(0) \in \texttt{G} \quad \text{and} \quad \texttt{G} \neq \texttt{0}  
  \quad \text{using} \texttt{apply\_funtype} \quad \text{by auto} 
  \quad \text{ultimately show} \quad (\prod a) = a(0) \quad \text{using} \texttt{fold\_empty Fold1\_def} \quad \text{by simp}
  qed
\end{verbatim}

What happens to the product of a list when we append an element to the list?

\begin{verbatim}
lemma (in semigr0) prod\_append: assumes A1: n \in \texttt{nat} and  
  A2: a : \text{succ}(n) \to \texttt{G} \quad \text{and} \quad A3: x \in \texttt{G}  
  shows \((\prod a \leftarrow x) = (\prod a) \cdot x\)  
  proof -  
  from A1 A2 have I: \text{Tail}(a) : n \to \texttt{G} \quad a(0) \in \texttt{G}  
  \quad \text{using} \texttt{tail\_props empty\_in\_every\_succ apply\_funtype} \quad \text{by auto}
  qed
\end{verbatim}
from assms have $(\prod a \leftarrow x) = \text{Fold}(f, a(0), \text{Tail}(a) \leftarrow x)$
using head_of_append tail_append_commute Fold1_def
by simp
also from A1 A3 I have $\ldots = (\prod a) \cdot x$
using semigr_binop fold_append 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 semigr0) prod_conc_distr:
assumes A1: $n \in \mathbb{N}$ $k \in \mathbb{N}$ 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 $k \in \mathbb{N}$ by simp
moreover have $\forall b \in \text{succ}(0) \rightarrow G. (\prod a) \cdot (\prod b) = \prod (a \sqcup b)$
proof -
{ fix b assume A3: $b : \text{succ}(0) \rightarrow G$
  with A1 A2 have $\text{succ}(n) \in \mathbb{N}$ $a : \text{succ}(n) \rightarrow G$ $b : 1 \rightarrow G$
  by auto
  then have $a \sqcup b = a \leftarrow b(0)$ by (rule append_ielem_nice)
  with A1 A2 A3 have $(\prod a) \cdot (\prod b) = \prod (a \sqcup b)$
  using apply_funtype prod_append semigr_binop prod_of_ielem
  by simp
} thus thesis by simp
qed

moreover have $\forall j \in \mathbb{N}$.
$(\forall b \in \text{succ}(j) \rightarrow G. (\prod a) \cdot (\prod b) = \prod (a \sqcup b)) \rightarrow$
$(\forall b \in \text{succ}($successor$(j)) \rightarrow G. (\prod a) \cdot (\prod b) = \prod (a \sqcup b))$
proof -
{ fix j assume A4: $j \in \mathbb{N}$ and
  A5: $(\forall b \in \text{succ}(j) \rightarrow G. (\prod a) \cdot (\prod b) = \prod (a \sqcup b))$
  { fix b assume A6: $b : \text{succ}($successor$(j)) \rightarrow G$
    let $c = \text{Init}(b)$
    from A4 A6 have $T: b($successor$(j)) \in G$ and
    I: $c : \text{succ}(j) \rightarrow G$ and II: $b = c \leftarrow b($successor$(j))$
    using apply_funtype init_props by auto
    from A1 A2 A4 A6 have
    $\text{succ}(n) \in \mathbb{N}$ $\text{succ}(j) \in \mathbb{N}$
    $a : \text{succ}(n) \rightarrow G$ $b : \text{succ}($successor$(j)) \rightarrow G$
    by auto
    then have III: $(a \sqcup c) \leftarrow b($successor$(j)) = a \sqcup b$
    by (rule concat_init_last)
    from A4 I T have $(\prod c \leftarrow b($successor$(j))) = (\prod c) \cdot b($successor$(j))$
    by (rule prod_append)
}
with II have
\((\prod a) \cdot (\prod b) = (\prod a) \cdot ((\prod a) \cdot b(succ(j)))\)
by simp

moreover from A1 A2 A4 T I have
\((\prod a) \in G \quad (\prod c) \in G \quad b(succ(j)) \in G\)
using prod_type by auto

ultimately have
\((\prod a) \cdot (\prod b) = ((\prod a) \cdot (\prod c)) \cdot b(succ(j))\)
using semigr_assoc by auto

with A5 I have \((\prod a) \cdot (\prod b) = (\prod (a \sqcup c)) \cdot b(succ(j))\)
by simp

moreover
from A1 A2 A4 T I have
\(T1: succ(n) \in \text{nat} \quad succ(j) \in \text{nat}\)
and \(a : succ(n) \rightarrow G \quad c : succ(j) \rightarrow G\)
by auto

then have \(Concat(a,c) : succ(n) \#+ succ(j) \rightarrow G\)
by (rule concat_props)

with A1 A4 T have
\(succ(n \#+ j) \in \text{nat}\)
\(a \sqcup c : succ(succ(n \#+j)) \rightarrow G\)
\(b(succ(j)) \in G\)
using succ_plus by auto
then have
\((\prod (a \sqcup c) \leftarrow b(succ(j))) = (\prod (a \sqcup c)) \cdot b(succ(j))\)
by (rule prod_append)

with III have \((\prod (a \sqcup c)) \cdot b(succ(j)) = \prod (a \sqcup b)\)
by simp
ultimately have \((\prod a) \cdot (\prod b) = \prod (a \sqcup b)\)
by simp
hence \((\forall b \in succ(succ(j)) \rightarrow G. \quad (\prod a) \cdot (\prod b) = \prod (a \sqcup b))\)
by simp
thus thesis by blast
qed

26.2 Products over sets of indices

In this section we study the properties of expressions of the form \(\prod_{i \in \Lambda} a_i = a_{i_0} \cdot a_{i_1} \cdots a_{i_{n-1}}\), 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 \(a : X \rightarrow G\) 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 \(\prod_{i \in \Lambda} a_i = a_{i_0} \cdot a_{i_1} \cdot \cdots a_{i_{n-1}}\). 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 SetFold(f,a,Λ,r) = Fold1(f,a O Enumeration(Λ,r))

For a finite subset Λ of a linearly ordered set X we will write σ(Λ) to denote the enumeration of the elements of Λ, i.e. the only order isomorphism |Λ| → Λ, where |Λ| ∈ N is the number of elements of Λ. 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 Λ ⊆ X of indices of a sequence a : X → G (i.e. ∏_{i∈Λ} a_i) is denoted ∏(Λ,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 +

  fixes X r
  assumes linord: IsLinOrder(X,r)

  fixes a
  assumes a_is_fun: a : X → G

  fixes σ
  defines σ_def [simp]: σ(Λ) ≡ Enumeration(Λ,r)

  fixes setpr (∏)
  defines setpr_def [simp]: ∏(Λ,b) ≡ SetFold(f,b,Λ,r)

We can use the enums locale in the semigr0 context.

lemma (in semigr1) enums_valid_in_semigr1: shows enums(X,r)
  using linord enums_def by simp

Definition of product over a set expressed in notation of the semigr0 locale.

lemma (in semigr1) setproddef:
  shows ∏(Λ,a) = ∏ (a 0 σ(Λ))
  using SetFold_def by simp

A composition of enumeration of a nonempty finite subset of 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: Λ ∈ FinPow(X) and A2: Λ≠0
  shows
  ∃n ∈ nat . |Λ| = succ(n) ∧ a 0 σ(Λ) : succ(n) → G

235
and \( \prod(\Lambda, a) \in G \)

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 0 \(\sigma(\Lambda) : |\Lambda| \rightarrow G\)
using a_is_fun FinPow_def comp_fun_subset
by simp
with \(<n \in \text{nat}>\) and \(|\Lambda| = \text{succ}(n)>\) show
\(\exists n \in \text{nat} \cdot |\Lambda| = \text{succ}(n) \land a 0 \sigma(\Lambda) : \text{succ}(n) \rightarrow G\)
by auto
from \(<n \in \text{nat}>\) \(<|\Lambda| = \text{succ}(n)>\) \(<a 0 \sigma(\Lambda) : |\Lambda| \rightarrow G>\)
show \(\prod(\Lambda, a) \in G\) using prod_type setproddef
by auto
qed

The \textit{enum_append} lemma from the \textit{Enumeration} theory specialized for natural numbers.

**Lemma**: \textit{(in semigr1) semigr1_enum_append:}

assumes \(\Lambda \in \text{FinPow}(X)\) and
\(n \in X - \Lambda\) and \(\forall k \in \Lambda. \langle k, n \rangle \in r\)
shows \(\sigma(\Lambda \cup \{n\}) = \sigma(\Lambda) \leftrightarrow n\)
using assms FinPow_def enums_valid_in_semigr1
enums.enum_append by simp

What is product over a singleton?

**Lemma**: \textit{(in semigr1) gen_prod_singleton:}

assumes A1: \(x \in X\)
s shows \(\prod(\{x\}, a) = a(x)\)

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 \textit{prod_append} to the products over sets of indices.

**Lemma**: \textit{(in semigr1) gen_prod_append:}

assumes
A1: \(\Lambda \in \text{FinPow}(X)\) and \(\Lambda \neq 0\) and
A2: \(n \in X - \Lambda\) and
A3: \(\forall k \in \Lambda. \langle k, n \rangle \in r\)
shows \(\prod(\Lambda \cup \{n\}, a) = (\prod(\Lambda, a)) \cdot a(n)\)

proof -
have \(\prod(\Lambda \cup \{n\}, a) = \prod (a 0 \sigma(\Lambda \cup \{n\}))\)
using setproddef by simp
also from A1 A3 A4 have \( \prod (a \circ \sigma(L) \leftrightarrow n) \)
using semigr1_enum_append by simp
also have \( \prod (a \circ \sigma(L) \leftrightarrow a(n)) \)
proof -
from A1 A3 have
| \( \Lambda \) | \in \text{nat} and \( \sigma(\Lambda) : \| \Lambda \| \rightarrow X \) and \( n \in X \)
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 \( \prod (a \circ \sigma(\Lambda)) \cdot a(n) \)
using a_is_fun setprod_type apply_funtype prod_append
by blast
also have \( \prod (\Lambda \cup \{n\}, a) = (\prod (\Lambda, a)) \cdot a(n) \)
using SetFold_def by simp
finally show \( \prod (\Lambda \cup \{n\}, a) = (\prod (\Lambda, a)) \cdot a(n) \)
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: \( A \in \text{FinPow}(X) \) and
A2: \( n \in A \) and A4: \( A - \{n\} \neq 0 \) and
A3: \( \forall k \in A. \langle k, n \rangle \in r \)
shows \( \prod (A - \{n\}, a) \cdot a(n) = \prod (A, a) \)
proof -
let \( \Lambda = A - \{n\} \)
from A1 A2 have \( \Lambda \in \text{FinPow}(X) \) and \( n \in X - \Lambda \)
using fin_rem_point_fin FinPow_def by auto
with A3 A4 have \( \prod (\Lambda \cup \{n\}, a) = (\prod (\Lambda, a)) \cdot a(n) \)
using a_is_fun gen_prod_append by simp
with A2 show thesis using rem_add_eq by simp
qed

26.3 Commutative semigroups

Commutative semigroups are those whose operation is commutative, i.e.
\( a \cdot b = b \cdot a \). This implies that for any permutation \( s : n \rightarrow n \) 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: \( \prod_{j=0}^{n} (a_j \cdot b_j) = (\prod_{j=0}^{n} a_j) \cdot (\prod_{j=0}^{n} b_j) \). 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 
\( a \in G \), \( b \in G \), \( c \in G \)

shows \( a \cdot b \cdot c = a \cdot c \cdot b \)

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 
\( A2: a \in G \), \( b \in G \), \( c \in G \), \( d \in G \)

shows \( a \cdot b \cdot (c \cdot d) = a \cdot c \cdot (b \cdot d) \)

proof -

from \( A2 \) have \( a \cdot b \cdot (c \cdot d) = a \cdot b \cdot c \cdot d \)

using semigr_closed semigr_assoc by simp

also have \( a \cdot b \cdot c \cdot d = a \cdot c \cdot (b \cdot d) \)

proof -

from \( A1 \) \( A2 \) have \( a \cdot b \cdot c \cdot d = c \cdot (a \cdot b) \cdot d \)

using IsCommutative_def semigr_closed by simp

also from \( A2 \) have \( ... = c \cdot a \cdot b \cdot d \)

using semigr_closed semigr_assoc by simp

also from \( A1 \) \( A2 \) have \( ... = a \cdot c \cdot b \cdot d \)

using IsCommutative_def semigr_closed by simp

also from \( A2 \) have \( ... = a \cdot c \cdot (b \cdot d) \)

using semigr_closed semigr_assoc by simp

finally show \( a \cdot b \cdot c \cdot d = a \cdot c \cdot (b \cdot d) \) by simp

qed

finally show \( a \cdot b \cdot (c \cdot d) = a \cdot c \cdot (b \cdot d) \) 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 \( A1: k \in \text{nat} \) and 
\( A2: a \in \text{succ}(\text{succ}(k)) \rightarrow G \)

shows \( (\prod a) = (\prod \text{Init}(a)) \cdot a(\text{succ}(k)) \)

proof -

let \( x = \text{Init}(a) \)

from assms have 
\( a(\text{succ}(k)) \in G \) and \( x : \text{succ}(k) \rightarrow G \)

using apply_funtype init_props by auto

with \( A1 \) have \( (\prod x \mapsto a(\text{succ}(k))) = (\prod x) \cdot a(\text{succ}(k)) \)

238
A lemma useful in the induction step of the main theorem.

**Lemma (in semigr0) prod_distr_ind_step:**

assumes A1: \( k \in \mathbb{N} \) and  
A2: \( a : \text{succ}(\text{succ}(k)) \rightarrow G \) and  
A3: \( b : \text{succ}(\text{succ}(k)) \rightarrow G \) and  
A4: \( c : \text{succ}(\text{succ}(k)) \rightarrow G \) and  
A5: \( \forall j \in \text{succ}(\text{succ}(k)). \ c(j) = a(j) \cdot b(j) \)

shows  
\( \text{Init}(a) : \text{succ}(k) \rightarrow G \)  
\( \text{Init}(b) : \text{succ}(k) \rightarrow G \)  
\( \text{Init}(c) : \text{succ}(k) \rightarrow G \)  
\( \forall j \in \text{succ}(k). \ \text{Init}(c)(j) = \text{Init}(a)(j) \cdot \text{Init}(b)(j) \)

**Proof:**

from A1 A2 A3 A4 show  
\( \text{Init}(a) : \text{succ}(k) \rightarrow G \)  
\( \text{Init}(b) : \text{succ}(k) \rightarrow G \)  
\( \text{Init}(c) : \text{succ}(k) \rightarrow G \)  
using init_props by auto

from A1 have T: \( \text{succ}(k) \in \mathbb{N} \) 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: \( n \in \mathbb{N} \)

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 \( \forall a \ b \ c. \)

\[
\begin{align*}
(a : \text{succ}(0) \to G \land b : \text{succ}(0) \to G \land c : \text{succ}(0) \to G) \land \\
(\forall j \in \text{succ}(0). \ c(j) = a(j) \cdot b(j)) \implies \\
(\prod c) = (\prod a) \cdot (\prod b)
\end{align*}
\]

proof

\[
\begin{align*}
\{ \text{fix } a \ b \ c \cdot \\
\text{assume } a : \text{succ}(0) \to G \land b : \text{succ}(0) \to G \land c : \text{succ}(0) \to G \land \\
(\forall j \in \text{succ}(0). \ c(j) = a(j) \cdot b(j))
\}
\]

then have

I: \( a : 1 \to G \land b : 1 \to G \land c : 1 \to G \land \)

II: \( c(0) = a(0) \cdot b(0) \) by auto

from I have

\( (\prod a) = a(0) \) and \( (\prod b) = b(0) \) and \( (\prod c) = c(0) \)

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 \( \forall k \in \text{nat}. \)

\[
\begin{align*}
(a : \text{succ}(k) \to G \land b : \text{succ}(k) \to G \land c : \text{succ}(k) \to G) \land \\
(\forall j \in \text{succ}(k). \ c(j) = a(j) \cdot b(j)) \implies \\
(\prod c) = (\prod a) \cdot (\prod b)
\end{align*}
\]

proof

fix k assume \( k \in \text{nat} \)

show \( (\forall a \ b \ c. \ a \in \text{succ}(k) \to G \land b \in \text{succ}(k) \to G \land c \in \text{succ}(k) \to G \land \\
(\forall j \in \text{succ}(k). \ c(j) = a(j) \cdot b(j)) \implies \\
(\prod c) = (\prod a) \cdot (\prod b)) \land \\
(\forall a \ b \ c. \ a \in \text{succ}(\text{succ}(k)) \to G \land b \in \text{succ}(\text{succ}(k)) \to G \land \\
(\forall j \in \text{succ}(\text{succ}(k)). \ c(j) = a(j) \cdot b(j)) \implies \\
(\prod c) = (\prod a) \cdot (\prod b)) \land \\
(\forall a \ b \ c. \ a \in \text{succ}(\text{succ}(\text{succ}(k))) \to G \land b \in \text{succ}(\text{succ}(\text{succ}(k))) \to G \land \\
(\forall j \in \text{succ}(\text{succ}(\text{succ}(k))). \ c(j) = a(j) \cdot b(j)) \implies \\
(\prod c) = (\prod a) \cdot (\prod b)) \land \\
(\forall a \ b \ c. \ a \in \text{succ}(\text{succ}(\text{succ}(\text{succ}(k)))) \to G \land b \in \text{succ}(\text{succ}(\text{succ}(\text{succ}(k)))) \to G \land \\
(\forall j \in \text{succ}(\text{succ}(\text{succ}(\text{succ}(k))). \ c(j) = a(j) \cdot b(j)) \implies \\
(\prod c) = (\prod a) \cdot (\prod b))
\]

proof

assume \( A3 : \forall a \ b \ c. \)

\[
\begin{align*}
a \in \text{succ}(k) \to G \land \\
b \in \text{succ}(k) \to G \land c \in \text{succ}(k) \to G \land \\
(\forall j \in \text{succ}(k). \ c(j) = a(j) \cdot b(j)) \implies \\
(\prod c) = (\prod a) \cdot (\prod b)
\end{align*}
\]

show \( \forall a \ b \ c. \)

\[
\begin{align*}
a \in \text{succ}(\text{succ}(k)) \to G \land \\
b \in \text{succ}(\text{succ}(k)) \to G \land \\
c \in \text{succ}(\text{succ}(k)) \to G \land \\
(\forall j \in \text{succ}(\text{succ}(k)). \ c(j) = a(j) \cdot b(j)) \implies \\
(\prod c) = (\prod a) \cdot (\prod b)
\end{align*}
\]

proof

assume \( A3 : \forall a \ b \ c. \)

\[
\begin{align*}
a \in \text{succ}(\text{succ}(k)) \to G \land \\
b \in \text{succ}(\text{succ}(k)) \to G \land \\
(\forall j \in \text{succ}(\text{succ}(k)). \ c(j) = a(j) \cdot b(j)) \implies \\
(\prod c) = (\prod a) \cdot (\prod b)
\end{align*}
\]

show \( \forall a \ b \ c. \)

\[
\begin{align*}
a \in \text{succ}(\text{succ}(\text{succ}(k))) \to G \land \\
b \in \text{succ}(\text{succ}(\text{succ}(k))) \to G \land \\
(\forall j \in \text{succ}(\text{succ}(\text{succ}(k))). \ c(j) = a(j) \cdot b(j)) \implies \\
(\prod c) = (\prod a) \cdot (\prod b)
\end{align*}
\]

proof

assume \( A3 : \forall a \ b \ c. \)

\[
\begin{align*}
a \in \text{succ}(\text{succ}(\text{succ}(\text{succ}(k)))) \to G \land \\
b \in \text{succ}(\text{succ}(\text{succ}(\text{succ}(k)))) \to G \land \\
(\forall j \in \text{succ}(\text{succ}(\text{succ}(\text{succ}(k))). \ c(j) = a(j) \cdot b(j)) \implies \\
(\prod c) = (\prod a) \cdot (\prod b)
\end{align*}
\]

proof
\( c \in \text{succ(succ(k))} \rightarrow G \land \\
(\forall j \in \text{succ(succ(k))}, c(j) = a(j) \cdot b(j)) \rightarrow \\
(\prod c) = (\prod a) \cdot (\prod b) \\
\text{proof} - \\
\{ \text{fix a b c} \\
\text{assume} \\
a \in \text{succ(succ(k))} \rightarrow G \land \\
b \in \text{succ(succ(k))} \rightarrow G \land \\
c \in \text{succ(succ(k))} \rightarrow G \land \\
(\forall j \in \text{succ(succ(k))}. c(j) = a(j) \cdot b(j)) \\
\text{with} <k \in \text{nat}> \text{ have I:} \\
a : \text{succ(succ(k))} \rightarrow G \\
b : \text{succ(succ(k))} \rightarrow G \\
c : \text{succ(succ(k))} \rightarrow G \\
\text{and II: } \forall j \in \text{succ(succ(k))}. c(j) = a(j) \cdot b(j) \\
\text{by auto} \\
\text{let} x = \text{Init}(a) \\
\text{let} y = \text{Init}(b) \\
\text{let} z = \text{Init}(c) \\
\text{from} <k \in \text{nat}> \text{ I have III:} \\
(\prod a) = (\prod x) \cdot a(\text{succ(k)}) \\
(\prod b) = (\prod y) \cdot b(\text{succ(k)}) \text{ and} \\
IV: (\prod c) = (\prod z) \cdot c(\text{succ(k)}) \\
\text{using} \text{ shorter_seq} \text{ by auto} \\
\text{moreover} \\
\text{from} <k \in \text{nat}> \text{ I II have} \\
x : \text{succ(k)} \rightarrow G \\
y : \text{succ(k)} \rightarrow G \\
z : \text{succ(k)} \rightarrow G \text{ and} \\
\forall j \in \text{succ(k)}. z(j) = x(j) \cdot y(j) \\
\text{using} \text{ prod_distr_ind_step} \text{ by auto} \\
\text{with A3 II IV have} \\
(\prod c) = (\prod x) \cdot (\prod y) \cdot (a(\text{succ(k)}) \cdot b(\text{succ(k)})) \\
\text{by simp} \\
\text{moreover from A1} <k \in \text{nat}> \text{ I III have} \\
(\prod x) \cdot (\prod y) \cdot (a(\text{succ(k)}) \cdot b(\text{succ(k)})) = \\
(\prod a) \cdot (\prod b) \\
\text{using} \text{ init_props prod_type apply_funtype} \\
\text{rearr4elems by simp} \\
\text{ultimately have} (\prod c) = (\prod a) \cdot (\prod b) \\
\text{by simp} \\
\} \text{ thus thesis by auto} \\
qued \\
qued \\
qued \\
\text{ultimately show thesis by (rule ind_on_nat)} \\
qued 

A reformulation of \text{prod_comm_distr} that is more convenient in applications.
A product of two products over disjoint sets of indices is the product over the union.
from \(<Q \in \text{Bisections}(A \cup \{n\})> \langle A \in \text{FinPow}(X) \rangle \langle n \in X-A \rangle \>
have \(\text{refl}(X,r) \ Q_0 \subseteq A \cup \{n\} \ Q_1 \subseteq A \cup \{n\}\)
\(A \subseteq X\) and \(n \in X\)
using \(\text{linord IsLinOrder_def total_is_refl Bisections_def FinPow_def}\)
by auto
from \(<\text{refl}(X,r) \ Q_0 \subseteq A \cup \{n\} \ A \subseteq X \ A \in X\ II\>
have \(\forall k \in Q_0. \ (k, n) \in r\) by (rule \(\text{refl}_\text{add_point}\))
from \(<\text{refl}(X,r) \ Q_1 \subseteq A \cup \{n\} \ A \subseteq X \ A \in X\ II\>
have \(\forall k \in Q_1. \ (k, n) \in r\) by (rule \(\text{refl}_\text{add_point}\))
from \(<n \in X-A \ Q \in \text{Bisections}(A \cup \{n\})\>
have \(Q_0 = \{n\} \lor Q_1 = \{n\} \lor (Q_0 - \{n\},Q_1-\{n\}) \in \text{Bisections}(A)\)
using \(\text{bisec_is_pair bisec_add_point by simp}\)
moreover
\{ assume \(Q_1 = \{n\}\) from \(<n \in X-A\) have \(n \notin A\) by auto
moreover
from \(<Q \in \text{Bisections}(A \cup \{n\})\>
have \(\langle Q_0,Q_1 \rangle \in \text{Bisections}(A \cup \{n\})\)
using \(\text{bisec_is_pair by simp}\)
with \(<Q_1 = \{n\}\) have \(\langle Q_0, \{n\}\rangle \in \text{Bisections}(A \cup \{n\})\)
by simp
ultimately have \(Q_0 = A\) and \(A \neq 0\)
using \(\text{set_point_bisec by auto}\)
with \(<A \in \text{FinPow}(X)\) \(\langle n \in X-A\ II \ Q_1 = \{n\}\>\)
have \(\prod\langle A \cup \{n\},a \rangle = (\prod\langle Q_0, a \rangle)(\prod\langle Q_1, a \rangle)\)
using \(\text{a_is_fun gen_produ_append gen_produ_singleton}\)
by simp
moreover
\{ assume \(Q_0 = \{n\}\) from \(<n \in X-A\) have \(n \in X\) by auto
then have \(\{n\} \in \text{FinPow}(X)\) and \(\{n\} \neq 0\)
using \(\text{singleton_in_finpow by auto}\)
from \(<n \in X-A\) have \(n \notin A\) by auto
moreover
from \(<Q \in \text{Bisections}(A \cup \{n\})\>
have \(\langle Q_0,Q_1 \rangle \in \text{Bisections}(A \cup \{n\})\)
using \(\text{bisec_is_pair by simp}\)
with \(<Q_0 = \{n\}\) have \(\langle \{n\}, Q_1 \rangle \in \text{Bisections}(A \cup \{n\})\)
by simp
ultimately have \(Q_1 = A\) and \(A \neq 0\) using \(\text{point_set_bisec by auto}\)
with \(A1 \langle A \in \text{FinPow}(X)\) \(\langle n \in X-A\ II\>
\langle n \in \text{FinPow}(X)\) \(\langle \{n\} \neq 0\ \langle Q_0 = \{n\}\>\)
have \(\prod\langle A \cup \{n\},a \rangle = (\prod\langle Q_0, a \rangle)(\prod\langle Q_1, a \rangle)\)
using \(\text{a_is_fun gen_produ_append gen_produ_singleton}\)
setprod_type \(\text{IsCommutative_def by auto}\)
moreover
\{ assume \(A4: \langle Q_0 - \{n\},Q_1 - \{n\}\rangle \in \text{Bisections}(A)\)
with \(<A \in \text{FinPow}(X)\> have
\[ Q_0 - \{n\} \in \text{FinPow}(X) \quad Q_0 - \{n\} \neq 0 \quad \text{and} \]
\[ Q_1 - \{n\} \in \text{FinPow}(X) \quad Q_1 - \{n\} \neq 0 \quad \text{using FinPow_def Bisections_def by auto} \]
with \(<n \in X - A> \) have
\[ \prod (Q_0 - \{n\},a) \in G \quad \prod (Q_1 - \{n\},a) \in G \quad \text{and} \]
\[ T: a(n) \in G \quad \text{using a_is_fun setprod_type apply_funtype by auto} \]
from \(<Q \in \text{Bisections}(A \cup \{n\})> \) A4 have
\[ (\langle Q_0, Q_1 - \{n\} \rangle \in \text{Bisections}(A) \land n \in Q_1) \lor \]
\[ (\langle Q_0 - \{n\}, Q_1 \rangle \in \text{Bisections}(A) \land n \in Q_0) \quad \text{using bisec_is_pair bisec_add_point_case3 by auto} \]
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 \(<A \in \text{FinPow}(X)> \) \(<n \in X - A> \) I II IV
\[ \langle Q_0, Q_1 - \{n\} \rangle \in \text{Bisections}(A) \quad \prod (Q_0, a) \in G > \]
\[ \prod (Q_1 - \{n\}, a) \in G \quad \prod (Q_1 \in \text{FinPow}(X) > \]
\[ n \in Q_1 \quad Q_1 - \{n\} \neq 0 \quad \text{have} \quad \prod (A \cup \{n\}, a) = (\prod (Q_0, a)) \cdot (\prod (Q_1, a)) \]
\quad \text{using gen_prod_append semigr_assoc gen_product_rem_point by simp} \}
moreover
\{ assume \( \langle Q_0 - \{n\}, Q_1 \rangle \in \text{Bisections}(A) \) and \( n \in Q_0 \) then have \( A \neq 0 \) using bisec_props by simp \}
with A1 A2 \(<A \in \text{FinPow}(X)> \) \(<n \in X - A> \) I II III T
\[ \langle Q_0 - \{n\}, Q_1 \rangle \in \text{Bisections}(A) \quad \prod (Q_0 - \{n\}, a) \in G > \]
\[ \prod (Q_1, a) \in G \quad Q_0 \in \text{FinPow}(X) > \quad n \in Q_0 \quad Q_0 - \{n\} \neq 0 \quad \text{have} \quad \prod (A \cup \{n\}, a) = (\prod (Q_0, a)) \cdot (\prod (Q_1, a)) \]
\quad \text{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 }
\[ \text{thus thesis by simp} \]
\]
\] thus thesis by simp
\]
\] thus thesis by simp
\]
\] thus thesis by simp
\]
\] moreover note A2 ultimately show thesis by (rule fin_ind_add_max)
\]
A better looking reformulation of prod_bisect.

**theorem (in semigr1) prod_disjoint:** assumes
\[ A1: f \{ \text{is commutative on} \} G \quad \text{and} \]
\[ A2: A \in \text{FinPow}(X) \quad A \neq 0 \quad \text{and} \]

244
A3: $B \in \text{FinPow}(X)$ $B \neq 0$ and
A4: $A \cap B = 0$
shows $\prod(A \cup B, a) = (\prod(A, a)) \cdot (\prod(B, a))$

proof -
from A2 A3 A4 have $(A, B) \in \text{Bisections}(A \cup B)$
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} $\rightarrow$
$(\prod\{\langle i, \prod(M(i), a) \rangle. i \in \text{succ}(n)\}) = (\prod(\bigcup i \in \text{succ}(n). M(i), a))$

proof -
note A2
moreover have $\forall M \in \text{succ}(0) \rightarrow \text{FinPow}(X)$.
M {is partition} $\rightarrow$
$(\prod\{\langle i, \prod(M(i), a) \rangle. i \in \text{succ}(0)\}) = (\prod(\bigcup i \in \text{succ}(0). M(i), a))$
using a_is_fun func1_1_L1 Partition_def apply_functype 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} $\rightarrow$
$(\prod\{\langle i, \prod(M(i), a) \rangle. i \in \text{succ}(k)\}) = (\prod(\bigcup i \in \text{succ}(k). M(i), a))$ $\rightarrow$
$(\forall M \in \text{succ}(\text{succ}(k)) \rightarrow \text{FinPow}(X)$.
M {is partition} $\rightarrow$
$(\prod\{\langle i, \prod(M(i), a) \rangle. i \in \text{succ}(\text{succ}(k))\}) = (\prod(\bigcup i \in \text{succ}(\text{succ}(k)). M(i), a))$ $\rightarrow$
proof -
{ fix $k$ assume $k \in \text{nat}$
assume A3: $\forall M \in \text{succ}(k) \rightarrow \text{FinPow}(X)$.
M {is partition} $\rightarrow$
$(\prod\{\langle i, \prod(M(i), a) \rangle. i \in \text{succ}(k)\})$ $\rightarrow$
$(\prod(\bigcup i \in \text{succ}(k). M(i), a))$
have $(\forall N \in \text{succ}(\text{succ}(k)) \rightarrow \text{FinPow}(X)$.
N {is partition} $\rightarrow$
$(\prod\{\langle i, \prod(N(i), a) \rangle. i \in \text{succ}(\text{succ}(k))\})$ $\rightarrow$
$(\prod(\bigcup i \in \text{succ}(\text{succ}(k)). N(i), a))$
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

245
let b = \{\langle i, \prod (N(i), a) \rangle \mid i \in \text{succ}(\text{succ}(k))\}
let c = \{\langle i, \prod (N(i), a) \rangle \mid 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 = \{\langle i, \prod (N(i), a) \rangle \mid 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)

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 \langle N : \text{succ}(\text{succ}(k)) \rightarrow \text{FinPow}(X)\rangle
  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 \langle M : \text{succ}(k) \rightarrow \text{FinPow}(X)\rangle have M \{is partition\}
    using func1_1_L1 Partition_def by simp
  with A3 \langle M : \text{succ}(k) \rightarrow \text{FinPow}(X)\rangle have
    (\prod (\{i, \prod (M(i), a) \mid i \in \text{succ}(k)\})) =
    (\prod \{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 \langle k \in \text{nat} \rangle have \text{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 \( \langle \text{succ}(k) \in \text{nat} \rangle \) A4 A5 have \( A \cap B = 0 \) by (rule list_partition)
moreover note A1
ultimately have \( \prod \langle A \cup B, a \rangle = (\prod \langle A, a \rangle) \cdot (\prod \langle B, a \rangle) \)
using prod_disjoint by simp
moreover have \( A \cup B = (\bigcup i \in \text{succ}(\text{succ}(k)). N(i)) \)
by auto
ultimately show thesis by simp
qed
finally have \( (\prod \{ \langle i, \prod \langle N(i), a \rangle \rangle. i \in \text{succ}(\text{succ}(k)) \}) = (\prod (\bigcup i \in \text{succ}(\text{succ}(k)). N(i), a)) \)
by simp
}\) thus thesis by auto
qed
} thus thesis by simp
qed
ultimately show thesis by (rule ind_on_nat)
qed

A more convenient reformulation of prod_list_of_lists.

theorem (in semigr1) prod_list_of_sets:
assumes A1: f \{is commutative on} G and
A2: n \in \text{nat} n \neq 0 and
A3: M : n \to \text{FinPow}(X) M \{is partition\}
shows
\( (\prod \{ \langle i, \prod \langle M(i), a \rangle \rangle. i \in n \}) = (\prod (\bigcup i \in n. M(i), a)) \)
proof -
from A2 obtain k where k \in \text{nat} and n = \text{succ}(k)
using Nat_ZF_1_L3 by auto
with A1 A3 show thesis using prod_list_of_lists
by simp
qed

The definition of the product \( \prod \langle A, a \rangle \equiv \text{SetFold}(f, a, A, r) \) 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 \( \prod \langle A, a \rangle = \prod a \circ \sigma(A) \) 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 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 setproddef. The proof is based on application of prod_list_of_sets to the finite collection of singletons that comprise \( A \).

theorem (in semigr1) prod_order_irr:
assumes A1: f \{is commutative on} G and
A2: A \in \text{FinPow}(X) A \neq 0 and
A3: b \in \text{bij}(|A|, A)
shows \( \prod (a \circ b) = \prod (A, a) \)

proof -
let \( n = |A| \)
let \( M = \{ \langle k, \{ b(k) \} \rangle. k \in n \} \)
have \( \prod (a \circ b) = \prod \{ \langle i, \prod (M(i), a) \rangle. i \in n \} \)
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 \( \text{bij_def inj_def apply_functype FinPow_def} \)
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 \( \text{bij_def comp_fun_apply} \)
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 \( \{ \langle i, \prod (M(i), a) \rangle. i \in n \} = \{ \langle i, (a \circ b)(i) \rangle. i \in n \} \)
by simp
moreover have \( \{ \langle i, (a \circ b)(i) \rangle. i \in n \} = a \circ b \)
proof -
  from \( A3 \) have \( b : n \to A \)
using \( \text{bij_def inj_def} \)
by simp
  moreover have \( A2 \ A3 \) 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 \( \text{bij_def inj_def apply_functype FinPow_def} \)
  \( \text{singleton_in_finpow} \) 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 \( \text{bij_def inj_def apply_functype FinPow_def} \)
  \( \text{singleton_in_finpow} \) 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 \( \ldots = (\prod (\bigcup i \in n. M(i), a)) \)
by simp

248
\[
(\prod \{i, \prod (M(i), a)\}. i \in n\} = (\prod (\bigcup i \in n. M(i), a))
\]
by (rule 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 \((\prod (a \circ b)) = (\prod (a \circ c))\)
using assms prod_order_irr by simp

27 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.

27.1 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 \(n = \{0, 1, \ldots, n-1\}\) 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 $\{\text{Fold1}(f, a \circ b). b \in \text{bij}(\mid A \mid, A)\}$ is a singleton and we
can extract its only value by taking its union.

definition
CommSetFold$(f, a, \Lambda) = \bigcup \{\text{Fold1}(f, a \circ b). b \in \text{bij}(\mid \Lambda \mid, \Lambda)\}$

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: $f \text{ is associative on } G$

assumes csgcomm: $f \text{ is commutative on } G$

fixes csgsum (infixl + 69)
defines csgsum_def[simp]: $x + y \equiv f(x, y)$

fixes $X$ $a$

assumes csgaisfun: $a : X \rightarrow G$

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 $(\sum(A, a)) = (\bigcup \{\sum(a \circ b). b \in \text{bij}(\mid A \mid, A)\})$
  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 func-
tion we sum up is linearly ordered, while in commsemigr we don’t have this
assumption.

lemma (in commsemigr) sum_over_set_bij:
assumes $A_1$: $A \in \text{FinPow}(X)$ $A \neq 0$ and $A_2$: $b \in \text{bij}(|A|, A)$

shows $(\sum (A, a)) = (\sum (a \ 0 \ b))$

proof -

have

\[ \forall c \in \text{bij}(|A|, A). \ \forall d \in \text{bij}(|A|, A). \ (\sum (a \ 0 \ c)) = (\sum (a \ 0 \ d)) \]

proof -

{ fix c assume $c \in \text{bij}(|A|, A)$
  fix d assume $d \in \text{bij}(|A|, A)$
  let $r = \text{InducedRelation}($\text{converse}(c), \text{Le})$
  have $\text{semigr1}(G, f, A, r, \text{restrict}(a, A))$
  proof -
    have $\text{semigr0}(G, f)$ using $\text{csgassoc}$ $\text{semigr0_def}$ by simp
    moreover from $A_1 < c \in \text{bij}(|A|, A) >$ have $\text{IsLinOrder}(A, r)$
      using $\text{bij_converse_bij}$ $\text{card_fin_is_nat}$
      $\text{natord_lin_on_each_nat}$ $\text{ind_rel_pres_lin}$ by simp
    moreover from $A_1$ have $\text{restrict}(a, A) : A \rightarrow G$
      using $\text{FinPow_def}$ $\text{csgaisfun}$ $\text{restrict_fun}$ by simp
    ultimately show thesis using $\text{semigr1_axioms.intro}$ $\text{semigr1_def}$
      by simp
    qed
    moreover have $f$ \{is commutative on\} $G$ using $\text{csgcomm}$
      by simp
    moreover from $A_1$ have $A \in \text{FinPow}(A) A \neq 0$
      using $\text{FinPow_def}$ by auto
    moreover note $< c \in \text{bij}(|A|, A) > < d \in \text{bij}(|A|, A) >$
    ultimately have
    $\text{Fold1}(f, \text{restrict}(a, A) \ 0 \ c) = \text{Fold1}(f, \text{restrict}(a, A) \ 0 \ d)$
    by (rule $\text{semigr1.prod_bij_same}$)
    hence $(\sum (\text{restrict}(a, A) \ 0 \ c)) = (\sum (\text{restrict}(a, A) \ 0 \ d))$
    by simp
    moreover from $A_1 < c \in \text{bij}(|A|, A) > < d \in \text{bij}(|A|, A) >$
    have
    $\text{restrict}(a, A) \ 0 \ c = a \ 0 \ c$ and $\text{restrict}(a, A) \ 0 \ d = a \ 0 \ d$
    using $\text{bij_def}$ $\text{surj_def}$ $\text{csgaisfun}$ $\text{FinPow_def}$ $\text{comp_restrict}$
    by auto
    ultimately have $(\sum (a \ 0 \ c)) = (\sum (a \ 0 \ d))$ by simp
    } thus thesis by blast
  qed

with $A_2$ have $(\bigcup \{ \sum (a \ 0 \ b) \}. \ b \in \text{bij}(|A|, A)) = (\sum (a \ 0 \ b))$
  by (rule $\text{singleton_comprehension}$)

then show thesis using $\text{CommSetFolddef}$ by simp

qd

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 coresponds to adding a value.

lemma (in $\text{commsemigr}$) sum_over_set_add_point:
assumes $A_1$: $A \in \text{FinPow}(X) \ A \neq 0$
shows $\sum (A,a) \in G$ and
\[ \forall x \in X-A. \sum (A \cup \{x\},a) = (\sum (A,a)) + a(x) \]

proof
- from A1 obtain $b$ where $b \in \text{bij}(|A|,A)$
  using fin_bij_card by auto
with A1 have $\sum (A,a) = (\sum (a \circ b))$
  using sum_over_set_bij by simp
from A1 have $|A| \in \text{nat}$ using card_fin_is_nat by simp
have semigr0(G,f) using csgassoc semigr0_def by simp
moreover from A1 obtain $n$ where $n \in \text{nat}$ and $|A| = \text{succ}(n)$
  using card_non_empty_succ by auto
with A1 have $\sum (A,a) = (\sum (a \circ b))$
  using sum_over_set_bij by simp
from A1 have $|A| \in \text{nat}$ using card_fin_add_one by simp
{ fix $x$ assume $x \in X-A$
  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 $\text{Append}(b,x) \in \text{bij}(|A|, A \cup \{x\})$
    using bij_def inj_def FinPow_def comp_fun_subset csgaisfun
    by auto
  ultimately have $\text{Fold1}(f,a \circ b) \in G$ by (rule semigr0.prod_type)
  with $\sum (A,a) = (\sum (a \circ b))$ show $\sum (A,a) \in G$
    by simp
  qed }
moreover have $\text{Append}(b,x) \in \text{bij}(\text{succ}(|A|), A \cup \{x\})$
  using bij_append_point by auto
with A1 have $\text{thesis}$ using list_compose_append by simp
qed

ultimately have $(\sum (A \cup \{x\},a)) = (\sum (a \circ \text{Append}(b,x)))$
  using sum_over_set_bij by simp
also have ... = $(\sum \text{Append}(a \circ b, a(x)))$
  proof
  - note $|A| \in \text{nat}$
  moreover from A1 have $b \in \text{bij}(|A|, A)$
    b : $|A| \rightarrow A$ and $A \subseteq X$
    using bij_def inj_def using FinPow_def by auto
  then have $b : |A| \rightarrow X$ by (rule func1_1_L1B)
  moreover from $x \in X-A$ have $x \in X$ and $a : X \rightarrow G$
  using csgaisfun by auto
  ultimately show thesis using list_compose_append
  by simp
  qed
  also have ... = $(\sum (A,a)) + a(x)$
  proof
  - note $\text{semigr0}(G,f)$
    $n \in \text{nat}$ $a \circ b : \text{succ}(n) \rightarrow G$

252
moreover from \( \langle x \in X-A \rangle \) have \( a(x) \in G \)

using `cs_gaisfun` apply_funtype by simp

ultimately have

\[
\text{Fold1}(f, \text{Append}(a \ O \ b, a(x))) = f(\text{Fold1}(f, a \ O \ b), a(x))
\]

by (rule `semigr0.prod_append`)

with \( A \ O b \in \text{bij}(|A|, A) \) show thesis

using `sum_over_set_bij` by simp

qed

finally have

\[
(\sum(A \cup \{x\}, a)) = (\sum(A, a)) + a(x)
\]

by simp

\}

thus \( \forall x \in X-A. \sum(A \cup \{x\}, a) = (\sum(A, a)) + a(x) \)

by simp

qed

end

28 Monoids

theory Monoid_ZF imports func_ZF Loop_ZF

begin

This theory provides basic facts about monoids.

28.1 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 \( \langle x, y \rangle \), 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**

\[
\text{IsAmonoid}(G,f) \equiv \\
\text{f (is associative on} \ G \ \land \\
(\exists e \in G. (\forall g \in G. ((f(\langle e, g \rangle) = g) \ \land \ (f(\langle g, e \rangle) = g))))
\]

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 \( \sqcup \) symbol to denote the monoid operation (for no particular reason).

locale monoid0 =

fixes \( G f \)
assumes monoidAsssum: IsAmonoid(G,f)

fixes monoper (infixl ⊕ 70)
defines monoper_def [simp]: a ⊕ b ≡ f(a,b)

The result of the monoid operation is in the monoid (carrier).

lemma (in monoid0) group0_1_L1:
  assumes a∈G b∈G shows a⊕b ∈ G
  using assms monoidAsssum IsAmonoid_def IsAssociative_def apply_funtype
  by auto

There is only one neutral element in a monoid.

lemma (in monoid0) group0_1_L2: shows
  ∃!e. e∈G ∧ (∀ g∈G. (e⊕g = g) ∧ g⊕e = g)
proof
  fix e y
  assume e ∈ G ∧ (∀g∈G. e ⊕ g = g ∧ g ⊕ e = g)
  and y ∈ G ∧ (∀g∈G. y ⊕ g = g ∧ g ⊕ y = g)
  then have y⊕e = y y⊕e = e by auto
  thus e = y by simp
next from monoidAsssum show
  ∃ e. e∈G ∧ (∀ g∈G. e⊕g = g ∧ g⊕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 ∈ G ∧ (∀ g∈G. e ⊕ g = g ∧ g ⊕ e = g)
proof
  let n = THE b. b∈G ∧ (∀ g∈G. b⊕g = g ∧ g⊕b = g)
  have ∃!b. b∈G ∧ (∀ g∈G. b⊕g = g ∧ g⊕b = g)
    using group0_1_L2 by simp
  then have n∈G ∧ (∀ g∈G. n⊕g = g ∧ g⊕n = g)
    by (rule thel)
  with A1 show thesis
    using TheNeutralElement_def by simp
qed

The monoid carrier is not empty.

lemma (in monoid0) group0_1_L3A: shows G≠0
proof
  have TheNeutralElement(G,f) ∈ G using unit_is_neutral
    by simp
  thus thesis by auto
qed

The range of the monoid operation is the whole monoid carrier.
lemma (in monoid0) group0_1_L3B: shows range(f) = G
proof
from monoidAsssum have f : G×G→G
  using IsAmonoid_def IsAssociative_def by simp
then show range(f) ⊆ G
  using func1_1_L5B by simp
show G ⊆ range(f)
proof
  fix g assume A1: g∈G
  let e = TheNeutralElement(G,f)
  from A1 have ⟨e,g⟩ ∈ G×G g = f⟨e,g⟩
    using unit_is_neutral by auto
  with <f : G×G→G> show g ∈ range(f)
    using func1_1_L5A by blast
qed
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×G) = G
  using monoidAsssum IsAmonoid_def IsAssociative_def
  group0_1_L3B range_image_domain by auto

In a monoid any neutral element is the neutral element.

lemma (in monoid0) group0_1_L4: assumes A1: e ∈ G ∧ (∀g∈G. e ⊕ g = g ∧ g ⊕ e = g)
  shows e = TheNeutralElement(G,f)
proof -
  let n = THE b. b∈ G ∧ (∀g∈G. b ⊕ g = g ∧ g ⊕ b = g)
  have ∃!b. b∈G ∧ (∀g∈G. b ⊕ g = g ∧ g ⊕ 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: 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
  \( g(e,h) = e \oplus h \land g(h,e) = h \oplus e \)
  using restrict_if by simp
with A3 A4 A6 A2 show
  \( g(e,h) = h \land g(h,e) = h \)
  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: \( H \subseteq G \)
and A3: TheNeutralElement(G,f) \( \in H \)
shows IsAmonoid(H,restrict(f,H \times H))
proof -
  let \( g = \) restrict(f,H \times H)
  let \( e = \) TheNeutralElement(G,f)
  from monoidAsssum have \( f \in G \times G \rightarrow G \)
  using IsAmonoid_def IsAssociative_def by simp
  moreover from A2 have \( H \times H \subseteq G \times G \) by auto
  moreover from A1 have \( \forall p \in H \times H. f(p) \in H \)
  using IsOpClosed_def by auto
  ultimately have \( g \in H \times H \rightarrow H \)
  using func1_2_L4 by simp
  moreover have \( \forall x \in H. \forall y \in H. \forall z \in H. x \oplus y \oplus z = g(g(x,g(y,z)) \)
  proof -
    from A1 have \( \forall x \in H. \forall y \in H. \forall z \in H. \)
    \( g(g(x,y),z) = x \oplus y \oplus z \).
    using IsOpClosed_def restrict_if by simp
    moreover have \( \forall x \in H. \forall y \in H. \forall z \in H. x \oplus y \oplus z = x \oplus (y \oplus z) \)
    proof -
      from monoidAsssum have \( \forall x \in G. \forall y \in G. \forall z \in G. \) \( x \oplus y \oplus z = x \oplus (y \oplus z) \)
      using IsAmonoid_def IsAssociative_def by simp
      with A2 show thesis by auto
    qed
    moreover from A1 have \( \forall x \in H. \forall y \in H. \forall z \in H. \) \( x \oplus (y \oplus z) = g(x,g(y,z)) \)
    using IsOpClosed_def restrict_if by simp
    ultimately show thesis by simp
  qed
moreover have \( \exists n \in H. (\forall h \in H. g(n,h) = h \land g(h,n) = h) \)
proof -
  from A1 have \( \forall x \in H. \forall y \in H. x \oplus y \in H \)
using IsOpClosed_def by simp

with A2 A3 have
  \( \forall h \in H. \, g(e,h) = h \land g(h,e) = h \)
  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 \subseteq G
  and A4: TheNeutralElement(G,f) \in H

shows TheNeutralElement(H,restrict(f,H \times H)) = TheNeutralElement(G,f)

proof -
  let e = TheNeutralElement(G,f)
  let g = restrict(f,H \times H)

  from assms have monoid0(H,g)
    using monoid0_def monoid0.group0_1_T1 by simp

  moreover have
    e \in H \land (\forall h \in H. \, g(e,h) = h \land g(h,e) = h)

  proof -
    { fix h assume h \in H
      with assms have
        monoid0(G,f) \forall x \in H. \forall y \in H. \, f(x,y) \in H
        H \subseteq G \, e = TheNeutralElement(G,f) \, g = restrict(f,H \times H)
        e \in H \, h \in H
      using monoid0_def IsOpClosed_def by auto
      then have g(e,h) = h \land g(h,e) = h
      by (rule monoid0.group0_1_L5)
      } hence \forall h \in H. \, g(e,h) = h \land 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

  qed

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 TheNeutralElement(G,f)

shows a \neq TheNeutralElement(G,f) \lor b \neq TheNeutralElement(G,f)

using assms unit_is_neutral by auto

257
29 Groups - introduction

theory Group_ZF imports Monoid_ZF

begin

This theory file covers basics of group theory.

29.1 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
IsAgroup(G,f) ≡ (IsAmonoid(G,f) ∧ (∀ g ∈ G. ∃ b ∈ G. f⟨g,b⟩ = TheNeutralElement(G,f)))

We define the group inverse as the set \{⟨x,y⟩ ∈ G × G : x · y = e\}, where \( e \) is the neutral element of the group. This set (which can be written as \((·)^{-1}\{e\}\)) 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 \( x · y = e \) this relation is in fact a function from \( G \) to \( G \).

definition
GroupInv(G,f) ≡ \{⟨x,y⟩ ∈ G × G. f⟨x,y⟩ = 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 ≡ TheNeutralElement(G,P)

fixes groper (infixl · 70)
defines groper_def[simp]: a · b ≡ P⟨a,b⟩

fixes inv (\( _{-1} \) [90] 91)
defines inv_def[simp]: \( x^{-1} \) ≡ GroupInv(G,P)(x)

First we show a lemma that says that we can use theorems proven in the monoid0 context (locale).
lemma (in group0) group0_2_L1: shows monoid0(G,P)  
  using groupAssum IsAgroup_def monoid0_def by simp

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 \( \forall g \in G. \exists b \in G. f(g,b) = \text{TheNeutralElement}(G,f) \)  
  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 \( 1 \in G \wedge (\forall g \in G. (1 \cdot g = g \wedge g \cdot 1 = g)) \)  
  using group0_2_L1 monoid0.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\in G b \in G  
  shows a\cdot b \in G using assms group0_2_L1 monoid0.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\in G b \in G c \in G  
  shows a \cdot (b \cdot c) = a \cdot b \cdot 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 conveniet 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.

definition_left_inverse:
  assumes A1: g\in G and A2: b\in G and A3: g \cdot b = 1  
  shows b \cdot g = 1  
proof -
  from A2 groupAssum obtain c where I: c \in G \wedge 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 \( b \cdot (g \cdot b \cdot c) = 1 \) using group0_2_L2 by simp
finally show \( b \cdot g = 1 \) by simp

qed

For every element of a group there is only one inverse.

lemma (in group0) group0_2_L4:
  assumes A1: \( x \in \text{G} \) shows \( \exists ! y. y \in \text{G} \land x \cdot y = 1 \)
proof
  from A1 groupAssum show \( \exists y. y \in \text{G} \land x \cdot y = 1 \)
    using IsAgroup_def by auto
fix y n
assume A2: \( y \in \text{G} \land x \cdot y = 1 \) and A3: \( n \in \text{G} \land x \cdot n = 1 \) show \( y = n \)
proof -
  from A1 A2 have T1: \( y \cdot x = 1 \)
    using group0_2_T1 by simp
from A2 A3 have \( y = y \cdot (x \cdot n) \)
    using group0_2_L2 by simp
also from A1 A2 A3 have \( \ldots = (y \cdot x) \cdot n \)
    using group_oper_assoc by blast
also from T1 A3 have \( \ldots = n \)
    using group0_2_L2 by simp
finally show \( y = n \) by simp
qed

qed

The group inverse is a function that maps \( \text{G} \) into \( \text{G} \).

theorem (in group0) group0_2_T2:
  assumes A1: IsAgroup(\( \text{G}, f \)) shows GroupInv(\( \text{G}, f \)) : \( \text{G} \rightharpoonup \text{G} \)
proof -
  have GroupInv(\( \text{G}, f \)) \subseteq \( \text{G} \times \text{G} \) using GroupInv_def by auto
moreover from A1 have \( \forall x \in \text{G}. \exists ! y. y \in \text{G} \land \langle x, y \rangle \in \text{GroupInv}(\text{G}, f) \)
    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(\( \text{G}, P \))
proof -
  from groupAssum have \( P : \text{G} \times \text{G} \rightarrow \text{G} \)
    using IsAgroup_def IsAmonoid_def IsAssociative_def by simp
  then show \( P^{-1} = \text{GroupInv}(\text{G}, P) \)

260
using func1_1_L14 GroupInv_def by auto
defauto

qed

The inverse is in the group.

lemma (in group0) inverse_in_group: assumes A1: x ∈ G shows x⁻¹ ∈ G
proof -
  from groupAssum have GroupInv(G,P) : G → 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 ∈ G shows x⁻¹ · x = 1 ∧ x · x⁻¹ = 1
proof
  from groupAssum have GroupInv(G,P) : G → G
    using group0_2_T2 by simp
  with A1 have ⟨x,x⁻¹⟩ ∈ GroupInv(G,P)
    using apply_Pair by simp
  then show x⁻¹ · x = 1 using GroupInv_def by simp
  with A1 show x · x⁻¹ = 1 using inverse_in_group group0_2_T1
    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 ∈ G and A2: b ∈ G and A3: a·b = a
  shows b = 1
proof -
  from A3 have a⁻¹ · (a·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 ∈ G and A2: b ∈ G and A3: a·b = b
  shows a = 1
proof -
  from A3 have (a·b)·b⁻¹ = b·b⁻¹ by simp
  with A1 A2 have a·(b·b⁻¹) = b·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.

261
lemma (in group0) group_inv_of_one: shows $1^{-1} = 1$
  using group0_2_L2 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: $a \in G$ and A2: $a^{-1} = 1$
  shows $a = 1$
proof -
  from A1 have $a \cdot a^{-1} = 1$ 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 $a \in G$ and $a \neq 1$
  shows $a^{-1} \neq 1$ 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 $a \in G$ and $a^{-1} \neq 1$
  shows $a \neq 1$
  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: $a \in G$ and A2: $b \in G$ and A3: $a \cdot b = 1$
  shows $a = b^{-1}$ and $b = a^{-1}$
proof -
  from A3 have $a \cdot b^{-1} = 1 \cdot b^{-1}$ by simp
  with A1 A2 have $a \cdot (b \cdot b^{-1}) = 1 \cdot b^{-1}$ 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 $a^{-1} \cdot (a \cdot b) = a^{-1} \cdot 1$ 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: $\forall g \in G. \ b(g) \in G \land g \cdot b(g) = 1$
  shows $\forall g \in G. \ b(g) = g^{-1}$
proof
  fix $g$ assume $g \in G$

262
moreover from A1 \(<g \in G>\) have \(b(g) \in G\) by simp
moreover from A1 \(<g \in G>\) have \(gb(g) = 1\) 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: \(a \in G\) and A2: \(b \in G\)
  shows \(b^{-1}a^{-1} = (a\cdot b)^{-1}\)
proof -
  from A1 A2 have \(b^{-1} \in G\) a \(\in G\) b \(\in G\) a \(\cdot b^{-1} \in G\)
  using inverse_in_group group_op_closed
  by auto
  with A1 show \(a\cdot b^{-1}a^{-1} = 1\)
  using group0_2_L6
  with \(<a \cdot b \in G>\) \(<b^{-1}a^{-1} \in G>\) show \(b^{-1}a^{-1} = (a\cdot b)^{-1}\)
  using group_oper_assoc
  by simp
  qed

What is the inverse of a product of three elements?

lemma (in group0) group_inv_of_three:
  assumes A1: \(a \in G\) b \(\in G\) c \(\in G\)
  shows \((a\cdot b\cdot c)^{-1} = c^{-1},(a\cdot b)^{-1}\)
\((a\cdot b\cdot c)^{-1} = c^{-1},(b^{-1}a^{-1})\)
\((a\cdot b\cdot c)^{-1} = c^{-1},b^{-1}a^{-1}\)
proof -
  from A1 have T:
  \(a\cdot b \in G\) a \(\in G\) b \(\in G\) c \(\in G\)
  using group_op_closed inverse_in_group by auto
  with A1 show \((a\cdot b\cdot c)^{-1} = c^{-1},(a\cdot b)^{-1}\) \(\text{and} (a\cdot b\cdot c)^{-1} = c^{-1},(b^{-1}a^{-1})\)
  using group_inv_of_two by auto
  with T show \((a\cdot b\cdot c)^{-1} = c^{-1},b^{-1}a^{-1}\) using group_oper_assoc
  by simp
  qed

The inverse of the inverse is the element.

lemma (in group0) group_inv_of_inv:
  assumes a \(\in G\) shows \(a = (a^{-1})^{-1}\)
Group inverse is nilpotent, therefore a bijection and involution.

**lemma (in group0) group_inv_bij:**
shows $\text{GroupInv}(G,P) \circ \text{GroupInv}(G,P) = \text{id}(G)$ and $\text{GroupInv}(G,P) \in \text{bij}(G,G)$
and $\text{GroupInv}(G,P) = \text{converse}(\text{GroupInv}(G,P))$

**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 $\text{id}(G): G \rightarrow G$
using comp_fun id_type by auto
moreover

- let $g \in G$
  - using comp_fun_apply and group_inv_of_inv by simp
- hence $\forall g \in G. (\text{GroupInv}(G,P) \circ \text{GroupInv}(G,P))(g) = \text{id}(G)(g)$ by simp
ultimately show $\text{GroupInv}(G,P) \circ \text{GroupInv}(G,P) = \text{id}(G)$
by (rule func_eq)

with I show $\text{GroupInv}(G,P) \in \text{bij}(G,G)$ using nilpotent_imp_bijective
by simp
with $\text{GroupInv}(G,P) \circ \text{GroupInv}(G,P) = \text{id}(G)$ 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 $V \subseteq G$ and $g \in \text{GroupInv}(G,P)(V)$
shows $g^{-1} \in V$
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 $\text{GroupInv}(G,P)(V) = \text{GroupInv}(G,P)^{-1}(V)$
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 $1 \in A$

264
shows \( 1 \in \text{GroupInv}(G,P)(A) \)
proof -
  have \( \text{GroupInv}(G,P):G\to G \) using \( \text{groupAssum \ group0_2_T2} \) by simp
  with \( \text{assms} \) have \( 1^{-1} \in \text{GroupInv}(G,P)(A) \) using \( \text{func_imagedef} \) by auto
  then show thesis using \( \text{group_inv_of_one} \) by simp
qed

The group inverse is onto.

lemma (in group0) group_inv_surj: shows \( \text{GroupInv}(G,P)(G) = G \)
  using \( \text{group_inv_bij \ bij_def \ surj_range_image_domain} \) by auto

If \( a^{-1} \cdot b = 1 \), then \( a = b \).

lemma (in group0) group0_2_L11:
  assumes \( A1: a \in G \ \ b \in G \) and \( A2: a^{-1} \cdot b = 1 \)
  shows \( a = b \)
proof -
  from \( A1 \ \ A2 \) have \( a^{-1} \in G \ \ b \in G \ \ a^{-1} \cdot b = 1 \)
  using \( \text{inverse_in_group \ by \ auto} \)
  then have \( b = (a^{-1})^{-1} \) by \( \text{rule group0_2_L9} \)
  with \( A1 \) show \( a = b \) using \( \text{group_inv_of_inv \ by \ simp} \)
qed

If \( a \cdot b^{-1} = 1 \), then \( a = b \).

lemma (in group0) group0_2_L11A:
  assumes \( A1: a \in G \ \ b \in G \) and \( A2: a \cdot b^{-1} = 1 \)
  shows \( a = b \)
proof -
  from \( A1 \ \ A2 \) have \( a \in G \ \ b^{-1} \in G \ \ a \cdot b^{-1} = 1 \)
  using \( \text{inverse_in_group \ by \ auto} \)
  then have \( a = (b^{-1})^{-1} \) by \( \text{rule group0_2_L9} \)
  with \( A1 \) show \( a = b \) using \( \text{group_inv_of_inv \ by \ simp} \)
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 \( A1: a \in G \) and \( A2: b^{-1} \neq a \)
  shows \( a^{-1} \neq b \)
proof -
  \{ assume \( a^{-1} = b \)
    then have \( (a^{-1})^{-1} = b^{-1} \) by simp
    with \( A1 \ \ A2 \) have \( \text{False \ using \ group_inv_of_inv} \)
    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 ∈ G  b ∈ G
shows
(a⋅b⁻¹)⁻¹ = b⋅a⁻¹
(a⁻¹⋅b)⁻¹ = b⁻¹⋅a
proof -
from A1 have
(a⋅b⁻¹)⁻¹ = (b⁻¹)⁻¹, a⁻¹ and (a⁻¹⋅b)⁻¹ = b⁻¹⋅(a⁻¹)⁻¹
using inverse_in_group group_inv_of_two by auto
with A1 show (a⋅b⁻¹)⁻¹ = b⋅a⁻¹  (a⁻¹⋅b)⁻¹ = b⁻¹⋅a
using group_inv_of_inv by auto
qed

A couple useful rearrangements with three elements: we can insert a b⋅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 ∈ G  b ∈ G  c ∈ G
shows
a⋅c⁻¹ = (a⋅b⁻¹)⋅(b⋅c⁻¹)
(a⁻¹⋅c⁻¹) = (a⁻¹⋅b)⋅(b⁻¹⋅c)
a⋅(b⋅c⁻¹) = a⋅c⁻¹⋅b⁻¹
a⋅(b⋅c⁻¹) = a⋅b⋅c⁻¹
(a⋅b⁻¹⋅c⁻¹)⁻¹ = c⋅b⋅a⁻¹
a⋅b⋅c⁻¹⋅(c⋅b⁻¹) = a
a⋅(b⋅c⁻¹) = a⋅b
proof -
from A1 have T:
a⁻¹ ∈ G  b⁻¹∈G  c⁻¹∈G
a⁻¹⋅b ∈ G  a⋅b⁻¹ ∈ G  a⋅b ∈ G
a⋅b⁻¹ ∈ G  b⋅c ∈ G
using inverse_in_group group_op_closed
by auto
from A1 T have
a⋅c⁻¹ = a⋅(b⁻¹⋅b)⋅c⁻¹
a⁻¹⋅c⁻¹ = a⁻¹⋅(b⋅b⁻¹)⋅c⁻¹
using group0_2_L2 group0_2_L6 by auto
with A1 T show
a⋅c⁻¹ = (a⋅b⁻¹)⋅(b⋅c⁻¹)
a⁻¹⋅c⁻¹ = (a⁻¹⋅b)⋅(b⁻¹⋅c)
using group_oper_assoc by auto
from A1 have a⋅(b⋅c⁻¹) = a⋅(c⁻¹⋅b⁻¹)
using group_inv_of_two by simp
with A1 T show a⋅(b⋅c⁻¹) = a⋅c⁻¹⋅b⁻¹
using group_oper_assoc by simp
from A1 T show a⋅(b⋅c⁻¹) = a⋅b⋅c⁻¹
using group_oper_assoc by simp
from A1 T show (a⋅b⁻¹⋅c⁻¹)⁻¹ = c⋅b⋅a⁻¹
using group_inv_of_three group_inv_of_inv
by simp

266
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 \(a \cdot b \cdot c^{-1} \cdot (c \cdot b^{-1}) = a\) by simp
from A1 T have \(a \cdot (b \cdot c) \cdot c^{-1} = a \cdot (b \cdot (c \cdot c^{-1}))\) 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 \(a \cdot b \cdot b^{-1} = c^{-1}\)
  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 \((a \cdot b) \cdot (c \cdot d)^{-1} = a \cdot (b \cdot d^{-1}) \cdot a^{-1} \cdot (a \cdot c^{-1})\)
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 \( \ldots = a \cdot (b \cdot d^{-1}) \cdot c^{-1} \)
using `group_oper_assoc` by simp
also from A1 T1 have \( \ldots = a \cdot (b \cdot d^{-1}) \cdot a^{-1} \cdot (a \cdot c^{-1}) \)
using `group0_2_L14A` by blast
finally show thesis by simp
qed

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} \cdot b = a \cdot (b \cdot b^{-1})
  \quad a^{-1} \cdot (a \cdot b) = a \cdot (a^{-1} \cdot b) = a \cdot (a^{-1} \cdot b)
proof -
  from A1 have
    a \cdot b \cdot b^{-1} \cdot b = a \cdot (b^{-1} \cdot b) \quad a \cdot b \cdot b^{-1} = a \cdot (b \cdot b^{-1})
    a^{-1} \cdot (a \cdot b) = a^{-1} \cdot a \cdot b \quad a \cdot (a^{-1} \cdot b) = a \cdot (a^{-1} \cdot b)
    using `inverse_in_group` `group_oper_assoc` by auto
with A1 show
  a \cdot b \cdot b^{-1} = a
  a \cdot b \cdot b^{-1} = a
  a^{-1} \cdot (a \cdot b) = a
  a \cdot (a^{-1} \cdot b) = b
    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^{-1}
proof -
  from A1 have (b \cdot a)^{-1} = a^{-1} \cdot b^{-1} \quad b^{-1} \in G
    using `group_inv_of_two` `inverse_in_group` by auto
with A1 show a \cdot (b \cdot a)^{-1} = b^{-1} using `inv_cancel_two`
    by simp
qed
```

Some other identities with three element 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 \cdot c^{-1}) \cdot c = a \cdot b
  a \cdot b^{-1} \cdot (b \cdot c^{-1}) = a \cdot c^{-1}
proof -
```

268
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: \( H \subseteq G \) and \( H \) {is closed under} \( P \)
shows \( (H \cup \{1\}) \) {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: \( a \in G \) \( b \in G \)
and A2: \( c = a \cdot b \)
shows \( c^{-1} \cdot b^{-1} = a^{-1} \cdot c = b \)
proof
from A2 A1 have \( c \cdot b^{-1} = a \cdot (b \cdot b^{-1}) \cdot a^{-1} \cdot c = (a^{-1} \cdot a) \cdot b \)
  using inverse_in_group group_oper_assoc by auto
moreover from A1 have \( a \cdot (b \cdot b^{-1}) = a \cdot (a^{-1} \cdot a) \cdot b = b \)
  using group0_2_L6 group0_2_L2 by auto
ultimately show \( c \cdot b^{-1} = a^{-1} \cdot c = b \)
  by auto
qed

We can cancel an element on the right from both sides of an equation.
lemma (in group0) cancel_right: assumes \( a \in G \) \( b \in G \) \( c \in G \) \( a \cdot b = c \cdot b \) 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
qed

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

29.2 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
\[
\text{IsAsubgroup}(H,P) \equiv \text{IsAgroup}(H, \text{restrict}(P,H\times H))
\]

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: IsAsubgroup(H,f) and A2: n = TheNeutralElement(H,restrict(f,H×H))

shows n ∈ H
∀ h∈H. restrict(f,H×H)(n,h) = h
∀ h∈H. restrict(f,H×H)(h,n) = h

**proof**

let b = restrict(f,H×H)
let e = TheNeutralElement(H,restrict(f,H×H))
from A1 have group0(H,b)
  using IsAsubgroup_def group0_def by simp
then have I:
  e ∈ H ∧ (∀h∈H. (b(e,h) = h ∧ b(h,e) = h))
by (rule group0.group0_2_L2)
with A2 show n ∈ H by simp
from A2 I show ∀ h∈H. b(n,h) = h and ∀ h∈H. b(h,n) = h by auto

qed

A subgroup is contained in the group.

**lemma (in group0) group0_3_L2:**

assumes A1: IsAsubgroup(H,P)

shows H ⊆ G

**proof**

fix h assume h∈H
let b = restrict(P,H×H)
let n = TheNeutralElement(H,restrict(P,H×H))
from A1 have b ∈ H×H→H
  using IsAsubgroup_def IsAgroup_def IsAmonoid_def IsAssociative_def by simp
moreover from A1 <h∈H> have ⟨ n,h ⟩ ∈ H×H
  using group0_3_L1 by simp
moreover from A1 <h∈H> have h = b(n,h )
  using group0_3_L1 by simp
ultimately have ⟨⟨n,h⟩,h⟩ ∈ b
  using func1_1_L5A by blast
then have ⟨⟨n,h⟩,h⟩ ∈ P using restrict_subset by auto
moreover from groupAssum have P:G×G→G
  using IsAgroup_def IsAmonoid_def IsAssociative_def by simp
ultimately show h∈G using func1_1_L5
  by blast

271
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 \( \text{IsAsubgroup}(H,P) \)
- 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 \( A1: \text{IsAsubgroup}(H,P) \)
- Shows \( \text{TheNeutralElement}(H, \text{restrict}(P, H \times H)) = 1 \)

**Proof -**

- Let \( n = \text{TheNeutralElement}(H, \text{restrict}(P, H \times H)) \)
- 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 \( A1: \text{IsAsubgroup}(H,P) \)
- 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 \( A1: \text{IsAsubgroup}(H,P) \)
  - and \( A2: \ a \in H \ b \in H \)
- Shows \( a \cdot b \in H \)

**Proof -**

- Let \( f = \text{restrict}(P, H \times H) \)
- From \( A1 \) have \( \text{monoid0}(H,f) \) 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 \( A1: \text{IsAgroup}(G,f) \)
  - and \( A2: \text{IsAsubgroup}(H,f) \) and \( A3: \ g = \text{restrict}(f,H \times H) \)
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 $\text{group0}(G,f)$ using group0_def by simp
- from A2 A3 have $\text{group0}(H,g)$ using IsAsubgroup_def group0_def by simp
- from $\langle \text{group0}(G,f) \rangle$ A2 A3 have $\text{GroupInv}(G,f) = f^{-\{e_1\}}$ 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 $\text{GroupInv}(H,g) = g^{-\{e_1\}}$ 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 $\text{GroupInv}(H,g) = \text{restrict}(\text{GroupInv}(G,P),H)$
  **proof** -
    from groupAssum have $\text{GroupInv}(G,P) : G \rightarrow G$ using group0_2_T2 by simp
    moreover from A1 A2 have $\text{GroupInv}(H,g) : H \rightarrow H$
    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 $\text{GroupInv}(G,P) \cap H \times H = \text{GroupInv}(H,g)$
    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 \( A1: \text{IsAsubgroup}(H, P) \) 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 \( A1: H \neq \emptyset \) and \( A2: H \subseteq G \) and \( A3: H \) (is closed under) \( P \) and \( A4: \forall x \in H. x^{-1} \in H \)

shows \( \text{IsAsubgroup}(H, P) \)

**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 group0_2_L1 monoid0.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 \( A2 \ A3 \) have \( 1 \in H \)

using IsAGroup_def group0_1_L6 by auto

moreover from \( A4 \) have \( g(h, h^{-1}) = h \cdot h^{-1} \)

using restrict_if by simp

ultimately have \( g(h, h^{-1}) = n \) by simp

with \( A4 \) have \( \exists b \in H. g(h, b) = n \) by auto

qed

ultimately show \( \text{IsAsubgroup}(H, P) \) using
Intersection of subgroups is a subgroup. This lemma is obsolete and should be replaced by subgroup_inter.

```plaintext
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)) 
proof - 
  let e = TheNeutralElement(G,f) 
  let g = restrict(f,H₁×H₁) 
  from A1 have I: group0(G,f) using group0_def by simp 
  from A2 have group0(H₁,g) using IsAsubgroup_def group0_def by simp 
  moreover have H₁∩H₂ ≠ 0 
    proof - 
      from A1 A2 A3 have e ∈ H₁∩H₂ using group0_3_L5 by auto 
      thus thesis by auto 
    qed 
  moreover have H₁∩H₂ ⊆ G using group0_3_L2 by auto 
  moreover have H₁∩H₂ {is closed under} g unfolding IsOpClosed_def using group0_3_L6 func_ZF_4_L7 func_ZF_4_L5 by simp 
  moreover have ∀ x ∈ H₁∩H₂. GroupInv(H₁,g)(x) ∈ H₁∩H₂ 
    using group0.group0_3_T2 group0.group0_3_T3A by simp 
  ultimately show thesis using group0.group0_3_T3 by simp 
qed
```

Intersection of subgroups is a subgroup.

```plaintext
lemma (in group0) subgroup_inter: assumes IsAsubgroup(H₁,P) and IsAsubgroup(H₂,P) shows IsAsubgroup(H₁∩H₂,P) 
proof - 
  from asssms have H₁∩H₂ ≠ 0 using group0_3_L5 by auto 
  moreover from asssms have H₁∩H₂ ⊆ G using group0_3_L2 by auto 
  moreover from asssms have H₁∩H₂ {is closed under} P unfolding IsOpClosed_def using group0_3_L6 func_ZF_4_L7 func_ZF_4_L5 by simp 
  moreover have ∀ x ∈ H₁∩H₂. x⁻¹ ∈ H₁∩H₂ 
    using group0_3_T2 group0_3_T3A by simp 
  ultimately show thesis using group0_3_T3 by auto 
qed
```
The range of the subgroup operation is the whole subgroup.

proof -
  from A1 have monoid0(H,restrict(P,H×H))
    using IsAsubgroup_def IsAgroup_def monoid0_def by simp
  then show thesis by (rule monoid0.range_carr)
qed

If we restrict the inverse to a subgroup, then the restricted inverse is onto
the subgroup.

lemma (in group0) restr_inv_onto: assumes A1: IsAsubgroup(H,P) shows restrict(GroupInv(G,P),H)(H) = H
proof -
  from A1 have GroupInv(H,restrict(P,H×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(H1,P) and IsAsubgroup(H2,P) shows IsAsubgroup(H1∪H2,P) ←→ (H1⊆H2 ∨ H2⊆H1)
proof
  assume H1⊆H2 ∨ H2⊆H1 show IsAsubgroup(H1∪H2,P)
  proof -
    from <H1⊆H2 ∨ H2⊆H1> have H2 = H1∪H2 ∨ H1 = H1∪H2 by auto
    with assms show IsAsubgroup(H1∪H2,P) by auto
  qed
next
  assume IsAsubgroup(H1∪H2,P) show H1⊆H2 ∨ H2⊆H1
  proof -
    { assume ¬ H1⊆H2
      then obtain x where x∈H1 and x∉H2 by auto
      with assms(1) have x⁻¹ ∈ H1 using group0_3_T3A by simp
      { fix y assume y∈H2
        let z = x·y
        from <x∈H1> <y∈H2> have x ∈ H1∪H2 and y ∈ H1∪H2 by auto
        with <IsAsubgroup(H1∪H2,P)> have z ∈ H1∪H2 using group0_3_L6
        by blast
        from assms <x ∈ H1∪H2> <y∈H2> have x∈G y∈G and y⁻¹∈H2
          using group0_3_T3A group0_3_L2 by auto
        then have z·y⁻¹ = x and x⁻¹·z = y using inv_cancel_two(2,3) by auto
        { assume z ∈ H2
          then x = z·y⁻¹ and y⁻¹·z = y by simp
          with H1⊆H2 have x ∈ H1 by auto
          with assms H1⊆H2 show thesis
        }
      }
    }
  qed
Transitivity for "is a subgroup of" relation. The proof (probably) uses the lemma restrict_restrict from standard Isabelle/ZF library which states that restrict(restrict(f,A),B) = restrict(f,A∩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(G_3,P) IsAsubgroup(G_2,P) IsAsubgroup(G_1,restrict(P,G_2×G_2))
  shows IsAsubgroup(G_1,P)
proof -
  from assms(2) have group0(G_2,restrict(P,G_2×G_2)) unfolding IsAsubgroup_def group0_def by simp
  with assms(3) have G_1⊆G_2 using group0.group0_3_L2 by simp
  hence G_2×G_2 ∩ G_1×G_1 = G_1×G_1 by auto
  with assms(3) show IsAsubgroup(G_1,P) unfolding IsAsubgroup_def by simp
qed

29.3 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 z_1 z_2 assume z_1∈G ∧ x·z_1 = y and z_2∈G ∧ x·z_2 = y
      with <x∈G> have z_1 = z_2 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
    with <IsAsubgroup(H_2,P)> y⁻¹∈H_2> have z·y⁻¹ ∈ H_2 using group0_3_L6 by simp
    with <z·y⁻¹ = x> <x∉H_2> have False by auto
    } hence z ∉ H_2 by auto
  with assms(1) <x⁻¹ ∈ H_1> <z ∈ H_1∪H_2> have x⁻¹·z ∈ H_1 using group0_3_L6 by simp
  hence H_2 ⊆ H_1 by blast
  thus thesis by blast
qed
then have $yx^{-1} \in G$ \land (yx^{-1}) \cdot x = y$ using \texttt{group_op_closed inverse_in_group inv_cancel_two(1)}

by simp

hence $\exists z. z \in G \land z \cdot x = y$ by auto

moreover

\{ fix $z_1 \ z_2$ assume $z_1 \in G \land z_1 \cdot x = y$ and $z_2 \in G \land z_2 \cdot x = y$

with $\langle x \in G \rangle$ have $z_1 = z_2$ using cancel_right by blast

\}

ultimately have $\exists ! z. z \in G \land z \cdot x = y$ by auto

} then show HasRightDiv(G,P) unfolding HasRightDiv_def by simp

qed

A group is a quasigroup and a loop.

\textbf{lemma} (in group0) group_is_loop: shows IsAquasigroup(G,P) and IsAloop(G,P)

\textbf{proof} -

show IsAquasigroup(G,P) unfolding IsAquasigroup_def HasLatinSquareProp_def

using gr_has_lr_div group_oper_fun by simp

then show IsAloop(G,P) unfolding IsAloop_def using group0_2_L2 by auto

qed

An associative loop is a group.

\textbf{theorem} assoc_loop_is_gr: assumes IsAloop(G,P) and $P \{\text{is associative on} G\}$

shows IsAgroup(G,P)

\textbf{proof} -

from assms(1) have $\exists e \in G. \ \forall x \in G. \ P\langle e, x \rangle = x$ \land $P\langle x, e \rangle = x$

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) $\langle x \in G \rangle$ have $y \in G$ \land $P\langle x, y \rangle = \text{TheNeutralElement}(G,P)$

using loop_loop0_valid loop0.lr_inv.props(3,4) by auto

hence $\exists y \in G. \ P\langle x, y \rangle = \text{TheNeutralElement}(G,P)$ by auto

\}

with $\langle \text{IsAmonoid}(G,P) \rangle$ show IsAgroup(G,P) unfolding IsAgroup_def by simp

qed

For groups the left and right inverse are the same as the group inverse.

\textbf{lemma} (in group0) lr_inv_gr_inv:

shows LeftInv(G,P) = GroupInv(G,P) and RightInv(G,P) = GroupInv(G,P)

\textbf{proof} -

have LeftInv(G,P):$G \rightarrow G$ using group_is_loop loop_loop0_valid loop0.lr_inv.props(3,4) 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 $y = \text{LeftInv}(G,P)(x)$

\}

\}

278
30 Groups 1

theory Group_ZF_1 imports Group_ZF

begin

In this theory we consider right and left translations and odd functions.

30.1 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 \( (a,b) \) as \( P(a,b) \) instead of the usual infix \( a \cdot b \) or \( a + b \).
definition
RightTranslation(G,P,g) ≡ {(a,b) ∈ G×G. P(a,g) = b}

A similar definition of the left translation.
definition
LeftTranslation(G,P,g) ≡ {(a,b) ∈ G×G. P(g,a) = b}

Translations map G into G. Two dimensional translations map G × G into itself.

lemma (in group0) group0_5_L1: assumes A1: g ∈ G
shows RightTranslation(G,P,g) : G → G and LeftTranslation(G,P,g) : G → G
proof -
  from A1 have ∀a∈G. a·g ∈ G and ∀a∈G. g·a ∈ G
    using group_oper_fun apply_funtype by auto
  then show RightTranslation(G,P,g) : G → 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 ∈ G a ∈ G
shows RightTranslation(G,P,g)(a) = a·g
LeftTranslation(G,P,g)(a) = g·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 ∈ G h ∈ G a ∈ G and
A2: T_g = LeftTranslation(G,P,g) T_h = LeftTranslation(G,P,h)
sows T_g(T_h(a)) = g·h·a
T_g(T_h(a)) = LeftTranslation(G,P,g·h)(a)
proof -
  from A1 have I: h·a ∈ G g·h ∈ G
    using group_oper_fun apply_funtype by auto
  with A1 A2 show T_g(T_h(a)) = g·h·a
    using group0_5_L2 group_oper_assoc by simp
  with A1 A2 I show
    T_g(T_h(a)) = LeftTranslation(G,P,g·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 ∈ G h ∈ G a ∈ G and

A2: \( T_g = \text{RightTranslation}(G,P,g) \) \( T_h = \text{RightTranslation}(G,P,h) \)

shows
\[
T_g(T_h(a)) = a \cdot h \cdot g
\]
\[
T_g(T_h(a)) = \text{RightTranslation}(G,P,h \cdot g)(a)
\]

proof -
from A1 have I: \( a \cdot h \in G \) \( h \cdot g \in G \)
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 \( T_g(T_h(a)) = \text{RightTranslation}(G,P,h \cdot g)(a) \)
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\( \in G \) h\( \in G \) 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 asssms have \( T_g : G \rightarrow G \) and \( T_h : G \rightarrow G \)
using group0_5_L1 by auto
then have \( T_g \circ T_h : G \rightarrow G \) using comp_fun by simp
moreover from asssms have RightTranslation(G,P,h·g):G→G
using group_op_closed group0_5_L1 by simp
moreover from asssms \( T_h : G \rightarrow G \) 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 asssms have \( T_g : G \rightarrow G \) and \( T_h : G \rightarrow G \)
using group0_5_L1 by auto
then have \( T_g \circ T_h : G \rightarrow G \) using comp_fun by simp
moreover from asssms have LeftTranslation(G,P,g·h):G→G
using group_op_closed group0_5_L1 by simp
moreover from asssms \( T_h : G \rightarrow G \) 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: \( g \in G \) \( h \in G \) and

281
A2: \( T_g = \text{LeftTranslation}(G,P,g) \) \( T_h = \text{LeftTranslation}(G,P,h) \)
shows \( T_g(T_h(A)) = \text{LeftTranslation}(G,P,g \cdot h)(A) \)
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: \( g \in G \) \( h \in G \) and A2: \( A \subseteq G \)
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 \( T_g : G \rightarrow G \) \( T_h : G \rightarrow G \)
using group0_5_L1 by auto
with assms \( \forall a \in A. \ a \in G \) 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
\( \text{RightTranslation}(G,P,1) = \text{id}(G) \) and \( \text{LeftTranslation}(G,P,1) = \text{id}(G) \)
proof -
have \( \text{RightTranslation}(G,P,1) : G \rightarrow G \) and \( \forall a \in G. \ \text{RightTranslation}(G,P,1)(a) = a \)
using group0_2_L2 group0_5_L1 group0_5_L2 by auto
then show \( \text{RightTranslation}(G,P,1) = \text{id}(G) \) by (rule indentity_fun)
have \( \text{LeftTranslation}(G,P,1) : G \rightarrow G \) and \( \forall a \in G. \ \text{LeftTranslation}(G,P,1)(a) = a \)
using group0_2_L2 group0_5_L1 group0_5_L2 by auto
then show \( \text{LeftTranslation}(G,P,1) = \text{id}(G) \) by (rule indentity_fun)
qed

Translation by neutral element does not move sets.

lemma (in group0) trans_neutral_image: assumes \( V \subseteq G \)
shows \( \text{RightTranslation}(G,P,1)(V) = V \) and \( \text{LeftTranslation}(G,P,1)(V) = V \)
using assms 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 \( g \in G \)
says \( \text{RightTranslation}(G,P,g) \circ \text{RightTranslation}(G,P,g^{-1}) = \text{id}(G) \) and \( \text{RightTranslation}(G,P,g^{-1}) \circ \text{RightTranslation}(G,P,g) = \text{id}(G) \) and \( \text{LeftTranslation}(G,P,g) \circ \text{LeftTranslation}(G,P,g^{-1}) = \text{id}(G) \) and \( \text{LeftTranslation}(G,P,g^{-1}) \circ \text{LeftTranslation}(G,P,g) = \text{id}(G) \)
Translations are bijective.

**Lemma (in group0) trans_bij:** assumes \( g \in G \) shows

\[
\text{RightTranslation}(G,P,g) \in \text{bij}(G,G) \text{ and } \text{LeftTranslation}(G,P,g) \in \text{bij}(G,G)
\]

**proof** -

from assms have

\[
\text{RightTranslation}(G,P,g) : G \to G \text{ and } \text{RightTranslation}(G,P,g^{-1}) : G \to G
\]

\[
\text{RightTranslation}(G,P,g) \circ \text{RightTranslation}(G,P,g^{-1}) = id(G)
\]

\[
\text{RightTranslation}(G,P,g^{-1}) \circ \text{RightTranslation}(G,P,g) = id(G)
\]

using inverse_in_group group0_2_L6 trans_comp group0_2_L6 trans_neutral by auto

then show \( \text{RightTranslation}(G,P,g) \in \text{bij}(G,G) \) using fg_imp_bijective

by simp

from assms have

\[
\text{LeftTranslation}(G,P,g) : G \to G \text{ and } \text{LeftTranslation}(G,P,g^{-1}) : G \to G
\]

\[
\text{LeftTranslation}(G,P,g) \circ \text{LeftTranslation}(G,P,g^{-1}) = id(G)
\]

\[
\text{LeftTranslation}(G,P,g^{-1}) \circ \text{LeftTranslation}(G,P,g) = id(G)
\]

using inverse_in_group group0_5_L1 trans_comp_id by auto

then show \( \text{LeftTranslation}(G,P,g) \in \text{bij}(G,G) \) using fg_imp_bijective

by simp

qed

Converse of a translation is translation by the inverse.

**Lemma (in group0) trans_conv_inv:** assumes \( g \in G \) shows

\[
\text{converse}(\text{RightTranslation}(G,P,g)) = \text{RightTranslation}(G,P,g^{-1}) \text{ and } \text{converse}(\text{LeftTranslation}(G,P,g)) = \text{LeftTranslation}(G,P,g^{-1})
\]

**proof** -

from assms have

\[
\text{RightTranslation}(G,P,g) \in \text{bij}(G,G) \text{ and } \text{RightTranslation}(G,P,g^{-1}) \in \text{bij}(G,G)
\]

and

\[
\text{LeftTranslation}(G,P,g) \in \text{bij}(G,G) \text{ and } \text{LeftTranslation}(G,P,g^{-1}) \in \text{bij}(G,G)
\]

using trans_bij inverse_in_group by auto

moreover from assms have

\[
\text{RightTranslation}(G,P,g^{-1}) \circ \text{RightTranslation}(G,P,g) = id(G) \text{ and } \text{LeftTranslation}(G,P,g^{-1}) \circ \text{LeftTranslation}(G,P,g) = id(G)
\]

\[
\text{LeftTranslation}(G,P,g^{-1}) \circ \text{LeftTranslation}(G,P,g) = id(G) \text{ and } \text{RightTranslation}(G,P,g^{-1}) \circ \text{RightTranslation}(G,P,g) = id(G)
\]

ultimately show

\[
\text{converse}(\text{RightTranslation}(G,P,g)) = \text{RightTranslation}(G,P,g^{-1}) \text{ and } \text{converse}(\text{LeftTranslation}(G,P,g)) = \text{LeftTranslation}(G,P,g^{-1})
\]

\[
\text{LeftTranslation}(G,P,g) = \text{converse}(\text{LeftTranslation}(G,P,g^{-1})) \text{ and } \text{RightTranslation}(G,P,g) = \text{converse}(\text{RightTranslation}(G,P,g^{-1}))
\]

using comp_id_conv by auto

283
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 \( g \in G \) 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 assms \ 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 \( g \in G \) 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 \( T : G \rightarrow G \) and \( F : G \rightarrow G \)  
using group0_5_L1 \ group_oper_fun \ fix_2nd_var_fun \ by \ auto  
moreover from assms have \( \forall a \in G. \ T(a) = F(a) \)  
using group0_5_L2 \ group_oper_fun \ fix_var_val \ by \ simp  
ultimately show \( T = F \) by \( \text{rule func_eq} \)

next  
let \( T = \) LeftTranslation\((G,P,g)\)  
let \( F = \) Fix1stVar\((P,g)\)  
from assms have \( T : G \rightarrow G \) and \( F : G \rightarrow G \)  
using group0_5_L1 \ group_oper_fun \ fix_1st_var_fun \ by \ auto  
moreover from assms have \( \forall a \in G. \ T(a) = F(a) \)  
using group0_5_L2 \ group_oper_fun \ fix_var_val \ by \ simp  
ultimately show \( T = F \) by \( \text{rule func_eq} \)

qed

A lemma demonstrating what is the left translation of a set

**Lemma (in group0) ltrans_image:** Assumes \( A1 : V \subseteq G \) and \( A2 : x \in G \) shows \( \text{LeftTranslation}(G,P,x)(V) = \{x \cdot v. \ v \in V\} \)

proof -  
from assms have \( \text{LeftTranslation}(G,P,x)(V) = \{\text{LeftTranslation}(G,P,x)(v). \ v \in V\} \)  
using group0_5_L1 \ func_imagedef \ by \ blast  
moreover from assms have \( \forall v \in V. \ \text{LeftTranslation}(G,P,x)(v) = x \cdot v \)  
using group0_5_L2 \ by \ auto  
ultimately show thesis by \( \text{auto} \)

qed

A lemma demonstrating what is the right translation of a set

**Lemma (in group0) rtrans_image:** Assumes \( A1 : V \subseteq G \) and \( A2 : x \in G \) shows \( \text{RightTranslation}(G,P,x)(V) = \{v \cdot x. \ v \in V\} \)

proof -  

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 \( x \in G \)

shows \( \text{LeftTranslation}(G,P,x)(V) \subseteq G \) and \( \text{RightTranslation}(G,P,x)(V) \subseteq G \)

**Proof**

- from assms have \( \text{LeftTranslation}(G,P,x) : G \rightarrow G \) and \( \text{RightTranslation}(G,P,x) : G \rightarrow G \)

  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(2) 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 \( A1 \) 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 
$(P \{\text{lifted to subsets of} \} \ G) \langle A,B \rangle = (\bigcup_{a \in A} \text{LeftTranslation}(G,P,a)(B))$
proof
from assms have I: $(P \{\text{lifted to subsets of} \} \ G) \langle A,B \rangle = \{a \cdot b \cdot \langle a,b \rangle \in A \times B\}$
using group_oper_fun lift_subsets_explained by simp
{ fix c assume $c \in (P \{\text{lifted to subsets of} \} \ G) \langle A,B \rangle$
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 $<a \in A>$ 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 auto
} thus $(P \{\text{lifted to subsets of} \} \ G) \langle A,B \rangle \subseteq (\bigcup_{a \in A} \text{LeftTranslation}(G,P,a)(B))$ by auto
{ fix $c$ assume $c \in (\bigcup_{a \in A} \text{LeftTranslation}(G,P,a)(B))$
then obtain $a \in A$ and $c \in \text{LeftTranslation}(G,P,a)(B)$ by auto
moreover from assms $-a \in A$ have $\text{LeftTranslation}(G,P,a)(B) = \{a \cdot b. b \in B\}$
using ltrans_image by auto
ultimately obtain $b$ where $b \in B$ and $c = a \cdot b$ by auto
with $<a \in A>$ have $c \in (P \{\text{lifted to subsets of} \} \ G) \langle A,B \rangle$ by auto
} thus $(\bigcup_{a \in A} \text{LeftTranslation}(G,P,a)(B)) \subseteq (P \{\text{lifted to subsets of} \} \ G) \langle A,B \rangle$ by auto
{ fix $c$ assume $c \in (\bigcup_{b \in B} \text{RightTranslation}(G,P,b)(A))$
then obtain $a \in A$ and $c \in \text{RightTranslation}(G,P,b)(A)$ by auto
moreover from assms $-b \in B$ have $\text{RightTranslation}(G,P,b)(A) = \{a \cdot b. a \in A\}$ using rtrans_image by auto
ultimately have $c \in \text{RightTranslation}(G,P,b)(A)$ by simp
with $<-b \in B>$ have $c \in (\bigcup_{b \in B} \text{RightTranslation}(G,P,b)(A))$ by auto
} thus $(P \{\text{lifted to subsets of} \} \ G) \langle A,B \rangle \subseteq (\bigcup_{b \in B} \text{RightTranslation}(G,P,b)(A))$ by auto
{ fix $c$ assume $c \in (\bigcup_{b \in B} \text{RightTranslation}(G,P,b)(A))$
then obtain $b \in B$ and $c \in \text{RightTranslation}(G,P,b)(A)$ by auto
moreover from assms $-b \in B$ have $\text{RightTranslation}(G,P,b)(A) = \{a \cdot b.$
} qed

The right translation version of image_ltrans_union
The proof follows the same schema.

lemma (in group0) image_rtrans_union: assumes $A \subseteq G \ B \subseteq G$ shows 
$(P \{\text{lifted to subsets of} \} \ G) \langle A,B \rangle = (\bigcup_{b \in B} \text{RightTranslation}(G,P,b)(A))$
proof
from assms have I: $(P \{\text{lifted to subsets of} \} \ G) \langle A,B \rangle = \{a \cdot b \cdot \langle a,b \rangle \in A \times B\}$
using group_oper_fun lift_subsets_explained by simp
{ fix c assume $c \in (P \{\text{lifted to subsets of} \} \ G) \langle A,B \rangle$
with I obtain $a \ b$ where $c = a \cdot b$ and $a \in A \ b \in B$ by auto
hence $c \in \{a \cdot b. a \in A\}$ by auto
moreover from assms $-b \in B$ have
$\text{RightTranslation}(G,P,b)(A) = \{a \cdot b. a \in A\}$ using rtrans_image by auto
ultimately have $c \in \text{RightTranslation}(G,P,b)(A)$ by simp
with $<-b \in B>$ have $c \in (\bigcup_{b \in B} \text{RightTranslation}(G,P,b)(A))$ by auto
} thus $(P \{\text{lifted to subsets of} \} \ G) \langle A,B \rangle \subseteq (\bigcup_{b \in B} \text{RightTranslation}(G,P,b)(A))$ by auto
{ fix $c$ assume $c \in (\bigcup_{b \in B} \text{RightTranslation}(G,P,b)(A))$
then obtain $b \in B$ and $c \in \text{RightTranslation}(G,P,b)(A)$ by auto
moreover from assms $-b \in B$ have $\text{RightTranslation}(G,P,b)(A) = \{a \cdot b.$
}
a ∈ A

using rtrans_image by auto
ultimately obtain a where a ∈ A and c = a · b by auto
with I < b < B have c ∈ (P {lifted to subsets of} G)(A, B) by auto
} thus ( ∪ b ∈ B. RightTranslation(G, P, b)(A)) ⊆ (P {lifted to subsets of} G)(A, B)
by auto
qed

If the neutral element belongs to a set, then an element of group belongs
the translation of that set.

lemma (in group0) neut_trans_elem:
assumes A1: A ⊆ G g ∈ G and A2: 1 ∈ A
shows g ∈ LeftTranslation(G, P, g)(A) g ∈ RightTranslation(G, P, g)(A)
proof -
from assms have g · 1 ∈ LeftTranslation(G, P, g)(A)
using ltrans_image by auto
with A1 show g ∈ LeftTranslation(G, P, g)(A) using group0_2_L2 by simp
from assms have 1 · g ∈ RightTranslation(G, P, g)(A)
using rtrans_image by auto
with A1 show g ∈ 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 A1: A ⊆ G and A2: g ∈ A
shows 1 ∈ LeftTranslation(G, P, g⁻¹)(A) 1 ∈ RightTranslation(G, P, g⁻¹)(A)
proof -
from assms have g⁻¹ ∈ G using inverse_in_group by auto
with assms have g⁻¹ · g ∈ LeftTranslation(G, P, g⁻¹)(A)
using ltrans_image by auto
moreover from assms have g⁻¹ · g = 1 using group0_2_L6 by auto
ultimately show 1 ∈ LeftTranslation(G, P, g⁻¹)(A) by simp
from g⁻¹ assms have g · g⁻¹ ∈ RightTranslation(G, P, g⁻¹)(A)
using rtrans_image by auto
moreover from assms have g · g⁻¹ = 1 using group0_2_L6 by auto
ultimately show 1 ∈ RightTranslation(G, P, g⁻¹)(A) by simp
qed

30.2 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 IsOdd(G, P, f) ≡ (∀ a ∈ 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 IsOdd(G,P,p) ←→ (∀a∈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 A1: p : G → G
  shows (∀a∈G. p(a⁻¹) = (p(a))⁻¹ ) ←→ (∀a∈G. (p(a⁻¹))⁻¹ = p(a))
proof
  assume (∀a∈G. p(a⁻¹) = (p(a))⁻¹ )
  with A1 show (∀a∈G. (p(a⁻¹))⁻¹ = p(a))
    using apply_funtype group_inv_of_inv by simp
next assume A2: (∀a∈G. (p(a⁻¹))⁻¹ = p(a))
  { fix a assume a∈G
    with A1 A2 have p(a⁻¹) ∈ G and ((p(a⁻¹))⁻¹)⁻¹ = (p(a))⁻¹
      using apply_funtype inverse_in_group by auto
    then have p(a⁻¹) = (p(a))⁻¹
      using group_inv_of_inv by simp
    } then show (∀a∈G. p(a⁻¹) = (p(a))⁻¹ ) by simp
qed

30.3 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 operation and how they relate to the concept of subgroups.

The next locale extends the group0 locale with notation related to interval arithmetics.

locale group4 = group0 +
fixes sdot (infixl · 70)
defines sdot_def [simp]: A·B ≡ (P {lifted to subsets of} G)<A,B>
fixes sinv (_⁻¹ [90] 91)
defines sinv_def[simp]: A⁻¹ ≡ 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⊆G B⊆G
  shows A·B = {x·y. ⟨x,y⟩ ∈ A×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 \( x \cdot y \in A \cdot B \)

**proof**
- from groupAssum have GroupInv(G,P):G→G using group0_2_T2 by simp
  with assms show \( A^{-1} = \{ x^{-1}.x \in A \} \) using func_imagedef by simp

**qed**

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 GroupInv(G,P):G→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 \( x \in A \ y \in B \) \( z\cdot x=y \) 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 \( (P \{ \text{lifted to subsets of} \ G \}) \{ \text{is associative on} \} \) 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 \( A \{\text{is closed under}\} P \) 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 \( A \subseteq A \) 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 \( \text{IsAsubgroup}(H,P) \)
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 \( x \)
with assms have \( (x^{-1})^{-1} \in \{y^{-1} \cdot y \in H \} \) using group0_3_T3A by auto
moreover from \( x \in H \) \( H \subseteq G \) have \( (x^{-1})^{-1} = x \) using group_inv_of_inv
by auto
ultimately have \( x \in \{y^{-1} \cdot y \in H \} \) by auto
with \( H \subseteq G \) have \( x \in H^{-1} \) using interval_inv by simp
} thus \( H \subseteq H^{-1} \) by auto
from assms have \( H \{\text{is closed under}\} P \) using group0_3_L6 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 \( x \)
with assms have \( x \in G \) using group0_3_L2 by auto
from assms \( H \subseteq G \) \( x \in H \) have \( x \cdot 1 \in H \cdot H \) using group0_3_L5 interval_prod
by auto
with \( x \in G \) have \( x \in H \cdot H \) using group0_2_L2 by simp
}
In the group4 interval_prod_inv: assumes A⊆G B⊆G

shows
\[(A \cdot B)^{-1} = \{(x \cdot y)^{-1}, (x, y) \in A \cdot B\}\]
\[(A \cdot B)^{-1} = \{y^{-1} \cdot x^{-1}, (x, y) \in A \cdot B\}\]
\[(A \cdot B)^{-1} = (B^{-1}) \cdot (A^{-1})\]

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: \((A \cdot B)^{-1} = \{(x \cdot y)^{-1}, (x, y) \in A \cdot B\}\)
proof
\{ fix \(p\) assume \(p \in (A \cdot B)^{-1}\)
  with I obtain \(z\) where \(p = z^{-1}\) and \(z \in A \cdot B\) by auto
  with assms obtain \(x\) \(y\) where \((x, y) \in A \cdot B\) and \(z = x \cdot y\) using interval_prod
  by auto
  with \(<p = z^{-1}>\) have \(p \in \{(x \cdot y)^{-1}, (x, y) \in A \cdot B\}\) by auto
\} thus \((A \cdot B)^{-1} \subseteq \{(x \cdot y)^{-1}, (x, y) \in A \cdot B\}\) by blast
\{ fix \(p\) assume \(p \in \{(x \cdot y)^{-1}, (x, y) \in A \cdot B\}\)
  then obtain \(x\) \(y\) where \(x \in A\) \(y \in B\) and \(p = (x \cdot y)^{-1}\) by auto
  with assms \(<(A \cdot B) \subseteq G\) have \(p \in (A \cdot B)^{-1}\) using interval_prod interval_inv
  by auto
\} thus \(\{(x \cdot y)^{-1}, (x, y) \in A \cdot B\} \subseteq (A \cdot B)^{-1}\) by blast
qed
have \(\{(x \cdot y)^{-1}, (x, y) \in A \cdot B\} = \{y^{-1} \cdot x^{-1}, (x, y) \in A \cdot B\}\)
proof
\{ fix \(p\) assume \(p \in \{(x \cdot y)^{-1}, (x, y) \in A \cdot B\}\)
  then obtain \(x\) \(y\) where \(x \in A\) \(y \in B\) and \(p = (x \cdot y)^{-1}\) by auto
  with assms have \(y^{-1} \cdot x^{-1} = (x \cdot y)^{-1}\) using group_inv_of_two by auto
  with \(<p = (x \cdot y)^{-1}>\) have \(p = y^{-1} \cdot x^{-1}\) by simp
  with \(<x \in A, y \in B>\) have \(p \in \{(y^{-1} \cdot x^{-1}, (x, y) \in A \cdot B\}\) by auto
\} thus \(\{(x \cdot y)^{-1}, (x, y) \in A \cdot B\} \subseteq \{y^{-1} \cdot x^{-1}, (x, y) \in A \cdot B\}\) by blast
qed
with II show III: \((A \cdot B)^{-1} = \{y^{-1} \cdot x^{-1}, (x, y) \in A \cdot B\}\) by simp
have \(\{y^{-1} \cdot x^{-1}, (x, y) \in A \cdot B\} = (B^{-1}) \cdot (A^{-1})\)
proof
\{ fix \(p\) assume \(p \in \{(y^{-1} \cdot x^{-1}, (x, y) \in A \cdot B\}\)
  then obtain \(x\) \(y\) where \(x \in A\) \(y \in B\) and \(p = y^{-1} \cdot x^{-1}\) by auto
  with assms have \(y^{-1} \in (B^{-1})\) and \(x^{-1} \in (A^{-1})\)
  using interval_inv by auto
  with \(<p = y^{-1} \cdot x^{-1}>\) have \(p \in (B^{-1}) \cdot (A^{-1})\) using interval_inv_cl interval_prod
\}
by auto
} thus \(\{y^{-1} \cdot x^{-1}. (x,y) \in A \times B\} \subseteq (B^{-1}) \cdot (A^{-1})\) by blast

\{ fix p assume \(p \in (A^{-1})\cdot (B^{-1})\)
   then obtain \(y \cdot x\) where \(y \in B^{-1}\) \(x \in A^{-1}\) and \(p = y \cdot x\)
   using interval_inv_cl interval_prod by auto
   with assms obtain \(x_1 \cdot y_1\) where \(x_1 \in A\) \(y_1 \in B\) and \(x = x_1^{-1}\) \(y = y_1^{-1}\)
   using interval_inv
   by auto
   with \(p = y \cdot x\) have \(p \in \{y^{-1} \cdot x^{-1}. (x,y) \in A \times B\}\) by auto
\} thus \((B^{-1}) \cdot (A^{-1}) \subseteq \{y^{-1} \cdot x^{-1}. (x,y) \in A \times B\}\) by blast
qed

with III show \((A^{-1}) \cdot (B^{-1}) = (B^{-1}) \cdot (A^{-1})\) by simp
qed

If \(H, K\) are subgroups then \(H \cdot K\) is a subgroup iff \(H \cdot K = K \cdot H\).

theorem (in group4) prod_subgr_subgr:
  assumes IsAsubgroup(H,P) and IsAsubgroup(K,P)
  shows IsAsubgroup(H \cdot K,P) \iff H \cdot K = K \cdot H
proof
  assume IsAsubgroup(H \cdot K,P)
  then have \((H \cdot K)^{-1} = H \cdot K\) using subgroup_inv_sq(1) by simp
  with assms show \(H \cdot K = K \cdot H\) using group0_3_L2 interval_prod_inv subgroup_inv_sq(1) by auto
next
from assms have \(H \subseteq G\) and \(K \subseteq G\) using group0_3_L2 by auto
  have I: \(H \cdot K \neq 0\)
  proof -
    let \(x = 1\) let \(y = 1\)
    from assms have \(x \cdot y \in (H \cdot K)\) using group0_3_L5 group0_3_L2 interval_prod
    by auto
    thus thesis by auto
  qed
from \(<H \subseteq G> <K \subseteq G>\) have II: \(H \cdot K \subseteq G\) using interval_prod_closed by simp

assume \(H \cdot K = K \cdot H\)
have III: \((H \cdot K)\{\text{is closed under}\} P\)
proof -
  have \((H \cdot K) \cdot (H \cdot K) = H \cdot K\)
  proof -
    from \(<H \subseteq G> <K \subseteq G>\) have \((H \cdot K) \cdot (H \cdot K) = H \cdot (K \cdot H)\)
    using interval_prod_rearr1 by simp
    also from \(<H \cdot K = K \cdot H>\) have \(H \cdot (H \cdot K) = H \cdot (K \cdot K)\) by simp
    also from \(<H \subseteq G> <K \subseteq G>\) have \(H \cdot (H \cdot K) = (H \cdot H) \cdot (K \cdot K)\)
    using interval_prod_rearr1 by simp
    finally show thesis by simp
  qed
  with \(<H \subseteq G>\) show thesis using subset_gr_op_cl by simp

292
have IV: \( \forall x \in H \cdot K. \ x^{-1} \in H \cdot K \)

proof -

\{
  \text{fix } x \text{ assume } x \in H \cdot K
  \text{ with } <H \subseteq G> \text{ have } x^{-1} \in (H \cdot K)^{-1} \text{ using interval_inv by auto}
  \text{ with assms } <H \subseteq G> <K \subseteq G> <H \cdot K = K \cdot H> \text{ have } x^{-1} \in H \cdot K
  \text{ using interval_prod_inv subgroup_inv_sq(1) by simp}
\)
thus thesis by auto
qed

from I II III IV show IsAsubgroup(H \cdot K,P) using group0_3_T3 by simp
qed
end

31 Groups - and alternative definition

theory Group_ZF_1b imports Group_ZF

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 "," 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/ZF.thy. The informal proofs come from an Aug. 14, 2005 post by buli on the matematyka.org forum.

31.1 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 \( \langle a, b \rangle \).

locale group2 =  
fixes P  
fixes dot (infixl \( \cdot \))  
defines dot_def [simp]: \( a \cdot b \equiv P\langle a, b \rangle \)

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: \( G \neq 0 \) and A2: \( P \) {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 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 \( \langle a \in G \rangle \) obtain y where y\( \in G \) and y\( \cdot a = a \)  
by auto  
have I: \( \forall b \in G. b = b \cdot x \wedge b = y \cdot b \)

proof  
fix b assume b\( \in G \)  
with A4 \( \langle a \in G \rangle \) obtain y\( \_ \) where y\( \_ \)\( \in G \)  
and y\( \_ \)\( \cdot a = b \) by auto  
from A3 \( \langle a \in G \rangle \) \( \langle b \in G \rangle \) obtain x\( \_ \) where x\( \_ \)\( \in G \)  
and a\( \cdot x\( \_ \) = b \) by auto  
from \( \langle a \cdot x = a \rangle \) \( \langle y \cdot a = a \rangle \) \( \langle y\_ \cdot a = b \rangle \) \( \langle a \cdot x\_ = b \rangle \)  
have b = y\( \_ \)\( \cdot (a \cdot x\_ \) and b = \( (y \cdot a) \cdot x\_ \)  
by auto  
moreover from A2 \( \langle a \in G \rangle \) \( \langle x \in G \rangle \) \( \langle y \in G \rangle \) \( \langle x\_ \in G \rangle \) \( \langle y\_ \in G \rangle \) have  
\( (y \cdot a) \cdot x\_ = y \cdot (a \cdot x\_ \) y\( \_ \)\( \cdot (a \cdot x\_ \)  
using IsAssociative_def by auto  
moreover from \( \langle y\_ \cdot a = b \rangle \) \( \langle a \cdot x\_ = b \rangle \) have  
\( (y\_ \cdot a) \cdot x = b \cdot x \) \( y \cdot (a \cdot x\_ \) = y \cdot b \)  
by auto  
ultimately show b = b \( \cdot x \) \( \wedge b = y \cdot b \) by simp

qed  
moreover have x = y


proof -
from \(<x \in G> I have x = x \cdot y = y by simp
also from \(<y \in G> I have y \cdot x = y by simp
finally show x = y by simp
qed
ultimately have \(\forall b \in G. b \cdot x = x \land x \cdot b = b \) by simp
with \(A2 \) have IsAmonoid\((G,P)\) using IsAmonoid_def by auto
with \(A3 \) show IsAgroup\((G,P)\)
  using monoid0_def monoid0.unit_is_neutral IsAgroup_def 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 \(P\langle a,b \rangle\) 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
  \(\forall a \in G. \forall b \in G. \exists x \in G. a \cdot x = b \) and
  \(\forall a \in G. \forall b \in G. \exists y \in G. y \cdot a = b \)
proof -
  \{ fix a b assume a \in G b \in G
  let x = a^{-1} \cdot b
  let y = b \cdot a^{-1}
  from \(<a \in G> \) \(<b \in G> \) have
  \(x \in G \land y \in G \) and \(a \cdot x = b \) \(y \cdot a = b \)
  using inverse_in_group group_op_closed inv_cancel_two
  by auto
  hence \(\exists x \in G. a \cdot x = b \) and \(\exists y \in G. y \cdot a = b \) by auto
  \}
thus
  \(\forall a \in G. \forall b \in G. \exists x \in G. a \cdot x = b \) and
  \(\forall a \in G. \forall b \in G. \exists y \in G. y \cdot a = b \)
  by auto
qed

end 

32 Abelian Group

theory AbelianGroup_ZF imports Group_ZF

begin

A group is called “abelian“ if its operation is commutative, i.e. \(P\langle a,b \rangle = P\langle a,b \rangle\) 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.

### 32.1 Rearrangement formulae

This section is not interesting and should not be read. Here we will prove formulas in 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 parantheses 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 parantheses, then rearrange the elements in proper order, then put the parantheses 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 parantheses is concerned Isabelle does its job automatically.

```isar
lemma (in group0) group0_4_L2:
  assumes A1:P {is commutative on} G
  and A2:a\in G b\in G c\in G d\in G E\in G F\in G
  shows (a\cdot b)\cdot (c\cdot d)\cdot (E\cdot F) = (a\cdot (d\cdot F))\cdot (b\cdot (c\cdot E))
proof -
  from A2 have (a\cdot b)\cdot (c\cdot d)\cdot (E\cdot F) = a\cdot b\cdot c\cdot d\cdot E\cdot F
    using group_op_closed group_oper_assoc
    by simp
  also have a\cdot b\cdot c\cdot d\cdot E\cdot F = a\cdot d\cdot F\cdot b\cdot c\cdot E
    proof -
      from A1 A2 have a\cdot b\cdot c\cdot d\cdot E\cdot F = F\cdot (a\cdot b\cdot c\cdot d\cdot E)
        using IsCommutative_def group_op_closed
        by simp
      also from A2 have F\cdot (a\cdot b\cdot c\cdot d\cdot E) = F\cdot a\cdot b\cdot c\cdot d\cdot E
        using group_op_closed group_oper_assoc
        by simp
      also from A1 A2 have F\cdot a\cdot b\cdot c\cdot d\cdot E = d\cdot (F\cdot a\cdot b\cdot c)\cdot E
        using IsCommutative_def group_op_closed
        by simp
      also from A2 have d\cdot (F\cdot a\cdot b\cdot c)\cdot E = d\cdot F\cdot a\cdot b\cdot c\cdot E
    qed
  qed
```

296
using group_op_closed group_oper_assoc by simp
also from A1 A2 have d·F·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
  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)·(d⁻¹·c⁻¹)·(F⁻¹·E⁻¹)
  using group_inv_of_two by simp
also from A1 A2 T1 have
(a·b)·(d⁻¹·c⁻¹)·(F⁻¹·E⁻¹) = (a·(c⁻¹·E⁻¹))·(b·(d⁻¹·F⁻¹))
  using group0_4_L2 by simp
also from A2 A3 have
(a·(c⁻¹·E⁻¹))·(b·(d⁻¹·F⁻¹)) = (a·(E·c))⁻¹·(b·(F·d))⁻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
    (a·b)⁻¹ = a⁻¹·b⁻¹

297
Another useful rearrangement.

proof -
rom $A_2$ have $T_1$: $b^{-1} \in G$ $a^{-1} \in G$ using inverse_in_group by auto
with $A_1$ show $b^{-1}a^{-1} = a^{-1}b^{-1}$ using IsCommutative_def by simp
with $A_2$ show $(a\cdot b)^{-1} = a^{-1}b^{-1}$ using group_inv_of_two by simp
from $A_2$ $T_1$ have $(a\cdot b^{-1})^{-1} = (b^{-1})^{-1}a^{-1}$ using group_inv_of_two by simp
with $A_1$ $A_2$ $T_1$ show $(a\cdot b^{-1})^{-1} = a^{-1}b$
  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 $A_1$: $P$ is commutative on $G$
  and $A_2$: $a \in G$ $b \in G$ $c \in G$
  shows
  $a \cdot b \cdot c = c \cdot a \cdot b$
  $a^{-1} \cdot (b^{-1} \cdot c^{-1})^{-1} = (a \cdot (b \cdot c)^{-1})^{-1}$
  $a \cdot (b \cdot c)^{-1} = a \cdot b^{-1} \cdot c^{-1}$
  $a \cdot (b^{-1} \cdot c)^{-1} = a \cdot b^{-1} \cdot c$
  $a \cdot b^{-1} \cdot c^{-1} = a \cdot c^{-1} \cdot b^{-1}$

proof -
rom $A_1$ $A_2$ have $a \cdot b \cdot c = c \cdot (a \cdot b)$
  using IsCommutative_def group_op_closed
  by simp
with $A_2$ show $a \cdot b \cdot c = c \cdot a \cdot b$
  group_op_closed group_op_assoc
  by simp
from $A_2$ have $T$:
  $b^{-1} \in G$ $c^{-1} \in G$ $b^{-1} \cdot c^{-1} \in G$ $a \cdot b \in G$
  using inverse_in_group group_op_closed
  by auto
with $A_1$ $A_2$ 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 $A_1$ $A_2$ $T$ have $a \cdot (b \cdot c)^{-1} = a \cdot (b^{-1} \cdot c^{-1})$
  using group_inv_of_two IsCommutative_def by simp
with $A_2$ $T$ show $a \cdot (b^{-1} \cdot c)^{-1} = a \cdot b^{-1} \cdot c^{-1}$
  using group_op_assoc by simp
from $A_1$ $A_2$ $T$ have $a \cdot (b^{-1} \cdot c)^{-1} = a \cdot (b^{-1} \cdot (c^{-1})^{-1})$
  using group_inv_of_two IsCommutative_def by simp
with $A_2$ $T$ show $a \cdot (b^{-1} \cdot c)^{-1} = a \cdot b^{-1} \cdot c$
  using group_op_assoc group_inv_of_inv by simp
from $A_1$ $A_2$ $T$ have $a \cdot b^{-1} \cdot c^{-1} = a \cdot (c^{-1} \cdot b^{-1})$
  using group_op_assoc IsCommutative_def by simp
with $A_2$ $T$ show $a \cdot b^{-1} \cdot c^{-1} = a \cdot c^{-1} \cdot b^{-1}$
  using group_op_assoc by simp
qed

Another useful rearrangement.
lemma (in group0) group0_4_L4B:
assumes P {is commutative on} G
and a∈G b∈G c∈G
shows a·b⁻¹·(b·c⁻¹) = a·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∈G b∈G c∈G
shows a·b·c = c·a·b
a·b·c = a·(c·b)
a·b·c = c·(a·b)
a·b·c = c·b·a

proof -
from A1 A2 show I: a·b·c = c·a·b
using group0_4_L4A by simp
also from A1 A2 have c·a·b = a·c·b
using IsCommutative_def by simp
also from A2 have a·c·b = a·(c·b)
using group_oper_assoc by simp
finally show a·b·c = a·(c·b) by simp
from A2 I show a·b·c = c·(a·b)
using group_oper_assoc by simp
also from A1 A2 have c·(a·b) = c·(b·a)
using IsCommutative_def by simp
also from A2 have c·(b·a) = c·b·a
using group_oper_assoc by simp
finally show a·b·c = c·b·a by simp
qed

Some rearrangement with three elements and inverse.

lemma (in group0) group0_4_L4D:
assumes A1: P {is commutative on} G
and A2: a∈G b∈G c∈G
shows a⁻¹·b⁻¹·c = c·a⁻¹·b⁻¹
b⁻¹·a⁻¹·c = c·a⁻¹·b⁻¹
(a⁻¹·b·c)⁻¹ = a·b⁻¹·c⁻¹

proof -
from A2 have T:
a⁻¹ ∈ G b⁻¹ ∈ G c⁻¹∈G
using inverse_in_group by auto
with A1 A2 show
a⁻¹·b⁻¹·c = c·a⁻¹·b⁻¹
b⁻¹·a⁻¹·c = c·a⁻¹·b⁻¹
using group0_4_L4A by auto

299
from A1 A2 T show \((a^{-1} \cdot b \cdot c)^{-1} = a \cdot b^{-1} \cdot c^{-1}\)
using `group_inv_of_three group_inv_of_inv group0_4_L4C`
by `simp`
qed

Another rearrangement lemma with three elements and equation.

lemma (in group0) group0_4_L5: assumes A1: \(P\) `is commutative on\) \(G\)
and A2: \(a \in G\) \(b \in G\) \(c \in G\)
and A3: \(c = a \cdot b^{-1}\)
shows \(a = b \cdot c\)
proof -
from A2 A3 have \(c \cdot (b^{-1})^{-1} = a\)
using `inverse_in_group group0_2_L18` 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: \(a \in G\) \(b \in G\)
shows \(a \cdot b \cdot a^{-1} = b\)
\(a^{-1} \cdot b \cdot a = b\)
\(a^{-1} \cdot (b \cdot a) = b\)
\(a \cdot (b \cdot a^{-1}) = b\)
proof -
from A1 A2 have \(a \cdot b \cdot a^{-1} = a^{-1} \cdot a \cdot b\)
using `inverse_in_group group0_4_L4A` by `blast`
also from A2 have \(\ldots = b\)
using `group0_2_L6 group0_2_L2` by `simp`
finally show \(a \cdot b \cdot a^{-1} = b\) by `simp`
from A1 A2 have \(a^{-1} \cdot b \cdot a = a \cdot a^{-1} \cdot b\)
using `inverse_in_group group0_4_L4A` by `blast`
also from A2 have \(\ldots = b\)
using `group0_2_L6 group0_2_L2` by `simp`
finally show \(a^{-1} \cdot b \cdot a = b\) by `simp`
moreover from A2 have \(a^{-1} \cdot b \cdot a = a^{-1} \cdot (b \cdot a)\)
using `inverse_in_group group_oper_assoc` by `simp`
ultimately show \(a^{-1} \cdot (b \cdot a) = b\) by `simp`
from A1 A2 have \(a \cdot (b \cdot a^{-1}) = b\)
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\in G \ b\in G
  shows a\cdot 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\in G \ b\in G
  shows a\cdot (a\cdot b)\cdot (a^{-1}) = b \ b\cdot (b\cdot a^{-1}) = b
  proof -
    from A2 have a\cdot (a\cdot b)\cdot (a^{-1}) = a\cdot (b\cdot a^{-1})
      using group_inv_of_two by simp
    also from A2 have \ldots = a\cdot b^{-1} a^{-1}
      using inverse_in_group group_oper_assoc by simp
    also from A1 A2 have \ldots = b^{-1}
      using group0_4_L6AA by simp
    finally show a\cdot (a\cdot b)\cdot (a^{-1}) = b^{-1} by simp
    from A1 A2 have a\cdot (b\cdot a^{-1}) = a\cdot (a^{-1}\cdot b)
      using inverse_in_group IsCommutative_def by simp
    also from A2 have \ldots = b
      using inverse_in_group group_oper_assoc group0_2_L6 group0_2_L2
        group_inv_of_inv 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
  shows a\cdot (a\cdot b^{-1})^{-1} = b
  using assms inverse_in_group group0_4_L6AB group_inv_of_inv
    group_oper_assoc 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
  shows 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
        group_oper_assoc by auto
    with A1 A2 show
\[ ab^{-1}c^{-1} = bc \]
\[ a^{-1}bc = bc \]
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.

\textbf{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 shows \[ a \cdot b \cdot c \cdot d \cdot a^{-1} = b \cdot c \cdot d \]
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.

\textbf{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 \[ a \cdot b^{-1} \cdot (a \cdot c^{-1})^{-1} = c \cdot b^{-1} \]
\[ (a \cdot c^{-1})^{-1} \cdot (b \cdot c) = a^{-1} \cdot b \]
\[ a \cdot (b \cdot (c \cdot a^{-1} \cdot b^{-1})) = c \]
\[ a \cdot b \cdot c^{-1} \cdot (c \cdot a^{-1}) = b \]
proof -
from A2 have T:
\[ a^{-1} \in G \quad b^{-1} \in G \quad c^{-1} \in G \]
\[ a \cdot b \in G \quad ab^{-1} \in G \quad c^{-1} \cdot a^{-1} \in G \quad c \cdot a^{-1} \in G \]
using inverse_in_group group_op_closed by auto
with A1 A2 show \( a \cdot b^{-1} \cdot (a \cdot c^{-1})^{-1} = c \cdot b^{-1} \)
using group0_2_L12 group_oper_assoc group0_4_L6B IsCommutative_def by simp
from A2 T have \( (a \cdot c^{-1})^{-1} \cdot (b \cdot c) = c^{-1} \cdot a^{-1} \cdot b \cdot c \)
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 \( (a \cdot c^{-1})^{-1} \cdot (b \cdot c) = a^{-1} \cdot b \)
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_L6B group0_4_L6A
by simp
from T have \( a \cdot b \cdot c^{-1} \cdot (c \cdot a^{-1}) = a \cdot b \cdot (c^{-1} \cdot (c \cdot a^{-1})) \)
using group_oper_assoc by simp
also from A1 A2 T have \( ... = b \)

302
using group_oper_assoc group0_2_L6 group0_2_L2 group0_4_L6A
by simp

finally show \( a \cdot b \cdot c^{-1} \cdot (c \cdot a^{-1}) = b \) 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 \( a \cdot b \cdot (a \cdot c) = b \cdot c \)

proof -
  from A2 have T: \( b^{-1} \in G \) \( c^{-1} \in G \)
  using inverse_in_group by auto
  with A1 A2 have \( a \cdot (b^{-1}) \cdot (a \cdot (c^{-1})^{-1})^{-1} = c^{-1} \cdot (b^{-1})^{-1} \)
  using group0_4_L6D by simp
  with A1 A2 T show \( a \cdot b \cdot (a \cdot c)^{-1} = b \cdot c^{-1} \)
  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 A2 have \( b^{-1} \in G \)
  using inverse_in_group by simp
  with A1 A2 have \( a \cdot b^{-1} \cdot (a \cdot (b^{-1})^{-1})^{-1} = b^{-1} \cdot b^{-1} \)
  using group0_4_L6D by simp
  with A2 show \( a \cdot b^{-1} \cdot (a^{-1})^{-1} = b^{-1} \cdot b^{-1} \)
  using group_inv_of_inv by simp

qed

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 \( a \cdot b \cdot c \cdot d = a \cdot d \cdot b \cdot c \)
  \( a \cdot b \cdot c \cdot d = a \cdot c \cdot b \cdot d \)

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

303
Some rearrangements with four elements and inverse that are applications of \texttt{rearr\_ab\_gr\_4\_elem}

\begin{verbatim}

definition rearr\_ab\_gr\_4\_elemB: 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 \cdot^{-1} = a \cdot d \cdot b \cdot c \cdot d \cdot^{-1} using rearr\_ab\_gr\_4\_elemA by auto

\end{verbatim}

Some rearrangement lemmas with four elements.

\begin{verbatim}

definition group0\_4\_L7: 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 \cdot^{-1} = a \cdot c \cdot (b \cdot d \cdot^{-1}) using rearr\_ab\_gr\_4\_elemA by auto

\end{verbatim}
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_op_closed group_oper_assoc0_4_L4A by simp
  also from A2 T have a·(d⁻¹·b·c) = a·d⁻¹·b·c
    using group_oper_assoc by simp
  finally show a·b·c·d⁻¹ = a·d⁻¹·b·c by simp
  from A2 T have a·d·(b·d·(c·d))⁻¹ = a·d·(d⁻¹·(b·d·c)⁻¹)
    using group_oper_assoc group_inv_of_two by simp
  also from A2 T have ... = a·(b·d·c)⁻¹
    using group_oper_assoc inv_cancel_two by simp
  also from A1 A2 have ... = a·(d·(b·c))⁻¹
    using IsCommutative_def group_oper_assoc by simp
  also from A2 T have ... = a·((b·c)⁻¹·d⁻¹)
    using group_inv_of_two by simp
  also from A2 T have ... = a·(b·c)⁻¹·d⁻¹
    using group_oper_assoc by simp
  finally show a·d·(b·d·(c·d))⁻¹ = a·(b·c)⁻¹·d⁻¹
    by simp
  from A2 have a·(b·c)·d = a·(b·(c·d))
    using group_op_closed group_oper_assoc by simp
  also from A1 A2 have ... = a·(b·(d·c))
    using IsCommutative_def group_op_closed by simp
  also from A2 have ... = a·b·d·c
    using group_oper_assoc by simp
  finally show a·(b·c)·d = a·b·d·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∈G b∈G c∈G d∈G
  shows
    a·(b·c)⁻¹ = (a·d⁻¹·c⁻¹)·(d·b⁻¹)
a·b·(c·d) = c·a·(b·d)
a·b·(c·d) = a·c·(b·d)
a·(b·c)⁻¹·d = a·b·d·c⁻¹
    (a·b)·(c·d)⁻¹·(d·b⁻¹)⁻¹ = a·c⁻¹
proof -
  from A2 have T:
Some other rearrangements with four elements.

lemma (in group0) group0_4_L8A:
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^{-1}) = a \cdot c \cdot (b \cdot d^{-1})^{-1}
a \cdot b \cdot (c \cdot d^{-1}) = a \cdot c \cdot b^{-1} \cdot d^{-1}

proof -
from A2 have
T: a \in G b^{-1} \in G c \in G d^{-1} \in G
using inverse_in_group by auto
with A1 show a \cdot b \cdot (c \cdot d^{-1}) = a \cdot c \cdot (b \cdot d^{-1})^{-1}
by (rule group0_4_L8)
with A2 T show a \cdot b \cdot (c \cdot d^{-1}) = a \cdot c \cdot b^{-1} \cdot d^{-1}
using group_op_closed group_oper_assoc
by simp
qed

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 \cdot (d^{-1})
  shows
d = b \cdot a^{-1} \cdot c^{-1}
d = a^{-1} \cdot b \cdot c^{-1}
b = a \cdot d^{-1} \cdot c^{-1}
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_L16 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 \( adc = b \) by simp 
thus \( b = adc \) by simp 
qed
end

33 Groups 2

theory Group_ZF_2 imports AbelianGroup_ZF func_ZF EquivClass1
begin

This theory continues Group_ZF.thy and considers lifting the group structure to function spaces and projecting the group structure to quotient spaces, in particular the quotient group.

33.1 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 result of the lifted operation is in the function space.

lemma (in monoid0) Group_ZF_2_1_L0A:
assumes 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 monoidAsssum 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 lifted operation is an operation on the function space.

lemma (in monoid0) Group_ZF_2_1_L0:
assumes A1: \( F = f \{\text{lifted to function space over} \} X \)
and A2: \( s : X \rightarrow G \) \( r : X \rightarrow G \)
shows \( F(s,r) : X \rightarrow G \)
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:**

assumes A1: F = f \{lifted to function space over\} X and A2: E = ConstantFunction(X, TheNeutralElement(G, f))

shows E : X\rightarrow G \land (\forall s \in 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 \in X\rightarrow G. F( E, s) = s \land F( s, E) = s
proof
fix s assume A3: s : X \rightarrow G
from monoidAsssum have T2: f : G \times G \rightarrow G
  using IsAmonoid_def IsAssociative_def by simp
from A3 A1 T1 have
  F( E, s) : X \rightarrow G F( s, E) : X \rightarrow G s : X \rightarrow G
  using Group\_ZF\_2\_1\_L0 by auto
moreover from T2 A1 T1 A2 A3 have
  \forall x \in X. (F( E, s))(x) = s(x)
  \forall x \in 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 \land F( s, E) = s
  using fun_extension_iff by auto
qed

**Theorem (in monoid0) Group\_ZF\_2\_1\_T1:**

assumes A1: F = f \{lifted to function space over\} X

shows IsAmonoid(X \rightarrow G, F)

**Proof**

from monoidAsssum A1 have
  F \{is associative on\} (X \rightarrow G)
  using IsAmonoid_def func\_ZF\_2\_L4 group0\_1\_L3B by auto
moreover from A1 have
  \exists E \in X \rightarrow G. \forall s \in X \rightarrow G. F( E, s) = s \land F( s, E) = s
  using Group\_ZF\_2\_1\_L1 by blast
ultimately show thesis using IsAmonoid_def by simp

**Lemma Group\_ZF\_2\_1\_L2:**

assumes A1: IsAmonoid(G, f)

---

309
and A2: \( F = f \) \{lifted to function space over\} \( X \)
and A3: \( E = \text{ConstantFunction}(X, \text{TheNeutralElement}(G,f)) \)
shows \( E = \text{TheNeutralElement}(X \rightarrow G, F) \)

**proof** -
from A1 A2 have
  \( \text{T1:monoid0}(G,f) \) \text{ and } \( \text{T2:monoid0}(X \rightarrow G, F) \)
  using monoid0_def monoid0.Group_ZF_2_1_T1 by auto
from T1 A2 A3 have
  \( E : X \rightarrow G \) \( \wedge \) \( (\forall s \in X \rightarrow G. \ F( E, s) = s \wedge F( s, E) = s) \)
  using monoid0.Group_ZF_2_1_L1 by simp
with T2 show thesis
  using monoid0.group0_1_L4 by auto
qeda

The lifted operation acts on the functions in a natural way defined by the monoid operation.

**lemma** \( \text{in monoid0} \) lifted_val:
assumes \( F = f \) \{lifted to function space over\} \( X \)
and \( s:X \rightarrow G \) \( r:X \rightarrow G \)
and \( x \in X \)
shows \( (F(s,r))(x) = s(x) \oplus r(x) \)
using monoidAsssum assms IsAmonoid_def IsAssociative_def
  \( \text{group0} \_1 \_3 \_B \) 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** \( \text{in group0} \) Group_ZF_2_1_L3:
assumes \( F = P \) \{lifted to function space over\} \( X \)
and \( s:X \rightarrow G \) \( r:X \rightarrow G \)
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** \( \text{in group0} \) 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
qeda

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→G
  and A3: i = GroupInv(G,P) O s
  shows i: X→G and F⟨s,i⟩ = TheNeutralElement(X→G,F)
proof -
  let E = ConstantFunction(X,1)
  have E : X→G
    using group0_2_L2 func1_3_L1 by simp
  moreover from groupAssum A2 A3 A1
    have F⟨s,i⟩: X→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 ∀x∈X. (F⟨s,i⟩)(x) = E(x)
      using group0_2_T2 comp_fun Group_ZF_2_1_L3
      comp_fun_apply apply_funtype group0_2_L6 func1_3_L2
      by simp
  moreover from groupAssum A1 have
    E = TheNeutralElement(X→G,F)
    using IsAgroup_def Group_ZF_2_1_L2 by simp
  ultimately show F⟨s,i⟩ = TheNeutralElement(X→G,F)
    using fun_extension_iff IsAgroup_def Group_ZF_2_1_L2
    by simp
  from groupAssum A2 A3 show i: X→G
    using group0_2_T2 comp_fun by simp
qed

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)
    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

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 \( \forall s \in (X \rightarrow G).\ \text{GroupInv}(X \rightarrow G,F)(s) = \text{GroupInv}(G,P) \circ s \)

proof -
  from A1 have \( \text{group0}(X \rightarrow G,F) \)
    using group0_def Group_ZF_2_1_T2
    by simp

  moreover from A1 have \( \forall s \in X \rightarrow G.\ \text{GroupInv}(G,P) \circ s : X \rightarrow G \land \)
    \( F(s,\text{GroupInv}(G,P) \circ s) = \text{TheNeutralElement}(X \rightarrow G,F) \)
    using Group_ZF_2_1_L5
    by simp

  ultimately have \( \forall s \in (X \rightarrow G).\ \text{GroupInv}(G,P) \circ s = \text{GroupInv}(X \rightarrow G,F)(s) \)
    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 \rightarrow G and x \in X
  shows \( (\text{GroupInv}(X \rightarrow G,F)(s))(x) = (s(x))^{-1} \)

  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 \times H) and
  A4: s \in H
  shows GroupInv(H,g)(s) = GroupInv(G,P) \circ s

proof -
  from A1 have T1: group0(X \rightarrow G,F)
    using group0_def Group_ZF_2_1_T2
    by simp

  with A2 A3 A4 have GroupInv(H,g)(s) = GroupInv(X \rightarrow G,F)(s)
    using group0.group0_3_T1 restrict by simp

  moreover from T1 A1 A2 A4 have
    GroupInv(X \rightarrow G,F)(s) = GroupInv(G,P) \circ s
    using group0.group0_3_L2 Group_ZF_2_1_L6 by blast

  ultimately show thesis by simp

qed

If a group is abelian, then its lift to a function space is also abelian.
lemma (in group0) Group_ZF_2_1_L7:
  assumes A1: F = P \{lifted to function space over\} X
  and A2: P \{is commutative on\} G
  shows F \{is commutative on\} (X\rightarrow G)
proof-
  from A1 A2 have F \{is commutative on\} (X\rightarrow \text{range}(P))
    using group_oper_on func_ZF_2_L2
    by simp
  moreover from groupAssum have range(P) = G
    using group0_2_L1 monoid0.group0_1_L3B
    by simp
  ultimately show thesis by simp
qed

33.2 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) Group_ZF_2_2_L1:
  assumes A1: equiv(G,r) and A2: Congruent2(r,f)
  and A3: F = ProjFun2(G,r,f)
  and A4: e = TheNeutralElement(G,f)
  shows \( r\{e\} \in G//r \wedge \forall c \in G//r. F\langle r\{e\},c \rangle = c \wedge F\langle c,r\{e\} \rangle = c \)
proof
  from A4 show T1: \( r\{e\} \in G//r \)
    using unit_is_neutral quotientI
    by simp
  show \( \forall c \in G//r. F\langle r\{e\},c \rangle = c \wedge F\langle c,r\{e\} \rangle = c \)
    proof
      fix c assume A5: \( c \in G//r \)
      then obtain g where D1: \( d1\in G \wedge c = r\{g\} \)
        using quotient_def by auto
      with A1 A2 A3 A4 D1 show
        F\langle r\{e\},c \rangle = c \wedge F\langle c,r\{e\} \rangle = c
        using unit_is_neutral EquivClass_1_L10
        by simp
    qed
qed

The projected structure is a monoid.

theorem (in monoid0) Group_ZF_2_2_T1:
  assumes A1: equiv(G,r) and A2: Congruent2(r,f)
  and A3: F = ProjFun2(G,r,f)
proof-
  from A1 A2 have
    \( F \{is commutative on\} (X\rightarrow range(P)) \)
    using group_oper_on func_ZF_2_L2
    by simp
  moreover from groupAssum have range(P) = G
    using group0_2_L1 monoid0.group0_1_L3B
    by simp
  ultimately show thesis by simp
q
shows IsAmonoid(G//r,F)

proof -
let E = r{TheNeutralElement(G,f)}
from A1 A2 A3 have
  E ∈ G//r ∧ (∀c∈G//r. F⟨E,c⟩ = c ∧ F⟨c,E⟩ = c)
  using Group_ZF_2_2_L1 by simp
hence
  ∃E∈G//r. ∀c∈G//r. F⟨E,c⟩ = c ∧ F⟨c,E⟩ = c
  by auto
with monoidAsssum 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 Group_ZF_2_2_L1:
assumes A1: IsAmonoid(G,f)
and A2: equiv(G,r)
and A3: Congruent2(r,f)
and A4: F = ProjFun2(G,r,f)
and A5: e = TheNeutralElement(G,f)
shows r{e} = TheNeutralElement(G//r,F)
proof -
from A1 A2 A3 A4 A5 have
  T1:monoid0(G,f) and T2:monoid0(G//r,F)
  using monoid0_def monoid0.Group_ZF_2_2_T1 by auto
from T1 T2 A1 A2 A3 A4 A5 have r{e} ∈ G//r ∧
  (∀c ∈ G//r. F⟨r{e},c⟩ = c ∧ F⟨c,r{e}⟩ = c)
  using T1 by simp
with T2 show thesis using monoid0.group0_1_L4 by auto
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:
assumes A1: equiv(G,r) and A2: Congruent2(r,P)
and A3: F = ProjFun2(G,r,P)
and A4: a ∈ G b ∈ G
shows F⟨r(a),r(b)⟩ = r{a·b}
proof -
from A1 A2 A3 A4 show thesis
  using EquivClass_1_L10 by simp
qed

The class of the inverse is a right inverse of the class.

lemma (in group0) Group_ZF_2_2_L3:
assumes A1: equiv(G,r) and A2: 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 inverse_in_group Group_ZF_2_2_L2 group0_2_L6
by simp
with groupAssum A1 A2 A3 show thesis
using IsAgroup_def Group_ZF_2_2_L1 by simp
qed

The group structure can be projected to the quotient space.

theorem (in group0) Group_ZF_3_T2:
assumes A1: \( \text{equiv}(G, r) \) 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 groupAssum A1 A2 have \( \text{IsAmonoid}(G//r, F) \)
using IsAgroup_def monoid0_def monoid0.Group_ZF_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: \( g \in G \ c = r \{g\} \)
using quotient_def by auto
let \( b = r \{g^{-1}\} \)
from D1 have \( b \in G//r \)
using inverse_in_group quotientI
by simp
moreover from A1 A2 D1 have
\( F( c, b) = E \)
using Group_ZF_2_2_L3 by simp
ultimately show \( \exists b \in G//r. F( c, b) = E \)
by auto
qed
ultimately show thesis
using IsAgroup_def by simp
qed

The group inverse (in the projected group) of a class is the class of the inverse.

lemma (in group0) Group_ZF_2_2_L4:
assumes A1: \( \text{equiv}(G, r) \) and A2: \( \text{Congruent2}(r, P) \) and A3: \( F = \text{ProjFun2}(G, r, P) \) and A4: \( a \in G \)

315
shows $r\{a^{-1}\} = \text{GroupInv}(G//r,F)(r\{a\})$

proof -
from $A1$ $A2$ $A3$ have $\text{group0}(G//r,F)$ using $\text{GroupZF}_3$$_2$T2 $\text{group0}$$_\text{def}$ by simp
moreover from $A4$ have
$r\{a\} \in G//r$ $r\{a^{-1}\} \in G//r$
using $\text{inverse}_\text{in}$$_\text{group}$ $\text{quotientI}$ by auto
moreover from $A1$ $A2$ $A3$ $A4$ have
$F(r\{a\},r\{a^{-1}\}) = \text{TheNeutralElement}(G//r,F)$
using $\text{GroupZF}_2$$_2$$_3$L3 by simp
ultimately show thesis
by (rule $\text{group0}$.$\text{group0}_2$$_L9$)
qed

33.3 Normal subgroups and quotient groups

If $H$ is a subgroup of $G$, then for every $a \in G$ we can cosider the sets
{$a \cdot h.h \in H$} and {$h \cdot a.h \in H$} (called a left and right “coset of $H$”,
resp.) These sets sometimes form a group, called the ”quotient group”.
This section discusses the notion of quotient groups.

A normal subgorup $N$ 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
$\text{QuotientGroupOp}(G,P,H) \equiv \text{ProjFun2}(G,\text{QuotientGroupRel}(G,P,H),P)$

Definition of a normal subgroup in a more readable notation.

lemma (in $\text{group0}$) $\text{GroupZF}_2$$_2$$_4$$_L0$:
assumes $\text{IsAnormalSubgroup}(G,P,H)$
and $g \in G$ $n \in H$
shows $g \cdot n \cdot g^{-1} \in H$
using assms $\text{IsAnormalSubgroup}_\text{def}$ by simp

The quotient group relation is reflexive.
lemma (in group0) Group_ZF_2_4_L1:
assumes IsAsubgroup(H,P)
sshows refl(G,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: IsAsubgroup(H,P)
sshows sym(QuotientGroupRel(G,P,H))
proof -
{fix a b assume A2: ⟨a,b⟩ ∈ QuotientGroupRel(G,P,H)
with A1 have (a⁻¹⁻¹)⁻¹ ∈ H
using QuotientGroupRel_def group0_3_T3A
by simp
moreover from A2 have (a⁻¹⁻¹)⁻¹ = b⁻¹⁻¹
using QuotientGroupRel_def group0_2_L12
by simp
ultimately have b⁻¹⁻¹ ∈ H by simp
with A2 have ⟨b,a⟩ ∈ 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: IsAsubgroup(H,P) and
A2: ⟨a,b⟩ ∈ QuotientGroupRel(G,P,H) and
A3: ⟨b,c⟩ ∈ QuotientGroupRel(G,P,H)
sshows ⟨a,c⟩ ∈ QuotientGroupRel(G,P,H)
proof -
let r = QuotientGroupRel(G,P,H)
from A2 A3 have T1:a∈G b∈G c∈G
using QuotientGroupRel_def by auto
from A1 A2 A3 have (a⁻¹⁻¹)·(b⁻¹⁻¹) ∈ H
using QuotientGroupRel_def group0_3_L6
by simp
moreover from T1 have
a⁻¹⁻¹ = (a⁻¹⁻¹)·(b⁻¹⁻¹)
using group0_2_L14A by blast
ultimately have a⁻¹⁻¹ ∈ 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($G, \text{QuotientGroupRel}(G, P, H))$

**proof** -
- let $r = \text{QuotientGroupRel}(G, P, H)$
- from $A1$
  - have $\forall a \ b \ c. \ ((a, b) \in r \land (b, c) \in r \rightarrow (a, c) \in r)$
    - 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 group0_2_L15

**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 $A2$: $\langle a1, a2 \rangle \in \text{QuotientGroupRel}(G, P, H)$
  - and $A3$: $\langle b1, b2 \rangle \in \text{QuotientGroupRel}(G, P, H)$
- **shows** $\langle a1 \cdot b1, a2 \cdot b2 \rangle \in \text{QuotientGroupRel}(G, P, H)$

**proof** -
- from $A2$ $A3$
  - have $T1$: $a1 \in G$ $a2 \in G$ $b1 \in G$ $b2 \in G$
    - $a1 \cdot b1 \in G$ $a2 \cdot b2 \in G$
    - $b1 \cdot b2^{-1} \in H$ $a1 \cdot a2^{-1} \in H$
    - using QuotientGroupRel_def group0_2_L1 group0_1_L1
    - by auto
  - with $A1$
    - show thesis
      - using IsAnormalSubgroup_def group0_3_L6 group0_2_L15

**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($\text{QuotientGroupRel}(G, P, H), P$)
  - using assms group0_def group0.Group_ZF_2_4_L4 Congruent2_def
  - by simp

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 ∈ G
shows ⟨a,1⟩ ∈ QuotientGroupRel(G,P,H) ←→ a ∈ 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∈G and
A3: r = QuotientGroupRel(G,P,H) and
A4: TheNeutralElement(G//r,QuotientGroupOp(G,P,H)) = e
shows r{a} = e ←→ ⟨a,1⟩ ∈ 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 by (rule 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
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_L5B by simp

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\( \in G \) and \( r = \text{QuotientGroupRel}(G,P,H) \) and TheNeutralElement(G//r,QuotientGroupOp(G,P,H)) = e

shows \( r\{a\} = e \iff a \in H \)

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: \( \text{IsAgroup}(G,P) \) and A2: \( P \) \{is commutative on\} \( G \) and A3: \( \text{IsAsubgroup}(H,P) \)

shows \( \text{IsAnormalSubgroup}(G,P,H) \) QuotientGroupOp(G,P,H) \{is commutative on\} (G//QuotientGroupRel(G,P,H))

**Proof:**

- from A1 A2 A3 show T1: \( \text{IsAnormalSubgroup}(G,P,H) \) 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
  - with A2 show QuotientGroupOp(G,P,H) \{is commutative on\} (G//QuotientGroupRel(G,P,H)) using EquivClass_2_T1 QuotientGroupOp_def by simp
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

33.4 Function spaces as monoids

On every space of functions \{f : X → 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 ∃I∈(X→X). ∀f∈(X→X). F⟨I,f⟩ = f ∧ F⟨f,I⟩ = f
proof-
let I = id(X)
from A1 have
  I ∈ X→X ∧ (∀f∈(X→X). F⟨I,f⟩ = f ∧ 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 composition 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
  I ∈ X→X ∧ (∀f∈(X→X). F⟨I,f⟩ = f ∧ F⟨f,I⟩ = f)
  using id_type func_ZF_6_L1A by simp
ultimately show I = TheNeutralElement(X→X,F)
  using monoid0.group0_1_L4 by auto
In this theory we consider notions in group theory that are useful for the construction of real numbers in the Real_ZF series of theories.

34.1 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 Real_ZF 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) Group_ZF_3_1_L1:
assumes A1: $F = P \{\text{lifted to function space over} \} X$
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 T1: $s:X \rightarrow G \ r:X \rightarrow G$
  using FinRangeFunctions_def by auto
with A1 have T2: $q : X \rightarrow G$
  using group0_2_L1 monoid0.Group_ZF_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 Finite1_L18 by auto
with A1 T1 T2 show thesis using
  group Oper_fun Finite1_L15 Group_ZF_2_1_L3 func_imagedef
  by simp
qed
ultimately show thesis using FinRangeFunctions_def
  by simp
qed

qed
end
The set of group valued finite range functions is closed with respect to the lifted group operation.

**Lemma (in group0) GroupZF_3_1_L2:**

**Assumes**
- $A_1$: $F = P \{\text{lifted to function space over} X \}$

**Shows**
- $\text{FinRangeFunctions}(X,G) \{\text{is closed under} \} F$

**Proof**
- Let $A = \text{FinRangeFunctions}(X,G)$
- From $A_1$ have $\forall x \in A. \forall y \in A. F(x,y) \in A$
- Using GroupZF_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) GroupZF_3_1_L3:**

**Assumes**
- $A_1$: $s \in \text{FinRangeFunctions}(X,G)$

**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 GroupZF_3_1_T1:**

**Assumes**
- $A_1$: $\text{IsAgroup}(G,P)$
- $A_2$: $F = P \{\text{lifted to function space over} X \}$
- $A_3$: $X \neq 0$

**Shows**
- $\text{IsAsubgroup}(\text{FinRangeFunctions}(X,G),F)$

**Proof**
- Let $e = \text{TheNeutralElement}(G,P)$
- Let $S = \text{FinRangeFunctions}(X,G)$
- From $A_1$ have $T_1$: $\text{group0}(G,P)$ using group0_def by simp
- With $A_1$ $A_2$ have $T_2$: $\text{group0}(X \rightarrow G,F)$
- Using group0.GroupZF_2_1_T2 group0_def by simp
- Moreover have $S \neq 0$
- **Proof**
  - From $T_1$ $A_3$ 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 $A_2$ $T_1$ have
  - $S \{\text{is closed under} \} F$
  - Using group0.GroupZF_3_1_L2 by simp
- Moreover from $A_1$ $A_2$ $T_1$ have
∀s ∈ S. GroupInv(X→G,F)(s) ∈ S
using FinRangeFunctions_def group0.Group_ZF_2_1_L6

ultimately show thesis
using group0.group0_3_T3 by simp
qed

34.2 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 Endoxus 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 intergers into themselves. We consider almost homomorphisms because we use slopes to define real numbers in the Real_ZF_x series.

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
HomDiff(G,f,s,x) ≡ f(s(f(fst(x),snd(x))),
<GroupInv(G,f)(f(s(fst(x)),s(snd(x)))))

Almost homomorphisms are defined as those maps s : G → G such that the homomorphism difference takes only finite number of values on G × G.

definition
AlmostHoms(G,f) ≡ {s ∈ G→G. (HomDiff(G,f,s,x). x ∈ G×G ) ∈ Fin(G)}

AlHomOp1(G,f) is the group operation on almost homomorphisms defined in a natural way by (s·r)(n) = s(n)·r(n). In the terminology defined in func1.thy this is the group operation f (on G) lifted to the function space G → G and restricted to the set AlmostHoms(G,f).

definition
AlHomOp1(G,f) ≡ restrict(f {lifted to function space over} G,
AlmostHoms(G,f)×AlmostHoms(G,f))

We also define a composition (binary) operator on almost homomorphisms in a natural way. We call that operator AlHomOp2 - the second operation on almost homomorphisms. Composition of almost homomorphisms is used to define multiplication of real numbers in Real_ZF series.

definition
This lemma provides more readable notation for the HomDiff definition. Not really intended to be used in proofs, but just to see the definition in the notation defined in the group0 locale.

lemma (in group0) HomDiff_notation:
shows HomDiff(G,P,s,⟨m,n⟩) = s(m\cdot n)\cdot (s(m)\cdot s(n))^{-1}
using HomDiff_def by simp

The next lemma shows the set from the definition of almost homomorphism in a different form.

lemma (in group0) Group_ZF_3_2_L1A: shows \{HomDiff(G,P,s,x). x ∈ G×G \} = {s(m\cdot n)\cdot (s(m)\cdot s(n))^{-1}. ⟨m,n⟩ ∈ G×G\}
proof -
  have ∀m∈G.∀n∈G. HomDiff(G,P,s,⟨m,n⟩) = s(m\cdot n)\cdot (s(m)\cdot s(n))^{-1}
    using HomDiff_notation by simp
  then show thesis by (rule ZF1_1_L4A)
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 (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 ∈ G\} being finite. We also add an assumption that the G is abelian as many needed properties do not hold without that.

locale group1 = group0 +
  assumes isAbelian: P {is commutative on} G
  fixes AH
  defines AH_def [simp]: AH ≡ AlmostHoms(G,P)
  fixes Op1
  defines Op1_def [simp]: Op1 ≡ AlHomOp1(G,P)
  fixes Op2
  defines Op2_def [simp]: Op2 ≡ AlHomOp2(G,P)
  fixes FR
  defines FR_def [simp]: FR ≡ FinRangeFunctions(G,G)
  fixes neg (\sim_ [90] 91)
defines neg_def [simp]: \sim s \equiv \text{GroupInv}(G,P) \circ s

fixes \delta
defines \delta_def [simp]: \delta(s,x) \equiv \text{HomDiff}(G,P,s,x)

fixes AHprod (infix \cdot 69)
defines AHprod_def [simp]: s \cdot r \equiv \text{AlHomOp1}(G,P)\langle s,r \rangle

fixes AHcomp (infix \circ 70)
defines AHcomp_def [simp]: s \circ r \equiv \text{AlHomOp2}(G,P)\langle s,r \rangle

fixes AlEq (infix \approx 68)
defines AlEq_def [simp]: s \approx r \equiv \langle s,r \rangle \in \text{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\rightarrow G r:G\rightarrow G
and A2: x \in G\times G
and A3: F = P \{lifted to function space over\} G
shows \delta(F\langle s,r \rangle,x) = \delta(s,x) \cdot \delta(r,x)
proof -
let p = F\langle s,r \rangle
from A2 obtain m n where
D1: x = \langle m,n \rangle m \in G n \in G
by auto
then have T1:m \cdot n \in G
using group0_2_L1 monoid0.group0_1_L1
by simp
with A1 D1 have T2:
s(m)\in G s(n)\in G r(m)\in G
r(n)\in G s(m\cdot n)\in G r(m\cdot n)\in G
using apply_funtype by auto
from A3 A1 have T3:p : G\rightarrow G
using group0_2_L1 monoid0.Group_ZF_2_1_L0
by simp
from D1 T3 have
\delta(p,x) = p(m\cdot n) \cdot ((p(n))^{-1} \cdot (p(m))^{-1})
using HomDiff_notation apply_funtype group_inv_of_two
by simp
also from A3 A1 D1 T1 isAbelian T2 have
... = \delta(s,x) \cdot \delta(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 \( p : G \rightarrow G \)
using 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 AlmostHoms_def by auto
with groupAssum A1 A2 show thesis
using IsAgroup_def IsAmonoid_def IsAssociative_def
Finite1_L15 AlmostHoms_def Group_ZF_3_2_L1
by auto
qed
ultimately show thesis using 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 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 \( s:G \rightarrow G \text{ and } m\in G \text{ n}\in G \)
shows \( m \cdot n \in G \)
\( s(m \cdot n) \in G \)
\( s(m) \in G \text{ s(n) } \in G \)
\( \delta(s,(m,n)) \in G \)
\( s(m) \cdot s(n) \in G \)
using assms group_op_closed inverse_in_group
apply_funtype HomDiff_def by auto

It is handy to have a version of Group_ZF_3_2_L4 specifically for almost homomorphisms.

corollary (in group1) Group_ZF_3_2_L4A:
assumes \( s \in AH \text{ and } m\in G \text{ n}\in G \)
shows \( m \cdot n \in G \)
\( s(m \cdot n) \in G \)
\( s(m) \in G \text{ s(n) } \in G \)
\( \delta(s,(m,n)) \in G \)
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 $A2:x:G \times G$

shows $\text{fst}(x) \cdot \text{snd}(x) \in G$

$s(\text{fst}(x) \cdot \text{snd}(x)) \in G$

$s(\text{fst}(x)) \in G$ $s(\text{snd}(x)) \in G$

$\delta(s,x) \in G$

$s(\text{fst}(x)) \cdot s(\text{snd}(x)) \in G$

**Proof:**

let $m = \text{fst}(x)$

let $n = \text{snd}(x)$

from $A1$ $A2$ show

$m \cdot n \in G$ $s(m \cdot n) \in G$

$s(m) \in G$ $s(n) \in G$

$s(m) \cdot s(n) \in G$

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 thesis

using Group_ZF_3_2_L4B HomDiff_def Group_ZF_3_2_L5 group0_4_L4A

by simp

What are the values of the inverse of an almost homomorphism?

**Lemma (in group1) Group_ZF_3_2_L5:**

assumes $s \in AH$ and $n:G$

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 $A2:x:G \times G$

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

using Group_ZF_3_2_L4B HomDiff_def

Group_ZF_3_2_L5 group0_4_L4A

by simp
The inverse of an almost homomorphism maps the group into itself.

**lemma (in group1) Group_ZF_3_2_L7:**
- assumes \( s \in AH \)
- shows \( \sim s : G \rightarrow G \)
  - 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 \( A1: F = P \{\text{lifted to function space over}\} G \)
  and \( A2: s \in AH \)
- shows \( \text{GroupInv}(G \rightarrow G,F)(s) \in AH \)
  - 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
            - \( \text{GroupInv}(G,P) : G \rightarrow G \)
              - 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
- \( \text{ConstantFunction}(G,1) \in AH \) and \( AH \neq 0 \)
  - proof -
let \( z = \text{ConstantFunction}(G,1) \)

have \( G \times G \neq \emptyset \) using group0_2_L1 monoid0.group0_1_L3A by blast

moreover have \( \forall x \in G \times G. \delta(z,x) = 1 \)

proof

  fix \( x \) assume \( A1: x \in G \times G \)

  then obtain \( m \ n \) where \( x = (m,n) \ m \in G \ n \in G \)

  by auto

  then show \( \delta(z,x) = 1 \)

  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 \( \{\delta(z,x). x \in G \times G\} = \{1\} \) by (rule ZF1_1_L5)

then show \( z \in AH \) using group0_2_L2 Finite1_L16

  func1_3_L1 group0_2_L2 AlmostHoms_def by simp

then show \( AH \neq 0 \) 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 \( A1: \text{IsAgroup}(G,P) \)

  and \( A2: P \text{ is commutative on } G \)

  and \( A3: F = P \text{ (lifted to function space over) } G \)

  shows \( \text{IsAsubgroup}(\text{AlmostHoms}(G,P),F) \)

proof -

  let \( AH = \text{AlmostHoms}(G,P) \)

  from \( A2 \) \( A1 \) have \( T1: \text{group1}(G,P) \)

  using group1_axioms.intro group0_def group1_def by simp

  from \( A1 \) \( A3 \) have \( \text{group0}(G \rightarrow G,F) \)

  using group0_def group0.Group_ZF_2_1_T2 by simp

  moreover from \( T1 \) have \( AH \neq 0 \)

  using group1.Group_ZF_3_2_L9 by simp

  moreover have \( T2: AH \subseteq G \rightarrow G \)

  using AlmostHoms_def by auto

  moreover from \( T1 \) \( A3 \) have

  \( AH \text{ (is closed under) } F \)

  using group1.Group_ZF_3_2_L3 by simp

  moreover from \( T1 \) \( A3 \) have

  \( \forall s \in AH. \text{GroupInv}(G \rightarrow G,F)(s) \in AH \)

  using group1.Group_ZF_3_2_L8 by simp

  ultimately show \( \text{IsAsubgroup}(\text{AlmostHoms}(G,P),F) \)

  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)`
  - `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_L3`
  - `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) 0 s`
  - `GroupInv(AH,Op1)(s) ∈ AH`
  - `GroupInv(G,P) 0 s ∈ AH`
- proof -
let $F = P \{\text{lifted to function space over}\} G$

from groupAssum isAbelian have IsAsubgroup$(AH,F)$
using Group$_{ZF\_3\_2\_L10}$ by simp

with $A1$ show $I: \text{GroupInv}(AH,Op1)(s) = \text{GroupInv}(G,P) \circ s$
using AlHomOp1_def Group$_{ZF\_2\_1\_L6A}$ by simp

from $A1$ show $\text{GroupInv}(AH,Op1)(s) \in AH$
using Group$_{ZF\_3\_2\_L10A}$ group0.inverse_in_group by simp

with $I$ show $\text{GroupInv}(G,P) \circ s \in AH$ 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 $s \in AH$ and $n \in G$

shows $(\text{GroupInv}(AH,Op1)(s))(n) = (s(n))^{-1}$

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 $s \cdot r^{-1}$ is also an almost homomorphism.

**Lemma Group$_{ZF\_3\_2\_L15}$:**

assumes $\text{IsAgroup}(G,f)$
and $f \{\text{is commutative on}\} G$
and $AH = \text{AlmostHoms}(G,f)$ $\text{Op1} = \text{AlHomOp1}(G,f)$
and $s \in AH$ $r \in AH$

shows $\text{Op1}(s,r) \in AH$
$\text{GroupInv}(AH,Op1)(r) \in AH$
$\text{Op1}(s,\text{GroupInv}(AH,Op1)(r)) \in AH$

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 context.
States that the product of almost homomorphisms is an almost homomorphism 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 $s \in AH$ $r \in AH$

shows $s \cdot r \in AH$ $s \cdot (^{-}r) \in AH$

using assms isAbelian group0_def group1_axioms group1_def
  Group$_{ZF\_3\_2\_L15}$ Group$_{ZF\_3\_2\_L13}$ by auto

### 34.3 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)
    using Finite1_L18 Finite1_L6B by auto
  have {s(fst(x)). x ∈ G×G} ∈ Fin(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)·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(fst(x)·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)
    using group0_def group0.group0_2_L1
  show IsAsubgroup(H1,H2)
    using group0_def group0.group0_2_L1
    group1_axioms.intro group1_def
by auto
with \( A_1 \) \( A_2 \) have IsAgroup\((G\rightarrow G,F)\)
   IsASubgroup\((H_1,F)\) IsASubgroup\((H_2,F)\)
   using group0.Group_ZF_2_1_T2 Group_ZF_3_2_L10
   monoid0.group0_1_L3A Group_ZF_3_1_T1
   by auto
then have
   IsASubgroup\((H_1 \cap H_2,\text{restrict}(F,H_1 \times H_1))\)
   using group0_3_L7 by simp
moreover from \( T_1 \) have \( H_1 \cap H_2 = H_2 \)
   using group1.Group_ZF_3_3_L1 by auto
ultimately show IsASubgroup\((H_2,\text{AlHomOp1}(G,f))\)
   using AlHomOp1_def by simp
with \( A_1 \) \( A_2 \) show IsANormalSubgroup\((\text{AlmostHoms}(G,f),\text{AlHomOp1}(G,f),\text{FinRangeFunctions}(G,G))\)
   using Group_ZF_3_2_L11 Group_ZF_2_4_L6
   by simp
qed

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\((AH//\text{QuotientGroupRel}(AH,\text{Op1},\text{FR}),\text{QuotientGroupOp}(AH,\text{Op1},\text{FR}))\)
and
   \( \text{QuotientGroupOp}(AH,\text{Op1},\text{FR}) \{\text{is commutative on}\} \)
   \( \text{(AH//QuotientGroupRel}(AH,\text{Op1},\text{FR})) \)
   using groupAssum isAbelian Group_ZF_3_3_L2 Group_ZF_3_2_L10A
   Group_ZF_2_4_T1 Group_ZF_3_2_L10A 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
   \( \text{QuotientGroupRel}(AH,\text{Op1},\text{FR}) \subseteq AH \times AH \) and
   equiv\((AH,\text{QuotientGroupRel}(AH,\text{Op1},\text{FR}))\)
   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 \( A_1 : s\approx r \)

shows \( r\approx s \)

**Proof:**

let \( R = \text{QuotientGroupRel}(AH,\text{Op1},\text{FR}) \)
from \( A_1 \) have equiv\((AH,R)\) and \( (s,r) \in R \)
   using Group_ZF_3_3_L3 by auto
then have \( (r,s) \in R \) by (rule equiv_is_sym)
then show $r = s$ 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 AH 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 
  $r = QuotientGroupRel(AH,Op1,FR)$ and 
  TheNeutralElement(AH//r,QuotientGroupOp(AH,Op1,FR)) = e 
shows $r\{s\} = e \leftrightarrow s \in FR$ 
  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 
  $r = QuotientGroupRel(AH,Op1,FR)$ and 
  $F = ProjFun2(AH,r,Op1)$ 
shows $r\{\sim s\} = GroupInv(AH//r,F)(r\{s\})$ 
proof - 
  from groupAssum isAbelian assms have 
    $r\{GroupInv(AH, Op1)(s)\} = GroupInv(AH//r,F)(r\ \{s\})$ 
    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

34.4 Compositions of almost homomorphisms

The goal of this section is to establish some facts about composition of almost homomorphisms, 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) Group_ZF_3_4_L1:**
- Assumes \( s \in AH \) and \( m \in G \ n \in G \)
- Shows \( s(mn) = s(m)s(n) \cdot \delta(s, (m,n)) \)
- Using isAbelian assms Group_ZF_3_2_L4A HomDiff_def group0_4_L5 by simp

What is the value of a composition of almost homomorphisms?

**Lemma (in group1) Group_ZF_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) Group_ZF_3_4_L3:**
- Assumes \( A1: s \in AH \ r \in AH \) and \( A2: m \in G \ n \in G \)
- Shows \( \delta(s \circ r, (m,n)) = \delta(s, (r(m), r(n))) \cdot \delta(s, (r(m) \cdot r(n), \delta(r, (m,n)))) \)
- Proof -
  - From \( A1 \) \( A2 \) have \( T1: s(r(m)) \cdot s(r(n)) \in G \)
  - \( \delta(s, (r(m), r(n))) \in G \)
  - \( \delta(s, (r(m) \cdot r(n)), \delta(r, (m,n))) \in G \)
  - Using Group_ZF_3_4_L2 AlmostHoms_def apply_funtype Group_ZF_3_2_L4A group0_2_L1 monoid0.group0_1_L1 by auto
  - From \( A1 \) \( A2 \) have \( \delta(s, (m,n)) = s(r(m) \cdot r(n)) \cdot \delta(r, (m,n)) \cdot (s(r(m))) \cdot (s(r(n)))^{-1} \)
  - Using HomDiff_def group0_2_L1 monoid0.group0_1_L1 Group_ZF_3_4_L2 Group_ZF_3_4_L1 by simp
  - Moreover from \( A1 \) \( A2 \) have \( s(r(m) \cdot r(n)) \cdot \delta(r, (m,n)) = s(r(m)) \cdot s(r(n)) \cdot \delta(s, (r(m) \cdot r(n)), \delta(r, (m,n)))) \)
  - Using Group_ZF_3_2_L4A Group_ZF_3_4_L1 by auto
  - Moreover from \( T1 \) isAbelian have \( s(r(m)) \cdot s(r(n)) \cdot \delta(s, (r(m), r(n))) \cdot (s(r(m))) \cdot (s(r(n)))^{-1} = \delta(s, (r(m) \cdot r(n))) \cdot \delta(s, (r(m), r(n))) \cdot \delta(s, (r(m) \cdot r(n)), \delta(r, (m,n)))) \)
  - 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:

- \( A = \delta(s, \langle r(fst(x)), r(snd(x)) \rangle) \)
- \( B = s(\delta(r,x)) \)
- \( C = \delta(s, \langle \langle r(fst(x)) \rangle \cdot r(snd(x)) \rangle, \delta(r,x)) \)

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

- \( \delta(s, \langle m, n \rangle) = \)
- \( \delta(s, \langle r(m), r(n) \rangle) \cdot s(\delta(r, \langle m, n \rangle)) \)
- \( \delta(s, \langle (r(m) \cdot r(n)), \delta(r, \langle m, n \rangle) \rangle) \)

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 \( \{ \delta(\text{Composition}(G) \langle s, r \rangle, x). x \in G \times G \} \in \text{Fin}(G) \)

**proof** -

from A1 have \( \forall x \in G \times G. \langle r(fst(x)), r(snd(x)) \rangle \in G \times G \)

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 have

- \( \{ \delta(s, \langle r(fst(x)), r(snd(x)) \rangle). x \in G \times G \} \in \text{Fin}(G) \)

by (rule Finite1_L6B)

moreover have \( \{ s(\delta(r,x)). x \in G \times G \} \in \text{Fin}(G) \)

**proof** -

from A1 have \( \forall m \in G. s(m) \in G \)

using AlmostHoms_def apply_funtype by auto

moreover from A1 have \( \{ \delta(r,x). x \in G \times G \} \in \text{Fin}(G) \)

using AlmostHoms_def by simp

ultimately show thesis

by (rule Finite1_L6C)

qed
ultimately have
\{\delta(s, \langle r(fst(x)), r(snd(x)) \rangle) \cdot s(\delta(r,x)) \cdot \delta(s, \langle (r(fst(x)) \cdot r(snd(x))), \delta(r,x) \rangle) \cdot s(\delta(r,x)) \cdot \delta(s, \langle (r(fst(x)) \cdot r(snd(x))), \delta(r,x) \rangle). x \in G \times G \} \in Fin(G)

moreover have
\{\delta(s, \langle r(fst(x)), r(snd(x)) \rangle) \cdot s(\delta(r,x)). x \in G \times G \} \in Fin(G)

proof -
from A1 have \(\forall x \in G \times G. \langle (r(fst(x)) \cdot r(snd(x))), \delta(r,x) \rangle \in G \times G\)
  using Group_ZF_3_2_L4B by simp
moreover from A1 have \(\{\delta(s,x). x \in G \times G \} \in Fin(G)\)
  using AlmostHoms_def by simp
ultimately show thesis by (rule Finite1_L6B)

qed
ultimately have
\{\delta(s, \langle r(fst(x)), r(snd(x)) \rangle) \cdot s(\delta(r,x)). x \in G \times G \} \in Fin(G)

moreover from A1 have \(\{\delta(s \circ r,x). x \in G \times G \} = \{\delta(s, \langle r(fst(x)), r(snd(x)) \rangle) \cdot s(\delta(r,x)). x \in G \times G \} \in Fin(G)\)
  using Group_ZF_3_4_L4 by simp
ultimately have \(\{\delta(s \circ r,x). x \in G \times G \} \in Fin(G)\) 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 s \in AH r \in AH
  shows Composition(G)\langle s, r \rangle \in AH s \circ r \in AH
proof -
from A1 have \(\langle s, r \rangle \in (G \to G) \times (G \to G)\)
  using AlmostHoms_def by simp
then have Composition(G)\langle s, r \rangle : G \to G
  using func_ZF_5_L1 apply_funtype by blast
with A1 show Composition(G)\langle s, r \rangle \in AH
  using Group_ZF_3_4_L5 AlmostHoms_def by simp
with A1 show s \circ r \in 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
  AH {is closed under} Composition(G)
  AlHomOp2(G,P) {is associative on} AH
proof -
show AH \{is closed under\} Composition(G)
using Group_ZF_3_4_T1 IsOpClosed_def by simp
moreover have 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 AlHomOp2(G,P) \{is associative on\} AH
using func_ZF_4_L3 AlHomOp2_def by simp
qed

Type information related to the situation of two almost homomorphisms.

lemma \(\text{in group1}) Group_ZF_3_4_L7:
assumes A1: \(s \in\) AH \(r \in\) AH and A2: \(n \in\) G
shows
\(s(n) \in G\) \((r(n))^{-1} \in G\)
\(s(n) \cdot (r(n))^{-1} \in G\) \(s(r(n)) \in G\)
proof -
from A1 A2 show
\(s(n) \in G\)
\((r(n))^{-1} \in G\)
\(s(r(n)) \in G\)
\(s(n) \cdot (r(n))^{-1} \in G\)
using AlmostHoms_def apply_type
by auto
qed

Type information related to the situation of three almost homomorphisms.

lemma \(\text{in group1}) Group_ZF_3_4_L8:
assumes A1: \(s \in\) AH \(r \in\) AH \(q \in\) AH and A2: \(n \in\) G
shows
\(q(n) \in G\)
\(s(r(n)) \in G\)
\(r(n) \cdot (q(n))^{-1} \in G\)
\(s(r(n)) \cdot (q(n))^{-1} \in G\)
\(\delta(s, (q(n), r(n) \cdot (q(n))^{-1})) \in G\)
proof -
from A1 A2 show
\(q(n) \in G\) \(s(r(n)) \in G\) \(r(n) \cdot (q(n))^{-1} \in G\)
using AlmostHoms_def apply_type
by auto
with A1 A2 show \(s(r(n) \cdot (q(n))^{-1}) \in G\)
\(\delta(s, (q(n), r(n) \cdot (q(n))^{-1})) \in G\)
using AlmostHoms_def apply_type Group_ZF_3_2_L4A
by auto
qed

A formula useful in showing that the composition of almost homomorphisms

339
is congruent with respect to the quotient group relation.

**Lemma (in group1) GroupZF_3_4_L9:**

assumes $A1: s1 \in AH \quad r1 \in AH \quad s2 \in AH \quad r2 \in AH$

and $A2: n \in G$

shows $(s1 \circ r1)(n) \cdot ((s2 \circ r2)(n))^{-1} = s1(r2(n)) \cdot (s2(r2(n)))^{-1} \cdot s1(r1(n) \cdot (r2(n))^{-1}) \cdot \delta(s1, (r2(n), r1(n) \cdot (r2(n))^{-1}))$

**Proof -**

from $A1 \ A2$ isAbelian have

$(s1 \circ r1)(n) \cdot ((s2 \circ r2)(n))^{-1} = s1(r2(n) \cdot (r1(n) \cdot (r2(n))^{-1})) \cdot (s2(r2(n)))^{-1}$

using GroupZF_3_4_L2 GroupZF_3_4_L7 group0_4_L6A group_oper_assoc by simp

with $A1 \ A2$ have $(s1 \circ r1)(n) \cdot (s2 \circ r2)(n))^{-1} = s1(r2(n)) \cdot s1(r1(n) \cdot (r2(n))^{-1}) \cdot \delta(s1, (r2(n), r1(n) \cdot (r2(n))^{-1})) \cdot (s2(r2(n)))^{-1}$

using GroupZF_3_4_L8 GroupZF_3_4_L1 by simp

with $A1 \ A2$ isAbelian show thesis using

GroupZF_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) GroupZF_3_4_L10:**

assumes $A1: s \in AH \quad r \in AH$

and $A2: n \in G$

shows $(s \cdot (\text{GroupInv}(AH, Op1)(r)))(n) = s(n) \cdot (r(n))^{-1}$

**Proof -**

from $A1 \ A2$ show thesis using

isAbelian GroupZF_3_2_L13 GroupZF_3_2_L12 GroupZF_3_2_L14 by simp

**QED**

A necessary condition for two a. h. to be almost equal.

**Lemma (in group1) GroupZF_3_4_L11:**

assumes $A1: s \approx r$

shows $\{s(n) \cdot (r(n))^{-1}. n \in G\} \in \text{Fin}(G)$

**Proof -**

from $A1$ have $s \in AH \quad r \in AH$

using QuotientGroupRel_def by auto

moreover from $A1$ have

$\{(s \cdot \text{GroupInv}(AH, Op1)(r))(n). n \in G\} \in \text{Fin}(G)$

using QuotientGroupRel_def Finite1_L18 by simp

ultimately show thesis using

GroupZF_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: \( s \in \text{AH} \)  \( r \in \text{AH} \) and A2: \( \{s(n) \cdot (r(n))^{-1}. n \in G\} \in \text{Fin}(G) \) shows \( s \approx r \)

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 A1: \( s \in \text{AH} \)  \( r \in \text{AH} \) and A2: \( s \cdot (\text{GroupInv}(\text{AH},\text{Op}_1)(r)) \in \text{FR} \) shows \( s \approx r \) \( r \approx s \)

proof -
  from A1 A2 show thesis
  using QuotientGroupRel_def by simp
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 \( s \approx r \) shows \( s \cdot (\text{GroupInv}(\text{AH},\text{Op}_1)(r)) \in \text{FR} \)

proof -
  have \( \{s \cdot (\text{GroupInv}(\text{AH},\text{Op}_1)(r)) \cdot n \in G\} \in \text{Fin}(G) \)
  proof -
    from A1 have \( \forall n \in G. r(n) \in G \)
      using QuotientGroupRel_def AlmostHoms_def apply_funtype by auto
    moreover from A1 have \( \{s(n) \cdot (s2(n))^{-1}. n \in G\} \in \text{Fin}(G) \)
      using Group_ZF_3_4_L11 by simp
    ultimately show thesis by (rule Finite1_L6B)
  qed
  moreover have \( \{s \cdot (\text{GroupInv}(\text{AH},\text{Op}_1)(r)) \cdot n \in G\} \in \text{Fin}(G) \)
  proof -
    from A1 have \( \forall n \in G. s1(n) \in G \)
      using QuotientGroupRel_def AlmostHoms_def apply_funtype by auto
  qed

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: \( s1 \approx s2 \) \( r1 \approx r2 \) shows \( (s1 \circ r1) \approx (s2 \circ r2) \)

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

341
moreover from A1 have \( \{r_1(n) \cdot (r_2(n))^{-1} \cdot n \in G \} \in \text{Fin}(G) \)
using Group_ZF_3_4_L11 by simp
ultimately show thesis by (rule Finite1_L6C)

qed

ultimately have \( \{s_1(r_2(n)) \cdot (s_2(r_2(n)))^{-1} \cdot s_1(r_1(n) \cdot (r_2(n))^{-1}) \cdot n \in G \} \in \text{Fin}(G) \)
using group_oper_fun Finite1_L15 by simp
moreover have \( \{\delta(s_1, (r_2(n), r_1(n) \cdot (r_2(n))^{-1})) \cdot n \in G \} \in \text{Fin}(G) \)
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 \( \{\delta(s_1, x) \cdot x \in G \times G \} \in \text{Fin}(G) \)
using QuotientGroupRel_def AlmostHoms_def by simp
ultimately show thesis by (rule Finite1_L6B)
qed

ultimately have \( \{s_1(r_2(n)) \cdot (s_2(r_2(n)))^{-1} \cdot s_1(r_1(n) \cdot (r_2(n))^{-1}) \cdot \delta(s_1, (r_2(n), r_1(n) \cdot (r_2(n))^{-1})) \cdot n \in G \} \in \text{Fin}(G) \)
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 Congruent2(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

342
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 monoid0.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 \(\text{IsAmonoid}(\text{AH}, \text{Op2})\)

\(\text{monoid0}(\text{AH}, \text{Op2})\)

\(\text{id}(G) = \text{TheNeutralElement}(\text{AH}, \text{Op2})\)

**Proof** -

let \(i = \text{TheNeutralElement}(G \to G, \text{Composition}(G))\)

have \(\text{IsAmonoid}(G \to G, \text{Composition}(G))\)

\(\text{monoid0}(G \to G, \text{Composition}(G))\)

using monoid0_def Group_ZF_2_5_L2 by auto 
moreover have \(\text{AH} \) \{is closed under\} \(\text{Composition}(G)\)

using Group_ZF_3_4_L6 by simp 
moreover have \(\text{AH} \subseteq G \to G\)

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 \(A1: R = \text{QuotientGroupRel}(\text{AH}, \text{Op1}, \text{FR})\)

shows 
\(\text{IsAmonoid}(\text{AH}//R, \text{ProjFun2}(\text{AH}, R, \text{Op2}))\)

\(R\{\text{id}(G)\} = \text{TheNeutralElement}(\text{AH}//R, \text{ProjFun2}(\text{AH}, R, \text{Op2}))\)

**Proof** -
have group0(AH,Op1) using Group_ZF_3_2_L10A group0_def by simp
with A1 groupAssum isAbelian show
R{id(G)} = TheNeutralElement(AH//R,ProjFun2(AH,R,Op2))
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

34.5 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 \( \forall x \in G. \ r(x) = s(x) \cdot c \)
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 \( \{\delta(r,p). \ p \in G \times G\} \in Fin(G) \)
proof -
from A1 A2 have
  \( \{\delta(s,p). \ p \in G \times G\} \in Fin(G) \)
  c^{-1} \in G
  using AlmostHoms_def inverse_in_group by auto
then have \( \{\delta(s,p) \cdot c^{-1}. \ p \in G \times G\} \in Fin(G) \)
  using group_op_closed Finite1_L16AA by simp
moreover from III have
  \( \{\delta(r,p). \ p \in G \times G\} = \{\delta(s,p) \cdot c^{-1}. \ p \in G \times G\} \)
  by (rule ZF1_1_L4B)
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
\( \forall n \in G. \ s(n) \cdot (r(n))^{-1} = c^{-1} \)
using AlmostHoms_def apply_funtype group0_4_L6AB
by auto
then have \( \{s(n) \cdot (r(n))^{-1}. \ n \in G\} = \{c^{-1}. \ n \in G\} \)
by (rule ZF1_1_L4B)
with A1 A2 IV show \( s \approx r \)
using group0_2_L1 monoid0.group0_1_L3A
inverse_in_group Group_ZF_3_4_L12 by simp
qed
end

35 Direct product

theory DirectProduct_ZF imports func_ZF

begin
This theory considers the direct product of binary operations. Contributed by Seo Sanghyeon.

35.1 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 \((g, h) \times (g', h') = (g \cdot g', h \circ h')\).

definition
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 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 DirectProduct(P,Q,G,H)
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

35.2 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 \times 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 \times H

**proof** -

- have \( \forall x \in G \times H. \forall y \in G \times H. \forall z \in G \times H. R(R(x,y),z) = \)
- \( \langle P(P(fst(x),fst(y)),fst(z)),Q(snd(x),snd(y)),snd(z)) \rangle \)
- using DirectProduct_ZF_1_L2 DirectProduct_ZF_1_L3 by auto

moreover have \( \forall x \in G \times H. \forall y \in G \times H. \forall z \in G \times H. R(x,R(y,z)) = \)

346
\langle P(\text{fst}(x),P(\text{fst}(y),\text{fst}(z))),Q(\text{snd}(x),Q(\text{snd}(y),\text{snd}(z)))\rangle \]

using DirectProduct_ZF_1_L2 DirectProduct_ZF_1_L3 by auto
ultimately have \( \forall x \in G \times H. \forall y \in G \times H. R(R(x,y),z) = R(x,R(y,z)) \)
using assms IsAssociative_def by simp
then show thesis
using DirectProduct_ZF_1_L1 IsAssociative_def by simp
qed

end

36 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.

36.1 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) \equiv
(\text{IsAgroup}(G,P) \land r \subseteq G \times G \land \text{IsPartOrder}(G,r) \land (\forall g \in G. \forall a \in b. \langle a,b \rangle \in r \longrightarrow (P( a,g),P( b,g )) \in r \land ( P( g,a),P( g,b )) \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) \equiv \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
iniquely defines an odd function of the whole group. This function is called an odd extension of \( f \)

**definition**

\[
\text{OddExtension}(G,P,r, f) \equiv (f \cup \{ (a, \text{GroupInv}(G,P)(f(\text{GroupInv}(G,P)(a)))) \} \\
\quad a \in \text{GroupInv}(G,P)(\text{PositiveSet}(G,P,r)) \} \cup \\
\{ (\text{TheNeutralElement}(G,P), \text{TheNeutralElement}(G,P)) \})
\]

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 \( f' \) 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 \((\text{infixl} \cdot 70)\)
**defines** groper_def [simp]: \( a \cdot b \equiv P\langle a,b\rangle \)

**fixes** inv \((^{-1} \text{ [90] 91})\)
**defines** inv_def [simp]: \( x^{-1} \equiv \text{GroupInv}(G,P)(x) \)

**fixes** lesseq \((\text{infix} \leq 68)\)
**defines** lesseq_def [simp]: \( a \leq b \equiv \langle a,b \rangle \in r \)

**fixes** sless \((\text{infix} < 68)\)
**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 \((- _ 72)\)
**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 \((\_ \text{')})
defines oddext_def [simp]: f' ≡ 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 ≠ 0
  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 ∈ G ++ ←→ 1 ≤ 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 ∈ G ++ ←→ (1 ≤ g ∧ g ≠ 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 {is total on} G and A2: a ∈ G-G
  shows a ≤ 1
proof -
  from A2 have a ∈ G 1 ∈ G ¬(1 ≤ 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 ∈ G
  shows g ≤ g
  using ordGroupAssum assms IsAnOrdGroup_def IsPartOrdOrder_def refl_def
  by simp

1 is nonnegative.

lemma (in group3) OrderedGroup_ZF_1_L3A: shows 1 ∈ G ++
  using OrderedGroup_ZF_1_L2 OrderedGroup_ZF_1_L3
  OrderedGroup_ZF_1_L1 group0.group0_2_L2 by simp

349
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 $a \in G \subseteq G$
- 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 $a \in G \subseteq G$
- Using ordGroupAssum assms IsAnOrdGroup_def by auto

It is good to have transitivity handy.

**Lemma (in group3) Group_order_transitive:**
- Assumes $A_1: a \leq b \land b \leq c$ shows $a \leq c$
- Proof:
  - From ordGroupAssum have $\text{trans}(r)$
  - Using IsAnOrdGroup_def IsPartOrder_def by simp
  - Moreover from $A_1$ 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 $A_1: a \leq b \land b \leq a$ shows $a = b$
- Proof:
  - From ordGroupAssum $A_1$ have $\text{antisym}(r)$
  - $(a,b) \in r \land (b,a) \in r$
  - 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 $A_1: a < b \land A_2: b \leq c$
- Shows $a < c$
- Proof:
  - From $A_1$ $A_2$ have $a \leq b \land b \leq c$ by auto
  - Then have $a \leq c$ by (rule Group_order_transitive)
  - Moreover from $A_1$ $A_2$ 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

**Lemma (in group3) group_strict_ord_transit:**
- Assumes $A_1: a \leq b \land A_2: b < c$

350
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

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 \( a < b \) and \( c \in G \)
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 \( r \) {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 \( r \) {is total on} \( G \)
  and \( a \in G^+ \) and \( b \in G-G^+ \)
shows \( b \leq a \)
proof -
  from assms have \( b \leq 1 \) \( 1 \leq a \)
  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
}
OrderedGroup_ZF_1_L4 IsAnOrdGroup_def IsPartOrder_def antisym_def
by auto
} thus thesis by auto
qed

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 $b \notin G - G^+$
    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
  group0.inverse_in_group by auto
  with A1 ordGroupAssum have a$a^{-1} \leq b^{-1}$ using IsAnOrdGroup_def
    by simp
  with T1 ordGroupAssum have $b^{-1} \cdot 1 \leq (b \cdot a^{-1})$
    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_L6 group0.group_oper_assoc
    group0.group0_2_L6 by simp
qed

If an element is smaller than 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 $\neg (a \leq 1 \wedge a \neq 1)$
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 $\neg (a \leq 1 \wedge a \neq 1)$ 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

36.2 Inequalities

This section develops 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 ∈ G and A2: a ≤ b⁻¹
shows b ≤ a⁻¹
proof -
  from A2 have (b⁻¹)⁻¹ ≤ a⁻¹
    using OrderedGroup_ZF_1_L5 by simp
  with A1 show b ≤ 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 ∈ G b ∈ G c ∈ G and A2: a·b ≤ a·c
shows b ≤ c
proof -
  from ordGroupAssum A1 A2 have a⁻¹·(a·b) ≤ a⁻¹·(a·c)
    using OrderedGroup_ZF_1_L1 group0.inverse_in_group IsAnOrdGroup_def by simp
  with A1 show b ≤ 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 ∈ G b ∈ G c ∈ G and a·b ≤ c·b
shows a ≤ c
proof -
  from assms(2,4) have (a·b)·b⁻¹ ≤ (c·b)·b⁻¹
    using OrderedGroup_ZF_1_L1 group0.inverse_in_group ord_transl_inv(1)
  by simp
  with assms(1,2,3) show a ≤ 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 ∈ G b ∈ G c ∈ G and A2: a·b⁻¹ ≤ a·c⁻¹
shows c ≤ b
proof -
  from A1 A2 have (c⁻¹)⁻¹ ≤ (b⁻¹)⁻¹
    using OrderedGroup_ZF_1_L1 group0.inverse_in_group
OrderedGroup_ZF_1_L5AE OrderedGroup_ZF_1_L5 by simp with A1 show c≤b
  using OrderedGroup_ZF_1_L1 group0.group_inv_of_inv by simp
qed

Taking negative on both sides reverses the inequality, another case with an
inverse on one side.

textual content omitted

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≤b·c 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 \( b \leq a \cdot b \) \( b \leq b \cdot a \)

**proof** -

from ordGroupAssum \( A1 \) \( A2 \) have \( 1 \cdot b \leq a \cdot b \) \( b \cdot 1 \leq b \cdot a \)

using IsAnOrdGroup_def by auto

with \( A2 \) show \( b \leq a \cdot b \) \( b \leq b \cdot a \)

using OrderedGroup_ZF_1_L1 group0.group0_2_L2

by auto

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 \( I: b \cdot c \leq c \cdot b \) \( a \cdot b \leq c \)

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 \( T: c \in G \) \( c^{-1} \in G \)

using OrderedGroup_ZF_1_L4 OrderedGroup_ZF_1_L1

group0.inverse_in_group by auto
from ordGroupAssum A1 A2 have \( a \cdot b^{-1} \cdot b \leq c \cdot b \)
using IsAnOrdGroup_def by simp
with A1 show \( a \leq c \cdot b \)
using OrderedGroup_ZF_1_L1 group0.inv_cancel_two by simp
with ordGroupAssum A2 T have \( c^{-1} \cdot a \leq c^{-1} \cdot (c \cdot b) \)
using IsAnOrdGroup_def by simp
with A1 T show \( c^{-1} \cdot a \leq b \)
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 \( a \leq b \) and \( c \leq d \)
shows \( a \cdot d^{-1} \leq b \cdot c^{-1} \)
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 \( a \in G \) \( b \in G \) and \( c \leq a \cdot b^{-1} \)
shows \( c \cdot b \leq a \)
proof -
from ordGroupAssum A1 A2 have \( c \cdot b \leq a \cdot b^{-1} \cdot b \)
using IsAnOrdGroup_def by simp
with A1 show \( c \cdot b \leq a \)
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 \( a \in G \) \( b \in G \) and \( c \leq a^{-1} \cdot b \)
shows \( a \cdot c \leq b \)
proof -
from ordGroupAssum A1 A2 have \( a \cdot c \leq a \cdot (a^{-1} \cdot b) \)
using IsAnOrdGroup_def by simp
with A1 show \( a \cdot c \leq b \)
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 \( A1: a \in G \) \( b \in G \) and \( A2: 1 \leq a \cdot b^{-1} \)

357
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: $r$ {is total on} $G$ and A2: $a \in G^+$
  shows $a \leq 1$ $a^{-1} \in G^+$ $\text{restrict}(\text{GroupInv}(G,P),G^{-} \cup G^+\{a\}) \in G^+$
proof -
from A2 have T1: $a \in G$ $a \notin G^+$ $1 \in G$
  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 $\text{restrict}(\text{GroupInv}(G,P),G^{-} \cup G^+\{a\}) \in G^+$
  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: $r$ {is total on} $G$
  and A2: $\forall a \in G^+. \forall b \in G^+. Q(a,b)$
  and A3: $\forall a \in G. \forall b \in G. Q(a,b) \rightarrow Q(a^{-1},b)$
  and A4: $\forall a \in G. \forall b \in G. Q(a,b) \rightarrow Q(a,b^{-1})$
  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 OrderedGroup_ZF_1_L6 OrderedGroup_ZF_1_L1 group0.inverse_in_group
  by simp
with $A5$ have $Q(a,b)$ using OrderedGroup_ZF_1_L1 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 OrderedGroup_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 OrderedGroup_ZF_1_L6 OrderedGroup_ZF_1_L1 group0.inverse_in_group by simp
with $A5$ have $Q(a,b)$ using OrderedGroup_ZF_1_L1 group0.group_inv_of_inv
  by simp }
  moreover
    assume $b \notin G^+$
    with $A1$ $A2$ $A3$ $A4$ $A5$ $T1$ have $Q((a^{-1})^{-1},(b^{-1})^{-1})$
      using OrderedGroup_ZF_1_L6 OrderedGroup_ZF_1_L1 group0.inverse_in_group by simp
with $A5$ have $Q(a,b)$ using OrderedGroup_ZF_1_L1 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 $1 \leq a \land 1 \leq b \lor a \leq 1 \land b \leq 1$
$1 \leq a \land 1 \leq b \land 1 \leq \ a \cdot b \lor a \leq 1 \land 1 \leq b \land a \cdot b \leq 1$
$1 \leq a \land b \leq 1 \land 1 \leq \ a \cdot b \lor 1 \leq a \land b \leq 1 \land a \cdot b \leq 1$
proof -
  from $A1$ $A2$ have $1 \leq a \lor a \leq 1$
  $1 \leq b \lor b \leq 1$
  $1 \leq a \cdot b \lor a \cdot b \leq 1$
  using OrderedGroup_ZF_1_L1 group0.group_op_closed group0.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
qed
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

361
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 \( a \cdot b \leq c \)
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 \( c \cdot b^{-1} \leq a \cdot b \cdot b^{-1} \)
\( a^{-1} \cdot c \leq a^{-1} \cdot (a \cdot b) \)
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 nonneg-ative.

lemma (in group3) OrderedGroup_ZF_1_L9D: assumes A1: \( a \leq b \)
shows \( 1 \leq b \cdot a^{-1} \)
proof -
from T: \( a \in G \) \( b \in G \) \( a^{-1} \in G \)
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) OrderedGroupZF_1_L9E:**

**Assumptions:**
- \( a \leq b \)
- \( a \neq b \)

**Shows:**
- \( 1 \leq b \cdot a^{-1} \)
  \( 1 \neq b \cdot a^{-1} \)
  \( b \cdot a^{-1} \in G_+ \)

**Proof:**
- From \( a \leq b \) and \( a \neq b \) have \( T: a \in G, b \in G \)
  by auto
- From \( a \leq b \) and \( a \neq b \) show \( 1 \leq b \cdot a^{-1} \)
  using OrderedGroupZF_1_L9D by simp

{ assume \( b \cdot a^{-1} = 1 \)
  with \( T \) have \( a=b \)
  using OrderedGroupZF_1_L1 group0.group0_2_L11A
  by auto
  with \( a \neq b \) have False by simp
}
- Then show \( 1 \neq b \cdot a^{-1} \)
  using OrderedGroupZF_1_L2A
  by simp

**QED**

If the difference is nonnegative, then \( a \leq b \).

**Lemma (in group3) OrderedGroupZF_1_L9F:**

**Assumptions:**
- \( a \in G \)
- \( b \in G \)
- \( 1 \leq b \cdot a^{-1} \)

**Shows:**
- \( a \cdot c \cdot b \leq a \cdot d \cdot b \)
  using ordGroupAssum assms IsAnOrdGroup_def

**QED**

If we increase the middle term in a product, the whole product increases.

**Lemma (in group3) OrderedGroupZF_1_L10:**

**Assumptions:**
- \( a \in G \)
- \( b \in G \)
- \( c \leq d \)

**Shows:**
- \( a \cdot c \cdot b \leq a \cdot d \cdot b \)
  using \( ordGroupAssum \) assms IsAnOrdGroup_def

A product of (strictly) positive elements is not the unit.

**Lemma (in group3) OrderedGroupZF_1_L11:**

**Assumptions:**
- \( 1 \leq a \)
- \( 1 \leq b \)
and A2: 1 ≠ a 1 ≠ b
shows 1 ≠ a·b
proof -
  from A1 have T1: a∈G b∈G
    using OrderedGroup_ZF_1_L4 by auto
  { assume 1 = a·b
    with A1 T1 have a≤1 1≤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 ≠ a·b by auto
qed

A product of nonnegative elements is nonnegative.

lemma (in group3) OrderedGroup_ZF_1_L12:
  assumes A1: 1 ≤ a 1 ≤ b
  shows 1 ≤ a·b
proof -
  from A1 have 1·1 ≤ a·b
    using OrderedGroup_ZF_1_L5B by simp
  then show 1 ≤ a·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 · a⁻¹.

lemma (in group3) OrderedGroup_ZF_1_L12A:
  assumes A1: a ≤ b
  shows 1 ≤ b·a⁻¹
proof -
  from A1 have T: 1 ∈ G a∈G b∈G
    using OrderedGroup_ZF_1_L4 OrderedGroup_ZF_1_L1 group0.group0_2_L2
    by auto
  with A1 have 1·a ≤ b
    using OrderedGroup_ZF_1_L1 group0.group0_2_L2
    by simp
  with T show 1 ≤ b·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∈G b∈G and A2: a·b⁻¹ < c
  shows a < c·b
proof -
  from A1 A2 have a·b⁻¹·b < c·b
    using group_strict_ord_transl_inv by auto
  moreover from A1 have a·b⁻¹·b = a
We can multiply the sides of two inequalities, first of them strict and we get a strict inequality.

**Lemma (in group3) OrderedGroup_ZF_1_L1C:**

Assumes A1: \( a < b \) and A2: \( c \leq d \)

Shows \( a \cdot c < b \cdot d \)

**Proof** -

From A1 A2 have T: \( a \in G \) \( b \in G \) \( c \in G \) \( d \in G \)

Using OrderedGroup_ZF_1_L4 by auto

With ordGroupAssum A2 have a \cdot c \leq a \cdot d

Using IsAnOrdGroup_def by simp

Moreover from A1 T have a \cdot d < b \cdot d

Using group_strict_ord_transl_inv by simp

Ultimately show a \cdot c < b \cdot 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_L1D:**

Assumes A1: \( a \leq b \) and A2: \( c < d \)

Shows \( a \cdot c < b \cdot d \)

**Proof** -

From A1 A2 have T: \( a \in G \) \( b \in G \) \( c \in G \) \( d \in G \)

Using OrderedGroup_ZF_1_L4 by auto

With A2 have a \cdot c < a \cdot d

Using group_strict_ord_transl_inv by simp

Moreover from ordGroupAssum A1 T have a \cdot d \leq b \cdot d

Using IsAnOrdGroup_def by simp

Ultimately show a \cdot c < b \cdot d

By (rule OrderedGroup_ZF_1_L4A)

**QED**

### 36.3 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 $G^+$ {is closed under} P
proof -
{ fix a b assume a$\in G^+$, b$\in G^+$
  then have T1: $1 \leq a \cdot b$ and $1 \neq a \cdot b$
    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 $G^+$ {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: $r$ {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 $a \neq 1 \land a^{-1} \notin G^+$
proof -
{ from A1 show T1: $a \neq 1$ using PositiveSet_def by auto
  assume a$^{-1} \in G^+$
  with A1 have $a \leq 1 \land 1 \leq a$
    using OrderedGroup_ZF_1_L5A 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 \( a \neq 1 \ a \notin G_+ \)

proof -
from \( A2 \) have \( a^{-1} \neq 1 \ \) \( (a^{-1})^{-1} \notin G_+ \)
  using OrderedGroup_ZF_1_L15 by auto
with \( A1 \) show \( a \neq 1 \ a \notin G_+ \)
  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 Exactly_1_of_3_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
with \( A3 \) show \( a^{-1} \in G_+ \)
    using OrderedGroup_ZF_1_L1 group0.group_inv_of_one
    PositiveSet_def
    by simp
ultimately show \( \text{Exactly}_1\text{-of}_3\text{-holds} (1,a \in G,a^{-1} \notin G_+) \)
  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 \( A3: a \neq 1 \ a \notin G_+ \)
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 \( A2: a \in G-G_+ \)
shows \( a \leq 1 \)
proof -
  \{ assume \( a=1 \)
    with \( A2 \) have \( a \leq 1 \) using OrderedGroup_ZF_1_L3 by simp \}
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)
sows 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 A2 have I: 1≤a and II: a≠1
    using PositiveSet_def by simp
  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+ by auto
    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

\[ \exists b \in G^+. \quad a \leq b \]

**proof**

\[
\{ \text{assume } A4: \quad a \in G^+ \text{ then have } a \leq a \\
\quad \text{using PositiveSet_def OrderedGroup_ZF_1_L3} \}
\]

by simp

with A4 have \( \exists b \in G^+. \quad a \leq b \) by auto

**moreover**

\[
\{ \text{assume } a \notin G^+ \}
\]

with A1 A3 have I: \( a \leq 1 \) 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 I \( \leq b \) using PositiveSet_def by simp

with I have \( a \leq b \) by (rule Group_order_transitive)

with II have \( \exists b \in G^+. \quad a \leq b \) 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 \( (-G^+) = \{a^{-1}, \ a \in G^+ \} \)

\( (-G^+) \subseteq G \)

**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 \( (-G^+) = \{a^{-1}, \ a \in G^+ \} \)

\( (-G^+) \subseteq G \)

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 \( a \in (-G^+) \)

**proof**

from A1 have \( a^{-1} \in G \quad a = (a^{-1})^{-1} \) using OrderedGroup_ZF_1_L1

group0.inverse_in_group group0.group_inv_of_inv

by auto

with A2 show \( a \in (-G^+) \) using OrderedGroup_ZF_1_L25

by auto

**qed**

370
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 $A1: r$ {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 $G^+ \cap (-G^+) \neq 0$
    then have $A\neq0$ by simp
    then obtain a where $a \in A$ 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: $a \cdot b^{-1} \in G_{+}$
show $b \leq a$ and (I: $a \neq b$)
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 show $b \leq a$ using OrderedGroup_ZF_1_L5K
by simp
from A1 I show $b \neq a$
using OrderedGroup_ZF_1_L1, group0.group0_2_L6
by auto
qed

A bit stronger than OrderedGroup_ZF_1_L29, adds case when two elements
are equal.

lemma (in group3) OrderedGroup_ZF_1_L30:
assumes a$G$, $b \in G$ and a$\neq b$
some have $a \leq b$
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: $a \in G$, $b \in G$
some have $a \leq b$
using assms OrderedGroup_ZF_1_L3, OrderedGroup_ZF_1_L29
by auto

moreover
\{ assume a$^{-1} = 1$
then have a$^{-1} \cdot b = 1 \cdot b$ by simp
with A2 have a$=$ b $\lor (a \leq b$ $\land$ a$\neq b$) $\lor (b \leq a$ $\land$ b$\neq a$)
using OrderedGroup_ZF_1_L1
by simp
\}
moreover
\{ assume a$^{-1} \in G_{+}$
with A2 have a$=$ b $\lor (a \leq b$ $\land$ a$\neq b$) $\lor (b \leq a$ $\land$ b$\neq a$)
group0.inv_cancel_two group0.group0_2_L2 by auto
\}
moreover
\{ assume a$^{-1} \in G_{+}$
with A2 have a$=$ b $\lor (a \leq b$ $\land$ a$\neq b$) $\lor (b \leq a$ $\land$ b$\neq a$)
using OrderedGroup_ZF_1_L29 by auto }
moreover
{ assume \((a \cdot b)^{-1}\) \in \(G\),
with A2 have \(b \cdot a^{-1} \in G\) using OrderedGroup_ZF_1_L1
  group0.group0_2_L12 by simp
with A2 have \(a=b \lor (a \leq b \land a \neq b) \lor (b \leq a \land b \neq a)\)
  using OrderedGroup_ZF_1_L29 by auto }
ultimately show \(a=b \lor (a \leq b \land a \neq b) \lor (b \leq a \land b \neq a)\)
  by auto
qed

36.4 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 \(a \leq M \cdot L^{-1} \leq a\)

proof -
from A3 have A\#0 using func1_1_L13A by fast
then obtain g where g \in A by auto
with A1 have T1: \(L \in G\) \(M \in G\) \(L^{-1} \in G\)
  using OrderedGroup_ZF_1_L4 OrderedGroup_ZF_1_L1
  group0.inverse_in_group by auto
with A2 have S : \(G \rightarrow G\) using OrderedGroup_ZF_1_L1 group0.group0_5_L1
  by simp
moreover from A1 have T2: \(A \subseteq G\) using OrderedGroup_ZF_1_L4 by auto
ultimately have \(S(A) = \{S(b). b \in A\}\) using func_imagedef
  by simp
with A3 obtain b where T3: \(b \in A \land a = S(b)\) by auto
with A1 ordGroupAssum T1 have \(b \cdot L^{-1} \leq M \cdot L^{-1} \leq b \cdot L^{-1}\)
  using IsAnOrdGroup_def by auto
with T3 A2 T1 T2 show \(a \leq M \cdot L^{-1} \leq a\)
  using OrderedGroup_ZF_1_L1 group0.group0_5_L2 group0.group0_2_L6
  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
  let B = \(0\)
  let g = \(1\)

373
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

\[ \exists B. \exists g \in G^+. \exists T \in G \rightarrow G. \ A = T(B) \wedge B \subseteq \text{Interval}(r,1,g) \]

by auto

moreover

\{ assume \( A3: A \neq 0 \)

with \( A1 \) have \( \exists L. \ \forall x \in A. \ L \leq x \) and \( \exists U. \ \forall x \in A. \ x \leq U \)

using IsBounded_def IsBoundedBelow_def IsBoundedAbove_def

by auto

then obtain \( L, U \) where \( D1: \ \forall x \in A. \ L \leq x \wedge x \leq U \)

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 \( T2: L \leq a \wedge a \leq U \) by auto

then have \( T3: L \in G \wedge L^{-1} \in G \wedge U \in G \)

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 \( T(B) = \{a \cdot L^{-1}. \ a \in A\} \)

using OrderedGroup_ZF_1_L1 group0.group0_5_L6

by simp

moreover from \( T3 \) \( T1 \) have \( \forall a \in A. \ a \cdot L^{-1} = a \cdot (L^{-1} \cdot L) \)

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 \( T(B) = \{a \cdot L. \ a \in A\} \)

using OrderedGroup_ZF_1_L1 group0.group0_2_L6 by simp

moreover from \( T1 \) have \( \forall a \in A. \ a \cdot L = a \)

using OrderedGroup_ZF_1_L1 group0.group0_2_L2 by simp

ultimately show thesis by simp

374
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 $\exists B. \exists g \in G^{+}. \exists T \in G \rightarrow G. \ A = T(B) \land B \subseteq \text{Interval}(r,1,g)$

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: IsBounded(A,r)

shows A $\in \text{Fin}(G)$

proof -

from A2 have $\exists B. \exists g \in G^{+}. \exists T \in G \rightarrow G. \ A = T(B) \land B \subseteq \text{Interval}(r,1,g)$

using OrderedGroup_ZF_2_L2 by simp

then obtain $B \ g \ T$ where D1: $g \in G^{+} \ B \subseteq \text{Interval}(r,1,g)$

and D2: $T : G \rightarrow G \ A = T(B)$ 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 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 A1: r {is total on} G and A2: G ≠ {1}
and A3: ∀a∈G. ∃b∈A. a≤b
shows
∀a∈G. ∃b∈A. a ≠ b ∧ a ≤ b
¬IsBoundedAbove(A,r)
A /∉ Fin(G)
proof -
{ fix a
  from A1 A2 obtain c where c ∈ G+
    using OrderedGroup_ZF_1_L21 by auto
  moreover assume a∈G
  ultimately have
    a·c ∈ G and I: a < a·c
    using OrderedGroup_ZF_1_L22 by auto
  with A3 obtain b where II: b∈A and III: a·c ≤ b
    by auto
  moreover from I III have a<b by (rule OrderedGroup_ZF_1_L4A)
  ultimately have ∃b∈A. a ≠ b ∧ a ≤ b by auto
} thus ∀a∈G. ∃b∈A. a ≠ b ∧ a ≤ b by simp
with ordGroupAssum A1 show
  ¬IsBoundedAbove(A,r)
  A /∉ 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 Fin(A) is the collection of finite subsets of A. In this lemma we show that G /∉ 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 A1: r {is total on} G and A2: G ≠ {1}
  shows
  G+ /∉ Fin(G)
  G /∉ Fin(G)
proof -
  from A1 A2 show I: G+ /∉ Fin(G)
    using OrderedGroup_ZF_1_L23 OrderedGroup_ZF_2_L2A
    by simp
  { assume G ∈ Fin(G)
    moreover have G+ ⊆ G using PositiveSet_def by auto
    ultimately have G+ ∈ Fin(G) using Fin_subset_lemma
    by blast
    with I have False by simp
  } then show G /∉ 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 
\begin{itemize}
  \item A1: \( r \) is total on \( G \) and \( A2: A \subseteq G \) and 
  \item A3: \( \neg \text{HasAmaximum}(r,A) \) and \( A4: x \in A \)
\end{itemize}

shows \( \exists y \in A. \ x < y \)

**Proof:**

from ordGroupAssum assms have
\begin{itemize}
  \item antisym(r)
  \item \( r \) is total on \( G \)
  \item \( A \subseteq G \)
  \item \( \neg \text{HasAmaximum}(r,A) \)
  \item \( x \in A \)
\end{itemize}

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 \( r \) is total on \( G \) shows \( \text{IsBoundedAbove}(G \setminus G^+,r) \)

**Proof:**

from A1 have \( \forall a \in G \setminus G^+. \ a \leq 1 \)

using OrderedGroup_ZF_2_L1_L17 by simp

then show \( \text{IsBoundedAbove}(G \setminus G^+,r) \)

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 \( r \) is total on \( G \) and \( A \subseteq G \) and \( A3: A \cap G^+ \in \text{Fin}(G) \)

shows \( \text{IsBoundedAbove}(A,r) \)

**Proof:**

have \( A \cap (G \setminus G^+) \subseteq G \setminus G^+ \) by auto

with A1 have \( \text{IsBoundedAbove}(A \cap (G \setminus G^+),r) \)

using OrderedGroup_ZF_2_L3 Order_ZF_3_L13 by blast

moreover from A1 have \( \text{IsBoundedAbove}(A \cap G^+,r) \)

using ord_group_fin_bounded IsBounded_def by simp

moreover from A1 ordGroupAssum have
\( r \) is total on \( G \) trans(r) \( r \subseteq G \times G \)

using IsAnOrdGroup_def IsPartOrder_def by auto

ultimately have \( \text{IsBoundedAbove}(A \cap (G \setminus G^+) \cup A \cap G^+,r) \)

using Order_ZF_3_L3 by simp

moreover from A2 have \( A = A \cap (G \setminus G^+) \cup A \cap G^+ \)

...
by auto
ultimately show \(IsBoundedAbove(A,r)\) by simp
qed

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 \(A2: IsBoundedAbove(-A,r)\)
shows \(IsBoundedBelow(A,r)\)

**proof**

{ assume \(A = 0\) then have \(IsBoundedBelow(A,r)\)
using IsBoundedBelow_def by auto }
moreover

{ assume \(A3: A \neq 0\)
from ordGroupAssum have \(I: GroupInv(G,P) : G \rightarrow G\)
using IsAnOrdGroup_def group0_2_T2 by simp
with \(A1\) \(A2\) \(A3\) obtain \(u\) where \(D: \forall a \in (-A). a \leq u\)
using func1_1_L15A IsBoundedAbove_def by auto

{ fix \(b\) assume \(b \in A\)
with \(A1\) \(I\) \(D\)
have \(b^{-1} \leq u\) and \(T: b \in G\)
using func_imagedef by auto

then have \(u^{-1} \leq (b^{-1})^{-1}\) using OrderedGroup_ZF_1_L5
by simp
with \(T\) have \(u^{-1} \leq b\)
using OrderedGroup_ZF_1_L1 group0.group_inv_of_inv
by simp
}

then have \(\forall b \in A. (u^{-1}, b) \in r\) by simp
then have \(IsBoundedBelow(A,r)\)
using Order_ZF_3_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 \(f(Interval(r,a,b)) \neq 0\)
using ordGroupAssum assms OrderedGroup_ZF_1_L4
Order_ZF_2_L6 Order_ZF_2_L2A
IsAnOrdGroup_def IsPartOrder_def func1_1_L15A
by auto

discloses

**37** More on ordered groups

theory OrderedGroup_ZF_1 imports OrderedGroup_ZF

begin

In this theory we continue the OrderedGroup_ZF theory development.

378
37.1 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:**

shows $\text{AbsoluteValue}(G,P,r) : G \rightarrow G$

**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 $g : G-G^+ \rightarrow G$

proof -

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:**

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 notation that
would be $|0| = 0$.

**Lemma (in group3) OrderedGroup_ZF_3_L2A:**

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:**


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 \( \text{domain}(\text{id}(G^+)) = G^+ \)
  - using \( \text{id_type} \) \( \text{func1_1_L1} \) by auto
  - with \( A1 \) show \( \text{thesis} \) using \( \text{fun_disjoint_apply2} \) \( \text{AbsoluteValue_def} \)
  - 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:**

- \( \{ \text{assume } a=1 \text{ then have } |a| = a^{-1} \} \)
  - using \( \text{OrderedGroupZF_3_L2A} \) \( \text{OrderedGroupZF_1_L1} \) \( \text{group0.group_inv_of_one} \)
    - by simp
- moreover
  - \( \{ \text{assume } a\neq 1 \} \)
    - with \( A1 \) have \( |a| = a^{-1} \) using \( \text{OrderedGroupZF_1_L4C} \) \( \text{OrderedGroupZF_3_L3} \)
      - 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 \( A1: r \text{ (is total on) } G \) \( \text{and } A2: a \in G \)

shows \( |a| \in G^+ \)

**Proof:**

- \( \{ \text{assume } a \in G^+ \text{ then have } |a| \in G^+ \} \)
  - using \( \text{OrderedGroupZF_3_L2} \) by simp
- moreover
  - \( \{ \text{assume } a \notin G^+ \} \)
    - with \( A1 \) \( A2 \) have \( |a| \in G^+ \) using \( \text{OrderedGroupZF_3_L3} \)
      - \( \text{OrderedGroupZF_1_L6} \) 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 \( \text{AbsoluteValue}(G,P,r) : G \rightarrow G^+ \)
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)
qed

If the absolute value is the unit, then the elemnent 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 A2 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
qed

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
qed

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 -
{ assume \( a \in G^+ \)
  with \( A2 \) have \( a \leq |a| \)
  using OrderedGroup_ZF_3_L2 OrderedGroup_ZF_1_L3 by simp }
moreover
{ assume \( a \notin G^+ \)
  with \( A1 \) \( A2 \) have \( a \leq |a| \)
  using 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 \(-a \leq |a|\).)

lemma (in group3) OrderedGroup_ZF_3_L6:
  assumes \( A1: a \in G \) shows \( a^{-1} \leq |a| \)
proof -
{ assume \( a \in G^+ \)
  then have \( T1: 1 \leq a \) and \( T2: |a| = a \) using OrderedGroup_ZF_1_L2
  OrderedGroup_ZF_3_L2 by auto
  then have \( a^{-1} \leq 1^{-1} \) using OrderedGroup_ZF_1_L5 by simp
  then have \( T3: a^{-1} \leq 1 \)
  using OrderedGroup_ZF_1_L1 group0.group_inv_of_one by simp
  from \( T3 \) \( T1 \) have \( a^{-1} \leq a \) by (rule Group_order_transitive)
  with \( T2 \) have \( a^{-1} \leq |a| \) by simp }
moreover
{ assume \( A2: a \notin G^+ \)
  from \( A1 \) have \( |a| \in G \)
  using OrderedGroup_ZF_3_L1 apply_funtype by auto
  with ordGroupAssum have \( |a| \leq |a| \)
  using IsAnOrdGroup_def IsPartOrder_def refl_def by simp
  with \( A1 \) \( A2 \) have \( a^{-1} \leq |a| \) using 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.

lemma (in group3) OrderedGroup_ZF_3_L6A:
  assumes \( r \{\text{is total on}\} G \) and \( a\in G \) \( b\in G \)
shows
  \( a\cdot b \leq |a|\cdot |b| \)
  \( a^{-1}\cdot b \leq |a|\cdot |b| \)
  \( a^{-1}\cdot b^{-1} \leq |a|\cdot |b| \)
  using assms OrderedGroup_ZF_3_L5 OrderedGroup_ZF_3_L6
  OrderedGroup_ZF_1_L5B by auto
  \( |a^{-1}| \leq |a| \).

lemma (in group3) OrderedGroup_ZF_3_L7:
  assumes \( r \{\text{is total on}\} G \) and \( a\in G \)
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 \cdot b| = |b \cdot 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| \cdot |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| \cdot |b| a^{-1} \cdot b^{-1} \leq |a| \cdot |b|
using OrderedGroup_ZF_1_L5B by auto
with A1 A3 show |a \cdot b| \leq |a| \cdot |b|
using OrderedGroup_ZF_1_L1 group0.group_inv_of_two IsCommutative_def
OrderedGroup_ZF_3_L4 by simp

383
We can multiply the sides of an inequality with absolute value.

**lemma (in group3) OrderedGroup_ZF_3_L7C:**

assumes A1: P {is commutative on} G
and A2: r {is total on} G and A3: a∈G b∈G
and A4: |a| ≤ c |b| ≤ d

shows |a·b| ≤ c·d

**proof -**

from A1 A2 A3 A4 have |a·b| ≤ |a|·|b|
  using OrderedGroup_ZF_1_L4 OrdGroup_triangle_ineq
  by simp
moreover from A4 have |a|·|b| ≤ c·d
  using OrderedGroup_ZF_1_L5B by simp
ultimately show thesis by (rule Group_order_transitive)

qed

A version of the 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 and a∈G b∈G
and |a| ≤ c |b| ≤ d

shows |a·b| ≤ c·d

using assms OrderedGroup_ZF_1_L1 group0.inverse_in_group
OrderedGroup_ZF_3_L7A 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∈G b∈G c∈G

shows |a·b·c| ≤ |a|·|b|·|c|

**proof -**

from A3 have T: a·b ∈ G |c| ∈ G
  using OrderedGroup_ZF_1_L1 group0.group_op_closed
  OrderedGroup_ZF_3_L1 apply_funtype by auto
with A1 A2 A3 have |a·b·c| ≤ |a·b|·|c|
  using OrdGroup_triangle_ineq by simp
moreover from ordGroupAssum A1 A2 A3 T have
  |a·b|·|c| ≤ |a|·|b|·|c|
  using OrdGroup_triangle_ineq IsAnOrdGroup_def by simp
ultimately show |a·b·c| ≤ |a|·|b|·|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∈G b∈G

384
and A4: \(|ab^{-1}| \leq c\)
shows
|a| \leq c|b|
|a| \leq |b|·c
c^{-1}·a \leq b
a·c^{-1} \leq b
a \leq b·c

proof -
from A3 A4 have
T: \(ab^{-1} \in G\) |b| \in G c\in G c^{-1} \in 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| \leq |a·b^{-1}|·|b|
using OrdGroup_triangle_ineq by simp
with T A4 show |a| \leq c·|b| using OrderedGroup_ZF_1_L5C
by blast
with T A1 show |a| \leq |b|·c
using IsCommutative_def by simp
from A2 T have a·b^{-1} \leq |a·b^{-1}|
using OrderedGroup_ZF_1_L5 by simp
moreover note A4
ultimately have I: a·b^{-1} \leq c
by (rule Group_order_transitive)
with A3 show c^{-1}·a \leq b
using OrderedGroup_ZF_1_L5H by simp
with A1 A3 T show a·c^{-1} \leq b
using IsCommutative_def by simp
from A1 A3 T I show a \leq 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: \(\text{P is commutative on}\ G\)
and A2: \(r \text{ is total on}\ G\) and A3: \(a\in G\ b\in G\)
and A4: \(|ab^{-1}| \leq c\)
shows \(b·c^{-1} \leq a\)

proof -
from A3 have \((a·b^{-1})^{-1} \in G\)
using OrderedGroup_ZF_1_L1
\(\text{group0.inverse_in_group}\ group0\text{-group_op_closed}\)
by auto
with A2 have ||(a·b^{-1})^{-1}| = |a·b^{-1}|
using `OrderedGroup_ZF_3_L7A` by simp
moreover from A3 have \((a^{-1}b)^{-1} = b^{-1}a^{-1}\)
  using `OrderedGroup_ZF_1_L1`, `group0.group0_2_L12` by simp
ultimately have \(|b^{-1}a| = |a^{-1}b|\)
  by simp
with A1 A2 A3 A4 show \(b^{-1}c \leq a\)
  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: \(P\) is commutative on \(G\)
  and A2: \(r\) is total on \(G\) and
  A3: \(a \in G\) \(b \in G\) \(c \in G\) \(d \in G\)
  shows \(|a^{-1}c| \leq |a^{-1}b| \cdot |c^{-1}d| \cdot |b^{-1}d|\)
proof -
  from A3 have T:
    \(a^{-1}c \in G\) \(a^{-1}b \in G\) \(c^{-1}d \in G\) \(b^{-1}d^{-1} \in G\)
    using `OrderedGroup_ZF_1_L1`, `group0.inverse_in_group`, `group0.group_op_closed` by auto
  with A1 A2 have \(|(a^{-1}b) \cdot (c^{-1}d)^{-1} \cdot (b^{-1}d)^{-1}| \leq |a^{-1}b| \cdot |(c^{-1}d)^{-1}| \cdot |(b^{-1}d)^{-1}|\)
    using `OrdGroup_triangle_ineq3` by simp
moreover from A2 T have \(|(c^{-1}d)^{-1}| = |c^{-1}d|\) and \(|(b^{-1}d)^{-1}| = |b^{-1}d|\)
  using `OrderedGroup_ZF_3_L7A` by auto
moreover from A1 A3 have \((a^{-1}b) \cdot (c^{-1}d)^{-1} \cdot (b^{-1}d)^{-1} = a^{-1}c\)
  using `OrderedGroup_ZF_1_L1`, `group0.group0_4_L8` by simp
ultimately show \(|a^{-1}c| \leq |a^{-1}b| \cdot |c^{-1}d| \cdot |b^{-1}d|\)
  by simp
qed

\(|a| \leq L\) 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: \(|a| \leq L\)
  shows \(L^{-1} \leq a\)
proof -
  from A1 have I: \(a^{-1} \leq |a|\) using `OrderedGroup_ZF_3_L6` by simp
from I A2 have \(a^{-1} \leq L\) (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 \(|a| \leq L\) 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 \( a \leq L \)
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 \( \leq |a| \) \( |a| \leq L \)
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 \( c^{-1} \cdot L^{-1} \leq 1 \cdot a \)
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_1_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 \( 1 \leq c \)
with T A2 have \( 1 \leq c \)

using OrderedGroup_ZF_1_L1
group0.group0_2_L6 group0.inv_cancel_two
by simp
ultimately have a·b⁻¹ ≤ c
  by (rule Group_order_transitive)
with T show (b·a⁻¹)⁻¹ ≤ c
  using OrderedGroup_ZF_1_L1 group0.group0_2_L12
by simp
qed
ultimately show |b·a⁻¹| ≤ c
  using OrderedGroup_ZF_3_L4 by simp
qed

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 {is total on} G
  and A2: A ⊆ G and A3: ∀a∈A. |a| ≤ L
  shows IsBounded(A,r)
proof -
  from A1 A2 A3 have
    ∀a∈A. a≤L \forall a∈A. L⁻¹≤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 {is total on} G
  and A2: ∀x∈X. b(x)∈G \land |b(x)|≤L
  shows IsBounded({b(x). x∈X},r)
proof -
  from A2 have {b(x). x∈X} ⊆ G ∀a∈{b(x). x∈X}. |a| ≤ 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 {is total on} G
  and A2: f:X→G and A3: A⊆X
  and A4: ∀x∈A. |f(x)| ≤ L
  shows 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_func_type by auto
with A1 have IsBounded({f(x). x \in 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: \( l \leq a \leq u \)
shows \( |a| \leq \text{GreaterOf}(r, |l|, |u|) \)
proof -
from A2 have T1: \( |l| \in G \land |a| \in G \land |u| \in G \)
using OrderedGroup_ZF_1_L4 OrderedGroup_ZF_3_L1 apply_func_type by auto
{ assume A3: \( a \in G^+ \)
with A2 have \( 1 \leq a \leq u \)
    using OrderedGroup_ZF_1_L2 by auto
then have \( 1 \leq u \) by (rule Group_order_transitive)
with A2 A3 have \( |a| \leq |u| \)
    using OrderedGroup_ZF_1_L2 OrderedGroup_ZF_3_L2 by simp
moreover from A1 T1 have \( |u| \leq \text{GreaterOf}(r, |l|, |u|) \)
    using Order_ZF_3_L2 by simp
ultimately have \( |a| \leq \text{GreaterOf}(r, |l|, |u|) \)
    by (rule Group_order_transitive) }
moreover
{ assume A4: \( a \in G^+ \)
with A2 have T2:
    \( 1 \leq |l| \in G \land |a| \in G \land |u| \in G \land a \in G^+ \)
using OrderedGroup_ZF_1_L4 OrderedGroup_ZF_3_L1 apply_func_type by auto
with A2 have \( 1 \in G^+ \) using OrderedGroup_ZF_1_L4D by fast
with T2 A2 have \( |a| \leq |l| \)
    using OrderedGroup_ZF_3_L3 OrderedGroup_ZF_1_L5 by simp
moreover from A1 T2 have \( |l| \leq \text{GreaterOf}(r, |l|, |u|) \)
    using Order_ZF_3_L2 by simp
ultimately have \( |a| \leq \text{GreaterOf}(r, |l|, |u|) \)
    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 $A_1$: $r$ {is total on} $G$
and $A_2$: $\text{IsBounded}(A,r)$
sows $\exists L. \forall a \in A. |a| \leq L$

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 $\text{IsBounded_def} \text{IsBoundedAbove_def} \text{IsBoundedBelow_def}$
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 $\text{OrderedGroup_ZF_3_L10}$ 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) $\text{OrderedGroup_ZF}_3.\:\text{L11}$:
assumes $r$ {is total on} $G$
and $\text{IsBounded}({\{b(x). x \in X}\},r)$
sows $\exists L. \forall x \in X. |b(x)| \leq L$
using $\text{assms} \text{OrderedGroup_ZF}_3.\:\text{L11A}$ 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) $\text{OrderedGroup_ZF}_3.\:\text{L11A}$:
assumes $A_1$: $r$ {is total on} $G$
and $A_2$: $X \neq 0$ and $A_3$: $\{b(x). x \in X\} \in \text{Fin}(G)$
sows $\exists L \in G. \forall x \in X. |b(x)| \leq L$
proof -

from $A_1 \ A_3$ have $\exists L. \forall x \in X. |b(x)| \leq L$
using $\text{ord_group_fin_bounded} \text{OrderedGroup_ZF}_3.\:\text{L11}$ by simp
then obtain $L$ where $I$: $\forall x \in X. |b(x)| \leq L$
using $\text{OrderedGroup_ZF}_3.\:\text{L11}$ by auto
from $A_2$ obtain $x$ where $x \in X$ by auto
with $I$ show thesis using $\text{OrderedGroup_ZF}_1.\:\text{L4}$ by blast

qed

In totally ordered groups the absolute value of a nonunit element is in $G_+$.  

lemma (in group3) $\text{OrderedGroup_ZF}_3.\:\text{L12}$:
assumes $A_1$: $r$ {is total on} $G$
and $A_2$: $a \in G$ and $A_3$: $a \neq 1$
sows $|a| \in G_+$
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

37.2 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)|)
sows 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
sows |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 \( A \subseteq G \)
- HasAmaximum\( (r,A) \) HasAminimum\( (r,A) \)
- Shows \( b \in \text{AbsoluteValue}(G,P,r)(A) \)

**Proof:**

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 \( A_1: r \) is total on \( G \) and \( A_2: A \subseteq G \)
- \( A_3: \) HasAmaximum\( (r,A) \) HasAminimum\( (r,A) \)

**Proof:**

Let \( M = \text{GreaterOf}(r,|\text{Minimum}(r,A)|,|\text{Maximum}(r,A)|) \)

From \( A_2 A_3 \) have \( M \in \text{AbsoluteValue}(G,P,r)(A) \)

Using OrderedGroup_ZF_4_L1 by simp

Moreover from \( A_1 A_2 A_3 \) have

\[ \forall b \in \text{AbsoluteValue}(G,P,r)(A). \quad b \leq M \]

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 \( A_1: r \) is total on \( G \) and \( A_2: A \subseteq G \)
- \( A_3: \) HasAmaximum\( (r,A) \) HasAminimum\( (r,A) \)
- \( A_4: a \in A \)

**Proof:**

From \( A_2 A_4 \) have \( |a| \in \text{AbsoluteValue}(G,P,r)(A) \)

Using OrderedGroup_ZF_3_L1 func_imagedef by auto

With ordGroupAssum \( A_1 A_2 A_3 \) show thesis using

IsAnOrdGroup_def IsPartOrder_def OrderedGroup_ZF_4_L4

Order_ZF_4_L3 by simp

**QED**

### 37.3 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:**

\[
\text{assumes } A1: r = \{p \in G \times G. \text{fst}(p) = \text{snd}(p) \lor \text{fst}(p)^{-1} \cdot \text{snd}(p) \in H\}
\text{shows } (a,b) \in r \iff a \in G \land b \in G \land a^{-1}b \in H \cup \{1\}
\]

**proof**

- Assume \((a,b) \in r\)
  - With \(A1\) show \(a \in G \land b \in G \land a^{-1}b \in H \cup \{1\}\) using group0_2_L6 by auto
- Next assume \(a \in G \land b \in G \land a^{-1}b \in H \cup \{1\}\)
  - Then have \(a \in G \land b \in G \land b = (a^{-1})^{-1} \lor a \in G \land b \in G \land a^{-1}b \in H\)
    - Using inverse_in_group group0_2_L9 by auto
  - With \(A1\) show \((a,b) \in r\) using group_inv_of_inv by auto

**qed**

The relation defined by a positive set is antisymmetric.

**lemma (in group0) OrderedGroup_ZF_5_L2:**

\[
\text{assumes } A1: r = \{p \in G \times G. \text{fst}(p) = \text{snd}(p) \lor \text{fst}(p)^{-1} \cdot \text{snd}(p) \in H\}
\text{and } A2: H \subseteq G \text{ H \{is closed under\} P}
\text{shows } \text{antisym}(r)
\]

**proof**

- Fix \(a\) \(b\) assume \(A3: (a,b) \in r \land (b,a) \in r\)
  - With \(A1\) have \(T: a \in G \land b \in G\) by auto
    - Assume \(A4: a \neq b\)
      - With \(A1\) \(A3\) have \(a^{-1}b \in G \land a^{-1}b \in H \land (a^{-1}b)^{-1} \in H\)
        - Using inverse_in_group group0_2_L9 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 \(\text{antisym}(r)\) by (rule antisymI)

**qed**

The relation defined by a positive set is transitive.

**lemma (in group0) OrderedGroup_ZF_5_L3:**

\[
\text{assumes } A1: r = \{p \in G \times G. \text{fst}(p) = \text{snd}(p) \lor \text{fst}(p)^{-1} \cdot \text{snd}(p) \in H\}
\text{and } A2: H \subseteq G \text{ H \{is closed under\} P}
\text{shows } \text{trans}(r)
\]

**proof**

- Fix \(a\) \(b\) \(c\) assume \((a,b) \in r \land (b,c) \in r\)
with A1 have 
  \(a \in G \land b \in G \land a^{-1}b \in H \cup \{1\}\) 
  \(b \in G \land c \in G \land b^{-1}c \in H \cup \{1\}\) 
  using OrderedGroup_ZF_5_L1 by auto 

with A2 have 
  I: \(a \in G \land b \in G \land c \in G\) 
  and \((a^{-1} \cdot b \cdot (b^{-1} \cdot c)) \in H \cup \{1\}\) 
  using inverse_in_group group0_2_L17 IsOpClosed_def by auto

moreover from I have \(a^{-1} \cdot c = (a^{-1} \cdot b \cdot (b^{-1} \cdot c))\) 
by (rule group0_2_L14A)

ultimately have \(\langle a, c \rangle \in G \times G\) 
by blast

then have \(\forall a \ b \ c. \ (a, b) \in r \land (b, c) \in r \rightarrow (a, c) \in r\) 
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 group0) OrderedGroup_ZF_5_L4:**

assumes \(A1: r = \{p \in G \times G. \; \text{fst}(p) = \text{snd}(p) \lor \text{fst}(p)^{-1} \cdot \text{snd}(p) \in H\}\)
and \(A2: P \; \text{is commutative on} \; G\)
and \(A3: \langle a, b \rangle \in r\) and \(A4: c \in G\)
shows \(\langle a \cdot c, b \cdot c \rangle \in r\)

proof

from A1 A3 A4 have 
  I: \(a \in G \land b \in G \land a \cdot c \in G \land b \cdot c \in G\) 
  and II: \(a^{-1} \cdot b \in H \cup \{1\}\) 
  using OrderedGroup_ZF_5_L1 group_op_closed by auto

with A2 A4 have \((a \cdot c)^{-1} \cdot (b \cdot c) \in H \cup \{1\}\) 
by group0_4_L6D by simp

with A1 I show \((a \cdot c, b \cdot c) \in r\) using OrderedGroup_ZF_5_L1 
by auto

with A2 A4 I show \((c \cdot a, c \cdot b) \in r\)

using IsCommutative_def by simp

qed

If \(H \subseteq G\) 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 relation \(\leq\) defined by \(a \leq b \iff a^{-1}b \in H\) 
orders the group \(G\). In such order \(H\) may be the set of positive 
or nonnegative elements.

**Lemma (in group0) OrderedGroup_ZF_5_L5:**

assumes \(A1: P \; \text{is commutative on} \; G\)
and \(A2: H \subseteq G \land H \; \text{is closed under} \; P\)
and \(A3: \forall a \in G. \; a \neq 1 \rightarrow (a \in H) \lor (a^{-1} \in H)\)
and A4: \( r = \{ p \in G \times G. \text{fst}(p) = \text{snd}(p) \lor \text{fst}(p)^{-1} \cdot \text{snd}(p) \in H \} \)

shows

\( \text{IsAnOrdGroup}(G,P,r) \)

\( r \) (is total on) \( G \)

\( \text{Nonnegative}(G,P,r) = \text{PositiveSet}(G,P,r) \cup \{1\} \)

**proof**

- from `groupAssum` A2 A3 A4 have

  \( \text{IsAgroup}(G,P) \subseteq G \times G \) \( \text{IsPartOrder}(G,r) \)

  \( \text{IsPartOrder_def} \)

  by auto

moreover from A1 A4 have

\( \forall g \in G. \forall a \ b. (a,b) \in r \longrightarrow (a \cdot g, b \cdot g) \in r \)

using `OrderedGroup_ZF_5_L4` by blast

ultimately show \( \text{IsAnOrdGroup}(G,P,r) \)

using `IsAnOrdGroup_def` by simp

then show \( \text{Nonnegative}(G,P,r) = \text{PositiveSet}(G,P,r) \cup \{1\} \)

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 (\( a,b \) / \( r \))

    with A4 T have \( \text{I}: a \neq b \) and \( \text{II}: a^{-1} \cdot b \notin H \)

    by auto

    from A3 T T1 I have \( (a^{-1} \cdot b \in H) \) Xor \( ((a^{-1} \cdot b)^{-1} \in H) \)

    using `group0_2_L11` by simp

  }

  then have \( (a,b) \in r \lor (b,a) \in r \) 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 \( r = \{ p \in G \times G. \text{fst}(p) = \text{snd}(p) \lor \text{fst}(p)^{-1} \cdot \text{snd}(p) \in H \} \)

shows \( \text{PositiveSet}(G,P,r) = H \)

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**

\( \text{OrderFromPosSet}(G,P,H) \equiv \)

\( \{ p \in G \times G. \text{fst}(p) = \text{snd}(p) \lor P(\text{GroupInv}(G,P)(\text{fst}(p)), \text{snd}(p)) \in H \} \)
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).

\[\text{theorem (in group0) Group_ord_by_positive_set}:\]
\[\text{assumes } P \text{ is commutative on } G \]
\[\text{and } H \subseteq G \text{ is closed under } P \]
\[\text{and } \forall a \in G. \ a \neq 1 \rightarrow (a \in H) \text{ xor } (a^{-1} \in H) \]
\[\text{shows}\]
\[\text{IsAnOrdGroup}(G,P,\text{OrderFromPosSet}(G,P,H))\]
\[\text{OrderFromPosSet}(G,P,H) \text{ is total on } G\]
\[\text{PositiveSet}(G,P,\text{OrderFromPosSet}(G,P,H)) = H\]
\[\text{Nonnegative}(G,P,\text{OrderFromPosSet}(G,P,H)) = H \cup \{1\}\]
\[\text{using asms OrderFromPosSet_def OrderedGroup_ZF_5_L5 OrderedGroup_ZF_5_L6 by auto}\]

### 37.4 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.

\[\text{lemma (in group3) OrderedGroup_ZF_6_L1}:\]
\[\text{shows } f^o = f \cup \{(a, (f(a^{-1}))^{-1}) \mid a \in G_+ \} \cup \{(1,1)\}\]
\[\text{using OddExtension_def by simp}\]

A technical lemma that states that from a function defined on \( G_+ \) with values in \( G \) we have \( (f(a^{-1}))^{-1} \in G \).

\[\text{lemma (in group3) OrderedGroup_ZF_6_L2}:\]
\[\text{assumes } f : G_+ \rightarrow G \text{ and } a \in G_+\]
\[\text{shows}\]
\[f(a^{-1}) \in G\]
\[(f(a^{-1}))^{-1} \in G\]
\[\text{using asms OrderedGroup_ZF_1_L27 apply_funtype OrderedGroup_ZF_1_L1 group0.inverse_in_group by auto}\]

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: \( r \) \{is total on\} \( G \) and A2: \( f: G_+ \rightarrow G \)
shows
\( f': G \rightarrow G \)
\( \forall a \in G_+. \quad (f')(a) = f(a) \)
\( \forall a \in (-G_+). \quad (f')(a) = (f(a^{-1}))^{-1} \)
\( (f')(1) = 1 \)
proof -
from A1 A2 have I:
\( f: G_+ \rightarrow G \)
\( \forall a \in -G_+. \quad (f(a^{-1}))^{-1} \in G \)
\( G_+ \cap (-G_+) = 0 \)
\( 1 \notin G_+ \cup (-G_+) \)
\( f' = f \cup \{(a, (f(a^{-1}))^{-1}). \ a \in -G_+ \} \cup \{(1,1)\} \)
using OrderedGroup_ZF_6_L2 OrdGroup_decomp2 OrderedGroup_ZF_6_L1
by auto
then have f': \( G_+ \cup (-G_+) \cup \{1\} \rightarrow G \cup G \cup \{1\} \)
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 \( f': G \rightarrow G \) by simp
from 1 show \( \forall a \in G_+. \quad (f')(a) = f(a) \)
by (rule func1_1_L11E)
from 1 show \( \forall a \in (-G_+). \quad (f')(a) = (f(a^{-1}))^{-1} \)
by (rule func1_1_L11E)
from 1 show \( (f')(1) = 1 \)
by (rule func1_1_L11E)
qed

Odd extensions are odd, of course.

lemma (in group3) oddext_is_odd:
assumes A1: \( r \) \{is total on\} \( G \) and A2: \( f: G_+ \rightarrow G \)
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\( ^{-1} \in -G_+ \) and \( (f')(a) = f(a) \)
using OrderedGroup_ZF_1_L25 odd_ext_props by auto
with A1 A2 have
\( (f')(a^{-1}) = (f((a^{-1})^{-1}))^{-1} \) and \( (f(a))^{-1} = ((f')(a))^{-1} \)
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

397
by simp }
moreover
{ assume A4: a ∈ -G+
  with A1 A2 have a⁻¹ ∈ G+ and (f')(a) = (f(a⁻¹))⁻¹
  using OrderedGroup_ZF_1_L27 odd_ext_props
  by auto
  with A1 A2 A4 have (f')(a⁻¹) = ((f')(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')(a⁻¹) = ((f')(a))⁻¹
  using OrderedGroup_ZF_1_L1 group0.group0_6_L2
  by simp
}
ultimately show (f')(a⁻¹) = ((f')(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+→G
  and A3: a∈G
  shows ((f')(a⁻¹))⁻¹ = (f')(a)
proof -
  from A1 A2 have
  f': G → G
  ∀a∈G. (f')(a⁻¹) = ((f')(a))⁻¹
  using odd_ext_props oddext_is_odd by auto
  then have ∀a∈G. ((f')(a⁻¹))⁻¹ = (f')(a)
  using OrderedGroup_ZF_1_L1 group0.group0_6_L2 by simp
  with A3 show ((f')(a⁻¹))⁻¹ = (f')(a) by simp
qed

37.5 Functions with infinite limits

In this section we consider functions f : G → G with the property that for f(x) is arbitrarily large for large enough x. More precisely, for every a ∈ G there exist b ∈ G+ such that for every x ≥ b we have f(x) ≥ a. In a sense this means that lim_{x→∞} f(x) = ∞, hence the title of this section. We also prove dual statements for functions such that lim_{x→−∞} f(x) = −∞.

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 ≠ {1} and
  A3: f:G→G and
∀a∈G.∃b∈G+.∀x.b≤x→a≤f(x) and
A5: A⊆G and
A6: IsBoundedAbove(f(A),r)
shows IsBoundedAbove(A,r)

proof -
{ assume ¬IsBoundedAbove(A,r)
  then have I: ∀u.∃x∈A. ¬(x≤u)
    using IsBoundedAbove_def by auto
  have ∀a∈G.∃y∈f(A). a≤y
    proof -
    { fix a assume a∈G
    with A4 obtain b where
    II: b∈G+ and III: ∀x.b≤x→a≤f(x)
    by auto
    from I obtain x where IV: x∈A and ¬(x≤b)
    by auto
    with A1 A5 II have
    r {is total on} G
    x∈G b∈G ¬(x≤b)
    using PositiveSet_def by auto
    with III have a ≤ f(x)
    using OrderedGroup_ZF_1_L8 by blast
    with A3 A5 IV have ∃y∈f(A). a≤y
    using func_imagedef by auto
    } thus thesis by simp
  qed
  with A1 A2 A6 have False using OrderedGroup_ZF_2_L2A
  by simp
} thus thesis by auto
qed

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 {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→a≤f(y) and
A6: ∀x∈X. b(x) ∈ G ∧ f(b(x)) ≤ U
shows ∃u.∀x∈X. b(x) ≤ u

proof -
let A = {b(x). x∈X}
from A6 have I: A⊆G by auto
moreover note assms
moreover have IsBoundedAbove(f(A),r)
proof -
from A4 A6 I have ∀z∈f(A). ⟨z,U⟩ ∈ r
  using func_imagedef by simp
then show IsBoundedAbove(f(A),r)
  by (rule Order_ZF_3_L10)
ultimately have IsBoundedAbove(A,r) using OrderedGroup_ZF_7_L1 by simp
with A3 have \( \exists u. \forall y \in A. \ y \leq u \)
using IsBoundedAbove_def by simp
then show \( \exists u. \forall x \in X. \ b(x) \leq u \) 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 \neq \{1\} and
  A3: X \neq 0 and A4: f:G→G and
  A5: \forall a \in G. \exists b \in G. \forall y. b \leq y \rightarrow f(y^{-1}) \leq a and
  A6: \forall x \in X. b(x) \in G \land L \leq f(b(x))
  shows \( \exists l. \forall x \in X. l \leq b(x) \)
proof -
  let \( g = \text{GroupInv}(G,P) \circ f \circ \text{GroupInv}(G,P) \)
from ordGroupAssum have I: GroupInv(G,P) : G→G using IsAnOrdGroup_def group0_2_T2 by simp
with A4 have II: \forall x \in G. g(x) = (f(x^{-1}))^{-1}
  using func1_1_L18 by simp
note A1 A2 A3
moreover from A4 I have g : G→G using comp_fun by blast
moreover have \( \forall a \in G. \exists b \in G. \forall y. b \leq y \rightarrow a \leq g(y) \)
proof -
  { fix a assume A7: a\in G
    then have a^{-1} \in G
    using OrderedGroup_ZF_7_L1 group0.inverse_in_group by simp
    with A5 obtain b where
      III: b \in G. \forall y. b \leq y \rightarrow f(y^{-1}) \leq a^{-1}
      by auto
    with II A7 have \( \forall y. \ b \leq y \rightarrow a \leq g(y) \)
      using OrderedGroup_ZF_1_L5AD OrderedGroup_ZF_1_L4 by simp
    with III have \( \exists b \in G. \forall y. b \leq y \rightarrow a \leq g(y) \)
      by auto
  } then show \( \forall a \in G. \exists b \in G. \forall y. b \leq y \rightarrow a \leq g(y) \)
    by simp
qed
moreover have \( \forall x \in X. \ b(x)^{-1} \in G \land g(b(x)^{-1}) \leq L^{-1} \)
proof-
  { fix x assume x\in X
    with A6 have
      T: b(x) \in G \ b(x)^{-1} \in G and L \leq 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 A6 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
38 Rings - introduction

theory Ring_ZF imports AbelianGroup_ZF

begin

This theory file covers basic facts about rings.

38.1 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: \(\text{IsAring}(R, A, M)\)

fixes ringa (infixl + 90)
defines ringa_def [simp]: \(a + b \equiv A(a, b)\)

fixes ringminus (- _ 89)
defines ringminus_def [simp]: \((-a) \equiv \text{GroupInv}(R, A)(a)\)

fixes ringsub (infixl - 90)
defines ringsub_def [simp]: \(a - b \equiv a + (-b)\)
fixes ringm (infixl · 95)
defines ringm_def [simp]: a·b ≡ M⟨a,b⟩

fixes ringzero (0)
defines ringzero_def [simp]: 0 ≡ TheNeutralElement(R,A)

fixes ringone (1)
defines ringone_def [simp]: 1 ≡ TheNeutralElement(R,M)

fixes ringtwo (2)
defines ringtwo_def [simp]: 2 ≡ 1+1

fixes ringsq (_· 97)
defines ringsq_def [simp]: a² ≡ a·a

In the ring0 context we can use theorems proven in some other contexts.

lemma (in ring0) Ring_ZF_1_L1: shows
  monoid0(R,M)
group0(R,A)
  A {is commutative on} R
using
ringAssum IsAring_def group0_def monoid0_def by auto

The additive operation in a ring is distributive with respect to the multiplicative operation.

lemma (in ring0) ring_oper_distr: assumes A1: a∈R b∈R c∈R
shows
  a·(b+c) = a·b + a·c
  (b+c)·a = b·a + c·a
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.

lemma (in ring0) Ring_ZF_1_L2: shows
  0∈R 1∈R (-0) = 0
using
Ring_ZF_1_L1 group0.group0_2_L2 monoid0.unit_is_neutral
  group0.group_inv_of_one by auto

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∈R
shows
  (-a) ∈ R
  (-(-a)) = a
  a+0 = a
  0+a = a
  a·1 = a
  1·a = a
  a·a = 0
\( a - 0 = a \)
\( 2 \cdot a = a + a \)
\((-a) + a = 0 \)

Using axioms: `Ring_ZF_1_L1` `group0.inverse_in_group` `group0.group_inv_of_inv` `group0.group0_2_L6` `group0.group0_2_L2` `monoid0.unit_is_neutral` `Ring_ZF_1_L2` `ring_oper_distr`

By `auto`

Properties that require two elements of a ring.

Lemma (in `ring0`) `Ring_ZF_1_L4` assumes `A1: a \in \text{R} \ b \in \text{R}`

- \( a + b \in \text{R} \)
- \( a - b \in \text{R} \)
- \( a \cdot b \in \text{R} \)
- \( a + b = b + a \)

Using `ringAssum` axioms: `Ring_ZF_1_L1` `Ring_ZF_1_L3`

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 \text{R} \ b \in \text{R} \ a + b = a` shows `b = 0`

Using axioms: `Ring_ZF_1_L1` `group0.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 \text{R} \ \text{shows} \ 0 \cdot x = 0 \ \ x \cdot 0 = 0`

Proof:
- Let `a = x \cdot 1`
- Let `b = x \cdot 0`
- Let `c = 1 \cdot x`
- Let `d = 0 \cdot x`

From `A1` have
- \( a + b = x \cdot (1 + 0) \)
- \( c + d = (1 + 0) \cdot x \)

Using `Ring_ZF_1_L2` `ring_oper_distr` by `auto`

Moreover have \( x \cdot (1 + 0) = a \cdot (1 + 0) \cdot x = c \)

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
- \( a \in \text{R} \ b \in \text{R} \ \text{and} \ T2: c \in \text{R} \ d \in \text{R} \)
- 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 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∈R  b∈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 ∈ R (-a) ∈ R ((-a)\cdot b) ∈ R
    (-b) ∈ R a\cdot (-b) ∈ 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 Ring_ZF_1_L1 group0.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 Ring_ZF_1_L1 group0.group0_2_L9 by simp
moreover from I II show a\cdot (-b) = -(a\cdot b)
  using Ring_ZF_1_L1 group0.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∈R  b∈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∈R b∈R c∈R shows a·(b-c) = a·b - a·c (b-c)·a = b·a - c·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∈R b∈R shows (-b)-a = (-a)-b (-(a+b)) = (-a)-b (-(a-b)) = ((-a)+b) a-(-b) = a+b using assms ringAssum IsAring_def Ring_ZF_1_L1 group0.group0_4_L4 group0.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∈R b∈R and A2: a-b = 0 shows a=b proof - from A1 A2 have group0(R,A) a∈R b∈R A⟨a,GroupInv(R,A)(b)⟩ = TheNeutralElement(R,A) using Ring_ZF_1_L1 by auto then show a=b by (rule group0.group0_2_L11A) qed

Other basic properties involving three elements of a ring.

lemma (in ring0) Ring_ZF_1_L10: assumes a∈R b∈R c∈R shows a+(b+c) = a+b+c a-(b+c) = a-b-c a-(b-c) = a-b+c using assms ringAssum Ring_ZF_1_L1 group0.group_oper_assoc IsAring_def group0.group0_4_L4A by auto

Another property with three elements.

lemma (in ring0) Ring_ZF_1_L10A: assumes A1: a∈R b∈R c∈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: \[
\text{assumes } a \in R \text{ b} \in R \text{ c} \in R \text{ shows } \\
\text{a} + \text{b} + \text{c} = \text{a} + (\text{b} + \text{c}) \\
\text{a} \cdot \text{b} \cdot \text{c} = \text{a} \cdot (\text{b} \cdot \text{c})
\]
using assms ringAssum Ring_ZF_1_L1 group0.group_oper_assoc 
  IsAring_def IsAmonoid_def IsAssociative_def by auto

An interpretation of what it means that a ring has no zero divisors.

lemma (in ring0) Ring_ZF_1_L12: \[
\text{assumes } \text{HasNoZeroDivs}(R, A, M) \text{ and } \\
a \in R \text{ a} \neq 0 \text{ b} \in R \text{ b} \neq 0 \text{ shows } a \cdot b \neq 0
\]
using assms HasNoZeroDivs_def by auto

In rings with no zero divisors we can cancel nonzero factors.

lemma (in ring0) Ring_ZF_1_L12A: \[
\text{assumes } A1: \text{HasNoZeroDivs}(R, A, M) \text{ and } \\
a \in R \text{ b} \in R \text{ c} \in R \text{ and } \\
a \cdot c = b \cdot c \text{ and } c \neq 0 \text{ shows } a = b
\]
proof - 
  \begin{proof}
    \from A2 have T: a \cdot c \in R \text{ 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 \text{ b} \in R \text{ a-b} = 0 \\
    \by auto \\
    then show a=b by (rule Ring_ZF_1_L9A)
  \end{proof}
qed

In rings with no zero divisors if two elements are different, then after multiplying by a nonzero element they are still different.

lemma (in ring0) Ring_ZF_1_L12B: \[
\text{assumes } A1: \text{HasNoZeroDivs}(R, A, M) \text{ and } \\
a \in R \text{ b} \in R \text{ c} \in R \text{ a} \neq b \text{ c} \neq 0 \text{ shows } a \cdot c \neq b \cdot c
\]
using A1 Ring_ZF_1_L12A by auto

In rings with no zero divisors multiplying a nonzero element by a nonone element changes the value.

lemma (in ring0) Ring_ZF_1_L12C: \[
\text{assumes } A1: \text{HasNoZeroDivs}(R, A, M) \text{ and } \\
A2: a \in R \text{ b} \in R \text{ and } A3: 0 \neq a \text{ 1} \neq b \text{ shows } a \neq a \cdot b
\]
proof - 
  \begin{proof}
    \{ assume a = a \cdot b
  \end{proof}
with A1 A2 have a = 0 ∨ b-1 = 0
  using Ring_ZF_1_L3 Ring_ZF_1_L2 Ring_ZF_1_L8
Ring_ZF_1_L3 Ring_ZF_1_L2 Ring_ZF_1_L4 HasNoZeroDivs_def
  by simp
with A2 A3 have False
  using Ring_ZF_1_L2 Ring_ZF_1_L9A by auto
} then show a ≠ a·b by auto
qed

If a square is nonzero, then the element is nonzero.

lemma (in ring0) Ring_ZF_1_L13:
  assumes a∈R and a² ≠ 0
  shows a≠0
  using assms Ring_ZF_1_L2 Ring_ZF_1_L6 by auto

Square of an element and its opposite are the same.

lemma (in ring0) Ring_ZF_1_L14:
  assumes a∈R shows (-a)² = ((a)²)
  using assms Ring_ZF_1_L7A by simp

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.

lemma (in ring0) Ring_ZF_1_L15:
  assumes H ⊆ R and H {is closed under} A
  shows (H ∪ {0}) {is closed under} A
  using assms Ring_ZF_1_L1 group0.group0_2_L17 by simp

Adding zero to a set that is closed under multiplication results in a set that
is also closed under multiplication.

lemma (in ring0) Ring_ZF_1_L16:
  assumes A1: H ⊆ R and A2: H {is closed under} M
  shows (H ∪ {0}) {is closed under} M
  using assms Ring_ZF_1_L2 Ring_ZF_1_L6 IsOpClosed_def
  by auto

The ring is trivial if 0 = 1.

lemma (in ring0) Ring_ZF_1_L17: shows R = {0} ↔ 0=1
proof
  assume R = {0}
  then show 0=1 using Ring_ZF_1_L2
    by blast
next
  assume A1: 0 = 1
  then have R ⊆ {0}
    using Ring_ZF_1_L3 Ring_ZF_1_L6 by auto
  moreover have {0} ⊆ R using Ring_ZF_1_L2 by auto
  ultimately show R = {0} by auto
qed
The sets \( \{m \cdot x \in R\} \) and \( \{-m \cdot x \in R\} \) are the same.

**Lemma** (in ring0) *RingZF_1_L18*: assumes \( m \in R \)
shows \( \{m \cdot x \in R\} = \{-m \cdot x \in R\} \)

**Proof**

\[
\forall a \in \{m \cdot x \in R\} \\
\exists x \in R \text{ where } a = m \cdot x \\
\text{ by auto} \\
\text{ with } A1 \text{ have } (-x) \in R \text{ and } a = (-m) \cdot (-x) \\
\text{ using RingZF_1_L3, RingZF_1_L7A by auto} \\
\text{ then have } a \in \{-m \cdot x \in R\} \\
\text{ by auto} \\
\text{ then show } \{m \cdot x \in R\} \subseteq \{-m \cdot x \in R\} \\
\text{ by auto} \\
\]

next

\[
\forall a \in \{-m \cdot x \in R\} \\
\exists x \in R \text{ where } a = (-m) \cdot x \\
\text{ by auto} \\
\text{ with } A1 \text{ have } (-x) \in R \text{ and } a = m \cdot (-x) \\
\text{ using RingZF_1_L3, RingZF_1_L7 by auto} \\
\text{ then have } a \in \{m \cdot x \in R\} \\
\text{ by auto} \\
\text{ then show } \{-m \cdot x \in R\} \subseteq \{m \cdot x \in R\} \\
\text{ by auto} \\
\]

qed

### 38.2 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) *RingZF_2_L1*: assumes \( a \in R \) \( b \in R \)
shows \( a + b \cdot a = (b + 1) \cdot a \)

**Proof**

\[
\text{using asms RingZF_1_L2, ring_oper_distr, RingZF_1_L3, RingZF_1_L4 by simp} \\
\]

**Rearrangements with two elements and cancelling.**

**Lemma** (in ring0) *RingZF_2_L1A*: assumes \( a \in R \) \( b \in R \)
shows \( a - b + b = a \) \( a + b - a = b \) \( (-a) + b + a = b \) \( (-a) + (b + a) = b \) \( a + (b - a) = b \)

**Proof**

\[
\text{using asms RingZF_1_L1, group0.inv_cancel_two, group0.group0_4_L6A by auto} \\
\]

409
In commutative rings $a-(b+1)c = (a-d-c)+(d-bc)$. For unknown reasons we have to use the raw set notation in the proof, otherwise all methods fail.

**lemma (in ring0) Ring_ZF_2_L2:**

assumes $A1: a \in R \ b \in R \ c \in R \ d \in R$

shows $a-(b+1)c = (a-d-c)+(d-bc)$

**proof**

- let $B = b \cdot c$
  from ringAssum have $A \{\text{is commutative on}\} R$
  using IsAring_def by simp
  moreover from $A1$ have $a \in R \ B \in R \ c \in R \ d \in R$
  using Ring_ZF_1_L4 by auto
  ultimately have $A \langle a, GroupInv(R,A)(A \langle B, c \rangle) \rangle = A \langle A \langle a, GroupInv(R,A)(d) \rangle , GroupInv(R,A)(c) \rangle, A \langle d, GroupInv(R,A)(B) \rangle$
  using Ring_ZF_1_L1 group0.group0_4_L8 by blast
  with $A1$ show thesis
  using Ring_ZF_1_L2 ring_oper_distr Ring_ZF_1_L3 by simp

qed

Rerrangement 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
  group0(R,A)
  $A \{\text{is commutative on}\} R$
  $a \cdot x \in R \ b \in R \ c \cdot x \in R \ d \in R$
  using Ring_ZF_1_L1 group0.group0_4_L8 by auto
  then have $A \langle A \langle a \cdot x,b \rangle , A \langle c \cdot x,d \rangle \rangle = A \langle A \langle a \cdot x,c \cdot x \rangle , A \langle b,d \rangle \rangle$
  by (rule group0.group0_4_L8)
  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 $M \{\text{is commutative on}\} R$

and $a \in R \ b \in R \ c \in R$

shows $a \cdot (b \cdot c) = a \cdot c \cdot b$

using assms IsCommutative_def Ring_ZF_1_L11 by simp

Some other rearrangements with three elements.

**lemma (in ring0) ring_rearr_3_elemA:**

assumes $A1: M \{\text{is commutative on}\} R$ and
$A2: a \in R \ b \in R \ c \in R$

410
shows
\[ a \cdot (a \cdot c) - b \cdot (-b \cdot c) = (a \cdot a + b \cdot b) \cdot c \]
\[ a \cdot (-b \cdot c) + b \cdot (a \cdot c) = 0 \]

proof -
from A2 have T:
\[
\begin{align*}
 b \cdot c & \in R \\
 a \cdot a & \in R \\
 b \cdot b & \in R \\
 b \cdot (b \cdot c) & \in R \\
 a \cdot (b \cdot c) & \in R
\end{align*}
\]
using Ring_ZF_1_L4 by auto
with A2 show
\[
\begin{align*}
 a \cdot (a \cdot c) - b \cdot (-b \cdot c) &= (a \cdot a + b \cdot b) \cdot c \\
 a \cdot (-b \cdot c) + b \cdot (a \cdot c) &= 0
\end{align*}
\]
using Ring_ZF_1_L7 Ring_ZF_1_L3 Ring_ZF_1_L11
ring_oper_distr by simp
from A2 T have
\[
\begin{align*}
 a \cdot (-b \cdot c) + b \cdot (a \cdot c) &= (-a \cdot (b \cdot c)) + b \cdot a \cdot c \\
 a \cdot (-b \cdot c) + b \cdot (a \cdot c) &= 0
\end{align*}
\]
using IsCommutative_def Ring_ZF_1_L11 Ring_ZF_1_L3
by simp
finally show a \cdot (-b \cdot c) + b \cdot (a \cdot c) = 0
by simp

qed

Some rearrangements with four elements. Properties of abelian groups.

lemma (in ring0) Ring_ZF_2_L5:
assumes a\in R b\in R c\in R d\in R
shows
\[
\begin{align*}
 a - b - c - d &= a - d - b - c \\
 a + b + c - d &= a - d + b + c \\
 a + b - c - d &= a - c + (b - d) \\
 a + b + c + d &= a + c + (b + d)
\end{align*}
\]
using assms Ring_ZF_1_L1 group0.rearr_ab_gr_4_elemB
group0.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\in R b\in R c\in R d\in R e\in R f\in R
shows
\[
\begin{align*}
 a \cdot (c \cdot e - d \cdot f) - b \cdot (c \cdot f + d \cdot e) &= \\
 (a \cdot c - b \cdot d) \cdot e - (a \cdot d + b \cdot c) \cdot f &= \\
 a \cdot (c \cdot f + d \cdot e) + b \cdot (c \cdot e - d \cdot f) &= \\
 (a \cdot c - b \cdot d) \cdot f + (a \cdot d + b \cdot c) \cdot e &= \\
 a \cdot (c \cdot e) - b \cdot (d \cdot f) &= a \cdot c - b \cdot d + (a \cdot e - b \cdot f) \\
 a \cdot (d \cdot f) + b \cdot (c \cdot e) &= a \cdot d + b \cdot c + (a \cdot f + b \cdot e)
\end{align*}
\]
proof -
from A1 have T:
\[
\begin{align*}
 c \cdot e & \in R \\
 d \cdot f & \in R \\
 c \cdot f & \in R \\
 d \cdot e & \in R \\
 a \cdot c & \in R \\
 b \cdot d & \in R \\
 a \cdot d & \in R \\
 b \cdot c & \in R \\
 b \cdot f & \in R \\
 a \cdot e & \in R \\
 b \cdot e & \in R \\
 a \cdot f & \in R
\end{align*}
\]
a · c · e ∈ R  
a · d · f ∈ R
b · c · f ∈ R  
b · d · e ∈ R
b · c · e ∈ R  
b · d · f ∈ R
a · c · f ∈ R  
a · d · e ∈ R
a · c · e - a · d · f ∈ R
a · c · e - b · d · e ∈ R
a · c · f + a · d · e ∈ R
a · c · f - b · d · f ∈ R
a · c + a · e ∈ R
a · d + a · f ∈ R

using Ring_ZF_1_L4 by auto

with A1  show a · (c · e - d · f) - b · (c · f + d · e) =
(a · c - b · d) · e - (a · d + b · c) · f
using Ring_ZF_1_L8 ring_oper_distr Ring_ZF_1_L11
Ring_ZF_1_L10 Ring_ZF_2_L5 by simp

from A1  T  show a · (c · f + d · e) + b · (c · e - d · f) =
(a · c - b · d) · f + (a · d + b · c) · e
using Ring_ZF_1_L8 ring_oper_distr Ring_ZF_1_L11
Ring_ZF_1_L10A Ring_ZF_2_L5 Ring_ZF_1_L10
by simp

from A1  T  show a · (c + e) - b · (d + f) = a · c - b · d + (a · e - b · f)
a · (d + f) + b · (c + e) = a · d + b · c + (a · f + b · e)
using ring_oper_distr Ring_ZF_1_L10 Ring_ZF_2_L5
by auto

qed

end

39  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.

39.1  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) RingZF1_1_L1:**

assumes $A1: s \in AH \ r \in AH \ q \in AH$ and $A2: n \in G$ shows

$((s \circ (r \cdot q))(n)) \cdot (((s \circ r) \cdot (s \circ q))(n))^{-1} = \delta(s, \langle r(n), q(n) \rangle)$

$((r \cdot q) \circ s)(n) = ((r \circ s) \cdot (q \circ s))(n)$

**proof**

- from groupAssum isAbelian $A1$ have $T1$:
  
  - $r \cdot q \in AH$ $s \in AH$ $n \in G$
  - $s \circ r \in AH$ $s \circ q \in AH$
  - $s \circ (r \cdot q) \in AH$

  using GroupZF3_2_L15 GroupZF3_4_T1 by auto

- from $A1$ $A2$ have $T2$:
  
  - $s(n) \in G$ $r(n) \in G$
  - $s(r(n)) \in G$
  - $\delta(s, \langle r(n), q(n) \rangle) \in G$
  - $s(r(n)) \cdot s(q(n)) \in G$

  using AlmostHoms_def apply_funtype GroupZF3_2_L4B group0_2_L1 monoid0.group0_1_L1 by auto

with $T1$ $A1$ $A2$ isAbelian show

$((s \circ (r \cdot q))(n)) \cdot (((s \circ r) \cdot (s \circ q))(n))^{-1} = \delta(s, \langle r(n), q(n) \rangle)$

$((r \cdot q) \circ s)(n) = ((r \circ s) \cdot (q \circ s))(n)$

using GroupZF3_3_2_L12 GroupZF3_4_L2 GroupZF3_4_L1 group0_4_L6A by auto

qed

The sides of the distributive law equations for almost homomorphisms are almost equal.

**Lemma (in group1) RingZF1_1_L2:**

assumes $A1: s \in AH \ r \in AH \ q \in AH$ shows

$s \circ (r \cdot q) \approx (s \circ r) \cdot (s \circ q)$

$(r \cdot q) \circ s = (r \circ s) \cdot (q \circ s)$

**proof**

- from $A1$ have $\forall n \in G. \ \langle r(n), q(n) \rangle \in G \times G$
  
  using AlmostHoms_def apply_funtype by auto

moreover from $A1$ have $\{\delta(s, x). \ x \in G \times G\} \in Fin(G)$

using AlmostHoms_def by simp

ultimately have $\{\delta(s, \langle r(n), q(n) \rangle). \ n \in G\} \in Fin(G)$

by (rule Finite1_L6B)

with $A1$ have

$\{((s \circ (r \cdot q))(n)) \cdot (((s \circ r) \cdot (s \circ q))(n))^{-1}. \ n \in G\} \in Fin(G)$

using RingZF1_1_L1 by simp

moreover from groupAssum isAbelian $A1$ $A1$ have

$s \circ (r \cdot q) \in AH$ $(s \circ r) \cdot (s \circ q) \in AH$

using GroupZF3_3_2_L15 GroupZF3_4_T1 by auto

ultimately show $s \circ (r \cdot q) \approx (s \circ r) \cdot (s \circ q)$
using Group_ZF_3_4_L12 by simp
from groupAssum isAbelian A1 have
(r·q)os : G→G (ros)·(qos) : 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)os)(n) = ((ros)·(qos))(n)
using Ring_ZF_1_1_L1 by simp
ultimately show (r·q)os = (ros)·(qos)
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{(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{(q·r)} = R{(q·r)}
  proof -
  from T1 D1 have equiv(AH,R) so(q·r)≈(soq)·(sor)
    using Ring_ZF_1_1_L2 by auto
  with A1 show thesis using equiv_class_eq by simp
qed
also from A1 T1 D1 A3 have
  R{(soq)·(sor)} = A⟨M⟨a,b⟩,M⟨a,c⟩⟩
  using Group_ZF_3_3_L4 Group_ZF_3_4_T1 EquivClass_1_L10
  Group_ZF_3_3_L3 Group_ZF_3_4_L13A EquivClass_1_L10 Group_ZF_3_4_T1
  by simp
finally show M⟨a,A⟨b,c⟩⟩ = A⟨M⟨a,b⟩,M⟨a,c⟩⟩
  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) RingZF_1_1_L4:
assumes A1: R = QuotientGroupRel(AH,Op1,FR)
shows IsDistributive(AH//R,A,M)

proof -
  from A1 A2 have \( \forall a \in (AH//R). \forall b \in (AH//R). \forall c \in (AH//R).\)
  \( M(a,A\langle b,c \rangle) = A\langle M( a,b), M( a,c) \rangle \land \)
  \( M(A\langle b,c \rangle, a) = A\langle M( b,a), M( c,a) \rangle \)
  using RingZF_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) RingZF_1_1_T1:
assumes R = QuotientGroupRel(AH,Op1,FR)
shows IsAring(AH//R,A,M)

using assms QuotientGroupOp_def GroupZF_3_3_T1 GroupZF_3_4_T2
  RingZF_1_1_L4 IsAring_def by simp

end

40 Ordered rings

theory OrderedRing_ZF imports Ring_ZF OrderedGroup_ZF_1

begin

In this theory file we consider ordered rings.

40.1 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**
IsAnOrdRing(R,A,M,r) ≡
  (IsAring(R,A,M) ∧ (M {is commutative on} R) ∧
   r⊂R×R ∧ IsLinOrder(R,r) ∧
   (\( \forall a \ b. \forall c\in R. ( a,b) \in r \rightarrow (A( a,c),A( b,c) \in r) \land \)
    (Nonnegative(R,A,r) {is closed under} M)))
The next context (locale) defines notation used for ordered rings. We do that by extending the notation defined in the \texttt{ring0} locale and adding some assumptions to make sure we are talking about ordered rings in this context.

\begin{verbatim}
locale ring1 = ring0 +

    assumes mult_commut: M \{is commutative on\} R

    fixes r

    assumes ordincl: r ⊆ R×R

    assumes linord: IsLinOrder(R,r)

    fixes lesseq (infix ≤)
    defines lesseq_def [simp]: a ≤ b ≡ (a, b) ∈ r

    fixes sless (infix <)
    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)
\end{verbatim}

The next lemma assures us that we are talking about ordered rings in the \texttt{ring1} context.

\begin{verbatim}
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
\end{verbatim}

We can use theorems proven in the \texttt{ring1} context whenever we talk about an ordered ring.

\begin{verbatim}
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
\end{verbatim}

In the \texttt{ring1} context \(a ≤ b\) implies that \(a, b\) are elements of the ring.

\begin{verbatim}
lemma (in ring1) OrdRing_ZF_1_L3: assumes a ≤ b
    shows a ∈ R b ∈ R
    using assms ordincl by auto
\end{verbatim}

Ordered ring is an ordered group, hence we can use theorems proven in the \texttt{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 ∀g ∈ R. ∀a b. a ≤ b → a+g ≤ b+g ∧ g+a ≤ 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 ≤ b b ≤ c shows a ≤ c proof - from A1 have group3(R,A,r) ⟨a,b⟩ ∈ r ⟨b,c⟩ ∈ r using OrdRing_ZF_1_L4 by auto then have ⟨a,c⟩ ∈ r by (rule group3.Group_order_transitive) then show a ≤ c by simp qed

Transitivity for the strict order: if a < b and b ≤ c, then a < c. Property of ordered groups.

lemma (in ring1) ring_strict_ord_trans: assumes A1: a < b and A2: b ≤ c shows a < c proof - from A1 A2 have group3(R,A,r) ⟨a,b⟩ ∈ r ∧ a ≠ b ⟨b,c⟩ ∈ r using OrdRing_ZF_1_L4 by auto then have ⟨a,c⟩ ∈ r ∧ a ≠ c by (rule group3.OrderedGroup_ZF_1_L2A)
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
    \( \text{group3}(R,A,r) \)
    \( (a,b) \in r \) (\( b,c \) \( \in r \) \( \wedge b \neq c \)
    using OrdRing_ZF_1_L4 by auto
  then have \( (a,c) \in r \) by (rule group3.group_strict_ord_transit)
  then show \( a < c \) by simp
  qed

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: \( a \in R \) \( b \in R \)
  and A2: \( \neg (a \leq b) \)
  shows \( b \leq a \) \( (-a) \leq (-b) \) \( a \neq b \)
proof -
  from A1 A2 have I:
    \( \text{group3}(R,A,r) \)
    \( r \) (is total on) \( R \)
    \( a \in R \) \( b \in R \) \( (a,b) \notin r \)
    using OrdRing_ZF_1_L4 by auto
  then have \( (b,a) \in r \) by (rule group3.OrderedGroup_ZF_1_L8)
  then show \( b \leq a \) by simp
  from I have \( (\text{GroupInv}(R,A)(a),\text{GroupInv}(R,A)(b)) \in r \)
    by (rule group3.OrderedGroup_ZF_1_L8)
  then show \( (-a) \leq (-b) \) by simp
  from I show \( a \neq b \) by (rule group3.OrderedGroup_ZF_1_L8)
  qed

A special case of \( \text{OrdRing}_\text{ZF}_1\_L4A \) when one of the constants is 0. This is useful for many proofs by cases.

corollary (in ring1) ord_ring_split2: assumes A1: \( a \in R \)
  shows \( a \leq 0 \ \vee \ (0 \leq a \ \wedge \ a \neq 0) \)
proof -
  \{ from A1 have I: \( a \in R \) \( 0 \in R \)
    using Ring_ZF_1_L2 by auto
    moreover assume A2: \( \neg (a \leq 0) \)
    ultimately have \( 0 \leq a \) by (rule OrdRing_ZF_1_L4A)
    moreover from I A2 have \( a \neq 0 \) by (rule OrdRing_ZF_1_L4A)
    ultimately have \( 0 \leq a \ \wedge \ a \neq 0 \) by simp \}
  then show thesis by auto
Taking minus on both sides reverses an inequality.

**lemma (in ring1) OrdRing_ZF_1_L4B:** assumes $a \leq b$
shows $(-b) \leq (-a)$
using assms OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_1_L6
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 $0 \leq a \leq b$
shows $0 \leq a \cdot b$
using pos_mult_closed assms OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_1_L2
IsOpClosed_def by simp

Double nonnegative is nonnegative.

**lemma (in ring1) OrdRing_ZF_1_L5A:** assumes $A1: 0 \leq a$
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 $\text{IsAring}(R,A,M)$
$M \text{ is commutative on } R$
$\text{Nonnegative}(R,A,r) \text{ is closed under } M$
$\text{IsAnOrdGroup}(R,A,r)$
$r \text{ is total on } R$
shows $\text{IsAnOrdRing}(R,A,M,r)$
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 $a \in R \quad b \in R$
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 ≤ 0 and A2: 0 ≤ b
  shows a · b ≤ 0
proof -
  from A1 A2 have T1: a ∈ R b ∈ R a · b ∈ R
    using OrdRing_ZF_1_L3 Ring_ZF_1_L4 by auto
  from A1 A2 have 0 ≤ (-a) · b
    using OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_1_L5A OrdRing_ZF_1_L5
    by simp
  with T1 have a · b ≤ 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:
    a ∈ R b ∈ R c ∈ R 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 have 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 $A1: a \in \mathbb{R}$ shows $0 \leq (a^2)$

proof -

\{ assume $0 \leq a$
  then have $0 \leq (a^2)$ using OrdRing_ZF_1_L5 by simp \}

moreover

\{ assume $\neg (0 \leq a)$
  with $A1$ have $0 \leq (-a^2)$
  using OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_1_L8A
  OrdRing_ZF_1_L5 by simp
  with $A1$ have $0 \leq (a^2)$ using Ring_ZF_1_L14 by simp \}

ultimately show thesis by blast

qed

1 is nonnegative.

corollary (in ring1) ordring_one_is_nonneg: shows $0 \leq 1$

proof -

have $0 \leq (1^2)$ using Ring_ZF_1_L2 OrdRing_ZF_1_L10 by simp

then show $0 \leq 1$ using Ring_ZF_1_L2 Ring_ZF_1_L3 by simp

qed

In nontrivial rings one is positive.

lemma (in ring1) ordring_one_is_pos: assumes $0 \neq 1$

shows $1 \in \mathbb{R}$

using assms Ring_ZF_1_L2 ordring_one_is_nonneg PositiveSet_def by auto

Nonnegative is not negative. Property of ordered groups.

lemma (in ring1) OrdRing_ZF_1_L11: assumes $0 \leq a$

shows $\neg (a \leq 0 \land a \neq 0)$

using assms OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_1_L5AB by simp

A negative element cannot be a square.

lemma (in ring1) OrdRing_ZF_1_L12:

assumes $A1: a \leq 0 \quad a \neq 0$

shows $\neg (\exists b \in \mathbb{R}. \ a = (b^2))$

proof -

\{ assume $\exists b \in \mathbb{R}. \ a = (b^2)$
  with $A1$ have False using OrdRing_ZF_1_L10 OrdRing_ZF_1_L11 by auto \}

then show thesis by auto

qed

If $a \leq b$, then $0 \leq b - a$.

lemma (in ring1) OrdRing_ZF_1_L13: assumes $a \leq b$

shows $0 \leq b - a$
If $a < b$, then $0 < b - a$.

**Lemma (in ring1) OrdRingZF_1_L14**

**Assumes** $a \leq b$, $a \neq b$

**Shows**

- $0 \leq b - a$
- $b - a \in \mathbb{R}$

**Proof**

- Using Assms OrdRingZF_1_L4 Group3.OrderedGroupZF_1_L9D
- By simp

If the difference is nonnegative, then $a \leq b$.

**Lemma (in ring1) OrdRingZF_1_L15**

**Assumes** $a \in \mathbb{R}$, $b \in \mathbb{R}$ and $0 \leq b - a$

**Shows** $a \leq b$

**Proof**

- Using Assms OrdRingZF_1_L4 Group3.OrderedGroupZF_1_L9F
- By simp

A nonnegative number is does not decrease when multiplied by a number greater or equal 1.

**Lemma (in ring1) OrdRingZF_1_L16**

**Assumes** $A_1$: $0 \leq a$ and $A_2$: $1 \leq b$

**Shows** $a \leq a \cdot b$

**Proof**

- From $A_1$, $A_2$ have $T$: $a \in \mathbb{R}$, $b \in \mathbb{R}$, $a \cdot b \in \mathbb{R}$ using OrdRingZF_1_L3 RingZF_1_L4 by auto
- From $A_1$, $A_2$ have $0 \leq a \cdot (b - 1)$ using OrdRingZF_1_L13 OrdRingZF_1_L5 by simp
- With $T$ show $a \leq a \cdot b$
- Using RingZF_1_L8 RingZF_1_L2 RingZF_1_L3 OrdRingZF_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) OrdRingZF_1_L17**

**Assumes** $A_1$: $0 \leq a$ and $A_2$: $a \leq b$ and $A_3$: $1 \leq c$

**Shows** $a \leq b \cdot c$

**Proof**

- From $A_1$, $A_2$ have $0 \leq b$ by (rule ring_ord_transitive)
- With $A_3$ have $b \leq b \cdot c$ using OrdRingZF_1_L16 by simp
- With $A_2$ show $a \leq b \cdot c$ by (rule ring_ord_transitive)
- Qed

Strict order is preserved by translations.

**Lemma (in ring1) ring_strict_ord_trans_inv**

**Assumes** $a \prec b$ and $c \in \mathbb{R}$
shows  
\[ a + c < b + c \]
\[ c + a < c + b \]
using assms OrdRing_ZF_1_L4 group3.group_strict_ord_transl_inv  
by auto

We can put an element on the other side of a strict inequality, changing its sign.

lemma (in ring1) OrdRing_ZF_1_L18:  
assumes \[ a \in R \quad b \in R \quad \text{and} \quad a - b < c \]
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 \[ a < b \quad \text{and} \quad c \leq d \]
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 \[ a \leq b \quad \text{and} \quad c < d \]
shows \[ a + c < b + d \]
using assms OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_1_L12D  
by simp

40.2 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 \[ 0 \leq a \quad 0 \leq b \]
shows \[ |a \cdot b| = |a| \cdot |b| \]
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 \( a \in R \)
shows \( |-a| = |a| \)
using assms OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_3_L7A by simp

The next lemma states that \( |a \cdot (-b)| = |(-a) \cdot b| = |(-a) \cdot (-b)| = |a \cdot b| \).

lemma (in ring1) OrdRing_ZF_2_L3: assumes \( a \in R \quad b \in R \)
shows \( |-a \cdot b| = |a \cdot b| \)
\( |a \cdot (-b)| = |a \cdot b| \)
\( |(-a) \cdot (-b)| = |a \cdot b| \)
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.

lemma (in ring1) OrdRing_ZF_2_L4: assumes \( a \in R \quad \neg (0 \leq a) \)
shows \( 0 \leq (-a) \quad 0 \neq a \)
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 \( A1: a \in R \quad b \in R \)
shows \( |a \cdot b| = |a| \cdot |b| \)
proof -
\{ assume \( A2: 0 \leq a \) have \( |a \cdot b| = |a| \cdot |b| \)
proof -
\{ assume \( 0 \leq b \)
with \( A2 \) have \( |a \cdot b| = |a| \cdot |b| \)
using OrdRing_ZF_2_L1 by simp
moreover
\{ assume \( \neg (0 \leq b) \)
with \( A1 \ \ A2 \) have \( |a \cdot (-b)| = |a| \cdot |-b| \)
using OrdRing_ZF_2_L4 OrdRing_ZF_2_L1 by simp
with \( A1 \) have \( |a \cdot b| = |a| \cdot |b| \)
using OrdRing_ZF_2_L2 OrdRing_ZF_2_L3 by simp
ultimately show thesis by blast
qed
\}
moreover
\{ assume \( \neg (0 \leq a) \)
with \( A1 \) have \( A3: 0 \leq (-a) \)
using OrdRing_ZF_2_L4 by simp
have \( |a \cdot b| = |a| \cdot |b| \)
proof -
\{ assume \( 0 \leq b \)
with \( A3 \) have \( |(-a) \cdot b| = |(-a)| \cdot |b| \)
using OrdRing_ZF_2_L1 by simp
with \( A1 \) have \( |a \cdot b| = |a| \cdot |b| \)
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
  by simp

If \( a ≤ c \) and \( b ≤ c \), then \( a + b ≤ 2 · c \).

lemma (in ring1) OrdRing_ZF_2_L6: assumes a≤c b≤c shows a+b ≤ 2·c
  using assms OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_1_L5B
  OrdRing_ZF_1_L3 Ring_ZF_1_L3 by simp

40.3  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: shows \( R^+ \) {is closed under} \( A \)
  using OrdRing_ZF_1_L4 group3.OrdGroup_ZF_1_L13 by simp

Every element of a ring can be either in the postitive 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: assumes a∈R
  shows Exactly_1_of_3_holds (a=0, a∈R^+, (-a) ∈ R_+)
  using assms OrdRing_ZF_1_L4 group3.OrdGroup_decomp
  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: assumes a∈R a≠0 a ∉ R^+
  shows (-a) ∈ R_+
  using assms OrdRing_ZF_1_L4 group3.OrdGroup_cases
by simp

R+ is closed under multiplication iff the ring has no zero divisors.

lemma (in ring1) OrdRing_ZF_3_L3:
shows (R+ {is closed under} M) ←→ HasNoZeroDivs(R,A,M)
proof
assume A1: HasNoZeroDivs(R,A,M)
{ fix a b assume a∈R+ b∈R+
  then have 0≤a a≠0 0≤b b≠0
    using PositiveSet_def by auto
  with A1 have a·b ∈ R+
    using OrdRing_ZF_1_L5 Ring_ZF_1_L2 OrdRing_ZF_1_L3 Ring_ZF_1_L12
    OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_1_L2A
    by simp
} then show R+ {is closed under} M using IsOpClosed_def
  by simp

next assume A2: R+ {is closed under} M
{ fix a b assume A3: a∈R b∈R and a≠0 b≠0
  with A2 have |a·b| ∈ R+
    using OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_3_L12 IsOpClosed_def
    OrdRing_ZF_2_L5 by simp
  with A3 have a·b ≠ 0
    using PositiveSet_def Ring_ZF_1_L4
    OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_3_L2A
    by auto
} then show HasNoZeroDivs(R,A,M) using HasNoZeroDivs_def
  by auto
qed

Another (in addition to OrdRing_ZF_1_L6 sufficient condition that defines
order in an ordered ring starting from the positive set.

theorem (in ring0) ring_ord_by_positive_set:
assumes
A1: M {is commutative on} R and
A2: P ⊆ R P {is closed under} A 0 /∈ P and
A3: ∀a∈R. a≠0 → (a∈P) Xor ((-a) ∈ P) and
A4: P {is closed under} M and
A5: r = OrderFromPosSet(R,A,P)
shows
IsAnOrdGroup(R,A,r)
IsAnOrdRing(R,A,M,r)
r {is total on} R
PositiveSet(R,A,r) = P
Nonnegative(R,A,r) = P ∪ {0}
HasNoZeroDivs(R,A,M)
proof -
from A2 A3 A5 show
  I: IsAnOrdGroup(R,A,r) r {is total on} R and
  II: PositiveSet(R,A,r) = P and
III: Nonnegative(R,A,r) = P ∪ {0}
using Ring_ZF_1_L1 group0.Group_ord_by_positive_set
by auto
from A2 A4 III have Nonnegative(R,A,r) {is closed under} M
using Ring_ZF_1_L16 by simp
with ringAssum A1 I show IsAnOrdRing(R,A,M,r)
using OrdRing_ZF_1_L6 by simp
with A4 II show HasNoZeroDivs(R,A,M)
using OrdRing_ZF_1_L2 ring1.OrdRing_ZF_3_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 ≠ 1
shows R. ∉ Fin(R)
R ∈ Fin(R)
using assms Ring_ZF_1_L17 OrdRing_ZF_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_ZF_3_L4:
assumes 0 ≠ 1 and ∀ a ∈ R. ∃ b ∈ B. a ≤ b
shows ¬ IsBoundedAbove(B,r)
B /∈ Fin(R)
using assms Ring_ZF_1_L17 OrdRing_ZF_1_L4 group3.OrderedGroup_ZF_2_L2A
by auto

If m is greater or equal the multiplicative unit, then the set \{m · n : n ∈ R\} is infinite (unless the ring is trivial).

lemma (in ring1) OrdRing_ZF_3_L5: assumes A1: 0 ≠ 1 and A2: 1 ≤ m
shows
\{m · x ∈ R\} /∈ Fin(R)
\{m · x ∈ R\} /∈ Fin(R)
\{(-m) · x ∈ R\} /∈ Fin(R)
proof -
from A2 have T: m ∈ R using OrdRing_ZF_1_L3 by simp
from A2 have 0 ≤ 1 1 ≤ m
using ordring_one_is_nonneg by auto
then have I: 0 ≤ m by (rule ring_ord_transitive)
let B = {m · x ∈ R}
{ fix a assume A3: a ∈ R
  then have a ≤ 0 ∨ (0 ≤ a ∧ a ≠ 0)
using ord_ring_split2 by simp

moreover
{ assume A4: a ≤ 0
from A1 have m · 1 ∈ B using ordring_one_is_pos
by auto
  with T have m ∈ B using Ring_ZF_1_L3 by simp
moreover from A4 I have a ≤ m by (rule ring_ord_transitive)
ultimately have ∃ b ∈ B. a ≤ b by blast }
moreover
{ assume A4: 0 ≤ a ∧ a ≠ 0
with A3 have m a ∈ B using PositiveSet_def
by auto
moreover from A2 A4 have 1 · a ≤ m · a using OrdRing_ZF_1_L9
by simp
  with A3 have a ≤ m · a using Ring_ZF_1_L3
ultimately have ∃ b ∈ B. a ≤ b by auto
}
ultimately have ∃ b ∈ B. a ≤ b by auto

then have ∀ a ∈ R. ∃ b ∈ B. a ≤ b by simp
with A1 show B ∉ Fin(R) using OrdRing_ZF_3_L4
by simp
moreover have B ⊆ {m · x. x ∈ R}
  using PositiveSet_def by auto
ultimately show {m · x. x ∈ R} ∉ Fin(R) using Fin_subset
by auto
with T show {(-m) · x. x ∈ R} ∉ 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 
\( \{m \cdot n : n ∈ R\} \) is infinite (unless the ring is trivial).

**lemma (in ring1) OrdRing_ZF_3_L6:*** assumes A1: 0 ≠ 1 and A2: m ≤ -1
shows \( \{m \cdot x : x ∈ R\} \) does not belong to \( \text{Fin}(R) \)

**proof** -
  from A2 have \((-(-1)) \) ≤ -m
  using OrdRing_ZF_1_L4B by simp
  with A1 have \(\{(-m) \cdot x : x ∈ R\} \) ∉ Fin(R)
    using Ring_ZF_1_L2 Ring_ZF_1_L3 OrdRing_ZF_3_L5
  by simp
  with A2 show \( \{m \cdot x : x ∈ R\} \) ∉ 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 ∈ R_+ and A2: a ≤ b
shows b ∈ R_+

428
proof -

  from A1 A2 have
    group3(R,A,r)
    a ∈ PositiveSet(R,A,r)
    ⟨a,b⟩ ∈ r
    using OrdRing_ZF_1_L4 by auto
  then have b ∈ PositiveSet(R,A,r)
    by (rule group3.OrderedGroup_ZF_1_L19)
  then show b ∈ 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≠1 and A2: 1≤a
  shows a ∈ 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⁺
  shows 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
  shows 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
  shows a< b+1
proof -
  from A1 A2 have I: b < b+1
    using OrdRing_ZF_1_L3 OrdRing_ZF_3_L10 by auto
  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 429
shows
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
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
  GreaterOf(r,1,a) ∈ R⁺
using OrdRing_ZF_3_L8 by auto
with A1 show GreaterOf(r,1,a) + 1 ∈ R⁺
  using ordring_one_is_pos OrdRing_ZF_3_L1 IsOpClosed_def
  by simp
from A1 I show
  a ≤ GreaterOf(r,1,a) + 1  a ≠ GreaterOf(r,1,a) + 1
  using OrdRing_ZF_3_L11 by auto
qed

We can multiply strict inequality by a positive element.

lemma (in ring1) OrdRing_ZF_3_L13:
assumes A1: HasNoZeroDivs(R,A,M) and
  A2: a<b and A3: c∈R⁺
shows
  a·c < b·c
c·a < c·b

proof -
from A2 A3 have T: a∈R  b∈R  c∈R  c≠0
  using OrdRing_ZF_1_L3 PositiveSet_def by auto
from A2 A3 have a·c ≤ b·c using OrdRing_ZF_1_L9A
  by simp
moreover from A1 A2 T have a·c ≠ b·c
  using Ring_ZF_1_L12A by auto
ultimately show a·c < b·c by simp
moreover from mult_commut T have a·c = c·a and b·c = c·b
  using IsCommutative_def by auto
ultimately show c·a < c·b by simp
qed

A sufficient condition for an element to be in the set of positive ring elements.

lemma (in ring1) OrdRing_ZF_3_L14: assumes 0≤a and a≠0
shows a ∈ R⁺
If a ring has no zero divisors, the square of a nonzero element is positive.

**Lemma (in ring1) OrdRing_ZF_3_L15:**

- Assumes $\text{HasNoZeroDivs}(R,A,M)$ and $a \in R \land a \neq 0$.
- Shows $0 \leq a^2$.
- Assms: OrdRing_ZF_1_L3 PositiveSet_def
- Uses: OrdRing_ZF_1_L10 Ring_ZF_1_L12 OrdRing_ZF_3_L14

In rings with no zero divisors we can (strictly) increase a positive element by multiplying it by an element that is greater than 1.

**Lemma (in ring1) OrdRing_ZF_3_L16:**

- Assumes $\text{HasNoZeroDivs}(R,A,M)$ and $a \in R^+$ and $1 \leq b \land b \neq 1$.
- Shows $a \leq ab$.
- Assms: OrdRing_ZF_1_L16 OrdRing_ZF_1_L3
- Uses: OrdRing_ZF_3_L17
- By auto

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:**

- Assumes $\text{HasNoZeroDivs}(R,A,M)$ and $a \in R^+$ and $a \leq b$ and $1 \leq c$.
- Shows $ab \leq bc$.
- Assms: OrdRing_ZF_1_L12C

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:**

- Assumes $\text{HasNoZeroDivs}(R,A,M)$ and $a \in R \land b \in R$ and $a \neq 0 \lor b \neq 0$.
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 $a^2 + b^2 = 0 + b^2$
  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 $b \neq 0$ by simp
  with $A1$ $A2$ have $0 \leq b^2$ and $b^2 \neq 0$
  using OrdRing_ZF_3_L15 by auto
  hence $0 < b^2$ by auto
  ultimately have $0 < a^2 + b^2$ by simp }

ultimately have $0 < a^2 + b^2$
by auto

qed

end

41 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.

41.1 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**

\[
\text{LimitC}(i) \equiv \text{Card}(i) \land 0 < i \land (\forall y. (y < i \land \text{Card}(y)) \rightarrow \text{csucc}(y) < i)
\]

Simple fact used a couple of times in proofs.

**lemma** nat_less_infty: assumes \(n \in \text{nat} \) and \(\text{InfCard}(X)\) shows \(n < X\)

**proof** -

from assms have \(n \in \text{nat} \) and \(n \leq X\) using \(\text{lt_def InfCard_def}\) by auto

then show \(n < X\) using \(\text{lt_trans2}\) by blast

qed

There are three types of cardinals, the zero one, the successors of other cardinals and the limit cardinals.

**lemma** Card_cases_disj:

assumes \(\text{Card}(i)\)

shows \(i = 0 \lor (\exists j. \text{Card}(j) \land i = \text{csucc}(j)) \lor \text{LimitC}(i)\)

**proof** -

from assms have \(D: \text{Ord}(i)\) using \(\text{Card_is_Ord}\) by auto

\{ 
  assume \(F: i \neq 0\)
  assume Contr: \(\neg \text{LimitC}(i)\)
  from \(F\) \(D\) have \(0 < i\) using \(\text{Ord_0_lt}\) by auto
  with \(\text{Contr}\) assms have \(\exists y. y < i \land \text{Card}(y) \land \neg \text{csucc}(y) < i\)
    using \(\text{LimitC_def}\) by blast
  then obtain \(y\) where \(y < i \land \text{Card}(y) \land \neg \text{csucc}(y) < i\) by blast
  with \(D\) have \(y < i\) \(i \leq \text{csucc}(y)\) and \(0: \text{Card}(y)\)
    using \(\text{not_lt_imp_le}\) \(\text{lt_Ord}\) \(\text{Card_csucc}\) \(\text{Card_is_Ord}\)
    by auto
  with assms have \(\text{csucc}(y) \leq i \leq \text{csucc}(y)\) using \(\text{csucc_le}\) by auto
  then have \(i = \text{csucc}(y)\) using \(\text{le_anti_sym}\) by auto
  with \(0\) have \(\exists j. \text{Card}(j) \land i = \text{csucc}(j)\) 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 \(\text{Card}(Q)\)

shows \(A \leq Q \implies A \preceq Q\)

**proof** -

\[\text{433}\]
assume $A \leq Q$
then have $A < Q \vee A = Q$ using `le_iff` by auto
then have $A \approx Q \vee A < Q$ using `eqpoll_refl` by auto
with assms have $A \approx Q \vee A < Q$ using `lt_Card_imp_lesspoll` by auto
then show $A \lessapprox Q$ 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 $Q = \text{nat} \vee (\exists j. \text{csucc}(j) \lessapprox Q \land \text{InfCard}(j))$
proof-
  { assume $\forall j. \neg \text{csucc}(j) \lessapprox Q \vee \neg \text{InfCard}(j)$
    then have $D: \neg \text{csucc}(\text{nat}) \lessapprox Q$ using `InfCard_nat` by auto
    with $D$ assms have $\neg (\text{csucc}(\text{nat}) \leq Q)$ using `le_imp_lesspoll` `InfCard_is_Card` by auto
    with $D$ have $Q < (\text{csucc}(\text{nat}))$ using not_le_iff_lt `Card_is_Ord` `Card_csucc` `Card_is_Ord` `Card_is_Ord` `Card_nat` by auto
    with $D$ assms have $Q \leq \text{nat}$ using `Card_lt_csucc_iff` `InfCard_is_Card` `Card_nat` by auto
    with $D$ 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 `Ord_linear_lt`

lemma Ord_linear_lt_IML: assumes Ord(i) Ord(j)
  shows $i < j \vee i = j \vee j < i$
using assms `lt_def` `Ord_linear` `disjE` by simp

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.

lemma Card_less_csucc_eq_le:
  assumes Card(m)
  shows $A < \text{csucc}(m) \iff A \lessapprox m$
proof
  have $S: \text{Ord}(\text{csucc}(m))$ using `Card_csucc` `Card_is_Ord` assms by auto
  { assume $A: A < \text{csucc}(m)$
    with $S$ have $|A| \approx A$ using `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 $|A| \lessapprox (|A| \approx \text{csucc}(m))$ using `lesspoll_def` by auto
  }
with $S$ have $||A|| \leq \text{csucc}(m) |A| \neq \text{csucc}(m)$ using lepoll_cardinal_le by auto
then have $|A| \leq \text{csucc}(m) |A| \neq \text{csucc}(m)$ using Card_def Card_cardinal by auto
then have $I: (\text{csucc}(m)<|A|) |A| \neq \text{csucc}(m)$ using le_imp_not_lt by auto
from $S$ have $\text{csucc}(m)<|A| \lor |A|=\text{csucc}(m) \lor |A|<\text{csucc}(m)$
using Card_cardinal Card_is_Ord Ord_linearLt_IML by auto
with $I$ have $|A|<\text{csucc}(m)$ by simp
with assms have $|A| \leq m$ using Card_lt_csucc_iff Card_cardinal by auto
then have $|A|=m \lor |A|<m$ using le_iff by auto
then have $|A| \approx m \lor |A|<m$ using eqpoll_refl by auto
then have $T: |A| \leq m$ using lesspoll_def eqpoll_imp_lepoll by auto
from $A$ have $A \approx |A|$ using lesspoll_imp_eqpoll eqpoll_sym by auto
also from $T$ have $\ldots \leq m$ by auto
finally show $A \less m$ by simp
}

{ assume $A: A \less m$
from assms have $m<\text{csucc}(m)$ using lt_Card_imp_lesspoll Card_csucc Card_is_Ord
  lt_csucc by auto
with $A$ show $A < \text{csucc}(m)$ using lesspoll_trans1 by auto
}

qed

If the successor of a cardinal is infinite, so is the original cardinal.

lemma csucc_inf_imp_inf:
assumes $\text{Card}(j)$ and $\text{InfCard}(\text{csucc}(j))$
shows $\text{InfCard}(j)$
proof-
{
assumes $f: \text{Finite}(j)$
then obtain $n$ where $n \in \text{nat} j \approx n$ using Finite_def by auto
with assms(1) have $T: j=n n \in \text{nat}$
  using cardinal_cong nat_into_Card Card_def by auto
then have $Q: \text{succ}(j) \in \text{nat}$ using nat_succI by auto
with $f$ have $T: \text{Finite}(\text{succ}(j)) \text{Card}(\text{succ}(j))$
  using nat_into_Card nat_succI by auto
from $T(2)$ have $\text{Card}(\text{succ}(j)) \land j<\text{succ}(j)$ using Card_is_Ord by auto
moreover from this have $\text{Ord}(\text{succ}(j))$ using Card_is_Ord by auto
moreover
{
fix $x$
assume $A: x<\text{succ}(j)$
{
assume $\text{Card}(x) \land j<x$
with $A$ have $\text{False}$ using lt_trans1 by auto

435
Since all the cardinals previous to \( \text{nat} \) are finite, it cannot be a successor cardinal; hence it is a \textit{LimitC} cardinal.

corollary \text{LimitC\_nat}:
  shows \text{LimitC}(\text{nat})
proof
  note \text{Card\_nat}
  moreover have \( 0<\text{nat} \) using \( \text{lt\_def} \) by auto
  moreover
  \{ 
    fix \( y \)
    assume \text{AS}: \( y<\text{nat Card}(y) \)
    then have \text{ord}: \text{Ord}(y) unfolding \( \text{lt\_def} \) by auto
    then have \text{Cacsucc}: \text{Card}(\text{csucc}(y)) using \text{Card\_csucc} by auto
    \{ 
      assume \text{nat} \leq \text{csucc}(y)
      with \text{Cacsucc} have \text{InfCard}(\text{csucc}(y)) using \text{InfCard\_def} by auto
      with \text{AS}(2) have \text{InfCard}(y) using \text{csucc\_inf\_imp\_inf} by auto
      then have \text{nat} \leq y using \text{InfCard\_def} by auto
      with \text{AS}(1) have False using \text{lt\_trans2} by auto
    \}
    hence \( \neg(\text{nat} \leq \text{csucc}(y)) \) by auto
    then have \text{csucc}(y) \text{<nat} using \text{not\_le\_iff\_lt Ord\_nat Cacsucc Card\_is\Ord} by auto
  \}
  ultimately show thesis using \text{LimitC\_def} by auto
qed

41.2 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 no use in the case where \textit{LimitC}(Q) and it also makes use of the Axiom of Choice. The mentioned theorem is in the theory file \textit{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.

**Lemma** less_less_imp_un_less:
assumes $A \prec Q$ and $B \prec Q$ and $\text{InfCard}(Q)$
shows $A \cup B \prec Q$
proof-

- \{ 
  assume $\text{Finite} (A) \land \text{Finite}(B)$
  then have $\text{Finite}(A \cup B)$ using $\text{Finite}_\text{Un}$ by auto
  then obtain $n$ where $R$: $A \cup B \approx n \land n \in \mathbb{N}$ using $\text{Finite}_\text{def}$
    by auto
  then have $|A \cup B| \prec \mathbb{N}$ using $\text{lt}_\text{def}$ $\text{cardinal}_\text{cong}$
    $\text{nat}_\text{into}_\text{Card}$ $\text{Card}_\text{def}$ $\text{Card}_\text{nat}$ $\text{Card}_\text{is}_\text{Ord}$ by auto
  with assms(3) have $T$: $|A \cup B| \prec Q$ using $\text{InfCard}_\text{def}$ $\text{lt}_\text{trans2}$ by auto
  from $R$ have $\text{Ord}(n)A \cup B \preceq n$ using $\text{nat}_\text{into}_\text{Card}$ $\text{Card}_\text{is}_\text{Ord}$ $\text{eqpoll}_\text{imp}_\text{lepoll}$ by auto
  then have $A \cup B \approx |A \cup B|$ using $\text{lepoll}_\text{Ord}_\text{imp}_\text{eqpoll}$ $\text{eqpoll}_\text{sym}$ by auto
  also from $T$ assms(3) have $\ldots \prec Q$ using $\text{lt}_\text{Card}_\text{imp}_\text{lepoll}$ $\text{InfCard}_\text{is}_\text{Card}$
    by auto
  finally have $A \cup B \prec Q$ by simp
\}

moreover

- \{ 
  assume $\neg(\text{Finite} (A) \land \text{Finite}(B))$
  hence $A: \neg\text{Finite} (A) \lor \neg\text{Finite}(B)$ by auto
  from assms have $B$: $|A| \approx A \land |B| \approx B$ using $\text{lesspoll}_\text{imp}_\text{eqpoll}$ $\text{lesspoll}_\text{imp}_\text{eqpoll}$ $\text{InfCard}_\text{is}_\text{Card}$ $\text{Card}_\text{is}_\text{Ord}$ by auto
  from $B(1)$ have $Aeq$: $\forall x. (|A| \approx x) \rightarrow (A \approx x)$
    using $\text{eqpoll}_\text{sym}$ $\text{eqpoll}_\text{trans}$ by blast
  from $B(2)$ have $Beq$: $\forall x. (|B| \approx x) \rightarrow (B \approx x)$
    using $\text{eqpoll}_\text{sym}$ $\text{eqpoll}_\text{trans}$ by blast
  with $A Aeq$ have $\neg\text{Finite}(|A|) \lor \neg\text{Finite}(|B|)$ using $\text{Finite}_\text{def}$
    by auto
  then have $D$: $\text{InfCard}(|A|) \lor \text{InfCard}(|B|)$
    using $\text{InfCard}_\text{is}_\text{InfCard}$ $\text{InfCard}_\text{is}_\text{InfCard}$ $\text{Card}_\text{cardinal}$ by blast
\}

437
assume AS: \(|A| < |B|\)
{
    assume \(\neg \text{InfCard}(|A|)\)
    with D have InfCard(|B|) by auto
}
moreover
{
    assume InfCard(|A|)
    then have nat\(\leq |A|\) using InfCard_def by auto
    with AS have nat\(< |B|\) using lt_trans1 by auto
    then have nat\(\leq |B|\) 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: \(2^{\leq |B|}\) using lt_Card_imp_lesspoll Card_cardinal lesspoll_def
    by auto
from B(2) have \(|B| \equiv B\) by simp
also from assms(2) have \(...\prec Q\) by auto
finally have TTT: \(|B| \prec Q\) by simp
from B(1) have Card(|B|) \(\lesssim |A|\) 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: \(A \lesssim |B|\) using lesspoll_def by auto
from B(2) have I2: \(B \lesssim |B|\) using eqpoll_sym eqpoll_imp_lepoll by auto
have A \(\cup B \lesssim A + B\) using Un_lepoll_sum by auto
also from I1 I2 have \(...\lesssim |B| + |B|\) using sum_lepoll_mono by auto
also from AG have \(...\lesssim |B| \Rightarrow |B|\) * \(|B|\) using sum_lepoll_prod by auto
also from assms(3) INFB have \(...\approx |B|\) using InfCard_square_eqpoll
    by auto
finally have A \(\cup B \lesssim |B|\) by simp
also from TTT have \(...\prec Q\) by auto
finally have A \(\cup B \prec Q\) by simp
}
moreover
{
    assume AS: \(|B| < |A|\)
    {
        assume \(\neg \text{InfCard}(|B|)\)
        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 \( \text{nat} \leq |A| \) using \( \text{leI} \) by auto
then have \( \text{InfCard}(|A|) \) using \( \text{InfCard_def Card_cardinal} \) by auto

ultimately have \( \text{INFB: InfCard}(|A|) \) by auto
then have \( 2 \leq |A| \) using \( \text{lt_Card_imp_lesspoll Card_cardinal lesspoll_def} \) by auto
from \( B(1) \) have \( |A| \approx A \) by simp
also from \( \text{assms(1)} \) have \( \ldots \prec Q \) by auto
finally have \( \text{TTT: } |A| \prec Q \) by simp
from \( B(2) \) have \( \text{Card}(|A|) \leq |B| \) using \( \text{eqpoll_sym Card_cardinal eqpoll_imp_lepoll} \) by auto
with \( \text{AS} \) have \( B \prec |A| \) using \( \text{lt_Card_imp_lesspoll lesspoll_trans1} \) by auto
then have \( I_1: B \leq |A| \) using \( \text{lesspoll_def} \) by auto
from \( B(1) \) have \( I_2: A \leq |A| \) using \( \text{eqpoll_sym eqpoll_imp_lepoll} \) by auto
have \( A \cup B \leq A+B \) using \( \text{Un_lepoll_sum by auto} \)
also from \( I_1 \) \( I_2 \) have \( \ldots \leq |A| + |A| \) using \( \text{sum_lepoll_mono by auto} \)
also from \( \text{AG} \) have \( \ldots \leq |A| * |A| \) using \( \text{sum_lepoll_prod by auto} \)
also from \( \text{INFB} \) \( \text{assms(3)} \) have \( \ldots \approx |A| \) using \( \text{InfCard_square_eqpoll} \) by auto
finally have \( A \cup B \prec Q \) by auto

moreover
\[
\begin{align*}
\text{assume } & \text{AS: } |A| = |B| \\
\text{with } & \text{D have } \text{INFB: } \text{InfCard}(|A|) \text{ by auto} \\
\text{then have } & \text{2} \leq |A| \text{ using } \text{nat_less_infty by simp} \\
\text{then have } & \text{AG: } 2 \leq |A| \text{ using } \text{lt_Card_imp_lesspoll Card_cardinal lesspoll_def} \\
& \text{by auto} \\
\text{from } & \text{B(1) have } |A| = A \text{ by simp} \\
\text{also from } & \text{assms(1) have } \ldots \prec Q \text{ by auto} \\
\text{finally have } & A \cup B \prec Q \text{ by simp} \\
\end{align*}
\]

}
by auto
} ultimately show A ∪ B ≺ Q by auto
qed

41.3 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}_{choice holds for subsets}_} where
{the axiom of} $Q$ {choice holds for subsets} $K$ ≡ Card($Q$) ∧ (∀ M N. (M ≲ $Q$ ∧ (∀ t∈M. Nt≠0 ∧ Nt⊆K)) → (∃ f. f:Pi(M,λt. Nt) ∧ (∀ t∈M. ft∈Nt)))

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}_{choice holds}} where
{the axiom of} $Q$ {choice holds} ≡ Card($Q$) ∧ (∀ M N. (M ≲ $Q$ ∧ (∀ t∈M. Nt≠0)) → (∃ f. f:Pi(M,λt. Nt) ∧ (∀ t∈M. ft∈Nt)))

The axiom of finite choice always holds.

definition finite_choice:
assumes n∈nat
shows {the axiom of} n {choice holds}
proof -
note assms(1)
moreover
{ fix M N assume M≤0 ∀ t∈M. Nt≠0
  then have M=0 using lepoll_0_is_0 by auto
  then have {(t,0). t∈M}:Pi(M,λt. Nt) unfolding Pi_def domain_def function_def Sigma_def by auto
  moreover from <M=0> have ∀ t∈M. {(t,0). t∈M}t∈Nt by auto
  ultimately have (∃ f. f:Pi(M,λt. Nt) ∧ (∀ t∈M. ft∈Nt)) by auto
}
  then have (∀ M N. (M ≤0 ∧ (∀ t∈M. Nt≠0)) → (∃ f. f:Pi(M,λt. Nt) ∧ (∀ t∈M. ft∈Nt))) by auto
  then have {the axiom of} 0 {choice holds} using AxiomCardinalChoiceGen_def
  nat_into_Card
  by auto

440
moreover { 
  fix x 
  assume as: x ∈ nat {the axiom of} x {choice holds} 
  
  { 
    fix M N assume ass: M ≲ succ(x) ∀ t ∈ M. N t ≠ 0 
    
    assume M ≲ x 
    from as(2) ass(2) have 
      (M ≲ x ∧ (∀ t ∈ M. N t ≠ 0)) → (∃ f ∈ Pi(M, λ t. N t) ∧ 
        (∀ t ∈ M. f t ∈ N t)) 
      unfolding AxiomCardinalChoiceGen_def by auto 
      with <M ≲ x> ass(2) have (∃ f ∈ Pi(M, λ t. N t) ∧ (∀ t ∈ M. f t ∈ N t)) 
      by auto 
  } 
  moreover 
  { 
    assume M ≲ succ(x) 
    then obtain f where f: f ∈ bij(succ(x), M) using eqpoll_sym eqpoll_def 
    by blast 
    moreover 
    have x ∈ succ(x) unfolding succ_def by auto 
    ultimately have restrict(f, succ(x) - {x}) ∈ bij(succ(x) - {x}, M - {fx}) 
    using bij_restrict_rem 
    by auto 
    moreover 
    have x ∈ x using mem_not_refl by auto 
    then have succ(x) - {x} = x unfolding succ_def by auto 
    ultimately have restrict(f, x) ∈ bij(x, M - {fx}) by auto 
    then have x ≈ M - {fx} unfolding eqpoll_def by auto 
    then have M - {fx} ≲ x using eqpoll_sym by auto 
    then have M - {fx} ≲ x using eqpoll_imp_lepoll by auto 
    with as(2) ass(2) have (∃ g. g ∈ Pi(M - {fx}, λ t. N t) ∧ (∀ t ∈ M - {fx}. g t ∈ N t)) 
    unfolding AxiomCardinalChoiceGen_def by auto 
    then obtain g where g: g ∈ Pi(M - {fx}, λ t. N t) ∀ t ∈ M - {fx}. g t ∈ N t 
    by auto 
    from f have ff: fx ∈ M using bij_def inj_def apply_functype by auto 
    with ass(2) have N(fx) ≠ 0 by auto 
    then obtain y where y: y ∈ N(fx) by auto 
    from g(1) have gg: g ∈ Sigma(M - {fx}, (λ N)) unfolding Pi_def by auto 
    with y ff have g ∪ ({fx, y}) ∈ Sigma(M, (λ N)) unfolding Sigma_def 
    by auto 
    moreover 
    from g(1) have dom: M - {fx} ⊆ domain(g) unfolding Pi_def by auto 
    then have M ⊆ domain(g ∪ {fx, y}) unfolding domain_def by auto 
  }
moreover 
from gg g(1) have noe: ~ (∃ t. ⟨fx, t⟩ ∈ g) and function(g)
unfolding domain_def Pi_def Sigma_def by auto
with dom have fg: function g ∪ {⟨fx, y⟩}
by blast
ultimately have PP: g ∪ {⟨fx, y⟩} ∈ Pi(M, λt. N t) unfolding Pi_def
by auto
have ⟨fx, y⟩ ∈ g ∪ {⟨fx, y⟩} by auto
from this fg have ⟨g ∪ {⟨fx, y⟩}⟩ (fx) = y by (rule function_apply_equality)
with y have (g ∪ {⟨fx, y⟩})(fx) ∈ N(fx) by auto
moreover
{ fix t assume A: t ∈ M-{fx}
with g(1) have ⟨t, gt⟩ ∈ g using apply_Pair by auto
then have ⟨g ∪ {⟨fx, y⟩}⟩ t = gt using apply_equality PP by auto
with A have ⟨g ∪ {⟨fx, y⟩}⟩ t ∈ Nt using g(2) by auto }
ultimately have ∀ t ∈ M. (g ∪ {⟨fx, y⟩}) t ∈ Nt by auto
with PP have ∃ g. g ∈ Pi(M, λt. N t) ∧ (∀ t ∈ M. gt ∈ Nt) by auto
ultimately have ∃ g. g ∈ Pi(M, λt. N t) ∧ (∀ t ∈ M. g t ∈ N t) using as(1) ass(1)
lepoll_suc_disj by auto
then have ∀ M. N. M ≤ₚ succ(x) ∧ (∀ t ∈ M. N t ≠ 0) → (∃ g. g ∈ Pi(M, λt. N t) ∧ (∀ t ∈ M. g t ∈ N t))
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} ←→ (∀ 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 ≤ Q1 Card(Q)
shows {the axiom of} Q1 {choice holds} → ({the axiom of} Q {choice holds}) using asms
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 \( f \in \text{surj}(A,B) \) \( A \subseteq \text{Ord}(Q) \)

shows \( B \preceq A \)

**proof**

let \( g = \{ (m, \mu j. j \in A \land f(j)=m) \mid m \in B \} \)

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 \( \exists y \in B. (y, x) \in g \) 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 \( P: z \in A \land f(z)=y \) \( \text{Ord}(z) \) unfolding surj_def
    by auto
  with \( T \) have \( x \in A \land f(x)=y \) using LeastI by simp
  hence \( x \in A \) by simp
}

hence \( \forall w \in B. \forall x \in B. g(w) = g(x) \rightarrow w = x \)

by auto

with fun show \( g \in \text{inj}(B,A) \) unfolding inj_def by auto

qed

then show thesis unfolding lepoll_def by auto
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.

\[ \text{theorem surj_fun_inv_2:} \]
\[ \text{assumes f:surj(A,B) A}\subseteq\text{Ord}(Q) \]
\[ \text{shows B}\subseteq\text{A} \]

\[ \text{proof-} \]
\[ \text{from assms(2) obtain h where h_def: h}\in\text{inj(A,Q) using lepoll_def by auto} \]
\[ \text{then have bij: h}\in\text{bij(A,range(h)) using inj_bij_range by auto} \]
\[ \text{then obtain h1 where h1}\in\text{bij(range(h),A) using bij_converse_bij by auto} \]
\[ \text{then have h1}\in\text{surj(range(h),A) using bij_def by auto} \]
\[ \text{with assms(1) have (f O h1)}\in\text{surj(range(h),B) using comp_surj by auto} \]
\[ \text{moreover} \]
\[ \{ \]
\[ \text{fix x} \]
\[ \text{assume p: x}\in\text{range(h)} \]
\[ \text{from bij have h}\in\text{surj(A,range(h)) using bij_def by auto} \]
\[ \text{with p obtain q where q}\in\text{A and h(q)}=x \text{ using surj_def by auto} \]
\[ \text{then have x}\in\text{Q using h_def inj_def by auto} \]
\[ \} \]
\[ \text{then have range(h)}\subseteq\text{Q by auto} \]
\[ \text{ultimately have B}\subseteq\text{range(h) using surj_fun_inv assms(3) by auto} \]
\[ \text{moreover have range(h)}\approx\text{A using bij eqpoll_def eqpoll_sym by blast} \]
\[ \text{ultimately show B}\subseteq\text{A using lepoll_eq_trans by auto} \]

\[ \text{qed} \]

42 Groups 4

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.

42.1 Conjugation of subgroups

The conjugate of a subgroup is a subgroup.
theorem (in group0) semigr0:
  shows semigr0(G,P)
  unfolding semigr0_def using groupAssum IsAgroup_def IsAmonoid_def by auto

theorem (in group0) conj_group_is_group:
  assumes IsAsubgroup(H,P) g∈G
  shows IsAsubgroup({g·(h·g⁻¹). h∈H},P)
proof-
  have sub:H⊆G using assms(1) group0_3_L2 by auto
  from assms(2) have g⁻¹∈G using inverse_in_group by auto
  { fix r assume r∈{g·(h·g⁻¹). h∈H}
    from h(1) have h⁻¹∈H using group0_3_T3A assms(1) by auto
    then have h⁻¹∈G using inverse_in_group by auto
    with <g⁻¹∈G> have (h⁻¹·(g⁻¹))∈G using group_op_closed by auto
    from h(2) have r⁻¹=(g·(h·(g⁻¹)))⁻¹ by auto moreover
    from <h∈G> <g⁻¹∈G> have s:h·(g⁻¹)∈G using group_op_closed by blast
    ultimately have r⁻¹=(h·(g⁻¹))⁻¹·(g⁻¹) using group_inv_of_two[of assms(2)]
  } by auto
  moreover
  from s assms(2) h(2) have r:r∈G using group_op_closed by auto
  have (h·(g⁻¹))⁻¹=(g⁻¹)⁻¹·h⁻¹ using group_inv_of_two[of <h∈G><g⁻¹∈G>]
by auto
  moreover have (g⁻¹)⁻¹·g using group_inv_of_inv[of assms(2)] by auto
  ultimately have r⁻¹=(h⁻¹·(g⁻¹)) by auto
  then have r⁻¹·g·(h⁻¹·(g⁻¹)) using group_oper_assoc[of assms(2) <h⁻¹∈G><g⁻¹∈G>]
by auto
  with <h⁻¹∈H> r have r⁻¹∈{g·(h·g⁻¹). h∈H} r∈G by auto
  then have ∀r∈{g·(h·g⁻¹). h∈H}. r⁻¹∈{g·(h·g⁻¹). h∈H} and {g·(h·g⁻¹). h∈H} ⊆ G by auto moreover
  { fix s t assume s:s∈{g·(h·g⁻¹). h∈H} and t:t∈{g·(h·g⁻¹). h∈H}
    then obtain hs ht where hs:hs∈H s=g·(hs·(g⁻¹)) and ht:ht∈H t=g·(ht·(g⁻¹))
  } by auto
  from hs(1) have hs∈G using sub by auto
  then have g·hs∈G using group_op_closed assms(2) by auto
  then have (g·hs)⁻¹∈G using inverse_in_group by auto
  from ht(1) have ht∈G using sub by auto
  with <g⁻¹∈G> have ht·(g⁻¹)∈G using group_op_closed by auto
  from hs(2) ht(2) have s·t=(g·(hs·(g⁻¹)))·(g·(ht·(g⁻¹))) by auto moreover
  have g·(hs·(g⁻¹))=g·hs·(g⁻¹) using group_oper_assoc[of assms(2) <hs∈G><g⁻¹∈G>]
by auto
  then have (g·(hs·(g⁻¹)))·(g·(ht·(g⁻¹)))=(g·hs·(g⁻¹))·(g·(ht·(g⁻¹))) by auto

445
then have \((g\cdot(h\cdot(g^{-1})))\cdot(g\cdot(h\cdot(g^{-1})))=(g\cdot(h\cdot(g^{-1})))\cdot(g^{-1}\cdot(h\cdot(g^{-1}))\)
using group_inv_of_inv[OF assms(2)] by auto
also have \(\cdots=g\cdot(hs\cdot(h\cdot(g^{-1})))\) using group0_2_L14A(2)[OF \((g\cdot(hs))^{-1}\in G\y g^{-1}\in G\y h\cdot(g^{-1})\in G\)]
group_inv_of_inv[OF \((g\cdot(hs))\in G\)]
by auto
ultimately have \(s\cdot=g\cdot(hs\cdot(h\cdot(g^{-1})))\) by auto moreover
have \((hs\cdot(h\cdot(g^{-1})))=(hs\cdot h\cdot(g^{-1}))\) using group_oper_assoc[OF \(hs\in G\y h\cdot(g\cdot(h^{-1}))\in G\)]
by auto moreover
have \((g\cdot(h\cdot(g^{-1})))\cdot(g\cdot(h\cdot(g^{-1})))\) using group_oper_assoc[OF \((g\cdot(G\y hs\in G\y h\cdot(g\cdot(h^{-1}))\in G\)]
by auto
ultimately have \(s\cdot=g\cdot((hs\cdot h\cdot(g^{-1})))\) by auto moreover
from \(hs(1)\) \(ht(1)\) have \(hs\cdot h\cdot G\) using assms(1) group0_3_L6 by auto
ultimately have \(s\cdot\in\{g\cdot(h\cdot(g^{-1})\}. h\in H\}\) 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 \(\{g\cdot(1\cdot g^{-1})\}. h\in H\}\) by auto
then have \(\{g\cdot(h\cdot(g^{-1})\}. h\in H\}\neq 0\) by auto ultimately
show thesis using group0_3_T3 by auto
qed

Every set is equipollent with its conjugates.

**Theorem (in group0) conj_set_is_eqpoll:**
assumes \(H\subseteq G\) \(g\in G\)
shows \(H=\{g\cdot(h\cdot(g^{-1})\}. h\in H\}\)
**proof**
have fun:\(\{h\cdot g\cdot(h^{-1})\}. h\in H\}:H\rightarrow\{g\cdot(h\cdot(g^{-1})\}. h\in H\}\) unfolding Pi_def function_def
domain_def by auto
\{
fix \(h1\) \(h2\) assume \(h1\in H\) \(h2\in H\)\(\{h\cdot g\cdot(h^{-1})\}. h\in H\)\(h1=\{h\cdot g\cdot(h^{-1})\}. h\in H\)\(h2\)
with fun have \(g\cdot(h\cdot(g^{-1}))=g\cdot(h\cdot(g^{-1}))\cdot g\cdot(h\cdot(g^{-1}))\)
\(Ghuman\h2\cdot Ghuman\h1\cdot Ghuman\h2\in Ghuman\) using apply_equality
assms(1)
group_op_closed[OF \(_\cdot\_\) inverse_in_group[OF assms(2)]\)] by auto
then have \(h\cdot(g\cdot(g^{-1}))\cdot h\cdot(g\cdot(g^{-1}))\) using group0_2_L19(2)[OF \(\cdot(h\cdot(g\cdot(g^{-1}))\cdot h\cdot(g\cdot(g^{-1}))\) assms(2)]
by auto
then have \(h1\cdot h2\) using group0_2_L19(1)[OF \(h1\in G\y h2\in G\) inverse_in_group[OF assms(2)]\)] by auto
\}
then have \(\forall h1\in H\). \(\forall h2\in H\). \(\{h\cdot g\cdot(h^{-1})\}. h\in H\)\(h1=\{h\cdot g\cdot(h^{-1})\}. h\in H\)\(h2\)
\(\mapsto h1\cdot h2\) by auto
with fun have \(\{h\cdot g\cdot(h^{-1})\}. h\in H\}\in inj(H,\{g\cdot(h\cdot(g^{-1})\}. h\in H\}\)
unfolding inj_def by auto moreover
\{
fix \(gh\) assume \(gh\in \{g\cdot(h\cdot(g^{-1})\}. h\in H\}\)
then obtain \(h\) where \(h\in H\) \(gh\cdot g\cdot(h\cdot(g^{-1}))\) by auto
then have \(\{h\cdot gh\in \{g\cdot(h\cdot(h^{-1})\}. h\in H\}\) by auto
then have \(\{h\cdot(g\cdot(h^{-1})\}. h\in H\}gh=gh\) using apply_equality fun by auto
with \(\cdot(h\cdot(g\cdot(h^{-1})\}. h\in H\}gh=gh\) by auto
\}

446
Every normal subgroup contains its conjugate subgroups.

**Theorem (in group0) norm_group_cont_conj:**

- **Assumes:** IsAnormalSubgroup(G,P,H) \( g \in G \)
- **Shows:** \{g \cdot (h \cdot g^{-1}) \cdot h \in H\} \subseteq H

**Proof:**

{ 
  \{ 
    \text{fix } r \text{ assume } r \in \{g \cdot (h \cdot g^{-1}) \cdot h \in H\} 
  \}
  \text{then show } \{g \cdot (h \cdot g^{-1}) \cdot h \in H\} \subseteq H 
}

qed

If a subgroup contains all its conjugate subgroups, then it is normal.

**Theorem (in group0) cont_conj_is_normal:**

- **Assumes:** IsAsubgroup(H,P) \( \forall \ g \in G. \ \{g \cdot (h \cdot g^{-1}) \cdot h \in H\} \subseteq H \)
- **Shows:** IsAnormalSubgroup(G,P,H)

**Proof:**

{ 
  \{ 
    \text{fix } g \text{ assume } h \in H \ g \in G 
  \}
  \text{with } \text{assms(2)} \text{ have } g^{-1} \in G \text{ using } inverse_in_group \text{ by auto} 
  \text{moreover have } h \in G \ g^{-1} \in G \text{ using } group_oper_assoc \text{ assms(1) by auto} 
  \text{ultimately have } g \cdot h \cdot g^{-1} \in H \text{ using } group_oper_assoc \text{ 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) \( \forall M. \ IsAsubgroup(M,P) \land H \approx M \rightarrow M=H \)
- **Shows:** IsAnormalSubgroup(G,P,H)

**Proof:**

{ 
  \text{fix } g \text{ assume } g : g \in G 
}

447
with assms(1) have IsAsubgroup\({g\cdot(h\cdot g^{-1}). h\in H}, P\) using conj_group_is_group by auto
moreover
from assms(1) g have H\approx\{g\cdot(h\cdot g^{-1}). h\in H\} using conj_set_is_eqpoll
group0_3_L2 by auto
ultimately have \{g\cdot(h\cdot g^{-1}). h\in H\}=H using assms(2) 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 then have P(a,b)=1-1 by auto then have P(a,b)=1 using group0_2_L2 by auto then have P(a,b)\in\{1\} by auto \}
then have \{1\}\{is closed under\}P unfolding IsOpClosed_def by auto moreover
\{ fix a assume a\in\{1\} then have a=1 by auto then have a^{-1}=a^{-1} by auto then have a^{-1}=1 using group_inv_of_one by auto then have a^{-1}\in\{1\} by auto \}
then have \forall a\in\{1\}. a^{-1}\in\{1\} by auto ultimately 
have IsAsubgroup(\{1\},P) using group0_3_T3 by auto moreover
\{ fix M assume M: IsAsubgroup(M,P) \{1\}\approx M then have 1\in M M\approx\{1\} using eqpoll_sym group0_3_L5 by auto then obtain f where f\in bij(M,\{1\}) unfolding eqpoll_def by auto then have inj:f\in inj(M,\{1\}) unfolding bij_def by auto then have fun:f:M\rightarrow\{1\} unfolding inj_def by auto \}
fix b assume b\in M\neq 1 then have fb\neq f1 unfolding inj_def by auto then have False using <b\in M> <1\in M> apply_type[OF fun] by auto
} then have M=\{1\} using <1\in M> by auto
ultimately show thesis using only_one_equipoll_sub by auto

448
Since the whole group and the trivial subgroup are normal, it is natural to define simplicity of groups in the following way:

**Definition**

\[
\text{IsSimple } ([\ldots\{\text{is a simple group}\} 89)\\
\text{where } [G,f]\{\text{is a simple group}\} \equiv \text{IsAgroup}(G,f) \land (\forall M. \text{IsAnormalSubgroup}(G,f,M) \rightarrow M=G \lor M=\{\text{TheNeutralElement}(G,f)\})
\]

From the definition follows that if a group has no subgroups, then it is simple.

**Corollary** (in group0) noSubgroup_imp_simple:

assumes \( \forall H. \text{IsAsubgroup}(H,G) \rightarrow H=G \lor H=\{1\} \)

shows \([G,P]\{\text{is a simple group}\}

proof-

have IsAgroup(G,P) using groupAssum. moreover

{ fix M assume IsAnormalSubgroup(G,P,M) then have IsAsubgroup(M,P) unfolding IsAnormalSubgroup_def by auto
  with assms have M=G \lor M=\{1\} by auto
  }
  ultimately show thesis unfolding IsSimple_def by auto

qed

Since every subgroup is normal in abelian groups, it follows that commutative simple groups do not have subgroups.

**Corollary** (in group0) abelian_simple_noSubgroups:

assumes \([G,P]\{\text{is a simple group}\} P{\text{is commutative on}}G\)

shows \( \forall H. \text{IsAsubgroup}(H,P) \rightarrow H=G \lor H=\{1\} \)

proof(safe)

fix H assume A:IsAsubgroup(H,P)H \neq \{1\}

then have IsAnormalSubgroup(G,P,H) using Group_ZF_2_4_L6(1) groupAssum assms(2)
  by auto
  with assms(1) A show H=G unfolding IsSimple_def by auto

qed

42.2 Finite groups

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) r=QuotientGroupRel(G,P,H)
  shows Finite(G//r)
proof -
  have fun:{(g,r{g}). g∈G}→(G//r) unfolding Pi_def function_def domain_def
  by auto
  { fix C assume C:C∈G//r
    then obtain c where c:c∈C using EquivClass_1_L5[OF Group_ZF_2_4_L1[OF assms(2)]
    by auto
    with C have r{c}=C using EquivClass_1_L2[OF Group_ZF_2_4_L3] assms(2,3)
    by auto
  }
  with fun have surj:{(g,r{g}). g∈G}∈surj(G,G//r) unfolding surj_def
  by auto
  moreover
  have sub2:r{g}⊆G using EquivClass_1_L2[OF Group_ZF_2_4_L3] assms(2,3)
  by auto
  have restr:(g,r{g}). g∈G using apply_equality fun
  by auto
  then have ∃c∈G. (g,r{g}). g∈G c=c using apply_equality fun
  by auto
  then have restr(RightTranslation(G,P,(g−1)·g2))(r{g1})∈bij(r{g1},RightTranslation(G,P,(g−1)·g2)(r{g1})) unfolding eqpoll_def
  by auto
  ultimately have restr(RightTranslation(G,P,(g−1)·g2),r{g1})∈bij(r{g1},RightTranslation(G,P,(g−1)·g2)(r{g1})) unfolding eqpoll_def
  by auto

All the cosets are equipollent.

lemma (in group0) cosets_equipoll:
  assumes IsAsubgroup(H,P) r=QuotientGroupRel(G,P,H) g1∈G g2∈G
  shows r{g1}≈r{g2}
proof -
  have sub2:r{g2}⊆G using inverse_in_group group_op_closed
  by auto
  then have RightTranslation(G,P,(g1−1)·g2)∈bij(G,G) using trans_bij(1)
  by auto
  moreover
  have sub2:r{g2}⊆G using inverse_in_group group_op_closed
  by auto
  then have RightTranslation(G,P,(g1−1)·g2)∈bij(G,G) using trans_bij(1)
  by auto
  ultimately have restr(RightTranslation(G,P,(g1−1)·g2),r{g1})∈bij(r{g1},RightTranslation(G,P,(g1−1)·g2)(r{g1})) unfolding eqpoll_def
  by auto
  then have restr(RightTranslation(G,P,(g1−1)·g2),r{g1})∈bij(r{g1},RightTranslation(G,P,(g1−1)·g2)(r{g1})) unfolding eqpoll_def
  by auto

450
then have $A_0 : \{ \text{RightTranslation}(G, P, (g_1^{-1}) \cdot g_2) \cdot t \in r(g_1) \}$
using func_imagedef [OF group0_5_L1(1)[OF GG] sub] by auto

\{ 
  \text{fix t assume } t \in \{ \text{RightTranslation}(G, P, (g_1^{-1}) \cdot g_2) \cdot t \in r(g_1) \} 
  then obtain q where q \cdot t = \text{RightTranslation}(G, P, (g_1^{-1}) \cdot g_2) \cdot q \cdot q \in r(g_1) 
  \}

by auto

then have $(g_1, q) \in r \cdot q \in G$ using image_iff sub by auto

then have $g_1 \cdot (q^{-1}) \in H \cdot q^{-1} \in G$ using assms(2) inverse_in_group unfolding QuotientGroupRel_def by auto

from q(1) have $t : t = q \cdot (g_1^{-1}) \cdot g_2$ using group0_5_L2(1)[OF GG] q(2)

by auto

then have $g_2 \cdot t^{-1} = g_2 \cdot (q \cdot ((g_1^{-1}) \cdot g_2)^{-1})$ by auto

then have $g_2 \cdot t^{-1} = g_2 \cdot (((g_1^{-1}) \cdot g_2)^{-1} \cdot q^{-1})$ using group_inv_of_two[OF $<q \in G> \cdot GG$]

by auto

then have $g_2 \cdot t^{-1} = g_2 \cdot (((g_2^{-1}) \cdot g_1^{-1}) \cdot q^{-1})$ using group_inv_of_two[OF inverse_in_group[OF assms(3)]

assms(4)] by auto

then have $g_2 \cdot t^{-1} = g_2 \cdot (((g_2^{-1}) \cdot g_1) \cdot q^{-1})$ using group_inv_of_inv assms(3)

by auto moreover

have $t \in G$ using $t : q \in G$ inverse_in_group[OF assms(3)] group_op_closed

by auto

have $(g_2^{-1}) \cdot g_1 \in G$ using assms(3) inverse_in_group[OF assms(4)] group_op_closed

by auto

with assms(4) $<q^{-1} \in G>$ have $g_2 \cdot (((g_2^{-1}) \cdot g_1) \cdot q^{-1}) = g_2 \cdot ((g_2^{-1}) \cdot g_1) \cdot q^{-1}$ using group_oper_assoc by auto

moreover have $g_2 \cdot ((g_2^{-1}) \cdot g_1) = g_2 \cdot (g_2^{-1}) \cdot g_1$ using assms(3) inverse_in_group[OF assms(4)] assms(4)

by auto

then have $g_2 \cdot ((g_2^{-1}) \cdot g_1) = g_1$ using group0_2_L6[OF assms(4)] group0_2_L2 assms(3) by auto ultimately

have $g_2 \cdot t^{-1} = g_1 \cdot q^{-1}$ by auto

with $<g_1(q^{-1}) \in H>$ have $g_2 \cdot t^{-1} \in H$ by auto

then have $(g_2, t) \in r$ using assms(2) unfolding QuotientGroupRel_def using assms(4) $<t \in G>$ by auto

then have $t \in r(g_2)$ using image_iff assms(4) by auto

\}

then have $A_1 : \{ \text{RightTranslation}(G, P, (g_1^{-1}) \cdot g_2) \cdot t \in r(g_1) \} \subseteq r(g_2)$ by auto

\{ 
  \text{fix t assume } t \in r(g_2) 
  then have $(g_2, t) \in r \cdot t \in G$ using sub2 image_iff by auto

  then have $H : g_2 \cdot t^{-1} \in H$ using assms(2) unfolding QuotientGroupRel_def by auto

  then have $G : g_2 \cdot t^{-1} \in G$ using group0_3_L2 assms(1) by auto

  then have $g_1 \cdot (g_1^{-1} \cdot (g_2 \cdot t^{-1})) = g_1 \cdot g_1^{-1} \cdot (g_2 \cdot t^{-1})$ using group_oper_assoc[OF assms(3) inverse_in_group[OF assms(3)]]

  by auto

  then have $g_1 \cdot (g_1^{-1} \cdot (g_2 \cdot t^{-1})) = g_2 \cdot t^{-1}$ using group0_2_L6[OF assms(3)]

  group0_2_L2 G by auto

  with $H$ have $H \cdot H : g_1 \cdot (g_1^{-1} \cdot (g_2 \cdot t^{-1})) \in H$ by auto

451
have \[ \text{GGG: } t \cdot g_2^{-1} \in G \text{ using } \langle t \in G \rangle \text{ inverse}_{-\text{in_group}}[\text{OF assms(4)}] \text{ group}_{-\text{op_closed}} \]
by auto
have \( (t \cdot g_2^{-1})^{-1} = g_2^{-1} \cdot t^{-1} \) using \( \text{group}_{-\text{inv_of_two}}[\text{OF } \langle t \in G \rangle \text{ inverse}_{-\text{in_group}}[\text{OF assms(4)}]] \) by auto
also have \( ... \cdot g_2 \cdot t^{-1} \) using \( \text{group}_{-\text{inv_of_two}}[\text{OF assms(4)}] \) by auto
ultimately have \( (t \cdot g_2^{-1})^{-1} = g_2^{-1} \cdot (g_2 \cdot t^{-1}) \) by auto
then have \( ((t \cdot g_2^{-1}) \cdot g_1)^{-1} = g_1^{-1} \cdot (g_2 \cdot t^{-1}) \) using \( \text{group}_{-\text{inv_of_two}}[\text{OF GGG assms(3)}] \) by auto
then have \( \text{HHH: } g_1 \cdot ((t \cdot g_2^{-1}) \cdot g_1)^{-1} \in H \) using \( \text{HH} \) by auto
have \( (t \cdot g_2^{-1}) \cdot g_1 \in G \) using \( \text{assms(3)} \) \( \langle t \in G \rangle \text{ inverse}_{-\text{in_group}}[\text{OF assms(4)}] \)
by auto
also have \( ... \cdot t \cdot (g_2^{-1} \cdot (g_1^{-1} \cdot g_2)) \) using \( \text{group}_{-\text{oper_assoc}}[\text{OF } \langle t \in G \rangle \text{ group}_{-\text{op_closed}}[\text{OF inverse}_{-\text{in_group}}[\text{OF assms(4)}] \text{ assms(3)}] \text{ GGG}] \)
by auto
also have \( ... \cdot t \cdot (g_2^{-1} \cdot (g_1^{-1} \cdot g_2)) \) using \( \text{group}_{-\text{oper_assoc}}[\text{OF inverse}_{-\text{in_group}}[\text{OF assms(4)}] \text{ assms(3)}] \text{ GGG}] \)
by auto
also have \( ... \cdot t \cdot g_2^{-1} \cdot (g_1^{-1} \cdot g_2) \) using \( \text{group}_{-\text{oper_assoc}}[\text{OF assms(3)}] \)
inverse_{-\text{in_group}}[\text{OF assms(4)}] by auto
also have \( ... \cdot t \cdot g_2^{-1} \cdot (g_1^{-1} \cdot g_2) \) using \( \text{group}_{-\text{oper_assoc}}[\text{OF assms(4)}] \text{ group0}_{-\text{L2}[2]}[\text{OF assms(3)}] \text{ group0}_{-\text{L2}[2]}[\text{OF assms(4)}] \)
\( \langle t \cdot g_2^{-1} \rangle \cdot g_1 \in G \) by auto
ultimately have \( t \cdot g_2^{-1} \cdot (g_1^{-1} \cdot g_2) = t \) by auto
then have \( \text{Right}_{-\text{Translation}}(G,P,(g_1^{-1} \cdot g_2) \cdot t \cdot g_2^{-1} \cdot g_1) = t \) using \( \text{group0}_{-\text{L2}[2]}(1)[\text{OF GGG}] \) \( \langle t \cdot g_2^{-1} \rangle \cdot g_1 \in G \) by auto
then have \( t \in \{ \text{Right}_{-\text{Translation}}(G,P,(g_1^{-1} \cdot g_2) \cdot t \cdot r \cdot g_1) \} \) using \( \text{rg1} \)
by force
\)
then have \( r(g_2) \subseteq \{ \text{Right}_{-\text{Translation}}(G,P,(g_1^{-1} \cdot g_2) \cdot t \cdot r \cdot g_1) \} \) by blast
with \( \text{A1} \) have \( r(g_2) = \{ \text{Right}_{-\text{Translation}}(G,P,(g_1^{-1} \cdot g_2) \cdot t \cdot r \cdot g_1) \} \) by auto
with \( \text{A0} \) show thesis by auto
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.

\textbf{theorem (in group0)} \text{Lagrange}:

assumes \( \text{Finite}(G) \) \( \text{IsAsubgroup}(H,P) \) \( r = \text{QuotientGroupRel}(G,P,H) \)
shows \( |G| = |H| \) #* \( |G//r| \)

\textbf{proof -}

have \( \text{Finite}(G//r) \) using \( \text{assms finite_cosets by auto moreover} \)
have \( \text{un: } \bigcup (G//r) = G \) using \( \text{Union}_{-\text{quotient}} \text{ Group}_{-2}_{-2}_{-4}_{-3}[\text{assms(2,3)}] \) by auto
then have \( \text{Finite}(\bigcup (G//r)) \) using \( \text{assms(1)} \) by auto moreover
have \( \forall c1 \in (G//r). \forall c2 \in (G//r). c1 \neq c2 \rightarrow c1 \cap c2 = 0 \) using \( \text{quotient}_{-\text{disj}}[\text{OF} \)

452
Given a subset of a group, we can ask ourselves which is the smallest group that contains that set; if it even exists.

**Lemma (in group0) inter_subgroups:**

assumes \( \forall H \in \mathcal{H}. \text{IsAsubgroup}(H, P) \neq 0 \)

shows \( \text{IsAsubgroup}(\bigcap \mathcal{H}, P) \)

**Proof:**
from assms have \( 1 \in \bigcap \mathcal{H} \) using group0_3_L5 by auto
then have \( \bigcap \mathcal{H} \neq 0 \) by auto moreover
{  
  fix \( A, B \) assume \( A \in \bigcap \mathcal{H}, B \in \bigcap \mathcal{H} \)
  then have \( \forall H \in \mathcal{H}. \ A \in H \land B \in H \) by auto
  then have \( \forall H \in \mathcal{H}. \ A \cdot B \in H \) using assms(1) group0_3_L6 by auto
  then have \( A \cdot B \in \bigcap \mathcal{H} \) using assms(2) by auto
}
then have \( (\bigcap \mathcal{H}) \{ \text{is closed under} \} P \) using IsOpClosed_def by auto moreover
{  
  fix \( A \) assume \( A \in \bigcap \mathcal{H} \)
  then have \( \forall H \in \mathcal{H}. \ A \in H \) by auto
  then have \( \forall H \in \mathcal{H}. \ A^{-1} \in H \) using assms(1) group0_3_T3A by auto
  then have \( A^{-1} \in \bigcap \mathcal{H} \) using assms(2) by auto
}
then have \( \forall A \in \bigcap \mathcal{H}. \ A^{-1} \in \bigcap \mathcal{H} \) by auto moreover
have \( \bigcap \mathcal{H} \subseteq G \) using assms(1, 2) group0_3_L2 by force
ultimately show thesis using group0_3_T3 by auto
As the previous lemma states, the subgroup that contains a subset can be defined as an intersection of subgroups.

**definition** (in `group0`)  
```
SubgroupGenerated (\_G 80)  
where \( \langle X \rangle_G \equiv \bigcap \{ H \in \text{Pow}(G). \ X \subseteq H \wedge \text{IsAsubgroup}(H,P) \} \)
```

**theorem** (in `group0`) `subgroupGen_is_subgroup`:  
assumes `X \subseteq G`  
shows `IsAsubgroup(\langle X \rangle_G,P)`  
proof  
- have `\text{restrict}(P,G \times G)=P` using `\text{group_oper_fun restrict_idem}` unfolding `Pi_def` by auto  
then have `IsAsubgroup(G,P)` unfolding `IsAsubgroup_def` using `\text{groupAssum}` by auto  
with assms have `G \in \{ H \in \text{Pow}(G). \ X \subseteq H \wedge \text{IsAsubgroup}(H,P) \}` by auto  
then have `(H \in \text{Pow}(G). \ X \subseteq H \wedge \text{IsAsubgroup}(H,P)) \neq 0` by auto  
then show thesis using `\text{inter_subgroups unfolding SubgroupGenerated_def}` by auto  
qed

### 42.4 Homomorphisms

A homomorphism is a function between groups that preserves group operations.

**definition**  
```
\text{Homomor}(\_\{ \text{is a homomorphism}\}(\_,\_)\rightarrow\{\_,\_\} 85)  
where \( \text{IsAgroup}(G,P) \implies \text{IsAgroup}(H,F) \implies \text{Homomor}(f,G,P,H,F) \equiv \forall \, g1\in G. \forall \, g2\in G. \ f(P\langle g1,g2 \rangle)=F\langle fg1,fg2 \rangle \)
```

Now a lemma about the definition:

**lemma** `homomor_eq`:  
assumes `IsAgroup(G,P) IsAgroup(H,F) Homomor(f,G,P,H,F) g1\in G g2\in G`  
suggests `f(P\langle g1,g2 \rangle)=F\langle fg1,fg2 \rangle`  
using assms `\text{Homomor_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}  
where \( \text{End}(G,P) \equiv \{ f:G\rightarrow G. \ \text{Homomor}(f,G,P,G,P) \} \)
```

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\in \text{End}(G,P) f2\in \text{End}(G,P)`  
suggests `\text{Composition}(G)\langle f1,f2 \rangle\in \text{End}(G,P)`  
proof-
from assms have \text{fun}: f_1: G \to G\text{ unfolding End_def by auto}
then have \text{fun2}: f_1 \circ f_2: G \to G\text{ using comp_fun by auto}
have \text{comp}: \text{Composition}(G)(f_1, f_2) = f_1 \circ f_2\text{ using func_ZF_5_L2 fun by auto}
\{
  fix g_1 g_2 assume AS2: g_1 \in G g_2 \in G
  then have \text{gig2}: g_1 \cdot g_2 \in G\text{ using group_op_closed by auto}
  from \text{fun2} have (f_1 \circ f_2)(g_1 \cdot g_2) = f_1(f_2(g_1 \cdot g_2))\text{ using comp_fun_apply}
  \}
  \text{fun(2) gig2} by auto
also have \ldots = f_1((f_2g_1) \cdot (f_2g_2))\text{ using assms(2) unfolding End_def Homomor_def[OF groupAssum groupAssum]}
  using AS2 by auto
moreover
  have f_2g_1 \in G f_2g_2 \in G\text{ using fun(2) AS2 apply_type by auto ultimately}
  have (f_1 0 f_2)(g_1 \cdot g_2) = (f_1(f_2g_1)) \cdot (f_1(f_2g_2))\text{ using assms(1) unfolding End_def Homomor_def[OF groupAssum groupAssum]}
  using AS2 by auto
  then have (f_1 0 f_2)(g_1 \cdot g_2) = ((f_1 0 f_2)g_1) \cdot ((f_1 0 f_2)g_2)\text{ using comp_fun_apply}
  \}
  \text{fun(2) AS2} by auto
}\then have \forall g_1 \in G. \forall g_2 \in G. (f_1 0 f_2)(g_1 \cdot g_2) = ((f_1 0 f_2)g_1) \cdot ((f_1 0 f_2)g_2)\text{ by auto}
then have (f_1 0 f_2) \in \text{End}(G, P)\text{ unfolding End_def Homomor_def[OF groupAssum groupAssum] using \text{fun2} by auto}
  with \text{comp} show \text{Composition}(G)(f_1, f_2) \in \text{End}(G, P)\text{ by auto qed}

\text{theorem}\text{ (in group0) end_comp_monoid:}
shows \text{IsAmonoid(End}(G, P), \text{restrict}(\text{Composition}(G), \text{End}(G, P) \times \text{End}(G, P)))\text{ and}
\text{TheNeutralElement}(\text{End}(G, P), \text{restrict}(\text{Composition}(G), \text{End}(G, P) \times \text{End}(G, P))) = \text{id}(G)
\text{proof-}
\text{have \text{fun}: id(G): G \to G unfolding id_def by auto}
\{
  fix g h assume gh \in G
  then have \text{id}: g \cdot h \in \text{id}(G) g = \text{gid}(G) h = h\text{ using group_op_closed by auto}
  then have id(G)(gh) = g \cdot h\text{ unfolding id_def by auto}
  with \text{id}(2, 3) have id(G)(gh) = (id(G)g) \cdot (id(G)h)\text{ by auto}
  \}
  \text{with \text{fun} have id(G) \in \text{End}(G, P)\text{ unfolding End_def Homomor_def[OF groupAssum groupAssum] by auto moreove}
from \text{GroupZF_2_5_L2(2)} have \text{A0: id}(G) = \text{TheNeutralElement}(G \to G, \text{Composition}(G))\text{ by auto ultimately}
have \text{A1: TheNeutralElement}(G \to G, \text{Composition}(G)) \in \text{End}(G, P)\text{ by auto}
moreover
  have \text{A2: End}(G, P) \subseteq G \to G\text{ unfolding End_def by auto moreover}
from \text{end_composition} have \text{A3: End}(G, P)\{\text{is closed under}\text{Composition}(G)}\text{ unfolding \text{IsOpClosed_def by auto}
ultimately show \text{IsAmonoid(End}(G, P), \text{restrict}(\text{Composition}(G), \text{End}(G, P) \times \text{End}(G, P)))\)
  using monoid0.group0_1_T1 unfolding monoid0_def using GroupZF_2_5_L2(1)
  by force

455
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), restrict(Composition(G), End(G,P) × End(G,P))) using group0_1_L6 by auto
qed

The set of endomorphisms is closed under pointwise addition. This is so because the group is abelian.

theorem (in group0) end_pointwise_addition:
  assumes f ∈ End(G,P) g ∈ End(G,P)
  P{is commutative on} G F = P{lifted to function space over} G
  shows F⟨f,g⟩ ∈ End(G,P)
proof-
  from assms(1,2) have fun: f ∈ G → G g ∈ G → G unfolding End_def by auto
  then have fun2: F⟨f,g⟩ : G → G using monoid0.Group_ZF_2_1_L0 group0_2_L1 assms(4) by auto
  { fix g1 g2 assume AS: g1 ∈ G g2 ∈ G
    then have elinv: g1⁻¹ ∈ G g2⁻¹ ∈ G using inverse_in_group by auto
    have (g1 · g2)⁻¹ = g2⁻¹ · g1⁻¹ using group_inv_of_two AS by auto
    then have (g1 · g2)⁻¹ = g1⁻¹ · (g2⁻¹) using assms(1) elinv unfolding IsCommutative_def by auto
  }
  with fun2 show thesis unfolding End_def Homomor_def [OF groupAssum groupAssum] by auto
qed

The inverse of an abelian group is an endomorphism.

lemma (in group0) end_inverse_group:
  assumes P{is commutative on} G
  shows GroupInv(G,P) ∈ End(G,P)
proof-
  { fix s t assume AS: s ∈ G t ∈ G
    then have elinv: s⁻¹ ∈ G t⁻¹ ∈ G using inverse_in_group by auto
    have (s⁻¹ · t⁻¹)⁻¹ = s⁻¹ · t⁻¹ using group_inv_of_two AS by auto
    then have (s⁻¹ · t⁻¹)⁻¹ = s⁻¹ · t⁻¹ using assms(1) elinv unfolding IsCommutative_def by auto
  }
  then have ∀ s ∈ G. ∀ t ∈ G. GroupInv(G,P)(s · t) = GroupInv(G,P)(s) · GroupInv(G,P)(t) by auto

456
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 multiplication.

**Theorem** (in `group0`)

End addition group:

assumes `P` (is commutative on) `G`  
`F = P` (lifted to function space over) `G`  
shows `IsAgroup(End(G,P),restrict(F,End(G,P)×End(G,P)))`  
(is commutative on) `End(G,P)`

**Proof**

- from `end_comp_monoid(1)`  
  `monoid0.group0_1_L3A` have `End(G,P)≠0`  
  unfolding `monoid0_def` by auto
- moreover have `End(G,P)⊆G→G`  
  unfolding `End_def` by auto
- moreover have `End(G,P){is closed under}F`  
  unfolding `IsOpClosed_def`  
  using `end_pointwise_addition`  
  `assms(1,2)` by auto
- moreover
  
  - fix `ff` assume `AS:ff∈End(G,P)`  
    then have `restrict(Composition(G),End(G,P)×End(G,P))(GroupInv(G,P),ff)∈End(G,P)`  
    unfolding `monoid0_def`  
    using `end_composition(1)`  
    `end_inverse_group[OF assms(1)]` by force
  
  - then have `Composition(G)(GroupInv(G,P), ff)∈End(G,P)`  
    using `AS end_inverse_group[OF assms(1)]` by auto
  
  - then have `GroupInv(G→G,F)ff∈End(G,P)`  
    using `Group_ZF_2_1_L6 assms(2)` `AS`  
    unfolding `End_def` by auto

ultimately show `IsAgroup(End(G,P),restrict(F,End(G,P)×End(G,P)))`  
using `group0.group0_3_T3 Group_ZF_2_1_T2[OF assms(2)]`  
unfolding `IsAsubgroup_def group0_def`  
by auto

show `restrict(F,End(G,P)×End(G,P)){is commutative on}End(G,P)`  
using `Group_ZF_2_1_L7[OF assms(2,1)]`  
unfolding `End_def IsCommutative_def` by auto

**QED**

**Lemma** (in `group0`)

```
with group0_2_T2 groupAssum show thesis unfolding End_def using Homomor_def by auto
```

**QED**
\textbf{fix} b c d \textbf{assume} \textbf{AS}: b ∈ \text{End}(G,P) \land c ∈ \text{End}(G,P) \land d ∈ \text{End}(G,P)

\textbf{have} \text{ig1}: \text{Composition}(G) \langle b, F \langle c, d \rangle \rangle = b \circ (F \langle c, d \rangle) \text{ using } \text{monoid0.Group_ZF_2_1_L0[OF group0_2_L1 assms(2)]}

\text{AS unfolding End_def using } \text{func_ZF_5_L2 by auto}

\text{have} \text{ig2}: F \langle \text{Composition}(G) \langle b , c \rangle, \text{Composition}(G) \langle b , d \rangle \rangle = F \langle b \circ c, b \circ d \rangle \text{ using } \text{AS unfolding End_def by force}

\text{have comp1fun: (b \circ (F \langle c, d \rangle)) : G → G using } \text{monoid0.Group_ZF_2_1_L0[OF group0_2_L1 assms(2)]}

\text{have} \text{comp2fun: (F \langle b \circ c, b \circ d \rangle): G → G using } \text{monoid0.Group_ZF_2_1_L0[OF group0_2_L1 assms(2)]}

\text{unfolding End_def using } \text{func_ZF_5_L2 by auto}

\text{take g ∈ G then have} (b \circ (F \langle c, d \rangle)) g = b((F \langle c, d \rangle) g) \text{ using } \text{comp_fun_apply monoid0.Group_ZF_2_1_L0[OF group0_2_L1 assms(2)]}

\text{AS(2,3)}

\text{ultimately have} (b \circ (F \langle c, d \rangle)) g = b(c(g) \cdot d(g)) \text{ using } \text{Group_ZF_2_1_L3[OF assms(2)] AS(2,3)}

\text{ultimately} \exists g ∈ G \text{ have} (b \circ (F \langle c, d \rangle)) g = (b(c)g) \cdot (b(d)g) \text{ using } \text{AS(1)}

\text{ultimately have} (b \circ (F \langle c, d \rangle)) g = (b \circ c)g \cdot (b \circ d)g \text{ using } \text{AS(1)}

\text{ultimately have} (b \circ (F \langle c, d \rangle)) g = F \langle (b \circ c), (b \circ d) \rangle \text{ using } \text{AS(1)}

\text{ultimately have} (b \circ (F \langle c, d \rangle)) g = F \langle (b \circ c), (b \circ d) \rangle \text{ using } \text{AS(1)}

\text{ultimately have} (b \circ (F \langle c, d \rangle)) g = F \langle (b \circ c), (b \circ d) \rangle \text{ using } \text{AS(1)}

\text{ultimately have} (b \circ (F \langle c, d \rangle)) g = F \langle (b \circ c), (b \circ d) \rangle \text{ using } \text{AS(1)}

\text{ultimately have} (b \circ (F \langle c, d \rangle)) g = F \langle (b \circ c), (b \circ d) \rangle \text{ using } \text{AS(1)}

\text{ultimately have} (b \circ (F \langle c, d \rangle)) g = F \langle (b \circ c), (b \circ d) \rangle \text{ using } \text{AS(1)}

\text{ultimately have} (b \circ (F \langle c, d \rangle)) g = F \langle (b \circ c), (b \circ d) \rangle \text{ using } \text{AS(1)}

\text{ultimately have} (b \circ (F \langle c, d \rangle)) g = F \langle (b \circ c), (b \circ d) \rangle \text{ using } \text{AS(1)}
\langle c, d \rangle = \text{restrict}(F, \text{End}(G, P) \times \text{End}(G, P)) \langle \text{restrict}(\text{Composition}(G), \text{End}(G, P) \times \text{End}(G, P)) \langle b, c \rangle, \text{restrict}(\text{Composition}(G), \text{End}(G, P) \times \text{End}(G, P)) \langle b, d \rangle \rangle

by auto

have ig1: \text{Composition}(G) \langle F \langle c, d \rangle, b \rangle = (F \langle c, d \rangle) 0 b \text{ using monoid0.Group_ZF_2_1_L0[OF group0_2_L1 assms(2)]}

AS unfolding \text{End_def} using \text{func_ZF_5_L2} by auto

have ig2: F \langle \text{Composition}(G) \langle c, b \rangle, \text{Composition}(G) \langle d, b \rangle \rangle = F \langle c 0 b, d 0 b \rangle \text{ using \text{monoid0.Group_ZF_2_1_L0}[OF group0_2_L1 assms(2)]}\text{ comp_fun AS unfolding End_def by force}

have comp1fun: ((F \langle c, d \rangle) \circ b): G \rightarrow G \text{ using \text{monoid0.Group_ZF_2_1_L0}[OF group0_2_L1 assms(2)]}\text{ comp_fun AS unfolding \text{End_def} using \text{func_ZF_5_L2} by auto}

have comp2fun: (F \langle c \circ b, d \circ b \rangle): G \rightarrow G \text{ using \text{monoid0.Group_ZF_2_1_L0}[OF group0_2_L1 assms(2)]}\text{ comp_fun AS unfolding End_def by force}

\{
fix g assume \text{gG: g \in G}

then have bg: \text{bg \in G} \text{ using AS(1) unfolding \text{End_def} using apply_type by auto}

from gG have \langle F\langle c, d \rangle \rangle 0 b = (F\langle c, d \rangle)(bg) \text{ using \text{comp_fun_apply AS(1)}}

unfolding \text{End_def by force}

also have \ldots = \langle (\text{c(bg)})\ldots (\text{d(bg)}) \rangle \text{ using Group_ZF_2_1_L3[OF assms(2)] AS(2,3) bg unfolding \text{End_def by auto}}

also have \ldots = (F\langle c 0 b, d 0 b \rangle) \text{g using \text{comp_fun_apply gG AS unfolding \text{End_def by auto}}}

also have \ldots = (\text{F\langle c \circ b, d \circ b \rangle}) \text{g using \text{gG Group_ZF_2_1_L3[OF assms(2)] AS \text{comp_fun comp_fun gG}}

AS unfolding \text{End_def by auto}

ultimately have \langle (F\langle c, d \rangle) 0 b \rangle g = (F\langle c 0 b, d 0 b \rangle) \text{g by auto}
\}

then have \forall \text{gG.} (\langle F\langle c, d \rangle \rangle 0 b) g = (F\langle c 0 b, d 0 b \rangle) \text{g by auto}

then have \langle F\langle c, d \rangle \rangle 0 b = F\langle c 0 b, d 0 b \rangle \text{ using \text{fun_extension[OF comp1fun comp2fun] by auto}}

with ig1 ig2 have Composition(G) \langle F \langle c, d \rangle, b \rangle = F \langle \text{Composition}(G) \langle c, b \rangle, \text{Composition}(G) \langle d, b \rangle \rangle \text{ by auto moreover}

have \text{F \langle c, d \rangle = \text{restrict}(F, \text{End}(G, P) \times \text{End}(G, P)) \langle c, d \rangle using AS(2,3) \text{ restrict by auto moreover}}

have Composition(G) \langle c, b \rangle = \text{restrict}(\text{Composition}(G), \text{End}(G, P) \times \text{End}(G, P))\langle c, \text{Composition}(G) \langle d, b \rangle = \text{restrict}(\text{Composition}(G), \text{End}(G, P) \times \text{End}(G, P))\langle d, b \rangle = \text{restrict \text{AS by auto moreover}}

have \text{Composition(G) \langle F \langle c, d \rangle, b \rangle = \text{restrict}(\text{Composition}(G), \text{End}(G, P) \times \text{End}(G, P)) \langle F \langle c, d \rangle, b \rangle using AS(1) \text{ end_pointwise_addition[OF AS(2,3) assms] by auto}}

moreover have \text{F \langle \text{Composition}(G) \langle c, b \rangle, \text{Composition}(G) \langle d, b \rangle \rangle = \text{restrict}(F, \text{End}(G, P) \times \text{End}(G, P)) \langle \text{Composition}(G) \langle c, b \rangle, \text{Composition}(G) \langle d, b \rangle \rangle using \text{end_composition[OF AS(2,1)] end_composition[OF AS(3,1)] by auto ultimately}}

have eq2: \text{restrict}(\text{Composition}(G), \text{End}(G, P) \times \text{End}(G, P)) \langle \text{restrict}(F, \text{End}(G, P) \times \text{End}(G, P)) \langle c, d, b \rangle = \text{restrict}(F, \text{End}(G, P) \times \text{End}(G, P)) \langle \text{restrict}(\text{Composition}(G), \text{End}(G, P) \times \text{End}(G, P)) \langle c, b \rangle, \text{restrict}(\text{Composition}(G), \text{End}(G, P) \times \text{End}(G, P)) \langle d, b \rangle \rangle by auto

459
with eq1 have (restrict((Composition(G),End(G,P) × End(G,P))) (b, restrict(F,End(G,P) × End(G,P))) (c, d)) = restrict(F,End(G,P) × End(G,P)) (restrict((Composition(G),End(G,P) × End(G,P))) (b, c), restrict((Composition(G),End(G,P) × End(G,P))) (c, d)) ∧ (restrict((Composition(G),End(G,P) × End(G,P))) (c, d), b) = restrict(F,End(G,P) × End(G,P)) (restrict((Composition(G),End(G,P) × End(G,P))) (c, b), restrict((Composition(G),End(G,P) × End(G,P))) (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 group0) end_is_ring:
  assumes P{is commutative on}G F = P {lifted to function space over} G shows IsAring(End(G,P),restrict(F,End(G,P) × End(G,P)),restrict((Composition(G),End(G,P) × End(G,P)))) unfolding IsAring_def using end_addition_group[OF assms] end_comp_monoid(1) distributive_comp_pointwise[OF assms] by auto

42.5 First isomorphism theorem

Now we will prove that any homomorphism \( f : G \rightarrow H \) defines a bijective homomorphism between \( G/H \) and \( f(G) \).

A group homomorphism sends the neutral element to the neutral element and commutes with the inverse.

lemma image_neutral:
proof-
  have g:TheNeutralElement(G,P)=P(\{TheNeutralElement(G,P),TheNeutralElement(G,P)\}) TheNeutralElement(G,P)∈G
    using assms(1) group0.group0_2_L2 unfolding group0_def by auto
  from g(1) have fTheNeutralElement(G,P)=f(P(\{TheNeutralElement(G,P),TheNeutralElement(G,P)\}) by auto
    also have ...=F(fTheNeutralElement(G,P),fTheNeutralElement(G,P))
      using assms(3) unfolding Homomor_def[OF assms(1,2)] using g(2) by auto
  ultimately have fTheNeutralElement(G,P)=F(fTheNeutralElement(G,P),fTheNeutralElement(G,P)) by auto moreover
  have h:fTheNeutralElement(G,P)∈H using g(2) apply_type[OF assms(4)] by auto
  then have fTheNeutralElement(G,P)=F(fTheNeutralElement(G,P),TheNeutralElement(H,F)) using assms(2) group0.group0_2_L2 unfolding group0_def by auto ultimately
  have F(fTheNeutralElement(G,P),TheNeutralElement(H,F))=F(fTheNeutralElement(G,P),fTheNeutralElement(G,P)) by auto
with h have \( \text{LeftTranslation}(H,F,f) \cdot \text{TheNeutralElement}(G,P) = \text{LeftTranslation}(H,F,H) \) using group0.group0_5_L2(2)[OF _ h] assms(2) group0.group0_2_L2 unfolding group0_def by auto

moreover have \( \text{LeftTranslation}(H,F,f) \cdot \text{TheNeutralElement}(G,P) \in \text{bij}(H,H) \)
using group0.trans_bij(2) assms(2) unfolding group0_def by auto

ultimately have \( \text{LeftTranslation}(H,F,f) \cdot \text{TheNeutralElement}(G,P) \in \text{inj}(H,H) \) by force

qed

lemma image_inv:
assumes IsAgroup(G,P) IsAgroup(H,F) Homomor(f,G,P,H,F) f:G \rightarrow H g \in G
shows f(\{GroupInv(G,P)g\}) = GroupInv(H,F) (fg)
proof
  have im:fg \in H using apply_type[OF assms(4,5)].
  have inv:GroupInv(G,P)g \in G using group0.inverse_in_group[OF _ assms(5)]
  unfolding group0_def by auto
  then have inv2:f(\{GroupInv(G,P)g\}) \in H using apply_type[OF assms(4)]
  by auto
  have fTheNeutralElement(G,P)\in f(P \cdot \langle g,\text{GroupInv}(G,P)g \rangle)
  using assms(1,5) group0.group0_2_L6 unfolding group0_def by auto
  also have \( \ldots = F(\{fg, f(\text{GroupInv}(G,P)g)\}) \) using assms(3) unfolding Homomor_def[OF assms(1,2)] using
  assms(5) inv by auto
  ultimately have \( \text{TheNeutralElement}(H,F) = F(\{fg, f(\text{GroupInv}(G,P)g)\}) \) using
  image_neutral[OF assms(1-4)] by auto
  then show thesis using group0.group0_2_L9(2)[OF _ im inv2] assms(2) unfolding group0_def by auto
qed

The kernel of an homomorphism is a normal subgroup.

theorem kerner_normal_sub:
assumes IsAgroup(G,P) IsAgroup(H,F) Homomor(f,G,P,H,F) f:G \rightarrow H
shows IsAnormalSubgroup(G,P,f-{\text{TheNeutralElement}(H,F)})
proof-
  have xy:\( \forall x \ y. \ (x, y) \in f \rightarrow (\forall y'. \ (x, y') \in f \rightarrow y = y') \) using assms(4)
  unfolding Pi_def function_def
  by force
  { fix g1 g2 assume g1\in f-\{\text{TheNeutralElement}(H,F)\}g2\in f-\{\text{TheNeutralElement}(H,F)\}
    then have \( \langle g1, \text{TheNeutralElement}(H,F) \rangle \in f \)\langle g2, \text{TheNeutralElement}(H,F) \rangle \in f
    using vimage_iff by auto
    moreover then have \( G: g1 \circ G \circ G2 \circ G \) using assms(4) unfolding Pi_def by auto
    then have \( \langle g1, fg1 \rangle \in f \)\langle g2, fg2 \rangle \in f
    using apply_Pair[OF assms(4)] by auto
  }
moreover note xy ultimately have \(fg_1 = \text{TheNeutralElement}(H,F)\) \(fg_2 = \text{TheNeutralElement}(H,F)\) by auto

moreover have \(f(P\langle g_1,g_2\rangle) = F\langle fg_1,fg_2\rangle\) using assms(3) \(G\) unfolding Homomor_def[OF assms(1,2)] by auto
 ultimately have \(f(P\langle g_1,g_2\rangle) = F(\text{TheNeutralElement}(H,F),\text{TheNeutralElement}(H,F))\) by auto
 also have \(\ldots = \text{TheNeutralElement}(H,F)\) using group0.group0_2_L2 assms(2) unfolding group0_def by auto

ultimately have \(f(P\langle g_1,g_2\rangle) = \text{TheNeutralElement}(H,F)\) by auto

moreover from \(G\) have \(P\langle g_1,g_2\rangle \in G\) using group0.group_op_closed assms(1) unfolding group0_def by auto
ultimately have \(\langle P\langle g_1,g_2\rangle,\text{TheNeutralElement}(H,F)\rangle \in f\) using apply_Pair[OF assms(4)] by force
 then have \(P\langle g_1,g_2\rangle \in f - \{\text{TheNeutralElement}(H,F)\}\) using vimage_iff by auto

then have \(f - \{\text{TheNeutralElement}(H,F)\}\) {is closed under} \(P\) unfolding IsOpClosed_def by auto
 moreover have \(A: f - \{\text{TheNeutralElement}(H,F)\} \subseteq G\) using func1_1_L3 assms(4) by auto
 moreover have \(f(\text{TheNeutralElement}(G,P)) = \text{TheNeutralElement}(H,F)\) using image_neutral assms by auto
 then have \(\langle \text{TheNeutralElement}(G,P),\text{TheNeutralElement}(H,F)\rangle \in f\) using apply_Pair[OF assms(4)]
 group0.group0_2_L2 assms(1) unfolding group0_def by force
 then have \(\text{TheNeutralElement}(G,P) \in f - \{\text{TheNeutralElement}(H,F)\}\) using vimage_iff by auto
 then have \(f - \{\text{TheNeutralElement}(H,F)\}\) \(\neq 0\) by auto moreover
 \{ fix \(x\) assume \(x \in f - \{\text{TheNeutralElement}(H,F)\}\) then have \(\langle x,\text{TheNeutralElement}(H,F)\rangle \in f\) and \(x \in G\) using vimage_iff and A by auto moreover
 from \(x\) have \(\langle x,fx\rangle \in f\) using apply_Pair[OF assms(4)] by auto ultimately have \(fx = \text{TheNeutralElement}(H,F)\) using \(xy\) by auto moreover
 have \(f(\text{GroupInv}(G,P)x) = \text{GroupInv}(H,F)(fx)\) using \(x\) image_inv assms by auto
 ultimately have \(f(\text{GroupInv}(G,P)x) = \text{GroupInv}(H,F)\) \(\text{TheNeutralElement}(H,F)\) by auto
 then have \(f(\text{GroupInv}(G,P)x) = \text{TheNeutralElement}(H,F)\) using group0.group_inv_of_one assms(2) unfolding group0_def by auto moreover
 have \(\langle \text{GroupInv}(G,P)x,\text{TheNeutralElement}(H,F)\rangle \in f\) by auto ultimately have \(\langle \text{GroupInv}(G,P)x,\text{TheNeutralElement}(H,F)\rangle \in f\) by auto
 then have \(\text{GroupInv}(G,P)x \in f - \{\text{TheNeutralElement}(H,F)\}\) using vimage_iff by auto

462
\[
\text{then have } \forall x \in f - \{\text{TheNeutralElement}(H,F)\}. \ GroupInv(G,P)x \in f - \{\text{TheNeutralElement}(H,F)\} \text{ by auto}
\]
ultimately have SS: IsAsubgroup(f - \{\text{TheNeutralElement}(H,F)\}, P) using group0.group0_3_T3
assms(1) unfolding group0_def by auto
\[
\{ 
\text{fix } g \ h \ \text{assume } AS: g \in G \ h \in f - \{\text{TheNeutralElement}(H,F)\}
\text{from } AS(1) \text{ have } im: fg \in H \text{ using assms(4) apply_type by auto}
\text{then have } iminv: \text{GroupInv}(H,F)(fg) \in H \text{ using assms(2) group0.inverse_in_group }
\text{unfolding group0_def by auto}
\text{from } AS \text{ have } h \in G \text{ and } inv: \text{GroupInv}(G,P)g \in G \text{ using A group0.inverse_in_group assms(1)}
\text{unfolding group0_def by auto}
\text{then have } P: P(h, \text{GroupInv}(G,P)g) \in G \text{ using assms(1) group0.group_op_closed}
\text{unfolding group0_def by auto}
\text{with } <g \in G> \text{ have } P(g, P(h, \text{GroupInv}(G,P)g) ) \in G \text{ using assms(1) group0.group_op_closed}
\text{unfolding group0_def by auto}
\text{then have } f(P(g, P(h, \text{GroupInv}(G,P)g) )) = f(fg, f(P(h, \text{GroupInv}(G,P)g) ))
\text{using assms(3) unfolding Homomor_def[OF assms(1,2)] using } <g \in G> P
\text{by auto}
\text{also have } \ldots = F(fg, F((fh, f(\text{GroupInv}(G,P)g)))\text{ using assms(3) unfolding Homomor_def[OF assms(1,2)]}
\text{using } <h \in G> \text{ inv by auto}
\text{also have } \ldots = F(fg, F((fh, \text{GroupInv}(H,F)(fg)))\text{ using image_inv[OF assms}
\text{<g \in G-] by auto}
\text{ultimately have } f(P(g, P(h, \text{GroupInv}(G,P)g) )) = f(fg, F((fh, \text{GroupInv}(H,F)(fg))))
\text{by auto}
\text{also have } \ldots = F(fg, \text{GroupInv}(H,F)(fg))\text{ using assms(2) im group0.group0_2_L2}
\text{unfolding group0_def}
\text{using iminv by auto}
\text{also have } \ldots = \text{TheNeutralElement}(H,F) \text{ using assms(2) group0.group0_2_L6}
\text{im}
\text{unfolding group0_def by auto}
\text{ultimately have } f(P(g, P(h, \text{GroupInv}(G,P)g) )) = \text{TheNeutralElement}(H,F)
\text{by auto moreover}
\text{from } P <g \in G> \text{ have } P(g, P(h, \text{GroupInv}(G,P)g)) \in G \text{ using group0.group_op_closed}
\text{assms(1) unfolding group0_def by auto}
\text{ultimately have } P(g, P(h, \text{GroupInv}(G,P)g) ) \in f - \{\text{TheNeutralElement}(H,F)\}
\text{using func1_i_L15[OF assms(4)]}
\text{by auto}
\}
\text{then have } \forall g \in G. \ P(g, P(h, \text{GroupInv}(G,P)g) ). \ h \in f - \{\text{TheNeutralElement}(H,F)\} \subseteq f - \{\text{TheNeutralElement}(H,F)\}
\text{by auto}
\text{then show thesis using group0.cont_conj_is_normal assms(1) SS unfolding group0_def by auto}
\]
The image of a homomorphism is a subgroup.

theorem image_sub:  
  assumes IsAgroup(G,P) IsAgroup(H,F) Homomor(f,G,P,H,F) f:G→H  
  shows IsAsubgroup(fG,F)  
proof  
  have TheNeutralElement(G,P)∈G using group0.group0_2_L2 assms(1) unfolding group0_def by auto  
  then have TheNeutralElement(H,F)∈fG using func_imagedef[OF assms(4),of G] image_neutral[OF assms] by force  
  then have fG≠0 by auto moreover  
  {  
    fix h1 h2 assume h1∈fGh2∈fG  
    then obtain g1 g2 where h1=fg1 h2=fg2 and p:g1∈Gg2∈G using func_imagedef[OF assms(4)] by auto  
    then have F(h1,h2)=F(fg1,fg2) by auto  
    also have ...=f(P(g1,g2)) using assms(3) unfolding Homomor_def[OF assms(1,2)] using p by auto  
    ultimately have F(h1,h2)=f(P(g1,g2)) by auto  
    moreover have P(g1,g2)∈G using p group0.group_op_closed assms(1) unfolding group0_def by auto  
    ultimately  
    have F(h1,h2)∈fG using func_imagedef[OF assms(4)] by auto  
  }  
  then have fG {is closed under} F unfolding IsOpClosed_def by auto  
  moreover have fG⊆H using func1_1_L6(2) assms(4) by auto moreover  
  {  
    fix h assume h∈fG  
    then obtain g where h=fg and p:g∈G using func_imagedef[OF assms(4)] by auto  
    then have GroupInv(H,F)h=GroupInv(H,F)(fg) by auto  
    then have GroupInv(H,F)h=f(GroupInv(G,P)g) using p image_inv[OF assms] by auto  
    then have GroupInv(H,F)h∈fG using p group0.inverse_in_group assms(1) unfolding group0_def  
    using func_imagedef[OF assms(4)] by auto  
  }  
  then have ∀h∈fG. GroupInv(H,F)h∈fG by auto ultimately  
  show thesis using group0.group0_3_T3 assms(2) unfolding group0_def by auto  
qed  

Now we are able to prove the first isomorphism theorem. This theorem states that any group homomorphism $f: G \to H$ gives an isomorphism between a quotient group of $G$ and a subgroup of $H$.

theorem isomorphism_first_theorem:  
  assumes IsAgroup(G,P) IsAgroup(H,F) Homomor(f,G,P,H,F) f:G→H
defines \( r \equiv \text{QuotientGroupRel}(G,P,f\{\text{TheNeutralElement}(H,F)\}) \) and 
\( PP \equiv \text{QuotientGroupOp}(G,P,f\{\text{TheNeutralElement}(H,F)\}) \)
shows \( \exists ff. \text{Homomor}(ff,G//r,PP,fG,\text{restrict}(F,(fG) \times (fG))) \wedge ff \in \text{bij}(G//r,fG) \)

proof-
let \( ff = \{ (r(g), fg) . g \in G \} \)
\{ 
  fix \( t \) assume \( t \in \{ (r(g), fg) . g \in G \} \)
  then obtain \( g \in G \) by auto
  moreover then have \( r(g) \in G//r \) unfolding \( r \_ \) def quotient \_ def by auto
  moreover from \( g \in G \) have \( fg \in fG \) using func\_imagedef[OF assms(4)] by auto
  ultimately have \( t \in (G//r) \times fG \) by auto
\}
then have \( ff \in \text{Pow}((G//r) \times fG) \) by auto
moreover have \( (G//r) \subseteq \text{domain}(ff) \) unfolding domain \_ def quotient \_ def by auto
\}
moreover {
  fix \( x \ y \ t \) assume \( A: (x,y) \in ff \) \( (x,t) \in ff \)
  then obtain \( gy \) \( gr \) where \( (x,y) = (r\{gy\}, fgy) \) \( (x,t) = (r\{gr\}, fgr) \) and \( p\): \( gr \in G \)
yielding \( gy \in G \)
by auto
  then have \( B: r\{gy\} = r\{gy\} \) \( fgy = fgr \) by auto
  from \( B(2,3) \) have \( q: y \in H \) \( t \in H \) using apply\_type \( p\) assms(4) by auto
  have \( (gy,gr) \in r \) using eq\_equiv\_class[OF \( B(1) \) \( p(1) \)] group0.Group\_ZF\_2\_4\_L3
  kerner\_normal\_sub[OF assms(1-4)]
  asssms(1) unfolding group0\_def IsAnormalSubgroup \( r \_ \) def by auto
  then have \( P(gy,\text{GroupInv}(G,P)gr) \in f\{\text{TheNeutralElement}(H,F)\} \) unfolding \( r \_ \) def QuotientGroupRel \( r \_ \) def by auto
  then have \( eq: f(P(gy,\text{GroupInv}(G,P)gr)) = \text{TheNeutralElement}(H,F) \) using func\_1\_1\_L15[OF assms(4)] by auto
  from \( B(2,3) \) have \( F(y,\text{GroupInv}(H,F)t) = F(fgy,\text{GroupInv}(H,F)(fgr)) \) by auto
  also have \( ... = f(fgy,\text{GroupInv}(G,P)gr) \) using image\_inv[OF assms(1-4)] \( p(1) \) by auto
  also have \( ... = f(P(gy,\text{GroupInv}(G,P)gr)) \) using assms(3) unfolding Homomor\_def[OF assms(1,2)] using \( p(2) \)
  group0.inverse\_in\_group asssms(1) \( p(1) \) unfolding group0\_def by auto
  ultimately have \( F(y,\text{GroupInv}(H,F)t) = \text{TheNeutralElement}(H,F) \) using eq
  by auto
  then have \( y = t \) using assms(2) group0.group0\_2\_L11A q unfolding group0\_def by auto
}\)
then have \( \forall x \ y . (x,y) \in ff \rightarrow (\forall y'. (x,y') \in ff \rightarrow y = y') \) by auto
ultimately have \( ff\_fun: ff: G//r \rightarrow fG \) unfolding Pi\_def function\_def by auto
\}
fix \( a1 \ a2 \) assume \( A: a1 \in G//ra2 \in G//r \)
then obtain \( g1 \ g2 \) where \( p\): \( g1 \in G \)
(2) \( g2 \) unfolding quotient\_def by auto
have equiv(G,r) using group0.Group\_ZF\_2\_4\_L3 kerner\_normal\_sub[OF}
assms(1-4)]
  assms(1) unfolding group0_def IsAnormalSubgroup_def r_def by auto
moreover  
  have Congruent2(r,P) using Group_ZF_2_4_L5A[OF assms(1) kerner_normal_sub[OF assms(1-4)]]
    unfolding r_def by auto
moreover  
  have PP=ProjFun2(G,r,P) unfolding PP_def QuotientGroupOp_def r_def
    by auto
moreover  
  note _ p  
  ultimately have PP⟨a1,a2⟩=r{P⟨g1,g2⟩} using group0.Group_ZF_2_2_L2
    assms(1)
    unfolding group0_def by auto
then have ⟨PP⟨a1,a2⟩,f(P⟨g1,g2⟩)⟩∈ ff using group0.group_op_closed[OF _ p] assms(1) unfolding group0_def
    by auto
then have eq:ff(PP⟨a1,a2⟩)=f(P⟨g1,g2⟩) using apply_equality ff_fun
    by auto
from _ p have r:∀a1∈G//r. ∀a2∈G//r. restrict(F,fG×fG)⟨ffa1,ffa2⟩=ff(PP⟨a1,a2⟩)
    by auto
have G:IsAgroup(G//r,PP) using Group_ZF_2_4_T1[OF assms(1) kerner_normal_sub[OF assms(1-4)]]
  unfolding r_def PP_def by auto
have H:IsAgroup(fG, restrict(F,fG×fG)) using image_sub[OF assms(1-4)] unfolding IsAsubgroup_def .
  have HOM:Homomor(ff,G//r,PP,fG,restrict(F,(fG)×(fG))) using r unfolding Homomor_def[OF G H] by auto
  
  fix b1 b2 assume AS:ffa1=ffa2=f(b1,f(b2))∈fG using apply_type AS by auto ultimately have
  restrict(F,fG×fG)⟨ffa1,ffa2⟩=ff(PP⟨a1,a2⟩) by auto
then have r:∀a1∈G//r. ∀a2∈G//r. restrict(F,fG×fG)⟨ffa1,ffa2⟩=ff(PP⟨a1,a2⟩)
  by auto
have G:IsAgroup(G//r,PP) using Group_ZF_2_4_T1[OF assms(1) kerner_normal_sub[OF assms(1-4)]]
  unfolding r_def PP_def by auto
have H:IsAgroup(fG, restrict(F,fG×fG)) using image_sub[OF assms(1-4)] unfolding IsAsubgroup_def .
  have HOM:Homomor(ff,G//r,PP,fG,restrict(F,(fG)×(fG))) using r unfolding Homomor_def[OF G H] by auto
  
  fix b1 b2 assume AS:ffa1=ffa2=f(b1,f(b2))∈fG using apply_type AS by auto ultimately have
  restrict(F,fG×fG)⟨ffa1,ffa2⟩=ff(PP⟨a1,a2⟩) by auto
then have r:∀a1∈G//r. ∀a2∈G//r. restrict(F,fG×fG)⟨ffa1,ffa2⟩=ff(PP⟨a1,a2⟩)
  by auto

As a last result, 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)IsAgroup(H,F)Homomor(f,G,P,H,F)
  shows Homomor(converse(f),H,F,G,P)
proof-
{  
  fix h1 h2 assume A:h1∈H h2∈H 
  from A(1) obtain g1 where g1:∈G fg1=h1 using assms(1) unfolding bij_def surj_def by auto 
  moreover 
  from A(2) obtain g2 where g2:∈G fg2=h2 using assms(1) unfolding bij_def surj_def by auto 
  ultimately 
  have F(fg1,fg2)=F(h1,h2) by auto 
  then have f(P(g1,g2))=F(h1,h2) using assms(2,3,4) homomor_eq g1(1) g2(1) by auto 
  then have converse(f)(f(P(g1,g2)))=converse(f)(F(h1,h2)) by auto 
  then have P(g1,g2)=converse(f)(F(h1,h2)) using left_inverse assms(1) 
  group0.group_op_closed 
  assms(2) g1(1) g2(1) unfolding group0_def bij_def by auto moreover 
  from g1(2) have converse(f)(fg1)=converse(f)h1 by auto 
  then have g1=converse(f)h1 using left_inverse assms(1) unfolding bij_def using g1(1) by auto moreover 
  from g2(2) have converse(f)(fg2)=converse(f)h2 by auto 
  then have g2=converse(f)h2 using left_inverse assms(1) unfolding bij_def using g2(1) by auto ultimately 
  have P(converse(f)h1,converse(f)h2)=converse(f)(F(h1,h2)) by auto 
  then show thesis using assms(2,3) Homomor_def by auto 
qed

end

43 Fields - introduction

theory Field_ZF imports Ring_ZF

begin

This theory covers basic facts about fields.

43.1 Definition and basic properties

In this section we define what is a field and list the basic properties of fields.

Field is a notrivial 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.
The field context extends the ring context adding field-related assumptions and notation related to the multiplicative inverse.

**locale** field = ring K A M

**assumes**
- mult_commute: M {is commutative on} K
- not_triv: 0 ≠ 1
- inv_exists: ∀a∈K. a≠0 → (∃b∈K. a·b = 1)

**fixes**
- non_zero (K_0)
**defines**
- non_zero_def[simp]: K_0 ≡ K-{0}

**fixes**
- inv (_^{-1})
**defines**
- inv_def[simp]: a^{-1} ≡ GroupInv(K_0,restrict(M,K_0×K_0))(a)

The next lemma assures us that we are talking fields in the field context.

**lemma** (in field) Field_ZF_1_L1: shows IsAfield(K,A,M)
**using**
- ringAssum mult_commute not_triv inv_exists IsAfield_def
  by simp

We can use theorems proven in the field context whenever we talk about a field.

**lemma** field_field0: assumes IsAfield(K,A,M)
  shows field0(K,A,M)
**using**
- assms IsAfield_def field0_axioms.intro ring0_def field0_def
  by simp

Let’s have an explicit statement that the multiplication in fields is commutative.

**lemma** (in field) field_mult_comm: assumes a∈K b∈K
  shows a·b = b·a
**using**
- mult_commute assms IsCommutative_def by simp

Fields do not have zero divisors.

**lemma** (in field) field_has_no_zero_divs: shows HasNoZeroDivs(K,A,M)

**proof** -
  { fix a b assume A1: a∈K b∈K and A2: a·b = 0 and A3: b≠0
    from inv_exists A1 A3 obtain c where I: c∈K and II: b·c = 1
      by auto
    from A2 have a·b·c = 0·c by simp
    with A1 I have a·(b·c) = 0
  }
using \text{Ring\_ZF\_1\_L11} \text{Ring\_ZF\_1\_L6} \text{by simp}
with A1 II have a=0 \text{using Ring\_ZF\_1\_L3} \text{by simp }
then have \( \forall a \in K. \forall b \in K. \ a \cdot b = 0 \implies a=0 \lor b=0 \) \text{by auto}
then show thesis \text{using HasNoZeroDivs\_def} \text{by auto}
qed

\( K_0 \) (the set of nonzero field elements is closed with respect to multiplication.

\text{lemma (in field0) Field\_ZF\_1\_L2}: 
\text{shows} \( K_0 \) \{is closed under\} \( M \)
\text{using Ring\_ZF\_1\_L4} \text{field\_has\_no\_zero\_divs} \text{Ring\_ZF\_1\_L12}
\text{IsOpClosed\_def} \text{by auto}

Any nonzero element has a right inverse that is nonzero.

\text{lemma (in field0) Field\_ZF\_1\_L3}: \text{assumes} A1: \( a \in K_0 \)
\text{shows} \( \exists b \in K_0. \ a \cdot b = 1 \)
proof -
from inv\_exists A1 obtain b where \( b \in K \) \text{and} \( a \cdot b = 1 \)
by auto
with not\_triv A1 show \( \exists b \in K_0. \ a \cdot b = 1 \)
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 \text{group0} context.

\text{theorem (in field0) Field\_ZF\_1\_L4}: \text{shows}
\text{IsAgroup} \( (K_0, \text{restrict}(M,K_0 \times K_0)) \)
\text{group0} \( (K_0, \text{restrict}(M,K_0 \times K_0)) \)
1 = \text{TheNeutralElement} \( (K_0, \text{restrict}(M,K_0 \times K_0)) \)
proof -
let \( f = \text{restrict}(M,K_0 \times K_0) \)
have \( M \) \{is associative on\} \( K \)
\( K_0 \subseteq K \) \( K_0 \) \{is closed under\} \( M \)
\text{using Field\_ZF\_1\_L1} \text{IsAfield\_def} \text{IsAring\_def} \text{IsAgroup\_def}
\text{IsAmonoid\_def} \text{Field\_ZF\_1\_L2} \text{by auto}
then have \( f \) \{is associative on\} \( K_0 \)
\text{using func\_ZF\_4\_L3} \text{by simp}
moreover
from not\_triv have 
I: \( 1 \in K_0 \land (\forall a \in K_0. \ f(1,a) = a \land f(a,1) = a) \)
\text{using Ring\_ZF\_1\_L2} \text{Ring\_ZF\_1\_L3} \text{by auto}
then have \( \exists n \in K_0. \forall a \in K_0. \ f(n,a) = a \land f(a,n) = a \)
\text{by blast}
ultimately have II: \text{IsAmonoid} \( (K_0,f) \) \text{using IsAmonoid\_def}
\text{by simp}
then have \text{monoid0} \( (K_0,f) \) \text{using monoid0\_def} \text{by simp}
moreover note I
ultimately show 1 = \text{TheNeutralElement} \( (K_0,f) \)
\text{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\cdot a^{-1} = 1 \ a^{-1}\cdot a = 1
proof -
  let f = restrict(M,K_0\times K_0)
  from A1 have group0(K_0,f)
  a \in K_0
  using Field_ZF_1_L4 by auto
  then have f\langle \text{GroupInv}(K_0,f)(a),a \rangle = \text{TheNeutralElement}(K_0,f) \wedge
  f\langle \text{GroupInv}(K_0,f)(a),a \rangle = \text{TheNeutralElement}(K_0,f)
  by (rule group0.group0_2_L6)
  with A1 show a\cdot a^{-1} = 1 \ a^{-1}\cdot a = 1
  using Field_ZF_1_L5 Field_ZF_1_L4 by auto

qed

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

43.2 Equations and identities

This section deals with more specialized identities that are true in fields.

\( a/(a^2) = 1/a \).

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 ... = \( (a \cdot a^{-1}) \cdot a^{-1} \)
  using Field_ZF_1_L5 Ring_ZF_1_L11
  by simp
also from A1 have ... = \( a^{-1} \)
  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: \( a \in K \)  \( b \in K \)  \( b \neq 0 \)  \( c \in K \)  and A2: \( a \cdot b \neq c \)
shows \( a \neq c \cdot b^{-1} \)
  proof -
    from A1 A2 have \( a \cdot b \cdot b^{-1} = c \cdot b^{-1} \)
      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
  Field_ZF_1_L7 Ring_ZF_1_L3 by simp

An identity with three field elements, inverse and cancelling.

472
lemma (in field0) Field_ZF_2_L6: assumes A1: \(a \in K\) \(b \in K\) \(b \neq 0\) \(c \in K\)
shows \(a \cdot b \cdot (c \cdot b^{-1}) = a \cdot c\)

proof -
from A1 have T: \(a \cdot b \in K\) \(b^{-1} \in K\)
using Ring_ZF_1_L4 Field_ZF_1_L5 by auto
with mult_commute A1 have a\(b \cdot (c \cdot b^{-1}) = a \cdot b \cdot (b^{-1} \cdot c)\)
using IsCommutative_def by simp
moreover
from A1 T have \(a \cdot b \in K\) \(b^{-1} \in K\) \(c \in K\)
by auto
then have \(a \cdot b \cdot b^{-1} \cdot c = a \cdot b \cdot (b^{-1} \cdot c)\)
using Field_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
qed

43.3 1/0=0

In ZF if \(f : X \rightarrow Y\) 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.

locale fielddd = field0 +
fixes division
defines division_def[simp]: division \(\equiv \{\langle p, \text{fst}(p) \cdot \text{snd}(p)^{-1}\rangle. p \in K \times K_0\}\)
fixes fdiv (infixl / 95)
defines fdiv_def[simp]: \(x/y \equiv \text{division}(x,y)\)

Division is a function on \(K \times K_0\) with values in \(K\).

lemma (in fielddd) div_fun: shows division: \(K \times K_0 \rightarrow K\)
proof -
have \(\forall p \in K \times K_0. \ \text{fst}(p) \cdot \text{snd}(p)^{-1} \in K\)
proof
fix p assume p \(\in K \times K_0\)
then have \(\text{fst}(p) \in K\) \(\text{snd}(p) \in K_0\)
by auto
ultimately have \(\text{fst}(p) \cdot \text{snd}(p)^{-1} \in K\)
using Ring_ZF_1_L4 Field_ZF_1_L5 by auto
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

```isarqed```

44 Ordered fields

theory OrderedField_ZF imports OrderedRing_ZF Field_ZF

begin

This theory covers basic facts about ordered fields.

44.1 Definition and basic properties

Here we define ordered fields and prove their basic properties.

Ordered field is a nontrivial ordered ring such that all non-zero elements have an inverse. We define the notion of being an 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**

\[
\text{IsAnOrdField}(K,A,M,r) \equiv (\text{IsAnOrdRing}(K,A,M,r) \land \\
(M \text{ is commutative on } K) \land \\
\text{TheNeutralElement}(K,A) \neq \text{TheNeutralElement}(K,M) \land \\
(\forall a \in K. a \neq \text{TheNeutralElement}(K,A) \rightarrow \\
(\exists b \in K. M(a,b) = \text{TheNeutralElement}(K,M))))
\]

The 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 `ring1` context to $K$, more appropriate for fields. Theoretically the Isar locale facility supports such renaming, but we experienced difficulties using some lemmas from `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 \implies (\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 \(-1\) (96 97)
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 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 field0 (about fields) context are valid in the 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 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 \text{ 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 \(-c\) \in R_+ }
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 $A1: 0 < a \ 0 < b$
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 $a \in R \ a \neq 0$
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 $a \in R \ a \neq 0$
shows $a \cdot (a^{-1}) = 1 \ (a^{-1}) \cdot a = 1$
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⋅b⁻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⋅b⁻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 < a² + b² and ∃c∈R. (a² + b²)⋅c = 1 proof - from A1 A2 show 0 < a² + b² using OrdField_ZF_1_L1B field0.Field_has_no_zero_divs OrdRing_ZF_3_L19 by simp then have (a² + b²)⁻¹ ∈ R and (a² + b²)⋅(a² + b²)⁻¹ = 1
using OrdRing_ZF_1_L3 PositiveSet_def OrdField_ZF_1_L8 by auto
then show \( \exists c \in R. (a^2 + b^2) \cdot c = 1 \) by auto
qed

44.2 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 \( c \in R^+ \) shows \( a \cdot c < b \cdot c \)
proof -
  from assms OrdField_ZF_1_L1B field0.field_has_no_zero_divs OrdRing_ZF_3_L13 by simp
qed

A special case of OrdField_ZF_2_L1 when we multiply an inverse by an element.

lemma (in field1) OrdField_ZF_2_L2: assumes \( a \in R^+ \) and \( a^{-1} < b \) shows \( 1 < b \cdot a \)
proof -
  from assms OrdField_ZF_2_L1 by simp
  with assms 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 \( a \leq b \) and \( c \in R^+ \) shows \( a \cdot (c^{-1}) \leq b \cdot (c^{-1}) \)
proof -
  using assms OrdField_ZF_1_L8 OrdRing_ZF_1_L9A by simp
qed

We can multiply a strict inequality by a positive element or its inverse.

lemma (in field1) OrdField_ZF_2_L4: assumes \( a < b \) and \( c \in R^+ \) shows
  \( a \cdot c < b \cdot c \)
  \( c \cdot a < c \cdot b \)
  \( a^{-1} < b^{-1} \)
proof -
  using assms OrdField_ZF_1_L1B field0.field_has_no_zero_divs OrdField_ZF_1_L8 OrdRing_ZF_3_L13 by auto
qed

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 $A1: a \in \mathbb{R}$ and $b \in \mathbb{R}^+$ 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 $A1: b \in \mathbb{R}$ and $c \in \mathbb{R}^+$ 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 $A1: a \in \mathbb{R}$ and $b \in \mathbb{R}^+$ and $A2: a \cdot b < c$
shows $a < c \cdot b^{-1}$

proof -
from $A1$ $A2$ have $a \cdot b \cdot b^{-1} < c \cdot b^{-1}$
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 $A1: b \in \mathbb{R}$ and $c \in \mathbb{R}^+$ and $A2: a < b \cdot c$
shows $a \cdot c^{-1} < b$

proof -
from $A1$ $A2$ have $a \cdot c^{-1} < b \cdot c \cdot c^{-1}$
using OrdField_ZF_2_L4 by simp
with $A1$ show $a \cdot c^{-1} < b$ 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 $A1: a \in \mathbb{R}^+$ and $A2: a \cdot a^{-1} \leq b$

shows $b^{-1} \leq a$

proof -
from A1 have $a^{-1} \in R_+$ using OrdField_ZF_1_L8
by simp
with A2 have $b \in R_+$ using OrdRing_ZF_3_L7
by blast
then have $T: b \in R_+$ $b^{-1} \in R_+$ using OrdField_ZF_1_L8
by auto
with A1 A2 have $b^{-1}a^{-1}a \leq b^{-1}b-a$
using OrdRing_ZF_1_L9A by simp
moreover
from A1 A2 T have
$b^{-1} \in R$ $a^{-1} \leq b$ $a \in R$ $b \notin 0$
using PositiveSet_def OrdRing_ZF_1_L3
by auto
with A2 have $b^{-1} \neq a$
using OrdField_ZF_1_L1B field0.Field_ZF_2_L4
by simp
with A1 A2 show $b^{-1} < a$
using OrdField_ZF_2_L7 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: $a \in R_+$ and A2: $a^{-1} < b$
shows $b^{-1} < a$
proof -
from A1 A2 have $a^{-1} \in R_+$ $a^{-1} \leq b$
using OrdField_ZF_1_L8 by auto
then have $b \in R_+$ using OrdRing_ZF_3_L7
by blast
then have $b \in R_+$ $b \neq 0$ using PositiveSet_def by auto
with A2 have $b^{-1} \neq a$
using OrdField_ZF_1_L1B field0.Field_ZF_1_L7
field0.Field_ZF_1_L6 Ring_ZF_1_L3
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: $a \in b$ and A2: $(b-a)^{-1} < c$
shows $1 + a \cdot c < b \cdot c$
proof -
from A1 A2 have $(b-a)^{-1} \in R_+$ $(b-a)^{-1} \leq c$
using OrdField_ZF_1_L9 by auto
then have $T_1: c \in R_+$ using OrdRing_ZF_3_L7 by blast
with A1 A2 have $T_2:$
$a \in R$ $b \in R$ $c \in R$ $c \neq 0$ $c^{-1} \in R$

480
44.3 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

\[ \text{IsAmodelOfReals}(K, A, M, r) \equiv \text{IsAnOrdField}(K, A, M, r) \land (r \text{ is complete}) \]

end

45 Integers - introduction

theory Int_ZF_IML imports OrderedGroup_ZF_1 Finite_ZF_1 ZF.Int Nat_ZF_IML

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
\(\mathbb{Z} \times \mathbb{Z}\). 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.

45.1 Addition and multiplication as ZF-functions.

In this section we provide definitions of addition and multiplication as sub-
sets of \((\mathbb{Z} \times \mathbb{Z}) \times \mathbb{Z}\). We use the (higher order) relation defined in the standard
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 OrderedGroup_ZF theory.

Definition of addition of integers as a binary operation on int. Recall that in standard Isabelle/ZF int is the set of integers and the sum of integers is denoted by prepending $+$ with a dollar sign.

definition
IntegerAddition $\equiv \{\langle x,c \rangle \in \text{int} \times \text{int}. \text{fst}(x) \#+ \text{snd}(x) = c\}$

Definition of multiplication of integers as a binary operation on int. In standard Isabelle/ZF product of integers is denoted by prepending the dollar sign to $\ast$.

definition
IntegerMultiplication $\equiv \{\langle x,c \rangle \in \text{int} \times \text{int}. \text{fst}(x) \#* \text{snd}(x) = c\}$

Definition of natural order on integers as a relation on int. In the standard Isabelle/ZF the inequality relation on integers is denoted $\leq$ prepended with the dollar sign.

definition
IntegerOrder $\equiv \{p \in \text{int} \times \text{int}. \text{fst}(p) \#\leq \text{snd}(p)\}$

This defines the set of positive integers.

definition
PositiveIntegers $\equiv \text{PositiveSet}(\text{int},\text{IntegerAddition},\text{IntegerOrder})$

IntegerAddition and IntegerMultiplication are functions on $\text{int} \times \text{int}$.

lemma Int_ZF_1_L1: shows

IntegerAddition : $\text{int} \times \text{int} \rightarrow \text{int}$

IntegerMultiplication : $\text{int} \times \text{int} \rightarrow \text{int}$

proof -

have

$\{\langle x,c \rangle \in (\text{int} \times \text{int}) \times \text{int}. \text{fst}(x) \#+ \text{snd}(x) = c\} \in \text{int} \times \text{int} \rightarrow \text{int}$

$\{\langle x,c \rangle \in (\text{int} \times \text{int}) \times \text{int}. \text{fst}(x) \#* \text{snd}(x) = c\} \in \text{int} \times \text{int} \rightarrow \text{int}$

using func1_1_L11A by auto

then show IntegerAddition : $\text{int} \times \text{int} \rightarrow \text{int}$

IntegerMultiplication : $\text{int} \times \text{int} \rightarrow \text{int}$

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 $m..n$ 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 OrderedGroup that assigns $x$ to itself if $x$ is positive and assigns the opposite of $x$ if $x \leq 0$. Unfortunately we

482
cannot use the $|\cdot|$ notation as in the OrderedGroup theory as this notation has been hogged by the standard Isabelle’s Int theory. The notation $-A$ where $A$ is a subset of integers means the set $\{-m : m \in A\}$. The symbol $\max f(M)$ denotes the maximum of function $f$ over the set $A$. We also introduce a similar notation for the minimum.

locale int0 =

  fixes ints (\mathbb{Z})
  defines ints_def [simp]: $\mathbb{Z} \equiv \text{int}$

  fixes ia (infixl + 69)
  defines ia_def [simp]: $a+b \equiv \text{IntegerAddition}(a,b)$

  fixes iminus (- _ 72)
  defines rminus_def [simp]: $-a \equiv \text{GroupInv}(\mathbb{Z},\text{IntegerAddition})(a)$

  fixes isub (infixl - 69)
  defines isub_def [simp]: $a-b \equiv a + (-b)$

  fixes imult (infixl · 70)
  defines imult_def [simp]: $a \cdot b \equiv \text{IntegerMultiplication}(a,b)$

  fixes setneg (- _ 72)
  defines setneg_def [simp]: $-A \equiv \text{GroupInv}(\mathbb{Z},\text{IntegerAddition})(A)$

  fixes izero (0)
  defines izero_def [simp]: $0 \equiv \text{TheNeutralElement}(\mathbb{Z},\text{IntegerAddition})$

  fixesione (1)
  defines ione_def [simp]: $1 \equiv \text{TheNeutralElement}(\mathbb{Z},\text{IntegerMultiplication})$

  fixes itwo (2)
  defines itwo_def [simp]: $2 \equiv 1+1$

  fixes ithree (3)
  defines ithree_def [simp]: $3 \equiv 2+1$

  fixes nonnegative ($\mathbb{Z}^+$)
  defines nonnegative_def [simp]: $\mathbb{Z}^+ \equiv \text{Nonnegative}(\mathbb{Z},\text{IntegerAddition},\text{IntegerOrder})$

  fixes positive ($\mathbb{Z}_+$)
  defines positive_def [simp]: $\mathbb{Z}_+ \equiv \text{PositiveSet}(\mathbb{Z},\text{IntegerAddition},\text{IntegerOrder})$

  fixes abs
  defines abs_def [simp]: $\text{abs}(m) \equiv \text{AbsoluteValue}(\mathbb{Z},\text{IntegerAddition},\text{IntegerOrder})(m)$
defines lesseq_def [simp]: \( m \leq n \equiv (m,n) \in \text{IntegerOrder} \)

defines interval_def [simp]: \( m..n \equiv \text{Interval(IntegerOrder},m,n) \)

defines maxf_def [simp]: \( \text{maxf}(f,A) \equiv \text{Maximum(IntegerOrder},f(A)) \)

defines minf_def [simp]: \( \text{minf}(f,A) \equiv \text{Minimum(IntegerOrder},f(A)) \)

Integer addition and multiplication are associative.

Integer addition and multiplication are commutative.

Zero is neutral for addition and one for multiplication.
lemma (in int0) Int_ZF_1_L5: assumes A1:x∈š
shows (# 0) + x = x ∧ x + (# 0) = x
(# 1)· x = x ∧ x· (# 1) = x
proof -
  from A1 show (# 0) + x = x ∧ x + (# 0) = x
    using Int_ZF_1_L2 zadd_int0 Int_ZF_1_L4 by simp
  from A1 have (# 1)· x = x
    using Int_ZF_1_L2 zmult_int1 by simp
  with A1 show (# 1)· x = x ∧ x· (# 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)∈š ∧
(∀ x∈š. (# 0)+ x = x ∧ x+ (# 0) = x)
(# 1)∈š ∧
(∀ x∈š. (# 1)· x = x ∧ x· (# 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(š,IntegerAddition)
  IsAmonoid(š,IntegerMultiplication)
proof -
  have ∃e∈š. ∀ x∈š. e+x = x ∧ x+ e = x
  ∃e∈š. ∀ x∈š. e· x = x ∧ x· e = x
    using int0.Int_ZF_1_L6 by auto
  then show IsAmonoid(š,IntegerAddition)
    IsAmonoid(š,IntegerMultiplication) using
    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 (# 1) = 1
proof -
  have monoid0(š,IntegerAddition)
    using Int_ZF_1_T1 monoid0_def by simp
  moreover have (# 0)∈Z ∧
    (∀ x∈Z. IntegerAddition(# 0 ,x) = x ∧
    IntegerAddition(x ,# 0) = x)
    using Int_ZF_1_L6 by auto
  ultimately have (# 0) = TheNeutralElement(Z,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 ($# 1) ∈ int ∧ 
(∀x∈int. IntegerMultiplication($# 1, x) = x ∧ 
IntegerMultiplication(x ,$# 1) = x) 
  using Int_ZF_1_L6 by auto 
ultimately have 
  ($# 1) = TheNeutralElement(int,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 ($# 0) ∈ Z  ($# 1) ∈ 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 ($# 0) ≠ ($# 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 ∃ b∈Z. g+b = 0 
proof - 
  from A1 have g+ $-g = 0 
  using Int_ZF_1_L2 Int_ZF_1_L8 by simp 
  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)
What is the additive group inverse in the group of integers?

lemma (in int0) Int_ZF_1_L9A: assumes A1: m\in\mathbb{Z}
shows \(-m = -m\)
proof -
  from A1 have m\in\mathbb{Z} \land \(-m \in \mathbb{Z}\) \land IntegerAddition(m,\(-m\)) =
    TheNeutralElement(int,IntegerAddition)
  using zminus_type Int_ZF_1_L2 Int_ZF_1_L8 by auto
  then have \(-m = \text{GroupInv}(int,IntegerAddition)(m)\)
    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\in\mathbb{Z} \land n\in\mathbb{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\in\mathbb{Z}
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\in\mathbb{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 subtracing 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)
\[ m \in \mathbb{Z}, \quad 1 \in \mathbb{Z} \]
\[
\text{IntegerAddition}(m, 1) = m \\
\text{using Int_ZF_1_T2 Int_ZF_1_L8A by auto} \\
\text{then have } 1 = \text{TheNeutralElement}(\mathbb{Z}, \text{IntegerAddition}) \\
\text{by (rule group0.group0_2_L7)} \\
\text{then have False using int_zero_not_one by simp} \\
\} \text{ then show } I: \text{I+1} \neq m \text{ by auto} \\
\{ \text{from A1 have } m - 1 + 1 = m \\
\text{ using Int_ZF_1_L8A Int_ZF_1_T2 group0.inv_cancel_two} \\
\text{ by simp} \\
\text{moreover assume } m - 1 = m \\
\text{ultimately have } m + 1 = m \text{ by simp} \\
\text{with I have False by simp} \\
\} \text{ then show } m - 1 \neq m \text{ by auto} \\
\text{qed}

If the difference is zero, the integers are equal.

**Lemma (in int0) Int_ZF_1_L15:**

\[ m \in \mathbb{Z}, \quad n \in \mathbb{Z} \text{ and } m - n = 0 \]
\[ \text{shows } m = n \]
\[ \text{proof -} \\
\text{let } G = \mathbb{Z} \\
\text{let } f = \text{IntegerAddition} \\
\text{from A1 A2 have} \\
\text{group0}(G, f) \\
\text{m } \in G \quad n \in G \\
\text{f(m, GroupInv(G, f)(n)) = TheNeutralElement(G, f)} \\
\text{using Int_ZF_1_T2 by auto} \\
\text{then show } m = n \text{ by (rule group0.group0_2_L11A)} \]
\[ \text{qed} \]

### 45.2 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:**

\[ m \in \mathbb{Z}, \quad n \in \mathbb{Z} \text{ and } m \leq n \]
\[ \text{shows } m \leq n \]
\[ \text{proof -} \\
\text{from A1 A2 have } (m, n) \in \{ x \in \mathbb{Z} \times \mathbb{Z}. \text{fst}(x) \leq \text{snd}(x) \} \\
\text{ by simp} \\
\text{then show thesis using IntegerOrder_def by simp} \]
\[ \text{qed} \]

The next lemma interprets the definition the other way.

**Lemma (in int0) Int_ZF_2_L1A:**

\[ \text{assumes A1: } m \leq n \]
shows \( m \leq n \) \( m \in \mathbb{Z} \) \( n \in \mathbb{Z} \)

proof -
  from A1 have \((m,n) \in \{p \in \mathbb{Z} \times \mathbb{Z}. \text{fst}(p) \leq \text{snd}(p)\}\)
    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 \( x \in \{p \in \text{int} \times \text{int}. \text{fst}(p) \leq \text{snd}(p)\}\)
    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 (in int0) Int_ZF_2_L1C:
  assumes A1: IsBoundedBelow(A,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 (in int0) int_ord_is_refl: shows refl(\( \mathbb{Z} \),IntegerOrder)
proof -
  have \( \forall m n. m \leq n \) \( n \leq m \rightarrow m=n \)
    using Int_ZF_2_L3 by auto
qed

The essential condition to show antisymmetry of the order on integers.

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
qed

The order on integers is antisymmetric.
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: \( (m,n) \in \text{IntegerOrder} \) \( (n,k) \in \text{IntegerOrder} \)

shows \( (m,k) \in \text{IntegerOrder} \)

**proof** -

from A1 have T1: \( m \leq n \) \( n \leq k \) 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: \( m \leq n \) \( n \leq k \)

shows \( m \leq k \)

**proof** -

from A1 have \( (m,n) \in \text{IntegerOrder} \) \( (n,k) \in \text{IntegerOrder} \)
by auto
then have \( (m,k) \in \text{IntegerOrder} \) by (rule Int_ZF_2_L5)
then show \( m \leq k \) by simp
qed

The order on integers is transitive.

**lemma Int_ZF_2_L6:** shows trans(IntegerOrder)

**proof** -

have \( \forall m \ n \ k. \)
\( (m, n) \in \text{IntegerOrder} \ \wedge \ (n, k) \in \text{IntegerOrder} \mapsto \)
\( (m, k) \in \text{IntegerOrder} \)
using Int_ZF_2_L5 by blast
then show thesis by (rule Fol1_L2)
qed

The order on integers is a partial order.

**lemma Int_ZF_2_L7:** shows IsPartOrder(int,IntegerOrder)

using int0.int_ord_is_refl int0.Int_ZF_2_L4
Int_ZF_2_L6 IsPartOrder_def by simp

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 \( m+k \leq n+k \) \( k+m \leq k+n \)

**proof** -

from A2 have \( m \leq n \) and \( m \in \mathbb{Z} \) \( n \in \mathbb{Z} \)
using Int_ZF_2_L1A by auto
with A1 show m+k ≤ n+k k+m≤ k+n
  using zadd_right_cancel_zle zadd_left_cancel_zle
Int_ZF_1_L2 Int_ZF_1_L1 apply_funtype
Int_ZF_1_L2 Int_ZF_2_L1 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 IsAnOrdGroup(\mathbb{Z},\text{IntegerAddition},\text{IntegerOrder})
  IntegerOrder {is total on} \mathbb{Z}
  group3(\mathbb{Z},\text{IntegerAddition},\text{IntegerOrder})
  IsLinOrder(\mathbb{Z},\text{IntegerOrder})
proof -
  have \( \forall k \in \mathbb{Z}. \ \forall m \in \mathbb{Z}. m \leq n \rightarrow m+k \leq n+k \land k+m \leq k+n \)
    using int_ord_transl_inv by simp
  then show T1: IsAnOrdGroup(\mathbb{Z},\text{IntegerAddition},\text{IntegerOrder}) using
    Int_ZF_1_T2 Int_ZF_2_L1B Int_ZF_2_L7 IsAnOrdGroup_def
    by simp
  then show group3(\mathbb{Z},\text{IntegerAddition},\text{IntegerOrder})
    using group3_def by simp
  have \( \forall n \in \mathbb{Z}. \ \forall m \in \mathbb{Z}. n \leq m \lor m \leq n \)
    using zle_linear Int_ZF_2_L1 by auto
  then show IntegerOrder {is total on} \mathbb{Z}
    using IsTotal_def by simp
  with T1 show IsLinOrder(\mathbb{Z},\text{IntegerOrder})
    using IsAnOrdGroup_def IsPartOrder_def 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 \( i \prec m \)
proof -
  from A1 have \( i \leq m \) \( i \in \mathbb{Z} \)
    using Int_ZF_2_L1A by auto
  with A2 show \( i \prec m \) using zle_def by simp
qed

This shows how Isabelle’s \( \prec \) operator translates to IsarMathLib notation.

lemma (in int0) Int_ZF_2_L9A: assumes A1: \( m \in \mathbb{Z} \) and A2: \( m \prec n \)
shows \( m \leq n \) \( m \neq n \)
  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∈Z and A2: m ≤ k - ($# 1)
  shows m+1 ≤ k
proof -
  from A2 have m+1 ≤ (k - ($# 1)) + 1
    using Int_ZF_1_L9A int_ord_transl_inv by simp
  with A1 show m+1 ≤ 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∈Z  m∈Z  k∈Z
  shows i+m ≤ k  if and only if  i ≤ k-m
proof -
  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∈Z  m∈Z and i-m ≤ k
  shows i ≤ k+m
proof -
  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 ≤ i
  shows (-i) ≤ (-k)
  $-i ≤ $-k
proof -
  using assms Int_ZF_2_L1A Int_ZF_1_L9A Int_ZF_2_T1
  group3.OrderedGroup_ZF_1_L5 by auto

Taking minus on both sides of inequality reverses it, version with a negative
on one side.

lemma (in int0) Int_ZF_2_L10AA: assumes n∈Z  m≤(-n)
  shows n≤(-m)
proof -
  using assms Int_ZF_2_T1 group3.OrderedGroup_ZF_1_L5AD
  by simp

We can cancel the same element on both sides of an inequality, a version
with minus on both sides.

lemma (in int0) Int_ZF_2_L10AB:
  assumes m∈Z  n∈Z  k∈Z and m-n ≤ m-k
  shows k≤n
proof -
  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 ≤ 0

492
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$\in$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 $+$ ($#$ succ(n))
proof -
  have $# succ(n) = $#1 $+$ n using int_succ_int_1 by blast
  then have i $+$ $# succ(n) = i $+$ ($# n $+$ $#1$) 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$\in$Z and A2: n$\in$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$nat n = 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 $\leq k$ $+$ $#1$ j $\leq k+1$
proof -
  from A1 have T1:j$\in$Z k$\in$Z j $\leq k$
    using Int_ZF_2_L1A by auto
  moreover from T1 have k $\leq k$ $+$ $#1$ using Int_ZF_2_L12 Int_ZF_2_L1A
    by simp
  ultimately have j $\leq k$ $+$ $#1$ using zle_trans by fast
  with T1 show j $\leq k$ $+$ $#1$ using Int_ZF_2_L1 by simp
  with T1 have j $\leq k+$$1$
    using Int_ZF_1_L2 by simp
qed

493
then show \( j \leq k + 1 \) using Int_ZF_1_L8 by simp

qed

Adding one increases integers, yet one more version.

lemma (in int0) Int_ZF_2_L12B: assumes A1: \( m \in \mathbb{Z} \) shows \( m \leq m + 1 \)
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: \( k \in \mathbb{Z} \)  m \in \mathbb{Z} \) and A2: \( n \in \text{nat} \)
and A3: \( k + (\# 1) = m + \# \text{succ}(n) \)
shows \( m \leq k \)
proof -
  from A1 have \( k \in \mathbb{Z} \)  m \in \mathbb{Z} \) \( n \in \mathbb{Z} \) by auto
  moreover from assms have \( k + \# 1 = m + \# \text{succ}(n) \)
    using Int_ZF_2_L11 by simp
  ultimately have \( k = m + \# \text{succ}(n) \) 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 \( m \in \mathbb{Z} \)
shows \( \text{abs}(m) \in \mathbb{Z} \)
proof -
  have \( \text{AbsoluteValue}(\mathbb{Z}, \text{IntegerAddition}, \text{IntegerOrder}) : \mathbb{Z} \rightarrow \mathbb{Z} \)
    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 \( (-m) \leq n \)
\( 0 \leq m + n \)
proof -
  from assms have \( 0 \leq m \)  \( 0 \leq n \)
    using assms Int_ZF_2_T1
    group3.OrderedGroup_ZF_3_L5AC group3.OrderedGroup_ZF_3_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 \( a, b, c, b_1, c_1 \in \text{IntegerOrder} \)
    \( \langle a, b, c \rangle \in \text{IntegerOrder} \)
    \( \langle b, b_1 \rangle \in \text{IntegerOrder} \)
    \( \langle c, c_1 \rangle \in \text{IntegerOrder} \)
  by auto
using Int_ZF_2_T1 by auto
then have \( \langle a, \text{IntegerAddition}(b_1, c_1) \rangle \in \text{IntegerOrder} \)
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 \( \text{group3}(\mathbb{Z}, \text{IntegerAddition}, \text{IntegerOrder}) \)
b \in \mathbb{Z}
\( \langle a, \text{IntegerAddition}(b, c) \rangle \in \text{IntegerOrder} \)
\( \langle c, c_1 \rangle \in \text{IntegerOrder} \)
using Int_ZF_2_T1 by auto
then have \( \langle \text{IntegerAddition}(a, \text{IntegerAddition}(b, c)), n \rangle \in \text{IntegerOrder} \)
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 incenses.

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 \( P = \text{IntegerAddition} \)
from assms have \( \text{group3}(\mathbb{Z}, P, \text{IntegerOrder}) \)
m \in \mathbb{Z}
\( \langle k, L \rangle \in \text{IntegerOrder} \)
using Int_ZF_2_T1 by auto
then have \( \langle P(P(m, k), n), P(P(m, L), n) \rangle \in \text{IntegerOrder} \)
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 \( 0 \leq n \) \( m \in \mathbb{Z} \)
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 \( m \in \mathbb{Z} \) \( n \in \mathbb{Z} \)
shows \( m+n \leq abs(m)+abs(n) \)
\( m-n \leq abs(m)+abs(n) \)
\( (-m)+n \leq abs(m)+abs(n) \)
\( (-m)-n \leq abs(m)+abs(n) \)
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 \( m \leq k \) and \( 0 \leq n \)
shows \( m \leq k+n \) \( m \leq n+k \)
using assms Int_ZF_2_T1 group3.OrderedGroup_ZF_1_L5G
by auto

Triangle inequality for integers.

**lemma (in int0) Int_triangle_ineq:**
assumes \( m \in \mathbb{Z} \) \( n \in \mathbb{Z} \)
shows \( abs(m+n) \leq abs(m)+abs(n) \)
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 \( abs(m) = m \)
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 \( abs(1) = 1 \)
proof -
have \((\# 0) \in \mathbb{Z} \) \((\# 1) \in \mathbb{Z} \) 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 \( 0 \leq 0 \) 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 \( abs(1) = 1 \) 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 $A1: 1 \leq a$

shows $0 \leq a$

$a \neq 0$

$2 \leq a+1$

$1 \leq a+1$

$0 \leq a+1$

**proof**

- from $A1$ have $0 \leq 1$ and $1 \leq a$
  using Int_ZF_2_L16A by auto
- have $0 \leq a$ by (rule Int_order_transitive)
- have $1 \leq 2$ using Int_ZF_2_L16B 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 $\text{abs}(-m) = \text{abs}(m)$

**proof**

- have $-m \in \mathbb{Z}$ by simp

**qed**

The absolute value of zero is zero.

**lemma (in int0) Int_ZF_2_L18:** shows $\text{abs}(0) = 0$

**proof**

- have $0 \in \mathbb{Z}$ by simp

**qed**

A different version of the triangle inequality.

**lemma (in int0) Int_triangle_ineq1:**

assumes $A1: m \in \mathbb{Z}$

$n \in \mathbb{Z}$

shows $\text{abs}(m-n) \leq \text{abs}(n)+\text{abs}(m)$

**proof**

- have $-n \in \mathbb{Z}$ by simp
- with $A1$ have $\text{abs}(m-n) \leq \text{abs}(m)+\text{abs}(-n)$

497
using Int_ZF_1_L9A Int_triangle_ineq by simp 
with A1 show 
  abs(m-n) ≤ abs(n)+abs(m) 
  abs(m-n) ≤ abs(m)+abs(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∈Z n∈Z 
  and abs(m-n) ≤ k 
  shows abs(m) ≤ abs(n)+k 
  m-k ≤ n 
  m ≤ n+k 
  n-k ≤ 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∈Z n∈Z k∈Z 
  shows abs(m+n+k) ≤ abs(m)+abs(n)+abs(k) 
proof - 
  from A1 have T: m+n ∈ Z abs(k) ∈ Z 
    using Int_ZF_1_T2 group0.group_op_closed Int_ZF_2_L14 
    by auto 
  with A1 have abs(m+n+k) ≤ abs(m+n) + abs(k) 
    using Int_triangle_ineq by simp 
  moreover from A1 T have 
    abs(m+n) + abs(k) ≤ abs(m) + abs(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∈Z n∈Z and A2: ¬(n≤m) 
  shows m≤n (¬n) ≤ (¬m) m≠n 
proof - 
  from A1 A2 show m≤n using Int_ZF_2_T1 IsTotal_def 
    by auto 
  then show (¬n) ≤ (¬m) using Int_ZF_2_L10
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 ≤ n and A2: m ≠ n
  shows ¬(n ≤ m)
proof -
  from A1 A2 have \langle m,n \rangle \in \text{IntegerOrder}
    m ≠ n
    using Int_ZF_2_T1 by auto
  then have \langle n,m \rangle /∈ \text{IntegerOrder}
    by (rule group3.OrderedGroup_ZF_1_L8AA)
  thus ¬(n ≤ m) by simp
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 ∈ Š and A2: ¬(0 ≤ m)
  shows m ≤ 0 0 ≤ (-m) m ≠ 0
proof -
  from A1 T: 0 ∈ Š
  using Int_ZF_2_L19 by simp
  from A1 T A2 show m ≤ 0 using Int_ZF_2_L19 by blast
  from A1 T A2 show m ≠ 0 by (rule Int_ZF_2_L19)
  then show 0 ≤ (-m)
   using Int_ZF_2_L19 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∈\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 = IntegerAddition
  let r = IntegerOrder
  let b = m
  from assms have group3(G, P, r)
   r \{is total on\} G
\[ b \in G \]
\[ Q(\text{TheNeutralElement}(G, P)) \]
\[ \forall a \in \text{PositiveSet}(G, P, r). \quad Q(a) \]
\[ \forall a \in \text{PositiveSet}(G, P, r). \quad Q(\text{GroupInv}(G, P)(a)) \]
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 \( A1: m \in \mathbb{Z} \)
shows \( m \leq \text{abs}(m) \)
\[ (-m) \leq \text{abs}(m) \]
using \( \text{assms Int_ZF_2_T1} \)
\[ \text{group3.OrderedGroup_ZF_3_L5} \quad \text{group3.OrderedGroup_ZF_3_L6} \]
by auto
\[ |m - n| = |n - m|. \]

**Lemma (in int0) Int_ZF_2_L20:** assumes \( m \in \mathbb{Z} \quad n \in \mathbb{Z} \)
shows \( \text{abs}(m-n) = \text{abs}(n-m) \)
using \( \text{assms Int_ZF_2_T1} \quad \text{group3.OrderedGroup_ZF_3_L7B} \)
by simp

We can add the sides of inequalities with absolute values.

**Lemma (in int0) Int_ZF_2_L21:**
assumes \( A1: m \in \mathbb{Z} \quad n \in \mathbb{Z} \)
and \( A2: \text{abs}(m) \leq k \quad \text{abs}(n) \leq l \)
shows \( \text{abs}(m+n) \leq k + l \quad \text{abs}(m-n) \leq k + l \)
using \( \text{assms Int_ZF_1_T2} \quad \text{Int_ZF_2_T1} \)
\[ \text{group3.OrderedGroup_ZF_3_L7C} \quad \text{group3.OrderedGroup_ZF_3_L7CA} \]
by auto

Absolute value is nonnegative.

**Lemma (in int0) Int_abs_nonneg:** assumes \( A1: m \in \mathbb{Z} \)
shows \( \text{abs}(m) \in \mathbb{Z}^+ \quad 0 \leq \text{abs}(m) \)
proof -
\[ \text{have } \text{AbsoluteValue}(\mathbb{Z}, \text{IntegerAddition}, \text{IntegerOrder}) : \mathbb{Z} \to \mathbb{Z}^+ \]
using \( \text{Int_ZF_2_T1} \quad \text{group3.OrderedGroup_ZF_3_L3C} \)
by simp
with \( A1 \) show \( \text{abs}(m) \in \mathbb{Z}^+ \)
using apply_functype
by simp
\[ \text{then show } 0 \leq \text{abs}(m) \]
using \( \text{Int_ZF_2_T1} \quad \text{group3.OrderedGroup_ZF_1_L2} \)
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 \(\text{abs}(m) \leq k\)
using assms Int_ZF_2_L16 by simp

45.3 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 A1: \(i \leq m\)
shows \(\exists n \in \text{nat}. m = i \# n\)
proof -
let \(n = 0\)
{ assume A2: \(i = m\)
  from A1 A2 have \(n \in \text{nat} \land m = i \# n\)
  using Int_ZF_2_L1A zadd_int0_right by auto
  hence \(\exists n \in \text{nat}. m = i \# n\) by blast }
moreover
{ assume A3: \(i \neq m\)
  with A1 have \(i < m\) \(\forall m \in \mathbb{Z}\)
  using Int_ZF_2_L9 Int_ZF_2_L1A by auto
  then obtain \(k\) where D1: \(k \in \text{nat} \land m = i \# \text{succ}(k)\)
  using zless_imp_succ_zadd_lemma by auto
  let \(n = \text{succ}(k)\)
  from D1 have \(n \in \text{nat} \land m = i \# n\) by auto
  hence \(\exists n \in \text{nat}. m = i \# n\) 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: \(\forall m. i \leq m \land Q(m) \rightarrow Q(m \# 1)\)
shows \(\forall k \in \text{nat}. Q(i \# k) \rightarrow Q(i \# \text{succ}(k))\)
proof
fix \(k\) assume A3: \(k \in \text{nat}\) show \(Q(i \# k) \rightarrow Q(i \# \text{succ}(k))\)
proof
  assume A4: \(Q(i \# k)\)
  from A1 A3 have \(i \leq i \# (\text{succ}(k))\) using Int_ZF_2_L12
  by simp
  with A4 A2 have \(Q(i \# (\text{succ}(k)) \# (\text{succ}(1)))\) by simp
  then show \(Q(i \# (\text{succ}(k)))\) using Int_ZF_2_L11 by simp
qed
qed

Induction on integers, version with higher-order increment function.

lemma (in int0) Int_ZF_3_L7:
assumes A1: i \leq k and A2: Q(i)
and A3: \forall m. i \leq m \land Q(m) \rightarrow Q(m + (# 1))
shows Q(k)

proof -
from A1 obtain n where D1: n \in \text{nat} and D2: k = i + # n
using Int_ZF_3_L2 by auto
from A1 have T1: m \in \text{nat} \land Q(m+1)
using Int_ZF_2_L1A by simp
note \langle \text{n \in nat} \rangle
moreover from A1 A2 have Q(i + #0)
using Int_ZF_2_L1A zadd_int0 by simp
moreover from T1 A3 have
\forall k \in \text{nat}. Q(i + (# k)) \rightarrow 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: \forall m. i \leq m \land Q(m) \rightarrow Q(m + 1)
shows \forall m. i \leq m \land Q(m) \rightarrow Q(m + (# 1))

proof -
{ \fix m assume i \leq m \land Q(m) 
with A1 have T1: m \in \text{Z} \land 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 \leq k and A2: Q(i)
and A3: \forall m. i \leq m \land Q(m) \rightarrow Q(m+1)
shows Q(k)

proof -
from A3 have \forall m. i \leq m \land Q(m) \rightarrow 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
Int_ZF_3_L7 substituting P(−k) for Q(k).

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 $A3: \forall m. \neg i \leq m \land P(-m) \rightarrow P(-(m - (1 \#)))$
shows $P(k)$

**Proof**
- from $A1$ have $T1: \neg i \leq \neg k$ using Int_ZF_2_L10 by simp
- from $A1$ $A2$ have $T2: P(-(-i))$ using Int_ZF_2_L1A zminus_zminus by simp
- from $T1$ $T2$ $A3$ have $P(-(-k))$ 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 $\forall m. \neg i \leq m \land P(-m) \rightarrow P(-(m + (1 \#)))$

**Proof**
fix $n$ show $\neg i \leq m \land P(-m) \rightarrow P(-(m + (1 \#)))$

**Proof**
- assume $A3: \neg i \leq m \land P(-m)$
- then have $\neg i \leq m$ by simp
- then have $\neg m \leq \neg(-i)$ 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 $T2: \forall m. \neg i \leq m \land P(-m) \rightarrow P(-(m + (1 \#)))$
  - 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 $A1: \forall n. n \leq i \land P(n) \rightarrow P(n - 1)$
shows $\forall n. n \leq i \land P(n) \rightarrow P(n + (1 \#))$

**Proof**
- \{ fix $n$ assume $n \leq i \land P(n)$
  - with $A1$ have $T1: n \in \mathbb{Z} \land P(n - 1)$ using Int_ZF_2_L1A by auto
  - then have $n - 1 = n - (1 \#)$ using Int_ZF_1_L8 by simp

503
with $T_1$ have $P(n \#-\#1)$ using \textit{IntZF\_1\_L10} by simp
} then show thesis by simp
qed

Backwards induction on integers.

\textbf{theorem} (in int0) \textit{Back\_induct\_on\_int}:
assumes \textbf{A1}: $k \leq i$ and \textbf{A2}: $P(i)$
and \textbf{A3}: \( \forall n. n \leq i \land P(n) \rightarrow P(n-1) \)
shows $P(k)$
proof -
  from \textbf{A3} have \( \forall n. n \leq i \land P(n) \rightarrow P(n \#-\#1) \)
  by (rule \textit{IntZF\_3\_L10})
  with \textbf{A1} \textbf{A2} show $P(k)$ by (rule \textit{IntZF\_3\_L9A})
qed

45.4 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 \textit{OrderedGroups\_ZF.thy}. To show the
other implication we show that all intervals starting at 0 are finite and then
use a result from \textit{OrderedGroups\_ZF.thy}.

There are no integers between $k$ and $k+1$.

\textbf{lemma} (in int0) \textit{IntZF\_4\_L1}:
assumes \textbf{A1}: $k \in \mathbb{Z}$ $m \in \mathbb{Z}$ $n \in \mathbb{N}$ and \textbf{A2}: $k \#+\#1 = m \#+\#n$
shows $m = k \#+\#1 \lor m \leq k$
proof -
  \{ assume \textbf{n}=0
    with \textbf{A1} \textbf{A2} have $m = k \#+\#1 \lor m \leq k$
    using \textit{zadd_int0} by simp \}
moreover
  \{ assume \textbf{n} \neq 0
    with \textbf{A1} obtain $j$ where \textbf{D1}: $j \in \mathbb{N}$ $n = \text{succ}(j)$
    using \textit{NatZF\_1\_L3} by auto
    with \textbf{A1} \textbf{A2} \textbf{D1} have $m = k \#+\#1 \lor m \leq k$
    using \textit{IntZF\_2\_L13} by simp \}
ultimately show thesis by blast
qed

A trivial calculation lemma that allows to subtract and add one.

\textbf{lemma} \textit{IntZF\_4\_L1A}:
assumes $m \in \mathbb{Z}$ shows $m \#-\#1 \#+\#1 = m$
using \textit{asms eq_zdiff_iff} by auto

There are no integers between $k$ and $k+1$, another formulation.

\textbf{lemma} (in int0) \textit{IntZF\_4\_L1B}:
assumes \textbf{A1}: $m \leq L$
shows

504
\[ m = L \lor m+1 \leq L \]
\[ m = L \lor m \leq L-1 \]

proof -
  let k = L $- $#1
  from A1 have T1: m \in \mathbb{Z} \land L = k \land \#1
    using Int_ZF_2_L1A Int_ZF_4_L1A by auto
  moreover from A1 obtain n where D1: n \in \mathbb{N} \land L = m \land \# n
    using Int_ZF_3_L2 by auto
  ultimately have m = L \lor m \leq k
    using Int_ZF_4_L1 by simp
  with T1 show m = L \lor m+1 \leq L
    using Int_ZF_2_L9A by auto
  with T1 show m = L \lor m \leq L-1
    using Int_ZF_1_L8A Int_ZF_2_L9B by simp
qed

If \( j \in m..k+1 \), then \( j \in m..n \) or \( j = k+1 \).

lemma (in int0) Int_ZF_4_L2: assumes A1: \( k \in \mathbb{Z} \)
  and A2: \( j \in m..(k + \#1) \)
  shows \( j \in m..k \lor j \in \{k + \#1\} \)
proof -
  from A2 have T1: m \leq j \leq (k + \#1) using Order_ZF_2_L1A by auto
  then have T2: m \in \mathbb{Z} \land j \in \mathbb{Z} using Int_ZF_2_L1A by auto
  from T1 have \( (m \leq j) \land (j \leq k) \lor j \in \{k \land \#1\} \)
    using Int_ZF_4_L1 by auto
  then show thesis using Order_ZF_2_L1B by auto
qed

Extending an integer interval by one is the same as adding the new endpoint.

lemma (in int0) Int_ZF_4_L3: assumes A1: \( i \leq m \)
  and A2: \( i..m \in \text{Fin}(\mathbb{Z}) \)
  shows \( i..m = i..(m+1) \)
proof -
  from A1 have T1: m \leq j \leq (k + \#1) using Order_ZF_2_L1A by auto
  then have T2: m \in \mathbb{Z} \land j \in \mathbb{Z} using Int_ZF_2_L1A by auto
  from T1 have \( (m \leq j) \land (j \leq k) \lor j \in \{k \land \#1\} \)
    using Int_ZF_4_L1 by auto
  then show thesis using Order_ZF_2_L1B 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 \( i..m \leq m \) using Int_ZF_2_T1 by auto
proof -
  from A1 have T1: m \leq j \leq (k + \#1) using Order_ZF_2_L1A by auto
  then have T2: m \in \mathbb{Z} \land j \in \mathbb{Z} using Int_ZF_2_L1A by auto
  from T1 have \( (m \leq j) \land (j \leq k) \lor j \in \{k \land \#1\} \)
    using Int_ZF_4_L1 by auto
  then show thesis using Order_ZF_2_L1B by auto
qed

505
shows $i..(m\#+\#1)\in\mathrm{Fin}(\mathbb{Z})$
using assms Int_ZF_4_L3 by simp

Integer intervals are finite.

**Lemma (in int0)** Int_ZF_4_L5: assumes $A_1: \ i\in\mathbb{Z} \ k\in\mathbb{Z}$
shows $i..k\in\mathrm{Fin}(\mathbb{Z})$
proof -
{ assume $A_2: i\leq k$
  moreover from $A_1$ have $i..i\in\mathrm{Fin}(\mathbb{Z})$
    using int_ord_is_refl Int_ZF_2_L4 Order_ZF_2_L4 by simp
  moreover from $A_2$ have $\forall m. i\leq m \land i..m\in\mathrm{Fin}(\mathbb{Z})\rightarrow i..(m\#+\#1)\in\mathrm{Fin}(\mathbb{Z})$
    using Int_ZF_4_L4 by simp
  ultimately have $i..k\in\mathrm{Fin}(\mathbb{Z})$ by (rule Int_ZF_3_L7) }
moreover
{ assume $\neg i\leq k$
  then have $i..k\in\mathrm{Fin}(\mathbb{Z})$ using Int_ZF_2_L6 Order_ZF_2_L5 by simp }
ultimately show thesis by blast
qed

Bounded integer sets are finite.

**Lemma (in int0)** Int_ZF_4_L6: assumes $A_1: \text{IsBounded}(A,\text{IntegerOrder})$
shows $A\in\mathrm{Fin}(\mathbb{Z})$
proof -
  have $T_1: \forall m\in\text{Nonnegative}(\mathbb{Z},\text{IntegerAddition},\text{IntegerOrder}).$
    $\#0..m\in\mathrm{Fin}(\mathbb{Z})$
  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 $\#0..m\in\mathrm{Fin}(\mathbb{Z})$ using Int_ZF_4_L5 by simp
  qed
  have group3($\mathbb{Z},\text{IntegerAddition},\text{IntegerOrder}$)
    using Int_ZF_2_T1 by simp
  moreover from $T_1$ have $\forall m\in\text{Nonnegative}(\mathbb{Z},\text{IntegerAddition},\text{IntegerOrder}).$
    $\text{Interval}(\text{IntegerOrder},\text{TheNeutralElement}(\mathbb{Z},\text{IntegerAddition}),m)\in\mathrm{Fin}(\mathbb{Z})$
    using Int_ZF_1_L8 by simp
  moreover note $A_1$
  ultimately show $A\in\mathrm{Fin}(\mathbb{Z})$ by (rule group3.OrderedGroup_ZF_2_T1)
qed

A subset of integers is bounded iff it is finite.

**Theorem (in int0)** Int_bounded_iff_fin:
shows $\text{IsBounded}(A,\text{IntegerOrder})\iff A\in\mathrm{Fin}(\mathbb{Z})$
using Int_ZF_4_L6 Int_ZF_2_T1 group3.ord_group_fin_bounded by blast

The image of an interval by any integer function is finite, hence bounded.
lemma (in int0) Int_ZF_4_L8:
assumes A1: \( i \in \mathbb{Z} \) \( k \in \mathbb{Z} \) and A2: \( f: \mathbb{Z} \rightarrow \mathbb{Z} \)
shows \( f(i..k) \in \text{Fin}(\mathbb{Z}) \)
IsBounded\( (f(i..k), \text{IntegerOrder}) \)
using assms Int_ZF_4_L5 Finite1_L6A Int_bounded_iff_fin
by auto

If for every integer we can find one in \( A \) that is greater or equal, then \( A \) is
is not bounded above, hence infinite.

lemma (in int0) Int_ZF_4_L9: assumes A1: \( \forall m \in \mathbb{Z}. \exists k \in A. m \leq k \)
shows \( \neg \text{IsBoundedAbove}(A, \text{IntegerOrder}) \)
\( A \notin \text{Fin}(\mathbb{Z}) \)
proof -
  have \( \mathbb{Z} \neq \{0\} \)
    using Int_ZF_1_L8A int_zero_not_one
  by blast
  with A1 show
    \( \neg \text{IsBoundedAbove}(A, \text{IntegerOrder}) \)
    \( A \notin \text{Fin}(\mathbb{Z}) \)
  using Int_ZF_2_T1 group3.OrderedGroup_ZF_2_L2A
  by auto
qed

end

46 Integers 1

theory Int_ZF_1 imports Int_ZF_IML OrderedRing_ZF
begin

This theory file considers the set of integers as an ordered ring.

46.1 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: assumes A1: \( a \in \mathbb{Z} \) \( b \in \mathbb{Z} \) \( c \in \mathbb{Z} \)
shows \( a \cdot (b+c) = a \cdot b + a \cdot c \)
\( (b+c) \cdot a = b \cdot a + c \cdot a \)
using assms Int_ZF_1_L2 zadd_zmult_distrib zadd_zmult_distrib2
by auto

507
Integers form a commutative ring, hence we can use theorems proven in 
ring0 context (locale).

**lemma (in int0) Int_ZF_1_1_L2:** shows
IsAring($\mathbb{Z}$, IntegerAddition, IntegerMultiplication)
IntegerMultiplication {is commutative on} $\mathbb{Z}$
proof -
  have $\forall a \in \mathbb{Z}. \forall b \in \mathbb{Z}. \forall c \in \mathbb{Z}.
      a \cdot (b+c) = a \cdot b + a \cdot c$ 
      $(b+c) \cdot a = b \cdot a + c \cdot a$
      using Int_ZF_1_1_L1 by simp
  then have IsDistributive($\mathbb{Z}$, IntegerAddition, IntegerMultiplication)
      using IsDistributive_def by simp
  then show IsAring($\mathbb{Z}$, IntegerAddition, IntegerMultiplication)
      ring0($\mathbb{Z}$, IntegerAddition, IntegerMultiplication)
      using Int_ZF_1_T1 Int_ZF_1_T2 IsAring_def ring0_def by auto
  have $\forall a \in \mathbb{Z}. \forall b \in \mathbb{Z}.
      a \cdot b = b \cdot a$ using Int_ZF_1_L4 by simp
  then show IntegerMultiplication {is commutative on} $\mathbb{Z}$
      using IsCommutative_def by simp
qed

Zero and one are integers.

**lemma (in int0) int_zero_one_are_int:** shows $0 \in \mathbb{Z}$  $1 \in \mathbb{Z}$
using Int_ZF_1_1_L2 ring0.Ring_ZF_1_L2 by auto

Negative of zero is zero.

**lemma (in int0) int_zero_one_are_intA:** shows $(-0) = 0$
using Int_ZF_1_1_L2 group0.group_inv_of_one by simp

Properties with one integer.

**lemma (in int0) Int_ZF_1_1_L4:** assumes $A1: a \in \mathbb{Z}$
shows
$a+0 = a$
$0+a = a$
$a \cdot 1 = a$
$1 \cdot a = a$
$0 \cdot a = 0$
$a \cdot 0 = 0$
$(-a) \in \mathbb{Z}$  $(-(-a)) = a$
$a-a = 0$
$a-0 = a$
$2 \cdot a = a+a$
proof -
  from $A1$ show
    $a+0 = a$
    $0+a = a$
    $a \cdot 1 = a$
    $1 \cdot a = a$
    $a-a = 0$
    $a-0 = 0$
    $(-a) \in \mathbb{Z}$  $2 \cdot a = a+a$
    $(-(-a)) = a$
      using Int_ZF_1_1_L2 ring0.Ring_ZF_1_L3 by auto
  from $A1$ show $0 \cdot a = 0$
    $a \cdot 0 = 0$
      using Int_ZF_1_1_L2 ring0.Ring_ZF_1_L6 by auto
qed

Properties that require two integers.

508
lemma (in int0) Int_ZF_1_1_L5: assumes $a \in \mathbb{Z} \quad b \in \mathbb{Z}$
shows $a+b \in \mathbb{Z} \\
 a-b \in \mathbb{Z} \\
 a \cdot b \in \mathbb{Z} \\
a+b = b+a \\
a \cdot b = b \cdot a \\
(-b)-a = (-a)-b \\
(-(a+b)) = (-a)-b \\
(-(a-b)) = ((-a)+b) \\
(-a) \cdot b = -(a \cdot b) \\
 a \cdot (-b) = -(a \cdot b) \\
(-a) \cdot (-b) = a \cdot b$
using assms Int_ZF_1_1_L2 ring0.Ring_ZF_1_L9 
ring0.Ring_ZF_1_L7A Int_ZF_1_L4 by auto

2 and 3 are integers.

lemma (in int0) Int_ZF_1_1_L5B: assumes $a \in \mathbb{Z} \quad b \in \mathbb{Z} \\
 \quad c \in \mathbb{Z}$
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 $a \in \mathbb{Z} \quad b \in \mathbb{Z} \\
 \quad c \in \mathbb{Z}$
shows $a-(b+c) = a-b-c \\
 a-(b-c) = a-b+c \\
 a \cdot (b-c) = a \cdot b - a \cdot c \\
(b-c) \cdot a = b \cdot a - c \cdot a$
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 $a \in \mathbb{Z} \quad b \in \mathbb{Z} \\
 \quad c \in \mathbb{Z}$
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 $a \in \mathbb{Z} \quad b \in \mathbb{Z} \\
 \quad c \in \mathbb{Z}$
shows $a+b+c = a+(b+c) \\
 a \cdot b \cdot c = a \cdot (b \cdot c)$ 
using assms Int_ZF_1_1_L2 ring0.Ring_ZF_1_L11 by auto
46.2 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 $0 \leq a$
shows $\text{abs}(a) + 1 = \text{abs}(a+1)$
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 $A1: a \in \mathbb{Z}$ and $A2: 0 \leq b$
shows $a + (\text{abs}(b) + 1) \cdot a = (\text{abs}(b+1) + 1) \cdot a$
proof -
  from $A2$ have $\text{abs}(b+1) \in \mathbb{Z}$
  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 $A1: a \in \mathbb{Z}$ $b \in \mathbb{Z}$
shows
  $a + b - a = b$
  $a + (b-a) = b$
  $a + b - b = a$
  $a - b + b = a$
  $(-a) + (a+b) = b$
  $a + (b-a) = b$
  $(-b) + (a+b) = a$
  $a - (b+a) = -b$
  $a - (a+b) = -b$
  $a - (a-b) = b$
  $a - b - (a + b) = (-b) - b$
using assms Int_ZF_1_2_L2 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 $A1: a \leq b$
shows $a - 1 \leq b$
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 $a - 1 \leq b+1 - 1$

510
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 ∈ ℤ 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_L14 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)
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_L2 Ring0.Ring_ZF_2_L1
  by simp
qed

A rearrangement with four integers.

lemma (in int0) Int_ZF_1_2_L6:
assumes $A1: a \in \mathbb{Z}, b \in \mathbb{Z}, c \in \mathbb{Z}, d \in \mathbb{Z}$
shows
\[ a-(b-1)c = (d-bc)-(d-a-c) \]
proof -
from $A1$ have $T1$:
\[ (d-bc) \in \mathbb{Z}, d-a \in \mathbb{Z}, \neg((bc)) \in \mathbb{Z} \]
using Int_ZF_1_1_L5 Int_ZF_1_1_L4 by auto
with $A1$ have
\[ (d-bc)-(d-a-c) = \neg((bc))+a+c \]
using Int_ZF_1_1_L6 Int_ZF_1_2_L3 by simp
also from $A1$ $T1$ have $\neg((bc))+a+c = a-(b-1)c$
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 $A1: a \in \mathbb{Z}, b \in \mathbb{Z}$
shows
\[ a \cdot b = (a-1) b + b \]
\[ a \cdot (b+1) = a b + a \]
\[ (b+1) a = b a + a \]
\[ (b+1) a = a + a b \]
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_L5 Int_ZF_1_1_L6 Int_ZF_1_1_L4 Int_ZF_1_1_L5
by auto

Another rearrangement with two integers.

lemma (in int0) Int_ZF_1_2_L8:
assumes $A1: a \in \mathbb{Z}, b \in \mathbb{Z}$
shows $a+1+(b+1) = b + a + 2$
using assms int_zero_one_are_int Int_ZF_1_1_L5 Int_ZF_1_1_L4 Int_ZF_1_1_L2
by simp

A couple of rearrangement with three integers.

lemma (in int0) Int_ZF_1_2_L9:
assumes $A1: a \in \mathbb{Z}, b \in \mathbb{Z}, c \in \mathbb{Z}$
shows
\[ (a-b)+(b-c) = a-c \]
\[ (a-b)-(a-c) = c-b \]
\[ a+(b+(c-a-b)) = c \]
\[ (-a)-b+c = c-a-b \]
\[ (-b)-a+c = c-a-b \]
\[ (-((-a)+b+c)) = a-b-c \]
\[ a+b+c-a = b+c \]
\[ a+b-(a+c) = b-c \]
using assms Int_ZF_1_1_T2
Int_ZF_1_1_L5 Int_ZF_1_1_L4 Int_ZF_1_1_L5 Int_ZF_1_1_L4 Int_ZF_1_1_L5
by auto

512
Another couple of rearrangements with three integers.

lemma (in int0) Int_ZF_1_2_L9A:

assumes A1: \( a \in \mathbb{Z} \) \( b \in \mathbb{Z} \) \( c \in \mathbb{Z} \)

shows \((-(a-b-c)) = c+b-a\)

proof -

from A1 have T:

\( a-b \in \mathbb{Z} \) \( (-(a-b)) \in \mathbb{Z} \) \( (-b) \in \mathbb{Z} \) 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 \( \ldots = c+b-a\)

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: \( a \in \mathbb{Z} \) \( b \in \mathbb{Z} \) \( c \in \mathbb{Z} \)

shows \((a+1)\cdot b + (c+1)\cdot b = (c+a+2)\cdot b\)

proof -

from A1 have \( a+1 \in \mathbb{Z} \) \( c+1 \in \mathbb{Z} \)

using int_zero_one_are_int Int_ZF_1_1_L5 by auto

with A1 have \((a+1)\cdot b + (c+1)\cdot b = (a+1+(c+1))\cdot b\)

using Int_ZF_1_1_L1 by simp

also from A1 have \( \ldots = (c+a+2)\cdot b\)

using Int_ZF_1_2_L8 by simp

finally show thesis by simp

qed

A technical rearrangement involving inequalities with absolute value.

lemma (in int0) Int_ZF_1_2_L10A:

assumes A1: \( a \in \mathbb{Z} \) \( b \in \mathbb{Z} \) \( c \in \mathbb{Z} \) \( e \in \mathbb{Z} \)

and A2: \( \text{abs}(a-b-c) \leq d \) \( \text{abs}(b-a-e) \leq f \)

shows \( \text{abs}(c-e) \leq f+d \)

proof -

from A1 A2 have T1:

\( d \in \mathbb{Z} \) \( f \in \mathbb{Z} \) \( a-b \in \mathbb{Z} \) \( a-b-c \in \mathbb{Z} \) \( b-a-e \in \mathbb{Z} \)

using Int_ZF_2_L1A Int_ZF_1_1_L5 by auto

with A2 have \( \text{abs}((b-a-e)-(a-b-c)) \leq f +d \)

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∈Z
shows a+1+2 = a+3
a = 2·a - a
proof -
from A1 show a+1+2 = a+3
using int_zero_one_are_int int_two_three_are_int Int_ZF_1_1_L4C
by simp
from A1 show a = 2·a - a
using int_zero_one_are_int Int_ZF_1_1_L6 Int_ZF_1_1_L5
by simp
qed

A simple rearrangement with three integers.

**lemma** (in int0) Int_ZF_1_2_L12: assumes a∈Z, b∈Z, c∈Z
shows (b-c)·a = a·b - a·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∈Z, b∈Z, c∈Z, d∈Z, x∈Z
shows (x+(a·x+b)+c)·d = d·(a+1)·x + (b·d+c·d)
proof -
from A1 have T1: a·x ∈ Z (a+1)·x ∈ Z
(a+1)·x + b ∈ Z
using Int_ZF_1_1_L5 int_zero_one_are_int by auto
with A1 have (x+(a·x+b)+c)·d = ((a+1)·x + b)·d + c·d
using Int_ZF_1_1_L1 Int_ZF_1_2_L7 Int_ZF_1_1_L1
by simp
also from A1 T1 have ... = (a+1)·x·d + b · d + c·d
using Int_ZF_1_1_L1 by simp
finally have (x+(a·x+b)+c)·d = (a+1)·x·d + b·d + c·d
by simp
moreover from A1 T1 have (a+1)·x·d = d·(a+1)·x
using int_zero_one_are_int Int_ZF_1_1_L5 Int_ZF_1_1_L7 by simp
ultimately have (x+(a·x+b)+c)·d = d·(a+1)·x + b·d + c·d
by simp
moreover from A1 T1 have d·(a+1)·x ∈ Z, b·d ∈ Z, c·d ∈ 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:
  \( \text{group0} (G, f) \) \( f \) \{is commutative on\} \( G \)
  \( a \in G \) \( b \in G \) \( c \in G \) \( d \in G \)
  \( a = f(f(b,\text{GroupInv}(G, f)(c)),\text{GroupInv}(G, f)(d)) \)
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 I have d = f(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 I have b = f(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 \( a+b-c+(c-b) = a \) \( a+(b+c)-c = a+b \)
using assms Int_ZF_1_1_T2 group0.group0_4_L8 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 \( G = \mathbb{Z} \)
let \( f = \text{IntegerAddition} \)

515
from A1 have I:
  group0(G, f)
  a ∈ G  b ∈ G c ∈ G
  using Int_ZF_1_T2 by auto
then have
  f(f(f(a,b),GroupInv(G, f)(c)),f(c,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)),GroupInv(G, f)(c)) = f(a,b)
  by (rule group0.group0_2_L14A)
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∈ℤ  b∈ℤ  c∈ℤ
  shows a+b-c+(c-a) = b
proof -
from A1 have
  group0(G, f)  f {is commutative on} G
  a ∈ G  b ∈ G c ∈ G
  using Int_ZF_1_T2 by auto
then have
  f(f(f(a,b),GroupInv(G, f)(c)),f(c,GroupInv(G, f)(a))) = b
  by (rule group0.group0_4_L6D)
thus a+b-c+(c-a) = b by simp
qed

46.3 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≤a  0≤b
  and A3: 0 ≤ a·b
  shows 0 ≤ a·(b+1)
proof -
from A1 A3 have 0·0 ≤ a·b+a
  using int.ineq_add_sides by simp
with A1 show 0 ≤ a·(b+1)
  using int_zero_one_are_int Int_ZF_1_1_L4 Int_ZF_2_L1A Int_ZF_1_2_L7

516
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 \(0 \leq a\) 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 \(\forall m. 0 \leq m \wedge 0 \leq a \cdot m \rightarrow 0 \leq a \cdot (m+1)\)
    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_T1 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_L6 OrdRing_ZF_1_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: \(1 \leq a\) \(1 \leq b\)
  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 that one is greater than one.

lemma (in int0) Int_ZF_1_3_L3:
  assumes A1: $1 \leq a$
  shows $1 \leq a \cdot b$
proof -
  from A1 have $1 \leq b$
  proof -
    using Int_ZF_2_L1A Int_ZF_1_1_L4 by auto
  moreover from A1 have
    $\forall m. 1 \leq m \rightarrow 1 \leq a \cdot (m+1)$
    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 $a \in \mathbb{Z}$ $b \in \mathbb{Z}$
  shows $\text{abs}((-a) \cdot b) = \text{abs}(a \cdot b)$
  $\text{abs}(a \cdot (-b)) = \text{abs}(a \cdot b)$
  $\text{abs}((-a) \cdot (-b)) = \text{abs}(a \cdot b)$
  using assms Int_ZF_1_3_T1 ring1.OrdRing_ZF_2_L5 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 $a \in \mathbb{Z}$ $b \in \mathbb{Z}$
  shows $\text{abs}(a \cdot b) = \text{abs}(a) \cdot \text{abs}(b)$
  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_Leq bitten 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 $a = b$ $\lor$ $a+1 \leq b$
    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 \( b \leq a \lor a+1 \leq b \)
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 \( T: a \in \mathbb{Z} \ b \in \mathbb{Z} \) 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 \( a+1 \leq b \)
  \( a \leq b-1 \)
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 \( \text{A1: } a \in \mathbb{Z} \) \( \text{b} \in \mathbb{Z} \)
shows \( a = b \lor (a \leq b - 1) \lor (b + 1 \leq a) \)
proof -
  from \( \text{A1} \) have \( a = b \lor (a \leq b - 1) \lor (b + 1 \leq a) \)
  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 \( \text{A1: } a \in \mathbb{Z} \)
shows \( a = 0 \lor (a \leq -1) \lor (1 \leq a) \)
proof -
  from \( \text{A1} \) have \( a = 0 \lor (a \leq -1) \lor (1 \leq a) \)
  using int_zero_one_are_int Int_ZF_1_3_L6 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 \( \text{a} \in \mathbb{Z} \)
shows \( \neg (a \leq 0) \iff 1 \leq a \)
using assms int_zero_one_are_int 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 \( \text{a} \in \mathbb{Z} \) \( \text{b} \in \mathbb{Z} \)
  and \( \neg (a \leq 0) \) \( \neg (b \leq 0) \)
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 \( \text{A1: } a \in \mathbb{Z} \) \( \text{b} \in \mathbb{Z} \)
  and \( \text{A2: } \neg (b \leq 0) \) \( \text{A3: } a \cdot b \leq 0 \)
shows \( a \leq 0 \)
proof -
  { assume \( \neg (a \leq 0) \)
    with \( \text{A1} \) \( \text{A2} \) have \( \neg ((a \cdot b) \leq 0) \) using Int_ZF_1_3_L8

520
by simp 
} with A3 show a≤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∈\mathbb{Z}  b∈\mathbb{Z}"  
shows "a≤b ↔ a-b ≤ 0"  
using assms 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≤b"  
shows "0 ≤ b-a"  
using assms 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 "a∈\mathbb{Z}  b∈\mathbb{Z}  c∈\mathbb{Z}"  
and A2: "a·c ≤ b·c" and A4: "¬(c≤0)"  
shows "a ≤ b"  
proof -  
  from A1 A4 have a-b ∈ \mathbb{Z}  c∈\mathbb{Z}  "¬(c≤0)"  
    using Int_ZF_1_1_L5 by auto  
  moreover from A1 A2 have (a-b)·c ≤ 0  
    using Int_ZF_1_1_L5 Int_ZF_1_3_L10 Int_ZF_1_1_L6 
    by simp  
  ultimately have a-b ≤ 0 by (rule Int_ZF_1_3_L9) 
  with A1 show a ≤ 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∈\mathbb{Z}  b∈\mathbb{Z}"  
and A2: "¬(abs(a) ≤ abs(b))"  
shows "¬(abs(a) ≤ 0)"  
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∈\mathbb{Z}_+
shows a·c ≤ b·c
c·a ≤ c·b
proof -
let R = \mathbb{Z}
let A = IntegerAddition
let M = IntegerMultiplication
let r = IntegerOrder
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
A rearrangement with four integers and absolute value.

**Lemma (in int0) Int_ZF_1_3_L14:**

- **Assumptions:**
  - \( a \in \mathbb{Z} \) \( b \in \mathbb{Z} \) \( c \in \mathbb{Z} \) \( d \in \mathbb{Z} \)
- **Shows:**
  - \( \text{abs}(a \cdot b) + (\text{abs}(a) + c) \cdot d = (d + \text{abs}(b)) \cdot \text{abs}(a) + c \cdot d \)

**Proof**:
- From A1 have T1:
  - \( \text{abs}(a) \in \mathbb{Z} \) \( \text{abs}(b) \in \mathbb{Z} \)
  - \( \text{abs}(a) \cdot \text{abs}(b) \in \mathbb{Z} \)
  - \( c \cdot d \in \mathbb{Z} \)
  - \( \text{abs}(b) + d \in \mathbb{Z} \)
  - Using Int_ZF_2_L14 Int_ZF_1_1_L5 by auto
- With A1 have \( \text{abs}(a \cdot b) + (\text{abs}(a) + c) \cdot d = \text{abs}(a) \cdot (\text{abs}(b) + d) + c \cdot d \)
  - 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:**

- **Assumptions:**
  - \( m \in \mathbb{Z} \) \( n \in \mathbb{Z} \)
  - \( \neg(\text{abs}(m) \leq \text{abs}(n)) \)
- **Shows:**
  - \( n \leq \text{abs}(m) \) \( m \neq 0 \)

**Proof**:
- From A1 have T1:
  - \( n \leq \text{abs}(n) \)
  - Using Int_ZF_2_L19C by simp
- From A1 have \( \text{abs}(n) \in \mathbb{Z} \) \( \text{abs}(m) \in \mathbb{Z} \)
  - Using Int_ZF_2_L14 by auto
- Moreover note A2
- Ultimately have \( \text{abs}(n) \leq \text{abs}(m) \)
  - By (rule Int_ZF_2_L19)
- With T1 show \( n \leq \text{abs}(m) \) by (rule Int_order_transitive)
- From A1 A2 show \( m \neq 0 \) 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:**
  - \( 0 \leq m \)
- **Shows:**
  - \( (-m) \leq 0 \)

**Proof**:
- 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:**
  - \( m \in \mathbb{Z} \)
shows

\[ (-\text{abs}(m)) \leq \text{abs}(m) \]
\[ (-\text{abs}(m)) \ldots \text{abs}(m) \neq 0 \]

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 \(\text{abs}(m) \in (-\text{abs}(m)) \ldots \text{abs}(m)\)
  using int_ord_is_refl Int_ZF_2_L1A Order_ZF_2_L2 by simp
thus \((-\text{abs}(m)) \ldots \text{abs}(m) \neq 0\) 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 \(m \in -n \ldots n\).

lemma (in int0) Int_ZF_1_3_L19: assumes A1: \(m \in \mathbb{Z}\) and A2: \(\text{abs}(m) \leq n\)
shows \((-n) \leq m\) \(m \leq n\)
\(m \in (-n) \ldots n\)
\(0 \leq n\)
using assms Int_ZF_2_T1 group3.OrderedGroup_ZF_3_L8
  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 \((-\text{n+k}) \leq m\)
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: \(\forall x \in X.\) \(b(x) \in \mathbb{Z}\) \(\wedge \text{abs}(b(x)) \leq L\)
shows IsBounded({\{b(x). \(x \in X\)}, \text{IntegerOrder}})
proof -
  let \(G = \mathbb{Z}\)
  let \(P = \text{IntegerAddition}\)
  let \(r = \text{IntegerOrder}\)
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)
  by (rule group3.OrderedGroup_ZF_3_L9B)
qed

A special case of the previous lemma for a function from integers to integers.

corollary (in int0) Int_ZF_1_3_L20C: assumes f:Z→Z and ∀m∈Z. abs(f(m)) ≤ L
  shows f(Z) ∈ Fin(Z)
proof -
  from assms have \( f: \mathbb{Z} \rightarrow \mathbb{Z} \subseteq \mathbb{Z} \quad \forall m \in \mathbb{Z}. \ abs(f(m)) \leq L \)
  by auto
  then have IsBounded(\( f(\mathbb{Z}) \)),IntegerOrder
  by (rule Int_ZF_1_3_L20B)
  then show \( f(\mathbb{Z}) \in \text{Fin}(\mathbb{Z}) \) using 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: \( a \in \mathbb{Z} \quad b \in \mathbb{Z} \quad c \in \mathbb{Z} \)
  shows \( \abs{a-b-c} \leq \abs{a} + \abs{b} + \abs{c} \)
proof -
  from A1 have T: \( a-b \in \mathbb{Z} \quad \abs{c} \in \mathbb{Z} \)
  using Int_ZF_1_1_L5 Int_ZF_2_L14 by auto
  with A1 have \( \abs{a-b-c} \leq \abs{a-b} + \abs{c} \)
  using Int_triangle_ineq1 by simp
  moreover from A1 T have \( \abs{a-b} + \abs{c} \leq \abs{a} + \abs{b} + \abs{c} \)
  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: \( a \leq c \quad b \leq c \)
  shows \( a+b \leq 2 \cdot c \)
using assms Int_ZF_1_3_T1 ring1.OrdRing_ZF_2_L6 by simp

If an integer \( a \) is between \( b \) and \( b + c \), then \( \abs{b-a} \leq c \). Property of ordered groups.

lemma (in int0) Int_ZF_1_3_L22:
  assumes A1: \( a \leq b \) and c\( \in \mathbb{Z} \) and \( b \leq c+a \)
  shows \( \abs{b-a} \leq c \)
using assms 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 A1: \( a \in \mathbb{Z} \quad b \in \mathbb{Z} \quad c \in \mathbb{Z} \quad d \in \mathbb{Z} \)
  shows \( \abs{a-c} \leq \abs{a+b} + \abs{c+d} + \abs{b-d} \)
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 \( \abs{b-a} \leq c \). Property of ordered groups. A version of Int_ZF_1_3_L22 with sligtly different assumptions.
lemma (in int0) Int_ZF_1_3_L23:
assumes A1: a≤b and A2: c∈\mathbb{Z} and A3: b≤ a+c
shows abs(b-a) ≤ c
proof -
  from A1 have a ∈ \mathbb{Z}
    using Int_ZF_2_L1A by simp
  with A2 A3 have b≤ c+a
    using Int_ZF_1_1_L5 by simp
  with A1 A2 show abs(b-a) ≤ c
    using Int_ZF_1_3_L22 by simp
qed

46.4 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 ∈ Fin(\mathbb{Z}) and A2: A≠0
shows
  HasAmaximum(IntegerOrder,A)
  HasAminimum(IntegerOrder,A)
  Maximum(IntegerOrder,A) ∈ A
  Minimum(IntegerOrder,A) ∈ A
  \forall x ∈ A. x ≤ Maximum(IntegerOrder,A)
  \forall x ∈ A. Minimum(IntegerOrder,A) ≤ x
  Maximum(IntegerOrder,A) ∈ \mathbb{Z}
  Minimum(IntegerOrder,A) ∈ \mathbb{Z}
proof -
  from A1 have
    A=0 ∨ HasAmaximum(IntegerOrder,A) and
    A=0 ∨ HasAminimum(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
    HasAmaximum(IntegerOrder,A)
    HasAminimum(IntegerOrder,A)
    by auto
  from A1 A2 show
    Maximum(IntegerOrder,A) ∈ A
    Minimum(IntegerOrder,A) ∈ A
    \forall x ∈ A. x ≤ Maximum(IntegerOrder,A)
    \forall x ∈ A. Minimum(IntegerOrder,A) ≤ x
    using Int_ZF_2_T1 Finite_ZF_1_1_T2 by auto
  moreover from A1 have A⊆\mathbb{Z} using FinD by simp
  ultimately show
    Maximum(IntegerOrder,A) ∈ \mathbb{Z}
    Minimum(IntegerOrder,A) ∈ \mathbb{Z}
Bounded nonempty integer subsets attain maximum and minimum.

**Theorem (in int0) Int bounded have max min:**

- Assumes `IsBounded(A,IntegerOrder)` and `A≠0`  
- Shows `HasAmaximum(IntegerOrder,A)` and `HasAminimum(IntegerOrder,A)`  
- Proves `∀x∈A. x ≤ Maximum(IntegerOrder,A)` and `∀x∈A. Minimum(IntegerOrder,A) ≤ x`  
- Uses `assms Int_fin_have_max_min Int_bounded_iff_fin`  
- Applies `auto` for proof.

Nonempty set of integers that is bounded below attains its minimum.

**Theorem (in int0) Int bounded below has min:**

- Assumes `IsBoundedBelow(A,IntegerOrder)` and `A≠0`  
- Shows `HasAminimum(IntegerOrder,A)` and `Minimum(IntegerOrder,A)∈A`  
- Proves `∀x∈A. Minimum(IntegerOrder,A) ≤ x`  
- Uses `assms Int_ZF_2_T1 Int_ZF_2_L6 Int_ZF_2_L1B Int_bounded_have_max_min`  
- Applies `auto` for proof.

Nonempty set of integers that is bounded above attains its maximum.

**Theorem (in int0) Int bounded above has max:**

- Assumes `IsBoundedAbove(A,IntegerOrder)` and `A≠0`  
- Shows `HasAmaximum(IntegerOrder,A)` and `Maximum(IntegerOrder,A)∈A`
Maximum(IntegerOrder,A) ∈ ℤ
∀x∈A. x ≤ Maximum(IntegerOrder,A)

proof -
from A1 A2 have
  IntegerOrder {is total on} ℤ
  trans(IntegerOrder) and
  I: IntegerOrder ⊆ ℤ×ℤ and
  ∀A. IsBounded(A,IntegerOrder) ∧ A≠0 → HasAmaximum(IntegerOrder,A)
  A≠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) ∈ A and
  ∀x∈A. x ≤ Maximum(IntegerOrder,A)
  using Int_ZF_2_L4 Order_ZF_4_L3 by auto
from I A1 have A ⊆ ℤ by (rule Order_ZF_3_L1A)
with II show Maximum(IntegerOrder,A) ∈ 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≠0
and A3: ∀q∈ℤ. F(q) ∈ ℤ
and A4: K = {F(q). q ∈ A}
shows
  HasAmaximum(IntegerOrder,K)
  HasAminimum(IntegerOrder,K)
  Maximum(IntegerOrder,K) ∈ K
  Minimum(IntegerOrder,K) ∈ K
  Maximum(IntegerOrder,K) ∈ ℤ
  Minimum(IntegerOrder,K) ∈ ℤ
  ∀q∈A. F(q) ≤ Maximum(IntegerOrder,K)
  ∀q∈A. Minimum(IntegerOrder,K) ≤ F(q)
  IsBounded(K,IntegerOrder)

proof -
from A1 have A ∈ Fin(ℤ) using Int_bounded_iff_fin
  by simp
with A3 have {F(q). q ∈ A} ∈ Fin(ℤ)
  by (rule fin_image_fin)
with A2 A4 have T1: K ∈ Fin(ℤ) K≠0 by auto
then show T2:
  HasAmaximum(IntegerOrder,K)
  HasAminimum(IntegerOrder,K)
  and Maximum(IntegerOrder,K) ∈ K
  Minimum(IntegerOrder,K) ∈ K
  Maximum(IntegerOrder,K) ∈ ℤ
A three element set has a maximum and minimum.

** lemma (in int0) Int_ZF_1_4_L1A: assumes A1: \( a \in \mathbb{Z} \quad b \in \mathbb{Z} \quad c \in \mathbb{Z} \) shows

Maximum(IntegerOrder,\{a,b,c\}) \in \mathbb{Z}

\( a \leq \) Maximum(IntegerOrder,\{a,b,c\})

\( b \leq \) Maximum(IntegerOrder,\{a,b,c\})

\( c \leq \) Maximum=IntegerOrder,\{a,b,c\})

using assms Int_ZF_2_T1 Finite_ZF_1_L2A by auto

Integer functions attain maxima and minima over intervals.

** lemma (in int0) Int_ZF_1_4_L2:

assumes A1: \( f: \mathbb{Z} \rightarrow \mathbb{Z} \) and A2: \( a \leq b \) shows

\( \text{maxf}(f,a..b) \in \mathbb{Z} \)

\( \forall c \in a..b. \quad f(c) \leq \text{maxf}(f,a..b) \)

\( \exists c \in a..b. \quad f(c) = \text{maxf}(f,a..b) \)

\( \text{minf}(f,a..b) \in \mathbb{Z} \)

\( \forall c \in a..b. \quad \text{minf}(f,a..b) \leq f(c) \)

\( \exists c \in a..b. \quad f(c) = \text{minf}(f,a..b) \)

proof -

from A2 have T: \( a \in \mathbb{Z} \quad b \in \mathbb{Z} \quad a..b \subseteq \mathbb{Z} \)

using Int_ZF_2_L1A Int_ZF_2_L1B Order_ZF_2_L6 by auto

with A1 A2 have

Maximum(IntegerOrder,\{a..b\}) \in f(a..b)

\( \forall x \in f(a..b). \quad x \leq \) Maximum(IntegerOrder,\{a..b\})

Maximum=IntegerOrder,\{a..b\}) \in \mathbb{Z}

Minimum=IntegerOrder,\{a..b\}) \in \mathbb{Z}

\( \forall x \in f(a..b). \quad \text{Minimum}(f,a..b) \leq x \)

\( \text{Minimum}(f,a..b) \in \mathbb{Z} \)

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 
\[ \forall c \in a..b. f(c) \leq \max f(f,a..b) \]
\[ \exists c \in a..b. f(c) = \max f(f,a..b) \]
\[ \min f(f,a..b) \in \mathbb{Z} \]
\[ \forall c \in a..b. \min f(f,a..b) \leq f(c) \]
\[ \exists c \in a..b. f(c) = \min f(f,a..b) \]
using func_imagedef by auto

qed

46.5 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 \( \mathbb{Z}^+ \) 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 \( \in \mathbb{Z}^+ \) and b \( \in \mathbb{Z}^+ \) shows a+b \( \in \mathbb{Z}^+ \) using assms pos_int_closed_add IsOpClosed_def by simp]

\( \mathbb{Z}^+ \) is bounded below.

[Lemma (in Int0) Int_ZF_1_5_L1: shows IsBoundedBelow(\( \mathbb{Z}^+ \),IntegerOrder) IsBoundedBelow(\( \mathbb{Z}^+ \),IntegerOrder) using Nonnegative_def PositiveSet_def IsBoundedBelow_def by auto]

Subsets of \( \mathbb{Z}^+ \) are bounded below.

[Lemma (in Int0) Int_ZF_1_5_L1A: assumes A \( \subseteq \mathbb{Z}^+ \) shows IsBoundedBelow(A,IntegerOrder) using assms Int_ZF_1_5_L1 Order_ZF_3_L12 by blast]

Subsets of \( \mathbb{Z}^+ \) are bounded below.

[Lemma (in Int0) Int_ZF_1_5_L1B: assumes A1: A \( \subseteq \mathbb{Z}^+ \) shows IsBoundedBelow(A,IntegerOrder) using A1 Int_ZF_1_5_L1 Order_ZF_3_L12 by blast]

Every nonempty subset of positive integers has a minimum.

[Lemma (in Int0) Int_ZF_1_5_L1C: assumes A \( \subseteq \mathbb{Z}^+ \) and A \( \neq 0 \) shows]
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) Int_ZF_1_5_L2:
assumes A1: $A \subseteq \mathbb{Z}^+$ and A2: $A \notin \text{Fin}(\mathbb{Z})$ and A3: $D \in \mathbb{Z}$
sows $\exists n \in A. \ D \leq n$
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 Nonnegative_def by auto
 ultimately have $\forall n \in A. \ n \leq D$
 using Int_ZF_2_L19 by blast
 hence $\forall n \in A. \ (n,D) \in \text{IntegerOrder}$ by simp
 then have IsBoundedAbove$(A,\text{IntegerOrder})$
 by (rule Order_ZF_3_L10)
 with A1 have IsBounded$(A,\text{IntegerOrder})$
 using Int_ZF_1_5_L1A IsBounded_def by simp
 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 postitive.

lemma (in int0) Int_decomp: assumes $m \in \mathbb{Z}$
Exactly 1 of 3 holds \( m=0, m \in \mathbb{Z}_+, (-m) \in \mathbb{Z}_+ \)

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 \( m \in \mathbb{Z} \)
shows \( m=0 \lor m \in \mathbb{Z}_+ \lor (-m) \in \mathbb{Z}_+ \)
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 \( m \in \mathbb{Z}_+ \iff 1 \leq m \)
proof
assume \( m \in \mathbb{Z}_+ \) then have \( 0 \leq m \) \( m \neq 0 \)
using PositiveSet_def by auto
then have \( 0+1 \leq m \)
using Int_ZF_4_L1B by auto
then show \( 1 \leq m \)
using int_zero_one_are_int Int_ZF_1_T2 group0.group0_2_L2
by simp
next assume \( 1 \leq m \)
then have \( m \in \mathbb{Z}_+ \) \( 0 \leq m \) \( m \neq 0 \)
using Int_ZF_2_L1A Int_ZF_2_L16C by auto
then show \( m \in \mathbb{Z}_+ \) 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 \( a \in \mathbb{Z}_+, b \in \mathbb{Z}_+ \)
shows \( a \cdot b \in \mathbb{Z}_+ \)
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 \( \mathbb{Z}_+ \) (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 HasNoZeroDivs(\( \mathbb{Z}, \text{IntegerAddition}, \text{IntegerMultiplication} \))
using pos_int_closed_mul Int_ZF_1_3_T1 ring1.OrdRing_ZF_3_L3 by simp

Nonnegative integers are positive ones plus zero.

**lemma (in int0) Int_ZF_1_5_L3A:** shows \( \mathbb{Z}^+ = \mathbb{Z}_+ \cup \{0\} \)
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 \( C = K - 1 \) - \( \max(f,1..(N-1)) \)
    assume \( 2 \leq N \)
    then have \( 2-1 \leq N-1 \)
      using int_zero_one_are_int Int_ZF_1_1_L4 int_ord_transl_inv
      by simp
    then have \( I: 1 \leq N-1 \)
      using int_zero_one_are_int_int_ZF_1_2_L3 by simp
    with \( A1 \) \( A2 \) have \( T: \max(f,1..(N-1)) \in \mathbb{Z} \), \( K-1 \in \mathbb{Z} \), \( C \in \mathbb{Z} \)
      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 \( n \in 1..(N-1) \)
      using Int_ZF_1_5_L3 Interval_def by auto
    with \( A1 \) \( T \) have \( f(n)+C \leq \max(f,1..(N-1)) + C \)
      using Int_ZF_1_4_L2 int_ord_transl_inv by simp
    with \( T \) have \( f(n)+C \leq K-1 \)
      using Int_ZF_1_2_L3 by simp
    with \( A5 \) have \( K \leq K-1 \)
  \}
  ultimately have \( \exists C \in \mathbb{Z}. \ \forall n \in \mathbb{Z}^+. \ K \leq f(n) + C \rightarrow N \leq n \)
  by auto

534
by (rule Int.order_transitive)
with \( A2 \) have \( \text{False} \) using Int.ZF_1.2_L3A by simp
\} \ then have \( K \leq f(n) + C \rightarrow N \leq n \)
by auto
\} \ then have \( \forall n \in \mathbb{Z}^+. \ K \leq f(n) + C \rightarrow N \leq n \)
by simp
ultimately have \( \exists C \in \mathbb{Z}. \ \forall n \in \mathbb{Z}^+. \ K \leq f(n) + C \rightarrow N \leq n \)
by auto \}
ultimately show thesis by auto

done
show \(0...(n-1) \subseteq \mathbb{Z}\)
using Int_ZF_2_L1B Order_ZF_2_L6 by simp
qed

Integers greater than one in \(\mathbb{Z}_+\) belong to \(\mathbb{Z}_+\). This is a property of ordered groups and follows from \texttt{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 \(a \in \mathbb{Z}_+\) and \(a \leq b\)
shows \(b \in \mathbb{Z}_+\)
proof-
from assms have \(1 \leq a\) \(a \leq b\)
  using Int_ZF_1_5_L3 by auto
then have \(1 \leq b\) by (rule Int_order_transitive)
then show \(b \in \mathbb{Z}_+\) using Int_ZF_1_5_L3 by simp
qed

Adding a positive integer increases integers.

lemma (in int0) Int_ZF_1_5_L7A: assumes \(a \in \mathbb{Z}_+\) \(b \in \mathbb{Z}_+\)
shows \(a \leq a+b\) \(a \neq a+b\) \(a+b \in \mathbb{Z}\)
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 \(a \in \mathbb{Z}_+\)
shows \(a \leq \text{GreaterOf}(\text{IntegerOrder},1,a)\)
\(\text{GreaterOf}(\text{IntegerOrder},1,a) + 1 \in \mathbb{Z}_+\)
\(a \leq \text{GreaterOf}(\text{IntegerOrder},1,a) + 1\)
\(a \neq \text{GreaterOf}(\text{IntegerOrder},1,a) + 1\)
using assms Int_zero_not_one Int_ZF_1_3_T1 ring1.OrdRing_ZF_3_L12 by auto

The opposite of an element of \(\mathbb{Z}_+\) cannot belong to \(\mathbb{Z}_+\).

lemma (in int0) Int_ZF_1_5_L8: assumes \(a \in \mathbb{Z}_+\)
shows \((-a) \notin \mathbb{Z}_+\)
using assms Int_ZF_2_T1 group3.OrderedGroup_ZF_1_L20 by simp

For every integer there is one in \(\mathbb{Z}_+\) that is greater or equal.

lemma (in int0) Int_ZF_1_5_L9: assumes \(a \in \mathbb{Z}\)
shows \(\exists b \in \mathbb{Z}_+. a \leq b\)
using assms Int_not_trivial Int_ZF_2_T1 group3.OrderedGroup_ZF_1_L23 by simp
A theorem about odd extensions. Recall from \texttt{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 $\mathbb{Z}$.

\textbf{Lemma} (in \texttt{int0}) \texttt{Int.ZF.1_5_L10}: assumes $f : \mathbb{Z}_+ \rightarrow \mathbb{Z}$ shows $\text{OddExtension}(\mathbb{Z}, \text{IntegerAddition}, \text{IntegerOrder}, f) : \mathbb{Z} \rightarrow \mathbb{Z}$ using assms \texttt{Int.ZF.2.T1 \ group3.odd_ext_props} by simp

On $\mathbb{Z}_+$, the odd extension of $f$ is the same as $f$.

\textbf{Lemma} (in \texttt{int0}) \texttt{Int.ZF.1_5_L11}: assumes $f : \mathbb{Z}_+ \rightarrow \mathbb{Z}$ and $a \in \mathbb{Z}_+$ and $g = \text{OddExtension}(\mathbb{Z}, \text{IntegerAddition}, \text{IntegerOrder}, f)$ shows $g(a) = f(a)$ using assms \texttt{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 $f(-a)$.

\textbf{Lemma} (in \texttt{int0}) \texttt{Int.ZF.1_5_L12}: assumes $f : \mathbb{Z}_+ \rightarrow \mathbb{Z}$ and $a \in (-\mathbb{Z}_+)$ and $g = \text{OddExtension}(\mathbb{Z}, \text{IntegerAddition}, \text{IntegerOrder}, f)$ shows $g(a) = -(f(-a))$ using assms \texttt{Int.ZF.2.T1 \ group3.odd_ext_props} by simp

Odd extensions are odd on $\mathbb{Z}$.

\textbf{Lemma} (in \texttt{int0}) \texttt{int_oddext_is_odd}: assumes $f : \mathbb{Z}_+ \rightarrow \mathbb{Z}$ and $a \in \mathbb{Z}$ and $g = \text{OddExtension}(\mathbb{Z}, \text{IntegerAddition}, \text{IntegerOrder}, f)$ shows $g(-a) = -(g(a))$ using assms \texttt{Int.ZF.2.T1 \ group3.oddext_is_odd} by simp

Alternative definition of an odd function.

\textbf{Lemma} (in \texttt{int0}) \texttt{Int.ZF.1_5_L13}: assumes $A1: f : \mathbb{Z} \rightarrow \mathbb{Z}$ shows $(\forall a \in \mathbb{Z}. f(-a) = -(f(a))) \iff (\forall a \in \mathbb{Z}. -(f(-a))) = f(a)$ using assms \texttt{Int.ZF.1.T2 \ group0.group0_6.L2} by simp

Another way of expressing the fact that odd extensions are odd.

\textbf{Lemma} (in \texttt{int0}) \texttt{int_oddext_is_odd_alt}: assumes $f : \mathbb{Z}_+ \rightarrow \mathbb{Z}$ and $a \in \mathbb{Z}$ and $g = \text{OddExtension}(\mathbb{Z}, \text{IntegerAddition}, \text{IntegerOrder}, f)$ shows $-(g(-a)) = g(a)$ using assms \texttt{Int.ZF.2.T1 \ group3.oddext_is_odd_alt} by simp

\subsection*{46.6 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 \texttt{OrderedGroup.ZF.thy} with the 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

\[
\forall a \in \mathbb{Z}. \exists b \in \mathbb{Z}^+. \forall x. \ b \leq x \rightarrow a \leq f(x) \quad \text{and} \quad A \subseteq \mathbb{Z} \quad \text{and}
\]

\[
\text{IsBoundedAbove}(f(A), \text{IntegerOrder})
\]

shows \( \text{IsBoundedAbove}(A, \text{IntegerOrder}) \)

using \( \text{assms int_not_trivial Int_ZF_2_T1 group3.OrderedGroup_ZF_7_L1} \)

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 \( A1: X \neq 0 \) and \( A2: f: \mathbb{Z} \rightarrow \mathbb{Z} \) and

\[
A3: \forall a \in \mathbb{Z}. \exists b \in \mathbb{Z}^+. \forall x. b \leq x \rightarrow a \leq f(x) \quad \text{and}
\]

\[
A4: \forall x \in X. \ b(x) \in \mathbb{Z} \quad \land \quad f(b(x)) \leq U
\]

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

\[
\text{group3}(G, P, r)
\]

\( r \) \{is total on\} \( G \)

\( G \neq \{\text{TheNeutralElement}(G, P)\} \)

\( X \neq 0 \quad f: G \rightarrow G \)

\( \forall a \in G. \exists b \in \text{PositiveSet}(G, P, r). \forall y. \langle b, y \rangle \in r \rightarrow \langle a, f(y) \rangle \in r \)

\( \forall x \in X. \ b(x) \in G \quad \land \quad \{f(b(x)), U\} \subseteq r \)

using \( \text{assms int_not_trivial Int_ZF_2_T1} \)

by auto

then have \( \exists u. \forall x \in X. \langle b(x), u \rangle \in r \) by \( \langle \text{rule group3.OrderedGroup_ZF_7_L2} \) thus thesis by simp

qed

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 \( A1: X \neq 0 \) and \( A2: f: \mathbb{Z} \rightarrow \mathbb{Z} \) and

\[
A3: \forall a \in \mathbb{Z}. \exists b \in \mathbb{Z}^+. \forall y. b \leq y \rightarrow f(-y) \leq a \quad \text{and}
\]

\[
A4: \forall x \in X. \ b(x) \in \mathbb{Z} \quad \land \quad L \leq f(b(x))
\]

shows \( \exists l. \forall x \in X. \ l \leq b(x) \)

proof -

let \( G = \mathbb{Z} \)

let \( P = \text{IntegerAddition} \)

let \( r = \text{IntegerOrder} \)

from \( A1 \) \( A2 \) \( A3 \) \( A4 \) have

538
group3(G, P, r)
r {is total on} G
G ≠ {TheNeutralElement(G, P)}
X≠0 f: G→G
∀a∈G. ∃b∈PositiveSet(G, P, r). ∀y. ⟨b, y⟩ ∈ r −→ ⟨f(Exponentiation(G, P), y), a⟩ ∈ r
∀x∈X. b(x) ∈ G ∧ ⟨L, f(b(x))⟩ ∈ r
using int_not_trivial Int_ZF_2_T1 by auto
then have ∃M. ∀x∈X. ⟨AbsoluteValue(G, P, r) b(x), M⟩ ∈ r
by (rule group3.OrderedGroup_ZF_7_L4)
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≠0 and A2: f: Š→Š and A3: ∀a∈Š. ∃b∈Š+. ∀x. b≤x −→ a≤f(x) and A4: ∀a∈Š. ∃b∈Š+. ∀y. b≤y −→ f(-y)≤a and A5: ∀x∈X. b(x) ∈ Š ∧ f(b(x)) ≤ U ∧ L ≤ f(b(x))
shows ∃M. ∀x∈X. abs(b(x)) ≤ M
proof -
let G = Š
let P = IntegerAddition
let r = IntegerOrder
from A1 A2 A3 A4 A5 have
  group3(G, P, r)
r {is total on} G
G ≠ {TheNeutralElement(G, P)}
X≠0 f: G→G
∀a∈G. ∃b∈PositiveSet(G, P, r). ∀y. ⟨b, y⟩ ∈ r −→ ⟨a, f(y)⟩ ∈ r
∀a∈G. ∃b∈PositiveSet(G, P, r). ∀y. ⟨b, y⟩ ∈ r −→ ⟨f(Exponentiation(G, P), y), a⟩ ∈ r
∀x∈X. b(x) ∈ G ∧ ⟨L, f(b(x))⟩ ∈ r ∧ ⟨f(b(x)), U⟩ ∈ r
using int_not_trivial Int_ZF_2_T1 by auto
then have ∃M. ∀x∈X. ⟨AbsoluteValue(G, P, r) b(x), M⟩ ∈ r
by (rule group3.OrderedGroup_ZF_7_L4)
thus thesis by simp
qed

If a function is larger than some constant for arguments large enough, then the image of a set defined by separation by a function with infinite limits 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: Š→Š and A2: N∈Š and
∀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_6_L1 by simp
with A3 A1 A2 have IsBoundedBelow(f(Z+),IntegerOrder)

540
by (rule Int_ZF_1_6_L5)
with A1 obtain l where I: ∀y∈f(Z⁺). l ≤ y
using Int_ZF_1_5_L5 IsBoundedBelow_def by auto
let c = K-l
from A3 have f(Z⁺) ≠ 0 using Int_ZF_1_5_L5
by simp
then have ∃y. y ∈ f(Z⁺) by (rule nonempty_has_element)
then obtain y where y ∈ f(Z⁺) by auto
with A4 I have T: l ∈ Z. c ∈ Z
using Int_ZF_2_L1A Int_ZF_1_1_L5 by auto
{ fix n assume A5: n ∈ Z⁺
  have Z⁺ ⊆ Z using PositiveSet_def by auto
  with A3 I T A5 have l + c ≤ f(n) + c
  using func_imagedef int_ord_transl_inv by auto
  with I T have l + c ≤ f(n) + c
  using int_ord_transl_inv by simp
  with A4 T have K ≤ f(n) + c
  using Int_ZF_1_2_L3 by simp
} then have ∀n∈Z⁺. K ≤ 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→Z and A2: K∈Z. N∈Z and
A3: ∀a∈Z. ∃b∈Z⁺. ∀x. b≤x −→ a ≤ f(x)
shows ∃C∈Z. N ≤ Minimum(IntegerOrder,{n∈Z⁺. K ≤ f(n)+C})
proof -
from A1 A2 have ∃C∈Z. ∀n∈Z⁺. K ≤ f(n) + C −→ N≤n
using Int_ZF_1_5_L4 by simp
then obtain C where I: C∈Z and
II: ∀n∈Z⁺. K ≤ f(n) + C −→ N≤n
by auto
have antisym(IntegerOrder) using Int_ZF_2_L4 by simp
moreover have HasMinimum(IntegerOrder,{n∈Z⁺. K ≤ f(n)+C})
proof -
from A2 A3 I have ∃n∈Z⁺. ∀x. n≤x −→ K-C ≤ f(x)
using Int_ZF_1_1_L5 by simp
then obtain n where
n∈Z⁺ and ∀x. n≤x −→ K-C ≤ f(x)
by auto
with A2 I have
{n∈Z⁺. K ≤ f(n)+C} ≠ 0
{n∈Z⁺. K ≤ f(n)+C} ⊆ Z⁺
using int_ord_is_refl refl_def PositiveSet_def Int_ZF_2_L9C by auto

541
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
⟨N,Minimum(IntegerOrder,\{n∈\mathbb{Z}_+. K ≤ f(n)+C\})⟩ ∈ 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\} ∉ 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\} ∉ 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

46.7 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 $A1: \forall q \in \mathbb{Z}. \ F(q) \in \mathbb{Z}$ and
$A2: \ \forall q \in \mathbb{Z}. \ F(q) \cdot \text{abs}(q) \leq A \cdot \text{abs}(q) + B$ and
$A3: A \in \mathbb{Z}. \ B \in \mathbb{Z}$
shows $\exists L. \ \forall p \in \mathbb{Z}. \ F(p) \leq L$

proof -

let $I = (-\text{abs}(B))..\text{abs}(B)$
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 $C1:
\text{IsBounded}(I, \text{IntegerOrder})$
$I \neq 0$
$\forall q \in \mathbb{Z}. \ F(q) \in \mathbb{Z}$
$K = \{F(q). \ q \in I\}$
using $\text{OrderZF}_3\_L11$ $\text{IntZF}_1\_3\_L17$ by auto
then have $M \in \mathbb{Z}$ by (rule $\text{IntZF}_1\_4\_L1$)
with $A3$ have $T1: M \leq L$ $A+1 \leq L$
using $\text{int_zero_one_are_int}$ $\text{IntZF}_1\_1\_L5$ $\text{IntZF}_1\_3\_L18$ by auto
from $C1$ have $T2: \ \forall q \in I. \ F(q) \leq M$
by (rule $\text{IntZF}_1\_4\_L1$)

{ fix $p$ assume $A4: p \in \mathbb{Z}$ have $F(p) \leq L$
proof -

{ assume $\text{abs}(p) \leq \text{abs}(B)$
with $A4 \ T1 \ T2$ have $F(p) \leq M$ $M \leq L$
using $\text{IntZF}_1\_3\_L19$ by auto
then have $F(p) \leq L$ by (rule $\text{Int_order_transitive}$)
moreover

{ assume $A5: \neg(\text{abs}(p) \leq \text{abs}(B))$
from $A3 \ A2 \ A4$ have
$A-\text{abs}(p) \in \mathbb{Z}. \ F(p)-\text{abs}(p) \leq A-\text{abs}(p) + B$
using $\text{IntZF}_2\_L14$ $\text{IntZF}_1\_1\_L5$ by auto
moreover from $A3 \ A4 \ A5$ have $B \leq \text{abs}(p)$
using $\text{IntZF}_1\_3\_L15$ by simp
ultimately have
$F(p)-\text{abs}(p) \leq A-\text{abs}(p) + \text{abs}(p)$
using $\text{IntZF}_2\_L15A$ by blast
with $A3 \ A4$ have $F(p)-\text{abs}(p) \leq (A+1)-\text{abs}(p)$
using $\text{IntZF}_2\_L14$ $\text{IntZF}_1\_2\_L7$ by simp
moreover from $A3 \ A1 \ A4 \ A5$ have
$F(p) \in \mathbb{Z}. \ A+1 \in \mathbb{Z}. \ \text{abs}(p) \in \mathbb{Z}$
$\neg(\text{abs}(p) \leq 0)$
using $\text{int_zero_one_are_int}$ $\text{IntZF}_1\_1\_L5$ $\text{IntZF}_2\_L14$ $\text{IntZF}_1\_3\_L11$
by auto
ultimately have $F(p) \leq A+1$
using $\text{Int_ineq_simpl_positive}$ by simp
moreover from $T1$ have $A+1 \leq L$ by simp
ultimately have $F(p) \leq L$ by (rule $\text{Int_order_transitive}$)

ultimately show thesis by blast

543
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

$$0 \leq a \land 0 \leq b \lor a \leq 0 \land b \leq 0 \lor$$

$$a \leq 0 \land 0 \leq b \lor 0 \leq a + b \lor a \leq 0 \land 0 \leq b \land a + b \leq 0 \lor$$

$$0 \leq a \land b \leq 0 \lor 0 \leq a + b \lor 0 \leq a \land b \leq 0 \land a + b \leq 0$$

using assms Int_ZF_2_T1 group3.OrdGroup_6cases by simp
from A2 have T: n ∈ ℤ.
  using Int_ZF_2_L1A by simp
from A2 have #0 $<$ n using Int_ZF_2_L9 Int_ZF_1_L8
  by auto
with T show 0 ≤ m zmod n
  m zmod n ≤ n
  m zmod n ≠ 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 zmod n ≤ n-1
  using Int_ZF_4_L1B by auto
qed

(m·k) div k = m.

lemma (in int0) IntDiv_ZF_1_L3:
  assumes m∈ ℤ. k∈ ℤ. and k ≠ 0
  shows (m·k) zdiv k = m
(k·m) zdiv 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 zdiv_mono1 from standard Isabelle to
our notation.

lemma (in int0) IntDiv_ZF_1_L4:
  assumes A1: m ≤ k and A2: 0 ≤ n n ≠ 0
  shows m zdiv n ≤ k zdiv n
proof -
  from A2 have #0 ≤ n #0 ≠ n
    using Int_ZF_1_L8 by auto
with A1 have m zdiv n $<$ k zdiv n
  m zdiv n ∈ ℤ. m zdiv k ∈ ℤ
    using Int_ZF_2_L1A Int_ZF_2_L9 zdiv_mono1
    by auto
then show (m zdiv n) ≤ (k zdiv 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 ∈ ℤ⁺ and A2: n ≤ k and A3: k·n ≤ m
  shows m = n·(m zdiv n) + (m zmod n)
(m zmod n)·n + (m zmod n)
(m zmod n) ∈ 0..(n-1)
k ≤ (m zdiv n)
\[ \text{m zdiv n} \in \mathbb{Z}_+ \]

**proof**

- **from A2 A3 have T:**
  
  \[ m \in \mathbb{Z} \quad n \in \mathbb{Z} \quad k \in \mathbb{Z} \quad m \text{ zdiv } n \in \mathbb{Z} \]
  
  using `Int_ZF_2_L1A` by auto
  
  then show \( m = n \cdot (m \text{ zdiv } n) + (m \text{ zmod } n) \)
  
  using `IntDiv_ZF_1_L1` by simp
  
  with T show \( m = (m \text{ zdiv } n) \cdot n + (m \text{ zmod } n) \)
  
  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 \( (m \text{ zmod } n) \in 0 \ldots (n-1) \)
  
  using `IntDiv_ZF_1_L2` `Order_ZF_2_L1` by simp

- **from A3 I have \( (k \cdot n \text{ zdiv } n) \leq (m \text{ zdiv } n) \)**
  
  using `IntDiv_ZF_1_L4` by simp
  
  with I T show \( k \leq (m \text{ zdiv } n) \)
  
  using `IntDiv_ZF_1_L3` by simp
  
  with A1 A2 show \( m \text{ zdiv } n \in \mathbb{Z}_+ \)
  
  using `Int_ZF_1_L5_L7` by blast

qed

**end**

### 48 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.

#### 48.1 Slopes

In this section we study basic properties of slopes - the integer almost homomorphisms. 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 equivalent 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 $\text{max}\delta(s)$ 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 "~" to denote it. As here this symbol seems to be hogged by the standard Isabelle, we will use "\sim" instead "\approx". 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 "\text{+}" denotes the first operation on almost homomorphisms. For slopes this is addition of functions defined in the natural way. The "\text{−1}" 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 $\varepsilon(f,(m,n)) = f^{−1}(m+n) - f^{−1}(m) - f^{−1}(n)$. Of course the intention is to use the fact that $\varepsilon(f,(m,n))$ 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 $m^S$ defined by $m^S(n) = m \cdot n$. The mapping $q \mapsto q^S$ embeds integers into $S$ preserving the order, (that is, maps positive integers into $S_+$).

locale int1 = int0 +

fixes slopes ($\mathcal{S}$)
defines slopes_def[simp]: $\mathcal{S} \equiv \text{AlmostHoms}(\mathbb{Z},\text{IntegerAddition})$

fixes posslopes ($\mathcal{S}_+$)
defines posslopes_def[simp]: $\mathcal{S}_+ \equiv \{s \in \mathcal{S}. (Z_+ \cap Z_+ \in \text{Fin}(Z))\}$

fixes $\delta$
defines $\delta$ _def[simp]: $\delta(s,m,n) \equiv s(m+n) - s(m) - s(n)$

fixes maxhomdiff ($\text{max}\delta$)
defines maxhomdiff_def[simp]: $\text{max}\delta(s) \equiv \text{Maximum}(\text{IntegerOrder},\{\text{abs}(\delta(s,m,n)). (m,n) \in \mathbb{Z} \times \mathbb{Z}\})$

fixes ALEqRelndefines ALEqRel_def[simp]:

547
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) 0 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 ∈ Z
\[ f(m+n) \in \mathbb{Z} \quad f(m) \in \mathbb{Z} \quad f(n) \in \mathbb{Z} \]
\[ f(m) + f(n) \in \mathbb{Z} \]
\[ \text{HomDiff}(\mathbb{Z}, \text{IntegerAddition}, f, (m, n)) \in \mathbb{Z} \]
using `assms` IntZF_2_1_L1 group1.GroupZF_3_2_L4 by auto

Slopes map integers into integers.

**Lemma (in int1) IntZF_2_1_L2B:**
- **Assumes** A1: \( f \in \mathcal{S} \) and A2: \( m \in \mathbb{Z} \)
- **Shows** \( f(m) \in \mathbb{Z} \)

**Proof:**
- from A1 have \( f: \mathbb{Z} \to \mathbb{Z} \) 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) IntZF_2_1_L3:**
- **Assumes** \( f: \mathbb{Z} \to \mathbb{Z} \) and \( m \in \mathbb{Z} \) \( n \in \mathbb{Z} \)
- **Shows** \( \text{HomDiff}(\mathbb{Z}, \text{IntegerAddition}, f, (m, n)) = \delta(f, m, n) \)

**Proof:**
- from \( A1 \ A2 \) have \( f: \mathbb{Z} \to \mathbb{Z} \) \( f(m) \in \mathbb{Z} \) \( f(n) \in \mathbb{Z} \) \( \delta(f, m, n) \in \mathbb{Z} \) and
- \( \text{HomDiff}(\mathbb{Z}, \text{IntegerAddition}, f, (m, n)) = \delta(f, m, n) \)
using `assms` IntZF_2_1_L2 AlmostHoms_def IntZF_2_1_L3 by auto
- with \( A1 \ A2 \) show \( f(m+n) = f(m) + (f(n) + \delta(f, m, n)) \)
using `assms` IntZF_2_1_L3 IntZF_1_L3 IntZF_2_1_L3 IntZF_2_1_L1 group1.GroupZF_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) IntZF_2_1_L3A:**
- **Assumes** \( f: \mathbb{Z} \to \mathbb{Z} \) and \( m \in \mathbb{Z} \) \( n \in \mathbb{Z} \)
- **Shows** \( f(m+n) = f(m) + (f(n) + \delta(f, m, n)) \)

**Proof:**
- from \( A1 \ A2 \) have
- \( T: f(m) \in \mathbb{Z} \) \( f(n) \in \mathbb{Z} \) \( \delta(f, m, n) \in \mathbb{Z} \) and
- \( \text{HomDiff}(\mathbb{Z}, \text{IntegerAddition}, f, (m, n)) = \delta(f, m, n) \)
using `assms` IntZF_2_1_L2 AlmostHoms_def IntZF_2_1_L3 by auto
- with \( A1 \ A2 \) show \( f(m+n) = f(m) + (f(n) + \delta(f, m, n)) \)
using `assms` IntZF_2_1_L3 IntZF_1_L3 IntZF_2_1_L3 IntZF_2_1_L1 group1.GroupZF_3_4_L1 by simp

**QED**

The homomorphism difference of any integer function is integer.

**Lemma (in int1) IntZF_2_1_L3B:**
- **Assumes** \( f: \mathbb{Z} \to \mathbb{Z} \) and \( m \in \mathbb{Z} \) \( n \in \mathbb{Z} \)
- **Shows** \( \delta(f, m, n) \in \mathbb{Z} \)

**Proof:**
- using `assms` IntZF_2_1_L2A IntZF_2_1_L3 by simp

**QED**

The value of an integer function at a sum expressed in terms of \( \delta \).
lemma (in int1) Int_ZF_2_1_L3C: assumes A1: f:\mathbb{Z}\rightarrow\mathbb{Z} and A2: m\in\mathbb{Z}\quad 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}\quad f(m+n) \in \mathbb{Z}\quad f(m) \in \mathbb{Z}\quad f(n) \in \mathbb{Z}
  using Int_ZF_1_1_L5 apply_funtype by auto
then show thesis using Int_ZF_1_2_L15 by simp
qed

The next lemma presents two ways the set of homomorphism differences can be written.

lemma (in int1) Int_ZF_2_1_L4: assumes A1: f:\mathbb{Z}\rightarrow\mathbb{Z}
shows \{abs(HomDiff(\mathbb{Z},\text{IntegerAddition},f,x)). x \in \mathbb{Z}\times\mathbb{Z}\} = \{abs(\delta(f,m,n)). \langle m,n \rangle \in \mathbb{Z}\times\mathbb{Z}\}
proof -
from A1 have \forall m\in\mathbb{Z}. \forall n\in\mathbb{Z}.
  abs(HomDiff(\mathbb{Z},\text{IntegerAddition},f,\langle m,n \rangle)) = abs(\delta(f,m,n))
  using Int_ZF_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) Int_ZF_2_1_L5: assumes A1: f:\mathbb{Z}\rightarrow\mathbb{Z} and A2: \forall m\in\mathbb{Z}.\forall n\in\mathbb{Z}. abs(\delta(f,m,n)) \leq L
shows f\in\mathcal{S}
proof -
  let Abs = \text{AbsoluteValue}(\mathbb{Z},\text{IntegerAddition},\text{IntegerOrder})
  have group3(\mathbb{Z},\text{IntegerAddition},\text{IntegerOrder})
    \text{IntegerOrder \{is total on\} \mathbb{Z}}
    using Int_ZF_2_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} \land
    (Abs(HomDiff(\mathbb{Z},\text{IntegerAddition},f,x)),L \rangle \in \text{IntegerOrder}
    using Int_ZF_2_1_L2A Int_ZF_2_1_L3 by auto
  ultimately have\n    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 \mathcal{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) Int_ZF_2_1_L7: assumes A1: s\in\mathcal{S} and A2: n\in\mathbb{Z}\quad m\in\mathbb{Z}
shows
$$\text{abs}(\delta(s,m,n)) \leq \max \delta(s)$$

$$\delta(s,m,n) \in \mathbb{Z} \quad \max \delta(s) \in \mathbb{Z}$$

$$(-\max \delta(s)) \leq \delta(s,m,n)$$

**proof**

- from A1 A2 show T: $\delta(s,m,n) \in \mathbb{Z}$
  - using Int_ZF_2_1_L2 Int_ZF_1_1_L5 by simp
- let $A = \{\text{abs}(\text{HomDiff}(\mathbb{Z},\text{IntegerAddition},s,x)). x \in \mathbb{Z} \times \mathbb{Z}\}$
- let $B = \{\text{abs}(\delta(s,m,n)). (m,n) \in \mathbb{Z} \times \mathbb{Z}\}$
- let $d = \text{abs}(\delta(s,m,n))$
- have $\text{IsLinOrder}(\mathbb{Z},\text{IntegerOrder})$ using Int_ZF_2_T1 by simp

moreover have $A \in \text{Fin}(\mathbb{Z})$

- have $\forall k \in \mathbb{Z}. \text{abs}(k) \in \mathbb{Z}$ using Int_ZF_2_L14 by simp
- moreover from A1 have
  - $\{\text{HomDiff}(\mathbb{Z},\text{IntegerAddition},s,x). x \in \mathbb{Z} \times \mathbb{Z}\} \in \text{Fin}(\mathbb{Z})$
  - using AlmostHoms_def by simp
  - ultimately show $A \in \text{Fin}(\mathbb{Z})$ by (rule Finite1_L6C)

qed

moreover have $A \neq 0$ by auto

ultimately have $\forall k \in A. \langle k, \text{Maximum}(\text{IntegerOrder},A) \rangle \in \text{IntegerOrder}$

by (rule Finite1_L6C)

moreover from A1 A2 have $d \in A$ using AlmostHoms_def Int_ZF_2_1_L4 by auto

ultimately have $d \leq \text{Maximum}(\text{IntegerOrder},A)$ by auto

with A1 show $d \leq \max \delta(s) \quad \max \delta(s) \in \mathbb{Z}$
  - using AlmostHoms_def Int_ZF_2_1_L4 Int_ZF_2_L1A by auto
  - with T show $(-\max \delta(s)) \leq \delta(s,m,n)$
  - 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: $s \in S$

shows

$$\text{abs}(s(0)) \leq \max \delta(s)$$

$$0 \leq \max \delta(s)$$

$$\text{abs}(s(0)) \in \mathbb{Z} \quad \max \delta(s) \in \mathbb{Z}$$

$$\text{abs}(s(0)) + \max \delta(s) \in \mathbb{Z}$$

**proof**

- from A1 have $s(0) \in \mathbb{Z}$
  - using int_zero_one_are_int Int_ZF_2_1_L2B by simp
  - then have I: $0 \leq \text{abs}(s(0))$
  - and $\text{abs}(\delta(s,0,0)) = \text{abs}(s(0))$
  - using int_abs_nonneg int_zero_one_are_int Int_ZF_1_1_L4 Int_ZF_2_L17 by auto

moreover from A1 have $\text{abs}(\delta(s,0,0)) \leq \max \delta(s)$
  - using int_zero_one_are_int Int_ZF_2_1_L7 by simp

551
ultimately show II: abs(s(0)) ≤ maxδ(s) 
  by simp
  with I I show 0 ≤ maxδ(s) by (rule Int_order_transitive)
with II show
  maxδ(s) ∈ ℤ  abs(s(0)) ∈ ℤ
  abs(s(0)) + maxδ(s) ∈ 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 \( A1: \text{s} \in S \)  \( r \in S \)
  and \( A2: \forall m \in ℤ. \text{abs}(s(m) - r(m)) \leq L \)
shows \( s \sim r \)
proof -
from \( A1 \) \( A2 \) have \( \forall m \in ℤ. s(m) - r(m) \in ℤ \land \text{abs}(s(m) - r(m)) \leq L \)
  using Int_ZF_2_1_L2B Int_ZF_1_1_L5 by simp
then have IsBounded({s(n)-r(n). n∈ℤ}, 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 neccessary condition for two slopes to be almost equal. For slopes the definition postulates the set \{ \( f(m) - g(m) : m \in ℤ \) \} 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 \( \exists L \in ℤ. \forall m \in ℤ. \text{abs}(s(m) - r(m)) \leq L \)
  \( s \in S \)  \( r \in S \)
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 \( AlEqRel \subseteq S \times S \)
equiv(S, A1EqRel) using Int_ZF_2_1_L1 group1.Group_ZF_3_3_L3 by auto

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 AlHomOp1(G, f)⟨s,GroupInv(AlmostHoms(G, f),AlHomOp1(G, f))(r)⟩ ∈ FinRangeFunctions(G, G) using Int_ZF_2_1_L1 group1.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 group1.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 group1.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 group1.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 group1.Group_ZF_3_2_L13 by simp

553
What is the value of a negative of a slope?

**Lemma (in Int)** Int_ZF_2_1_L12A: assumes \( s \in S \) and \( m \in \mathbb{Z} \) shows \((-s)(m) = -(s(m))\)

by simp

What are the values of a sum of slopes?

**Lemma (in Int)** Int_ZF_2_1_L12B: assumes \( s \in S \) \( r \in S \) 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 Int)** Int_ZF_2_1_L12C: assumes \( s \in S \) \( r \in S \)

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 Int)** Int_ZF_2_1_L13: assumes \( s \in S \) and \( n \in \mathbb{Z} \) \( m \in \mathbb{Z} \)

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 Int)** Int_ZF_2_1_L14: assumes A1: \( s \in S \) and A2: \( m \in \mathbb{Z} \)

shows \( s(-m) = s(0) - \delta(s,m,-m) - s(m) \)

abs(s(m)+s(-m)) \( \leq 2 \cdot \max \delta(s) \)

abs(s(-m)) \( \leq 2 \cdot \max \delta(s) + abs(s(m)) \)

\( s(-m) \leq abs(s(0)) + \max \delta(s) - s(m) \)

**Proof** -

- From A1 A2 have T:
  \((-m) \in \mathbb{Z} \) \( abs(s(m)) \in \mathbb{Z} \) \( s(0) \in \mathbb{Z} \) \( abs(s(0)) \in \mathbb{Z} \)
  \( \delta(s,m,-m) \in \mathbb{Z} \) \( s(m) \in \mathbb{Z} \) \( s(-m) \in \mathbb{Z} \)
  \((-s(m))) \in \mathbb{Z} \) \( s(0) - \delta(s,m,-m) \in \mathbb{Z} \)
  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
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 \( T: \\
\begin{align*}
 f(-m) &\in \mathbb{Z} \quad \delta(f,m,-m) \in \mathbb{Z} \\
 f(0) &\in \mathbb{Z} \\
 f(m) &\in \mathbb{Z} \\
 \end{align*}
\)

using Int_ZF_1_1_L4 Int_ZF_1_1_L5 int_zero_one_are_int apply_funtype

by \( \text{auto} \)

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 \( \text{simp} \)

**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:
assumes \( s \in S \) and \( M \in \mathbb{Z}_+ \)
shows \( \maxf(s,0..(M-1)) \in \mathbb{Z} \)
\( \forall n \in 0..(M-1). \, s(n) \leq \maxf(s,0..(M-1)) \)
\( \minf(s,0..(M-1)) \in \mathbb{Z} \)
\( \forall n \in 0..(M-1). \, \minf(s,0..(M-1)) \leq s(n) \)
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:
assumes \( A1: s \in S \) and \( A2: m \in \mathbb{Z} \) and \( A3: M \in \mathbb{Z}_+ \) and \( A4: k \in 0..(M-1) \)
shows \( s(m \cdot M) + (\minf(s,0..(M-1)) - \maxf(s)) \leq s(m \cdot M + k) \)
proof -
from \( A3 \) have \( 0..(M-1) \subseteq \mathbb{Z} \)
using Int_ZF_1_5_L6 by simp
with \( A1 \) \( A2 \) \( A3 \) \( A4 \) have \( T: m \cdot M \in \mathbb{Z} \)
\( k \in \mathbb{Z} \)
\( s(m \cdot M) \in \mathbb{Z} \)
using PositiveSet_def Int_ZF_1_1_L5 Int_ZF_2_1_L2B
by auto
with \( A1 \) \( A3 \) \( A4 \) have
\( s(m \cdot M) + (\minf(s,0..(M-1)) - \maxf(s)) \leq s(m \cdot M) + (s(k) + \delta(s,m \cdot M,k)) \)
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 \( \text{id}(\mathbb{Z}) \in S \)
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:
assumes \( A1: f: \mathbb{Z} \rightarrow \mathbb{Z} \) and \( A2: m \in \mathbb{Z} \)
\( n \in \mathbb{Z} \)
shows \( \abs(f(n) + f(m) - f(m+n)) = \abs(\delta(f,m,n)) \)
\( \abs(f(m) + f(n) - f(m+n)) = \abs(\delta(f,m,n)) \)
\( (-f(m)) - f(n) + f(m+n) = \delta(f,m,n) \)
\( (-f(n)) - f(m) + f(m+n) = \delta(f,m,n) \)
\( \abs((-f(m+n)) + f(m) + f(n)) = \abs(\delta(f,m,n)) \)
proof -
from \( A1 \) \( A2 \) have \( T: \)
\( f(m+n) \in \mathbb{Z} \)
\( f(m) \in \mathbb{Z} \)
\( f(n) \in \mathbb{Z} \)
\( f(m+n) - f(m) - f(n) \in \mathbb{Z} \)
\( (-f(m)) \in \mathbb{Z} \)
\( (-f(m+n)) + f(m) + f(n) \in \mathbb{Z} \)
using apply_funtype Int_ZF_1_1_L4 Int_ZF_1_1_L5 by auto
then have \( \abs((-f(m+n) - f(m) - f(n)) = \abs(f(m+n) - f(m) - f(n)) \)
using Int_ZF_2_L17 by simp
moreover from T have
\((-f(m+n) - f(m) - f(n)) = f(n) + f(m) - f(m+n)\)
using Int_ZF_1_2_L9A by simp
ultimately show \(\text{abs}(f(n) + f(m) - f(m+n)) = \text{abs}(\delta(f,m,n))\)
by simp
moreover from T have \(f(n) + f(m) = f(m) + f(n)\)
using Int_ZF_1_1_L5 by simp
ultimately show \(\text{abs}(f(m) + f(n) - f(m+n)) = \text{abs}(\delta(f,m,n))\)
by simp
from T show \(\text{abs}((-f(m)) + f(m+n) + f(n)) = \delta(f,m,n)\)
\(\text{abs}((-f(n)) + f(m+n) + f(m)) = \delta(f,m,n)\)
using Int_ZF_1_2_L9 by auto
from T have \(\text{abs}((-f(m+n)) + f(m) + f(n)) = \delta(f,m,n)\)
using Int_ZF_2_L17 by simp
also from T have \(\text{abs}((-f(m+n)) + f(m) + f(n)) = \delta(f,m,n)\)
using Int_ZF_1_2_L9 by simp
finally show \(\text{abs}((-f(m+n)) + f(m) + f(n)) = \delta(f,m,n)\)
by simp
qed

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: \(\forall x\in\mathbb{Z}. \ (-f(-x)) = f(x)\)
and A3: \(m\in\mathbb{Z} \ n\in\mathbb{Z}\)
shows \(\text{abs}(\delta(f,-m,m+n)) = \text{abs}(\delta(f,m,n))\)
\(\text{abs}(\delta(f,-n,m+n)) = \text{abs}(\delta(f,m,n))\)
\(\delta(f,n,-(m+n)) = \delta(f,m,n)\)
\(\delta(f,m,-(m+n)) = \delta(f,m,n)\)
\(\text{abs}(\delta(f,-m,-n)) = \text{abs}(\delta(f,m,n))\)
proof -
from A1 A2 A3 show
\(\text{abs}(\delta(f,-m,m+n)) = \text{abs}(\delta(f,m,n))\)
\(\text{abs}(\delta(f,-n,m+n)) = \text{abs}(\delta(f,m,n))\)
using Int_ZF_1_2_L3 Int_ZF_2_1_L18 by auto
from A3 have \(m+n \in \mathbb{Z}\) 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 \(\text{abs}(\delta(f,-m,-n)) = \text{abs}(f(-(m+n)) - f(-m) - f(-n))\)
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: $\forall a \in \mathbb{Z}^+. \ \forall b \in \mathbb{Z}^+. \ \text{abs}(\delta(f,a,b)) \leq L$ and
A3: $m \in \mathbb{Z}^+ \ \ n \in \mathbb{Z}^+
shows
$0 \leq L$
$\text{abs}(\delta(f,m,n)) \leq L + \text{abs}(f(0))$
proof -
from A1 A2 have
$\delta(f,1,1) \in \mathbb{Z}$ and $\text{abs}(\delta(f,1,1)) \leq L$
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:
$n \in \mathbb{Z} \ \ f(n) \in \mathbb{Z} \ \ f(0) \in \mathbb{Z}$
$\delta(f,m,n) \in \mathbb{Z} \ \ \text{abs}(\delta(f,m,n)) \in \mathbb{Z}$
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 ∨ m∈Z⁺ using Int_ZF_1_5_L3A by auto
moreover
{ assume m = 0
  with T I have $\text{abs}(\delta(f,m,n)) \leq L + \text{abs}(f(0))$
  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∈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_1_L15F by simp }
ultimately show $\text{abs}(\delta(f,m,n)) \leq L + \text{abs}(f(0))$
  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: $\forall a \in \mathbb{Z}^+. \ \forall b \in \mathbb{Z}^+. \ \text{abs}(\delta(f,a,b)) \leq L$ and
A3: $n \in \mathbb{Z}^+ \ \ m \in \mathbb{Z}^+$

558
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 \text{int_one_two_are_pos} \text{ PositiveSet_def} \text{ Nonnegative_def} \text{ IntZF_2_1_L3B} by auto

then have \( I: 0 \leq L \) using \text{IntZF_1_3_L19} by simp

from \( A1 \) \( A3 \) have

\[ m \in \mathbb{Z} \quad f(m) \in \mathbb{Z} \quad f(0) \in \mathbb{Z} \quad (-f(0)) \in \mathbb{Z} \]
\[ \delta(f,m,n) \in \mathbb{Z} \quad \text{abs}(\delta(f,m,n)) \in \mathbb{Z} \]

using \text{int_zero_one_are_int} \text{ apply_funtype} \text{ Nonnegative_def} \text{ IntZF_2_1_L3B} \text{ IntZF_2_L14} \text{ IntZF_1_1_L4} by auto

from \( A3 \) have \( n=0 \lor n \in \mathbb{Z}_+ \) using \text{IntZF_1_5_L3A} by auto

moreover

\{ assume \( n=0 \)

with \( T \) have \( \delta(f,m,n) = -f(0) \)

using \text{IntZF_1_1_L4} by simp

with \( T \) have \( \text{abs}(\delta(f,m,n)) = \text{abs}(f(0)) \)

using \text{IntZF_2_L17} by simp

with \( T \) I have \( \text{abs}(\delta(f,m,n)) \leq L + \text{abs}(f(0)) \)

using \text{IntZF_2_L15F} by simp \}

moreover

\{ assume \( n \in \mathbb{Z}_+ \)

with \( A2 \) \( A3 \) \( T \) have \( \text{abs}(\delta(f,m,n)) \leq L + \text{abs}(f(0)) \)

using \text{Int_abs_nonneg} \text{ IntZF_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) IntZF_2_1_L22:** assumes \( A1: f: \mathbb{Z} \rightarrow \mathbb{Z} \) and

\[ \forall a \in \mathbb{Z}_+. \forall b \in \mathbb{Z}_+. \text{abs}(\delta(f,a,b)) \leq L \]

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 \text{IntZF_2_1_L20} \text{ IntZF_2_1_L21} by simp

then show thesis by auto

qed

For odd functions we can do better than in \text{IntZF_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) IntZF_2_1_L23:** assumes \( A1: f: \mathbb{Z} \rightarrow \mathbb{Z} \) and

\[ \forall a \in \mathbb{Z}_+. \forall b \in \mathbb{Z}_+. \text{abs}(\delta(f,a,b)) \leq L \]

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
$0 \leq a \land 0 \leq b \lor a \leq 0 \land b \leq 0 \lor$
$a \leq 0 \land 0 \leq b \land 0 \leq a+b \lor a \leq 0 \land 0 \leq b \land a+b \leq 0 \lor$
$0 \leq a \land b \leq 0 \land 0 \leq a+b \lor 0 \leq a \land b \leq 0 \land a+b \leq 0$
using int_plane_split_in6 by simp
moreover
{ assume $0 \leq a \land 0 \leq b$
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 $a \leq 0 \land b \leq 0$
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 $a \leq 0 \land 0 \leq b \land 0 \leq a+b$
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 $a \leq 0 \land 0 \leq b \land a+b \leq 0$
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 $0 \leq a \land b \leq 0 \land 0 \leq a+b$
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 $0 \leq a \land b \leq 0 \land a+b \leq 0$
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 }

560
then have ∀m∈\mathbb{Z}. ∀n∈\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:š⁺→š and A2: ∀a∈š⁺. ∀b∈š⁺. \text{abs}(\delta(f,a,b)) \leq L 
shows OddExtension(š,IntegerAddition,IntegerOrder,f) \in S 
proof -
  let g = OddExtension(š,IntegerAddition,IntegerOrder,f) 
  from A1 have g : š→š 
    using Int_ZF_1_5_L10 by simp 
  moreover have ∀a∈š⁺. ∀b∈š⁺. \text{abs}(\delta(g,a,b)) \leq L 
    proof -
      { fix a b assume A3: a\inš⁺ b\inš⁺
        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 ∀x∈š. (-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 γ.

lemma (in int1) Int_ZF_2_1_L25:
  assumes A1: f:š→š and A2: m\inš n\inš 
shows δ(f,m,-n) ∈ š \ δ(f,n,-n) ∈ š \ (-δ(f,n,-n)) ∈ š 
  f(0) ∈ š 
  γ(f,m,n) ∈ š 
proof -
  from A1 A2 show T1: 
    δ(f,m,-n) \in š f(0) \in š 
    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 š 
    using Int_ZF_1_1_L4 by simp 
  with A1 A2 show δ(f,n,-n) \in š 
    using Int_ZF_2_1_L3B by simp 
  then show (-δ(f,n,-n)) \in š 
    using Int_ZF_1_1_L4 by simp 
  with T1 show γ(f,m,n) \in š 

A couple of formulae involving \( f(m - n) \) and \( \gamma(f, m, n) \).

**Lemma** (in int1) IntZF_2_1_L26:
assumes \( A1: f:Z\rightarrow Z \) and \( A2: m\in\mathbb{Z} \quad n\in\mathbb{Z} \)
shows
\[
  f(m-n) = \gamma(f,m,n) + f(m) - f(n)
\]
proof -
from \( A1 \) \( A2 \) have \( T: \)
\[
  (-n) \in \mathbb{Z} \quad \delta(f,m,-n) \in \mathbb{Z} \\
  f(0) \in \mathbb{Z} \quad f(m) \in \mathbb{Z} \quad f(n) \in \mathbb{Z} \quad (-f(n)) \in \mathbb{Z} \\
  (-\delta(f,n,-n)) \in \mathbb{Z} \\
  \gamma(f,m,n) \in \mathbb{Z}
\]
using IntZF_1_1_L4 IntZF_2_1_L25 apply_funtype IntZF_1_1_L5 by auto
with \( A1 \) \( A2 \) have \( f(m-n) = \)
\[
  \delta(f,m,-n) + ((-\delta(f,n,-n)) + f(0) - f(n)) + f(m)
\]
using IntZF_2_1_L3C IntZF_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 IntZF_1_2_L16 by simp
moreover from \( T \) have
\[
  \delta(f,m,-n) + ((-\delta(f,n,-n)) + f(0)) = \gamma(f,m,n)
\]
using IntZF_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 \( \ldots = f(m) \)
by simp
ultimately show \( f(m-n) + (f(n) - \gamma(f,m,n)) = f(m) \)
by simp
from \( T \) have \( \gamma(f,m,n) \in \mathbb{Z} \quad f(m) \in \mathbb{Z} \quad (-f(n)) \in \mathbb{Z} \)
by auto
then have
\[
  \gamma(f,m,n) + f(m) + (-f(n)) = \gamma(f,m,n) + (f(m) + (-f(n)))
\]
by (rule IntZF_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) IntZF_2_1_L26A:
assumes \( A1: f:Z\rightarrow Z \) 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 $... = \gamma(f,m,n) + (f(m) - f(n) + (-f(k)))$
  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 $A1: s \in S$
  shows $\exists L \in \mathbb{Z}. \ \forall m \in \mathbb{Z}. \ \forall n \in \mathbb{Z}. \ \text{abs}(\gamma(s,m,n)) \leq L$
proof -
  let $L = \text{max}\delta(s) + \text{max}\delta(s) + \text{abs}(s(0))$
from A1 have T:
  $\text{max}\delta(s) \in \mathbb{Z}$ $\text{abs}(s(0)) \in \mathbb{Z}$ $L \in \mathbb{Z}$
  using Int_ZF_2_1_L8 int_zero_one_are_int Int_ZF_2_1_L28
  Int_ZF_2_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:
      $(-n) \in \mathbb{Z}$
      $\delta(s,m,-n) \in \mathbb{Z}$
      $\delta(s,n,-n) \in \mathbb{Z}$
      $(-\delta(s,n,-n)) \in \mathbb{Z}$
      $s(0) \in \mathbb{Z}$ $\text{abs}(s(0)) \in \mathbb{Z}$
      using Int_ZF_1_1_L4 AlmostHoms_def Int_ZF_2_1_L25 Int_ZF_2_L14
      by auto
  }
with \( T \) have
\[
\begin{align*}
&\text{abs}(\delta(s,m,-n) - \delta(s,n,-n) + s(0)) \leq \\
&\text{abs}(\delta(s,m,-n)) + \text{abs}(-\delta(s,n,-n)) + \text{abs}(s(0))
\end{align*}
\]
using \textit{Int_triangle_ineq3} by simp
moreover from \( \text{A1} \ \text{A2} \ \text{T} \) have
\[
\begin{align*}
&\text{abs}(\delta(s,m,-n)) + \text{abs}(-\delta(s,n,-n)) + \text{abs}(s(0)) \leq L
\end{align*}
\]
by simp
ultimately have \( \text{abs}(\delta(s,m,-n) - \delta(s,n,-n) + s(0)) \leq L \)
by (rule \textit{Int_order_transitive})
then have \( \text{abs}(\gamma(s,m,n)) \leq L \)
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) \textit{Int_ZF_2.1_L28}: assumes \( \text{A1: } s \in S \)
shows \( \exists M \in \mathbb{Z}. \forall m \in \mathbb{Z}. \forall n \in \mathbb{Z}. \text{abs}(\gamma(s,m,n)) \leq L \)
proof -
from \( \text{A1} \) 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 \textit{Int_ZF_2.1_L27} by simp
then obtain \( L \) where
\( L \in \mathbb{Z} \) and \( \forall m \in \mathbb{Z}. \forall n \in \mathbb{Z}. \text{abs}(\gamma(s,m,n)) \leq L \)
using \textit{Int_ZF_2.1_L27} by auto
then have \( \text{I: } \forall m \in \mathbb{Z}. \text{abs}(\gamma(s,m,1)) \leq L \)
using \textit{int_zero_one_are_int} by simp
let \( M = s(1) + L \)
from \( \text{A1} \ \text{T} \) have
\( M \in \mathbb{Z} \)
using \textit{int_zero_one_are_int} \textit{Int_ZF_2.1_L2B} \textit{Int_ZF_1.1_L5} by simp
moreover
\{ fix \( m \) assume \( \text{A2: } m \in \mathbb{Z} \)
with \( \text{A1} \) have
\( \text{T1: } s : \mathbb{Z} \rightarrow \mathbb{Z} \ \ m \in \mathbb{Z} \ \ 1 \in \mathbb{Z} \ \text{and} \)
\( \text{T2: } \gamma(s,m,1) \in \mathbb{Z} \ \ s(1) \in \mathbb{Z} \)
using \textit{int_zero_one_are_int} \textit{AlmostHoms_def} \textit{Int_ZF_2.1_L2B} by auto
from \( \text{A2} \ \text{T1} \) have \( \text{T3: } s(m-1) \in \mathbb{Z} \)
using \textit{Int_ZF_1.1_L5} \textit{apply_funtype} by simp
from \( \text{T1} \ \text{T2} \) have
\( (\gamma(s,m,1)) \leq \text{abs}(\gamma(s,m,1)) \)
\( \text{abs}(\gamma(s,m,1)) \leq L \)
using \textit{Int_ZF_2.1_L19C} by auto
then have \( (\gamma(s,m,1)) \leq L \)
by (rule \textit{Int_order_transitive})
with \( \text{T2} \ \text{T3} \) have
\( s(m-1) + (s(1) - \gamma(s,m,1)) \leq s(m-1) + M \)
using \textit{int_ord_transl_inv} by simp
moreover from \( \text{T1} \) 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 \( \exists M \in \mathbb{Z}. \forall m \in \mathbb{Z}. \forall n \in \mathbb{Z}. \forall k \in \mathbb{Z}. \ \text{abs} (s(m-n-k) - (s(m)-s(n)-s(k))) \leq M \)
proof -
  from A1 have \( \exists L \in \mathbb{Z}. \forall m \in \mathbb{Z}. \forall n \in \mathbb{Z}. \ \text{abs} (\gamma(s,m,n)) \leq L \)
  using Int\_ZF\_2\_1\_L27 by simp
  then obtain \( L \) where I: \( L \in \mathbb{Z} \) and 
  II: \( \forall m \in \mathbb{Z}. \forall n \in \mathbb{Z}. \ \text{abs} (\gamma(s,m,n)) \leq L \)
  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: \( \text{abs} (\gamma(s,m-n,k) + \gamma(s,m,n)) \leq \text{abs} (\gamma(s,m-n,k)) + \text{abs} (\gamma(s,m,n)) \)
    using Int\_triangle\_ineq by simp
    from II A2 T have
    \( \text{abs} (\gamma(s,m-n,k)) \leq L \)
    \( \text{abs} (\gamma(s,m,n)) \leq L \)
    by auto
    then have \( \text{abs} (\gamma(s,m-n,k)) + \text{abs} (\gamma(s,m,n)) \leq L+L \)
    using int\_ineq\_add\_sides by simp
    with I have \( \text{abs} (\gamma(s,m-n,k) + \gamma(s,m,n)) \leq L+L \)
    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
\( \text{abs} (s(m-n-k) - (s(m)-s(n)-s(k))) \leq L+L \)
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 
\( \exists M \in \mathbb{Z}. \ \forall m \in \mathbb{Z}. \forall n \in \mathbb{Z}. \forall k \in \mathbb{Z}. \ abs(s(m-n-k) - (s(m)-s(n)-s(k))) \leq M \) 
proof -
  from A1 have 
  \( \exists M \in \mathbb{Z}. \ \forall m \in \mathbb{Z}. \forall n \in \mathbb{Z}. \forall k \in \mathbb{Z}. \ abs(s(m-n-k) - (s(m)-s(n)-s(k))) \leq M \) 
  using Int_ZF_2_1_L29 by simp
then obtain \( M \) where \( \mathbb{I}: M \in \mathbb{Z} \) and \( \mathbb{II}: \forall m \in \mathbb{Z}. \forall n \in \mathbb{Z}. \forall k \in \mathbb{Z}. \ abs(s(m-n-k) - (s(m)-s(n)-s(k))) \leq M \) 
  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_1_1_L5 Int_ZF_2_1_L2B by auto
  moreover from II A2 have \( abs(s(m-n-k) - (s(m)-s(n)-s(k))) \leq M \) 
  by simp
  ultimately have 
  \( s(m-n-k) \leq s(m)-s(n)-s(k)+M \) 
  \( s(m)-s(n)-s(k) - M \leq s(m-n-k) \) 
  using Int_triangle_ineq2 by simp
} then have 
  \( \forall m \in \mathbb{Z}. \forall n \in \mathbb{Z}. \forall k \in \mathbb{Z}. s(m-n-k) \leq s(m)-s(n)-s(k)+M \) 
  \( \forall m \in \mathbb{Z}. \forall n \in \mathbb{Z}. \forall k \in \mathbb{Z}. s(m)-s(n)-s(k) - M \leq s(m-n-k) \) 
  by auto
with I III show 
  \( \exists M \in \mathbb{Z}. \ \forall m \in \mathbb{Z}. \forall n \in \mathbb{Z}. \forall k \in \mathbb{Z}. \ abs(s(m-n-k) - (s(m)-s(n)-s(k))) \leq M \) 
  \( \exists K \in \mathbb{Z}. \ \forall m \in \mathbb{Z}. \forall n \in \mathbb{Z}. \forall k \in \mathbb{Z}. \ s(m)-s(n)-s(k)+K \leq s(m-n-k) \) 
  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: \( s \in S \) \( r \in S \) and A2: \( \forall m \in \mathbb{Z}_+. \ abs(s(m)-r(m)) \leq L \) 
shows \( s \sim r \)
proof -
  let a = \( abs(s(0) - r(0)) \)
  let c = \( 2 \cdot \max(\delta(s)) + 2 \cdot \max(\delta(r)) + L \)
  let M = \( \text{Maximum(IntegerOrder,} \{a,L,c\}) \)
  from A2 have \( abs(s(1)-r(1)) \leq L \) 
  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
moreover from A1 T have \( c \in \mathbb{Z} \)
moreover from A1 T have \( c \in \mathbb{Z} \)
ultimately have
I: \( a \leq M \) and
II: \( L \leq M \) and
III: \( c \leq M \)
using \( \text{Int}_\mathbb{Z} \text{.} \text{one}_\text{two}_\text{three}_\text{are}_\text{int}_ \text{Int}_\mathbb{Z}_\text{.} \text{one}_\text{L5} \) by \text{simp}

\[
\{ \text{fix } m \text{ assume } A5: m \in \mathbb{Z} \\
\quad \text{with } A1 \text{ have } T:
\quad s(m) \in \mathbb{Z}, \ r(m) \in \mathbb{Z}, \ s(m) - r(m) \in \mathbb{Z} \\
\quad s(-m) \in \mathbb{Z}, \ r(-m) \in \mathbb{Z} \\
\quad \text{using } \text{Int}_\mathbb{Z}_\text{.} \text{two}_\text{three}_\text{are}_\text{int}_ \text{Int}_\mathbb{Z}_\text{.} \text{one}_\text{L5} \\
\quad \text{by } \text{auto} \\
\quad \text{from } A5 \text{ have } m=0 \lor m \in \mathbb{Z}_+ \lor (-m) \in \mathbb{Z}_+ \\
\quad \text{using } \text{int}_\text{decomp}_\text{cases} \text{ by } \text{simp} \\
\quad \text{moreover} \\
\quad \{ \text{assume } m=0 \}
\quad \text{with } I \text{ have } \text{abs}(s(m) - r(m)) \leq M \\
\quad \text{by } \text{simp} \}
\quad \text{moreover} \\
\quad \{ \text{assume } m \in \mathbb{Z}_+ \\
\quad \text{with } A2 \text{ II have} \\
\quad \text{abs}(s(m) - r(m)) \leq L \text{ and } L \leq M \\
\quad \text{by } \text{auto} \\
\quad \quad \text{then have } \text{abs}(s(m) - r(m)) \leq M \\
\quad \quad \text{by } (\text{rule } \text{Int}_\text{order}_\text{transitive}) \}
\quad \text{moreover} \\
\quad \{ \text{assume } A6: (-m) \in \mathbb{Z}_+ \\
\quad \quad \text{from } T \text{ have } \text{abs}(s(m) - r(m)) \leq \text{abs}(s(m) + s(-m)) + \text{abs}(r(m) + r(-m)) + \text{abs}(s(-m) - r(-m)) \\
\quad \quad \text{using } \text{Int}_\mathbb{Z}_\text{.} \text{one}_\text{L14} \text{ Int}_\text{ineq}_\text{add}_\text{sides} \text{ by } \text{auto} \\
\quad \quad \text{then have} \\
\quad \quad \text{abs}(s(m) + s(-m)) + \text{abs}(r(m) + r(-m)) + \text{abs}(s(-m) - r(-m)) \leq c \\
\quad \quad \text{c} \leq M \\
\quad \quad \text{using } \text{Int}_\mathbb{Z}_\text{.} \text{two}_\text{one}_\text{L14} \text{ Int}_\text{ineq}_\text{add}_\text{sides} \text{ by } \text{auto} \\
\quad \quad \text{ultimately have } \text{abs}(s(m) - r(m)) \leq M \\
\quad \quad \text{by } (\text{rule } \text{Int}_\text{order}_\text{transitive}) \}
\quad \text{ultimately have } \text{abs}(s(m) - r(m)) \leq M \\
\quad \text{by } (\text{rule } \text{Int}_\text{order}_\text{transitive}) \}
\quad \text{ultimately have } \text{abs}(s(m) - r(m)) \leq M \\
\quad \text{by } \text{auto} \\
\quad \text{then have } \forall m \in \mathbb{Z}. \text{ abs}(s(m) - r(m)) \leq M \\
\quad \text{by } \text{simp} \\
\quad \text{with } A1 \text{ show } s \sim r \text{ by } (\text{rule } \text{Int}_\mathbb{Z}_\text{.} \text{two}_\text{one}_\text{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 \( A1: \ s \in S \) \( M \in \mathbb{Z} \)
and \( A2: \forall m \in \mathbb{Z}_+. \ m \leq s(m) \land s(m) \leq m + M \)
shows \( s \sim \text{id}(\mathbb{Z}) \)

**Proof** -
let \( r = \text{id}(\mathbb{Z}) \)
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_1_3_L23 PositiveSet_def id_conv by simp
ultimately show \( s \sim \text{id}(\mathbb{Z}) \) 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 \( A1 A2 A3 \) have \( I: \ \text{group1}(G, f) \)
\( s \in \text{AlmostHoms}(G, f) \)
\( c \in G \)
\( r = \{\langle x, f(s(x), c) \rangle. x \in G\} \)
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

568
48.2 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  
abs(f((p+1)q)-(p+1)f(q)) \leq abs(\delta(f,pq,q)) + abs(f(pq)-pf(q))  
abs(f((p-1)q)-(p-1)f(q)) \leq abs(\delta(f,(p-1)q,q)) + abs(f(pq)-pf(q))  
proof -
let \( R = \mathbb{Z} \)
let \( A = \text{IntegerAddition} \)
let \( M = \text{IntegerMultiplication} \)
let \( I = \text{GroupInv}(R,A) \)
let \( a = f((p+1)q) \)
let \( b = p \)
let \( c = f(q) \)
let \( d = f(pq) \)
from A1 A2 have T1:
  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 \( A\langle a,I(M\langle A\langle b, TheNeutralElement(R, M)\rangle, c)\rangle = A\langle A\langle a, I(d)\rangle, I(c)\rangle, A\langle d, I(M\langle b, c)\rangle\rangle \)
by (rule ring0.Ring_ZF_2_L2)
with A2 have \( f((p+1)q)-(p+1)f(q) = \delta(f,pq,q)+(f(pq)-pf(q)) \)
using int_zero_one_are_int Int_ZF_1_1_L1 L_int_ZF_1_1_L5 apply_funtype by auto
moreover from A1 A2 T1 have \( \delta(f,pq,q) \in \mathbb{Z} \ f(pq)-pf(q) \in \mathbb{Z} \)
using Int_ZF_1_1_L5 apply_functype by auto
ultimately show  
  abs(f((p+1)q)-(p+1)f(q)) \leq abs(\delta(f,pq,q)) + abs(f(pq)-pf(q))  
using Int_triangle_ineq by simp
from A1 A2 have \( T1: \)
f((p-1)q) \in \mathbb{Z} \ p \in \mathbb{Z} \ f(q) \in \mathbb{Z} \ f(pq) \in \mathbb{Z} 
using int_zero_one_are_int Int_ZF_1_1_L5 apply_functype by auto
then have \( f((p-1)q)-(p-1)f(q) = (f(pq)-pf(q))-(f(pq)-f((p-1)q)-f(q)) \)
by (rule Int_ZF_1_2_L6)
with A2 have \( f((p-1)q)-(p-1)f(q) = (f(pq)-pf(q)) - \delta(f,(p-1)q,q) \)
using Int_ZF_1_2_L7 by simp
moreover from A1 A2 have
\[ f(p \cdot q) - f(q) \in \mathbb{Z} \quad \delta(f, (p-1) \cdot q, q) \in \mathbb{Z} \]

using `Int_ZF_1_1_L5` `int_zero_one_are_int` `apply_funtype` by auto
ultimately show
\[ \text{abs}(f((p-1) \cdot q) - (p-1) \cdot f(q)) \leq \text{abs}(\delta(f, (p-1) \cdot q, q)) + \text{abs}(f(p \cdot q) - p \cdot f(q)) \]
using `Int_triangle_ineq1` by simp

qed

If \( f \) is a slope, then \( |f(p \cdot q) - p \cdot f(q)| \leq (|p| + 1) \cdot \text{max}\delta(f) \). 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 \( A2: 0 \leq p \quad q \in \mathbb{Z} \)
and \( A3: \text{abs}(f(p \cdot q) - p \cdot f(q)) \leq (\text{abs}(p)+1) \cdot \text{max}\delta(f) \)
shows \( \text{abs}(f((p+1) \cdot q) - (p+1) \cdot f(q)) \leq (\text{abs}(p+1)+1) \cdot \text{max}\delta(f) \)
proof -
from \( A2 \) have \( q \in \mathbb{Z} \quad p \cdot q \in \mathbb{Z} \)
using `Int_ZF_2_L1A` `int_zero_one_are_int` `Int_ZF_1_1_L5` by auto
with \( A1 \) have \( I: \text{abs}(\delta(f, p \cdot q, q)) \leq \text{max}\delta(f) \) by (rule `Int_ZF_2_1_L7`)
moreover note \( A3 \)
moreover from \( A1 \) \( A2 \) have
\[ \text{abs}(f((p+1) \cdot q) - (p+1) \cdot f(q)) \leq \text{abs}(\delta(f, p \cdot q, q)) + \text{abs}(f(p \cdot q) - p \cdot f(q)) \]
using `AlmostHoms_def` `Int_ZF_2_L1A` `Int_ZF_2_2_L1` by simp
ultimately have
\[ \text{abs}(f((p+1) \cdot q) - (p+1) \cdot f(q)) \leq \text{max}\delta(f) + (\text{abs}(p)+1) \cdot \text{max}\delta(f) \]
by (rule `Int_ZF_2_L15`)
moreover from \( I \) \( A2 \) have
\[ \text{max}\delta(f) + (\text{abs}(p)+1) \cdot \text{max}\delta(f) = (\text{abs}(p+1)+1) \cdot \text{max}\delta(f) \]
using `Int_ZF_2_L1A` `Int_ZF_2_2_L2` by simp
ultimately show
\[ \text{abs}(f((p+1) \cdot q) - (p+1) \cdot f(q)) \leq (\text{abs}(p+1)+1) \cdot \text{max}\delta(f) \]
by simp
qed

If \( f \) is a slope, then \( |f(p \cdot q) - p \cdot f(q)| \leq (|p| + 1) \cdot \text{max}\delta(f) \). 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 \( A2: p \leq 0 \quad q \in \mathbb{Z} \)
and \( A3: \text{abs}(f(p \cdot q) - p \cdot f(q)) \leq (\text{abs}(p)+1) \cdot \text{max}\delta(f) \)
shows \( \text{abs}(f((p-1) \cdot q) - (p-1) \cdot f(q)) \leq (\text{abs}(p-1)+1) \cdot \text{max}\delta(f) \)
proof -
from \( A2 \) have \( q \in \mathbb{Z} \quad (p-1) \cdot q \in \mathbb{Z} \)
using `Int_ZF_2_L1A` `int_zero_one_are_int` `Int_ZF_1_1_L5` by auto
with \( A1 \) have \( I: \text{abs}(\delta(f, (p-1) \cdot q, q)) \leq \text{max}\delta(f) \) 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  
\[ \text{abs}\left(f(p \cdot q) - (p-1) \cdot f(q)\right) \leq \text{max}\delta(f) + (\text{abs}(p)+1) \cdot \max\delta(f) \]  
by (rule Int_ZF_2_L15)

with I A2 show thesis using Int_ZF_2_L1A Int_ZF_1_2_L5 by simp

qed

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) Int_ZF_2_2_L4:

assumes \( A1: f \in S \) and \( A2: p \in \mathbb{Z} \), \( q \in \mathbb{Z} \)

shows \( \text{abs}(f(p \cdot q) - p \cdot f(q)) \leq (\text{abs}(p)+1) \cdot \max\delta(f) \)

proof -

\{ assume \( 0 \leq p \)

moreover from \( A1 \) \( A2 \) have
\[ \text{abs}(f(0 \cdot q) - 0 \cdot f(q)) \leq (\text{abs}(0)+1) \cdot \max\delta(f) \]
using int_zero_one_are_int Int_ZF_2_1_L2B Int_ZF_1_1_L4
Int_ZF_2_1_L8 Int_ZF_2_2_L18 by simp

moreover from \( A1 \) \( A2 \) have \n\[ \forall p. \ 0 \leq p \land \text{abs}(f(p \cdot q) - p \cdot f(q)) \leq (\text{abs}(p)+1) \cdot \max\delta(f) \rightarrow \]
\[ \text{abs}(f((p+1) \cdot q) - (p+1) \cdot f(q)) \leq (\text{abs}(p+1)+1) \cdot \max\delta(f) \]
using Int_ZF_2_2_L2 by simp

ultimately have \( \text{abs}(f(p \cdot q) - p \cdot f(q)) \leq (\text{abs}(p)+1) \cdot \max\delta(f) \)
by (rule Induction_on_int) \}

moreover
\{ assume \( \neg(0 \leq p) \)

with \( A2 \) have \( p \leq 0 \) using Int_ZF_2_L19A by simp

moreover from \( A1 \) \( A2 \) have \n\[ \text{abs}(f(0 \cdot q) - 0 \cdot f(q)) \leq (\text{abs}(0)+1) \cdot \max\delta(f) \]
using int_zero_one_are_int Int_ZF_2_1_L2B Int_ZF_1_1_L4
Int_ZF_2_1_L8 Int_ZF_2_2_L18 by simp

moreover from \( A1 \) \( A2 \) have \n\[ \forall p. \ p \leq 0 \land \text{abs}(f(p \cdot q) - p \cdot f(q)) \leq (\text{abs}(p)+1) \cdot \max\delta(f) \rightarrow \]
\[ \text{abs}(f((p+1) \cdot q) - (p+1) \cdot f(q)) \leq (\text{abs}(p+1)+1) \cdot \max\delta(f) \]
using Int_ZF_2_2_L3 by simp

ultimately have \( \text{abs}(f(p \cdot q) - p \cdot f(q)) \leq (\text{abs}(p)+1) \cdot \max\delta(f) \)
by (rule Back_induct_on_int) \}

ultimately show thesis by blast

qed

The next elegant result is Lemma 7 in the Arthan’s paper [2].

lemma (in int1) Arthan_Lem_7:

assumes \( A1: f \in S \) and \( A2: p \in \mathbb{Z} \), \( q \in \mathbb{Z} \)

shows \( \text{abs}(q \cdot f(p) - p \cdot f(q)) \leq (\text{abs}(p) + \text{abs}(q)+2) \cdot \max\delta(f) \)

proof -

from \( A1 \) \( A2 \) have \( T: \)
\[ q \cdot f(p) - p \cdot f(q) \in \mathbb{Z} \]
\[ f(p \cdot q) - p \cdot f(q) \in \mathbb{Z} \]
\[ f(q \cdot p) \in \mathbb{Z} \]
\[ f(p \cdot q) \in \mathbb{Z} \]
\[ q \cdot f(p) \in \mathbb{Z} \]
\[ p \cdot f(q) \in \mathbb{Z} \]
\[ \max\delta(f) \in \mathbb{Z} \]
\[ \text{abs}(q) \in \mathbb{Z} \]
\[ \text{abs}(p) \in \mathbb{Z} \]
using Int_ZF_1_1_L5 Int_ZF_2_1_L2B Int_ZF_2_1_L7 Int_ZF_2_L14 by auto
moreover have \(|q| \cdot f(p) - f(p)\cdot q| \leq (|q|+1) \max \delta(f)
proof -
from A1 A2 have \(|f(q \cdot p) - q \cdot f(p)| \leq (|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 from A1 A2 have \(|f(p \cdot q) - p \cdot f(q)| \leq (|p|+1) \max \delta(f)
using Int_ZF_2_2_L4 by simp
ultimately have \(|q| \cdot f(p) - f(p)\cdot q| + (f(p \cdot q) - p \cdot f(q))| \leq (|q|+1) \max \delta(f) + (|p|+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 \( \exists A \ B. \ A \in \mathbb{Z} \land B \in \mathbb{Z} \land (\forall p \in \mathbb{Z}. \ |f(p)| \leq A \cdot |p| + B) \)
proof -
let A = \( \max \delta(f) + |f(1)| \)
let B = 3 \( \max \delta(f) \)
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 \( \forall p \in \mathbb{Z}. \ |f(p)| \leq A \cdot |p| + B \)
proof
fix p assume A2: \( p \in \mathbb{Z} \)
with A1 have T:
\( f(p) \in \mathbb{Z} \land |p| \in \mathbb{Z} \land f(1) \in \mathbb{Z} \)
\( p \cdot f(1) \in \mathbb{Z} \land 3 \in \mathbb{Z} \land \max \delta(f) \in \mathbb{Z} \)
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 \( |p| \cdot f(p) - f(1) \cdot p| \leq (|p|+1) \max \delta(f) + (|p|+1) \max \delta(f) \)
using Int_ZF_2_L16A Int_ZF_1_1_L4 Int_ZF_1_2_L11
Int_triangle_ineq2 by simp
with A2 T show \( |f(p)| \leq A \cdot |p| + B \)
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: \( f \in S \land g \in S \)

572
shows \( f \circ g \sim g \circ f \)

**proof** -

from \( A_1 \) have

\[ \exists A B. A \in \mathbb{Z} \land B \in \mathbb{Z} \land (\forall p \in \mathbb{Z}. \, \text{abs}(f(p)) \leq A \cdot \text{abs}(p) + B) \]

\[ \exists C D. C \in \mathbb{Z} \land D \in \mathbb{Z} \land (\forall p \in \mathbb{Z}. \, \text{abs}(g(p)) \leq C \cdot \text{abs}(p) + D) \]

using \textit{Arthan}. by auto

then obtain \( A B C D \) where \( D_1: A \in \mathbb{Z} \land B \in \mathbb{Z} \land C \in \mathbb{Z} \land D \in \mathbb{Z} \) and \( D_2: \)

\[ \forall p \in \mathbb{Z}. \, \text{abs}(f(p)) \leq A \cdot \text{abs}(p) + B \]

\[ \forall p \in \mathbb{Z}. \, \text{abs}(g(p)) \leq C \cdot \text{abs}(p) + D \]

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 \( T_1: \)

\( g(p) \in \mathbb{Z} \land f(p) \in \mathbb{Z} \land \text{abs}(p) \in \mathbb{Z} \land 2 \in \mathbb{Z} \)

\( f(g(p)) \in \mathbb{Z} \land g(f(p)) \in \mathbb{Z} \land f(g(p)) - g(f(p)) \in \mathbb{Z} \)

\[ p : f(g(p)) \in \mathbb{Z} \land p : g(f(p)) \in \mathbb{Z} \]

\[ \text{abs}(f(g(p)) - g(f(p))) \in \mathbb{Z} \]

using \textit{IntZF_2_1_L2B IntZF_2_1_L10 IntZF_2_1_L5 IntZF_2_2_L14 int_two_three_are_int} by auto

with \( A_1 \) \( A_2 \) have

\[ \text{abs}(f(g(p)) - g(f(p))) \leq (\text{abs}(p) + \text{abs}(f(p)) + 2) \cdot \max \delta(g) + (\text{abs}(p) + \text{abs}(f(p)) + 2) \cdot \max \delta(f) \]

using \textit{Arthan_Lem_7 IntZF_2_1_L10A IntZF_1_2_L12} by simp

moreover have

\[ (\text{abs}(p) + \text{abs}(f(p)) + 2) \cdot \max \delta(g) + (\text{abs}(p) + \text{abs}(g(p)) + 2) \cdot \max \delta(f) \leq ((\max \delta(g) \cdot (A+1) + \max \delta(f) \cdot (C+1))) \cdot \text{abs}(p) + ((B \cdot \max \delta(g) + 2 \cdot \max \delta(g)) + (D \cdot \max \delta(f) + 2 \cdot \max \delta(f))) \]

**proof** -

from \( D_2 \) \( A_2 \) \( T_1 \) have

\[ \text{abs}(p) + \text{abs}(f(p)) + 2 \leq \text{abs}(p) + (A \cdot \text{abs}(p) + B) + 2 \]

\[ \text{abs}(p) + \text{abs}(g(p)) + 2 \leq \text{abs}(p) + (C \cdot \text{abs}(p) + D) + 2 \]

using \textit{IntZF_2_2_L15C} by auto

with \( A_1 \) have

\[ (\text{abs}(p) + \text{abs}(f(p)) + 2) \cdot \max \delta(g) \leq (\text{abs}(p) + (A \cdot \text{abs}(p) + B) + 2) \cdot \max \delta(g) \]

\[ (\text{abs}(p) + \text{abs}(g(p)) + 2) \cdot \max \delta(f) \leq (\text{abs}(p) + (C \cdot \text{abs}(p) + D) + 2) \cdot \max \delta(f) \]

using \textit{IntZF_2_1_L8 IntZF_1_3_L13} by auto

moreover from \( A_1 \) \( D_1 \) \( T_1 \) have

\[ (\text{abs}(p) + (A \cdot \text{abs}(p) + B) + 2) \cdot \max \delta(g) = \max \delta(g) \cdot (A+1) \cdot \text{abs}(p) + (B \cdot \max \delta(g) + 2 \cdot \max \delta(g)) \]

\[ (\text{abs}(p) + (C \cdot \text{abs}(p) + D) + 2) \cdot \max \delta(f) = \max \delta(f) \cdot (C+1) \cdot \text{abs}(p) + (D \cdot \max \delta(f) + 2 \cdot \max \delta(f)) \]

using \textit{IntZF_2_1_L8 IntZF_1_2_L13} by auto

ultimately have

\[ (\text{abs}(p) + \text{abs}(f(p)) + 2) \cdot \max \delta(g) + (\text{abs}(p) + \text{abs}(g(p)) + 2) \cdot \max \delta(f) \leq (\max \delta(g) \cdot (A+1) \cdot \text{abs}(p) + (B \cdot \max \delta(g) + 2 \cdot \max \delta(g))) + (\max \delta(f) \cdot (C+1) \cdot \text{abs}(p) + (D \cdot \max \delta(f) + 2 \cdot \max \delta(f))) \]

using \textit{ineq_add_sides} by simp

moreover from \( A_1 \) \( A_2 \) \( D_1 \) have \( \text{abs}(p) \in \mathbb{Z} \)
max\(\delta(g) \cdot (A+1)\) ∈ \(\mathbb{Z}\)  B·max\(\delta(g) + 2\cdot\max\delta(g)\) ∈ \(\mathbb{Z}\)
max\(\delta(f) \cdot (C+1)\) ∈ \(\mathbb{Z}\)  D·max\(\delta(f) + 2\cdot\max\delta(f)\) ∈ \(\mathbb{Z}\)

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)))\cdot 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))) ∈ \(\mathbb{Z}\)
using Int_ZF_1_3_L5 by auto

} then have
\(\forall p ∈ \mathbb{Z}. \ abs(f(g(p))-g(f(p))) ∈ \mathbb{Z}\)
\(\forall p ∈ \mathbb{Z}. \ abs(f(g(p))-g(f(p)))·abs(p) ≤ E·abs(p) + F\)
by auto
moreover from A1 D1 have E ∈ \(\mathbb{Z}\)  F ∈ \(\mathbb{Z}\)
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
\(∃ L. \ ∀ p ∈ \mathbb{Z}. \ abs(f(g(p))-g(f(p))) ≤ L\)
by (rule Int_ZF_1_7_L1)
with A1 obtain L where \(\forall p ∈ \mathbb{Z}. \ abs((f\circ g)(p)-(g\circ f)(p)) ≤ L\)
using Int_ZF_2_1_L10 by auto
moreover from A1 have f\circ g ∈ S  g\circ f ∈ S
using Int_ZF_2_1_L11 by auto
ultimately show f\circ g ≈ g\circ f using Int_ZF_2_1_L9 by auto
qed

eend

49  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.

49.1  Positive slopes

This section provides background material for defining the order relation on real numbers.
Positive slopes are functions (of course.)

**lemma (in int1) Int_ZF_2_3_L1:** assumes $A1: f \in S^+$ shows $f: \mathbb{Z} \to \mathbb{Z}$
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 $A1: f \in S^+$ 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 $f: \mathbb{Z} \to \mathbb{Z}^+ \subseteq \mathbb{Z}^+$
using AlmostHoms_def PositiveSet_def by auto
with $A2$ 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 $A1: f \in S^+$ and $A2: D \in \mathbb{Z}^+$
sends $\exists M \in \mathbb{Z}^+. \ \forall m \in \mathbb{Z}^+. \ (m+1) \cdot D \leq f(m \cdot M)$
proof -
let $E = \max \delta(f) + D$
let $A = f(\mathbb{Z}^+) \cap \mathbb{Z}^+$
from $A1$ and $A2$ have $D \leq E$
using Int_ZF_1_5_L3 Int_ZF_2_1_L8 Int_ZF_2_3_L1A Int_ZF_2_15D by simp
from $A1$ and $A2$ have $A \subseteq \mathbb{Z}^+$
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_1_5_L2A 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_1_5_L3 by simp
moreover from $II$ $III$ have $(1+1) \cdot E \leq f(1 \cdot M)$
using PositiveSet_def Int_ZF_1_1_L4 by simp
moreover have $\forall k. \ 1 \leq k \land (k+1) \cdot E \leq f(k \cdot M) \implies (k+1+1) \cdot E \leq f((k+1) \cdot M)$
proof -
{ fix $k$ assume $A5: 1 \leq k$ and $A6: (k+1) \cdot E \leq f(k \cdot M)$
with $A1$ $A2$ $II$ have $T: k \in \mathbb{Z}^+ \land k+1 \in \mathbb{Z}^+ \land (k+1) \cdot E \in \mathbb{Z}^+ \land 2 \cdot E \in \mathbb{Z}^+$
using Int_ZF_2_3_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
$\delta(f,k \cdot M, M) \in \mathbb{Z}^+$
using Int_ZF_2_3_L1A PositiveSet_def Int_ZF_1_1_L5
Int_ZF_2_1_L7 Int_ZF_2_16C by auto
with $III$ $A6$ have
$(k+1) \cdot E + (2 \cdot E - E) \leq f(k \cdot M) + (f(M) + \delta(f,k \cdot M,M))$
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 $f \in S_+$ $ f \notin 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 $f \notin S_+$ by auto

qed

We want to show that if $f$ is a slope and neither $f$ nor $-f$ are in $S_+$, then
$f$ 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 A1: $f \in S$ and A2: $f \notin S_+$
shows IsBoundedAbove($f(Z_+)$, IntegerOrder)
**proof** -
from A1 have f: $\mathbb{Z} \rightarrow \mathbb{Z}$ using AlmostHoms_def by simp
then have $f(Z_+) \subseteq \mathbb{Z}$ using func1_1_L6 by simp
moreover from A1 A2 have $f(Z_+) \cap Z_+ \in \text{Fin}(\mathbb{Z})$ by auto
ultimately show thesis using Int_ZF_2_T1 group3.OrderedGroupZF_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 A1: $f \in S$ and A2: $-f \notin S_+$
shows IsBoundedBelow($f(Z_+)$, IntegerOrder)
**proof** -
from A1 have T: $f : \mathbb{Z} \rightarrow \mathbb{Z}$ using AlmostHoms_def by simp
then have $(-f(Z_+)) = (-f)(Z_+)$ using Int_ZF_1_T2 group0_2_T2 PositiveSet_def func1_1_L15C by auto
with A1 A2 T show IsBoundedBelow($f(Z_+)$, IntegerOrder)
using Int_ZF_2_1_L12 Int_ZF_2_3_L2 PositiveSet_def func1_1_L6 Int_ZF_2_T1 group3.OrderedGroupZF_2_L5 by simp
qed

A slope that is bounded on $Z_+$ is bounded everywhere.

**lemma (in int1) Int_ZF_2_3_L4:**
assumes A1: $f \in S$ and A2: $m \in \mathbb{Z}$
and A3: $\forall n \in Z_+. \ abs(f(n)) \leq L$
shows $abs(f(m)) \leq 2 \max \delta(f) + L$
**proof** -
from A1 A3 have
$0 \leq abs(f(1)) \leq L$
using 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 Int_order_transitive)
note A2
moreover have $abs(f(0)) \leq 2 \max \delta(f) + L$
**proof** -
from A1 have
$abs(f(0)) \leq \max \delta(f)$  
0 $\leq \max \delta(f)$
and T: $\max \delta(f) \in \mathbb{Z}$
using Int_ZF_2_1_L8 by auto
with II have $abs(f(0)) \leq \max \delta(f) + \max \delta(f) + L$
using Int_ZF_2_L15F by simp
with T show thesis using Int_ZF_1_1_L4 by simp
qed
moreover from A1 A3 II have
∀n∈\mathbb{Z}_+. \abs(f(n)) \leq 2 \cdot \max\delta(f) + L

by simp

moreover have ∀n∈\mathbb{Z}_+. \abs(f(-n)) \leq 2 \cdot \max\delta(f) + L

proof

fix n assume n∈\mathbb{Z}_+

with A1 A3 have 2 \cdot \max\delta(f) ∈ Z

abs(f(n)) \leq 2 \cdot \max\delta(f) + abs(f(n))

abs(f(n)) \leq L

using int_two_three_are_int Int_ZF_2_1_L8 Int_ZF_1_1_L5

PositiveSet_def Int_ZF_2_1_L14 by auto

then show abs(f(-n)) \leq 2 \cdot \max\delta(f) + L

using Int_ZF_2_L15A by blast

qed

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∈S and A2: IsBounded(f(\mathbb{Z}_+), IntegerOrder)

shows f ∈ FinRangeFunctions(\mathbb{Z}, \mathbb{Z})

proof -

have T1: \mathbb{Z}_+ ⊆ Z using PositiveSet_def by auto

from A1 have T2: f:Z→Z using AlmostHoms_def by simp

from A2 obtain L where ∀a∈f(\mathbb{Z}_+). \abs(a) \leq L

using Int_ZF_1_3_L20A by auto

with T2 T1 have ∀n∈\mathbb{Z}_+. \abs(f(n)) \leq L

by (rule func1_1_L15B)

with A1 have ∀m∈\mathbb{Z}_+. \abs(f(m)) \leq 2 \cdot \max\delta(f) + L

using Int_ZF_2_3_L4 by simp

with T2 have f(\mathbb{Z}) ∈ Fin(\mathbb{Z})

by (rule Int_ZF_1_3_L20C)

with T2 show f ∈ 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∈S and IsBoundedBelow(f(\mathbb{Z}_+), IntegerOrder)

shows f ∈ FinRangeFunctions(\mathbb{Z}, \mathbb{Z}) ∨ f∈S_+

using assms Int_ZF_2_3_L2 IsBounded_def Int_ZF_2_3_L4A by 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∈S g∈S and
A2: \( \forall n \in \mathbb{Z}_+. \, f(n) \leq g(n) \)
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 \( \forall n \in \mathbb{Z}_+. \, \langle 0, h(n) \rangle \in \text{IntegerOrder} \)
using Int_ZF_2_1_L2B PositiveSet_def Int_ZF_1_3_L10A Int_ZF_2_1_L12 Int_ZF_2_1_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 have T1: \( \min(f,0..(M-1)) - \max(f) \in \mathbb{Z} \)
using Arthan_L_3_spec by auto
with A2 have T2: \( K - (\min(f,0..(M-1)) - \max(f)) - 1 \in \mathbb{Z} \)
using Int_ZF_2_1_L15 Int_ZF_2_1_L8 Int_ZF_1_1_L5 PositiveSet_def
by auto
with A2 have T3: \( K - (\min(f,0..(M-1)) - \max(f)) - 1 \leq j \)
using Int_ZF_1_3_L18 by auto
with A2 T1 T2 have IV: \( K \leq j+1 + (\min(f,0..(M-1)) - \max(f)) \)
using int_zero_one_are_int Int_ZF_2_L9C by simp

579
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 \( \forall m. N \leq m \implies K \leq f(m) \)
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: \( f \in S^+ \) and A2: \( K \in \mathbb{Z}^+ \)
shows \( \exists N \in \mathbb{Z}^+. \forall m. N \leq m \implies f(-m) \leq K \)

proof -
from A1 have T1: \( \text{abs}(f(0)) + \text{max}(f) \in \mathbb{Z} \)
using Int_ZF_2_L1A by auto
with A2 have \( \text{abs}(f(0)) + \text{max}(f) \in \mathbb{Z} \)
using Int_ZF_1_1_L5 by simp
with A1 have
\( \exists N \in \mathbb{Z}^+. \forall m. N \leq m \implies \text{abs}(f(0)) + \text{max}(f) - K \leq f(m) \)

580
using Int_ZF_2_3_L5 by simp
then obtain N where I: \( N \in \mathbb{Z}_+ \) and II:
\[ \forall m. \, N \leq m \longrightarrow \text{abs}(f(0)) + \text{max}\delta(f) - K \leq f(m) \]
by auto
\{ fix m assume A3: \( N \leq m \)
with A1 have
\[ f(-m) \leq \text{abs}(f(0)) + \text{max}\delta(f) - f(m) \]
using Int_ZF_2_L1A Int_ZF_2_1_L14 by simp
moreover from II T1 A3 have \( f(-m) \leq \text{abs}(f(0)) + \text{max}\delta(f) - f(m) \leq K \)
using Int_ZF_2_L10 int_ord_transl_inv by simp
ultimately have \( f(-m) \leq K \)
by (rule Int_order_transitive)
\}
then have \( \forall m. \, N \leq m \longrightarrow f(-m) \leq K \)
by simp
with I show thesis by auto
qed

A special case of Int_ZF_2_3_L5 where \( K = 1 \).

corollary (in int) Int_ZF_2_3_L6: assumes \( f \in \mathcal{S}_+ \)
shows \( \exists N \in \mathbb{Z}_+. \, \forall m. \, N \leq m \longrightarrow f(m) \in \mathbb{Z}_+ \)
using assms int_zero_one_are_int Int_ZF_2_3_L5 Int_ZF_1_5_L3 by simp

A special case of Int_ZF_2_3_L5 where \( m = N \).

corollary (in int) Int_ZF_2_3_L6A: assumes \( f \in \mathcal{S}_+ \) and \( K \in \mathbb{Z} \)
shows \( \exists N \in \mathbb{Z}_+. \, K \leq f(N) \)
proof -
from assms have \( \exists N \in \mathbb{Z}_+. \, \forall m. \, N \leq m \longrightarrow K \leq f(m) \)
using Int_ZF_2_3_L5 by simp
then obtain N where I: \( N \in \mathbb{Z}_+ \) and II: \( \forall m. \, N \leq m \longrightarrow K \leq f(m) \)
by auto
then show thesis using PositiveSet_def int_ord_is_refl refl_def by auto
qed

If values of a slope are not bounded above, then the slope is positive.

lemma (in int) Int_ZF_2_3_L7: assumes A1: \( f \in \mathcal{S} \)
and A2: \( \forall K \in \mathbb{Z}. \, \exists n \in \mathbb{Z}_+. \, K \leq f(n) \)
signs f \in \mathcal{S}_+ \)
proof -
\{ fix K assume K∈\mathbb{Z}
with A2 obtain n where n∈\mathbb{Z}_+ \, K \leq f(n)
by auto
moreover from A1 have \( \mathbb{Z}_+ \subseteq \mathbb{Z}. \, f: \mathbb{Z} \to \mathbb{Z} \)
using PositiveSet_def AlmostHoms_def by auto

581
ultimately have $\exists m \in f(Z_+). K \leq m$

using func1_1_L5D by auto

} then have $\forall K \in Z. \exists m \in f(Z_+). K \leq m$ by simp

with A1 show $f \in S_+$ using Int_ZF_4_L9 Int_ZF_2_3_L2 by auto

qed

For unbounded slope $f$ either $f \in S_+$ or $-f \in S_+$.

theorem (in int1) 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 -
\{ assume A3: $f \in S_+$ and A4: $(-f) \in S_+$
\}

have T1: $Z_+ \subseteq Z$ using PositiveSet_def by auto

from A1 have T2: $f: Z \rightarrow Z$ using AlmostHoms_def by simp

then have I: $f(Z_+) \subseteq Z$ using func1_1_L6 by auto

from A1 A2 have A12: $f \in S_+ \lor (-f) \in S_+$
using Int_ZF_2_3_L2 Int_ZF_2_3_L3 IsBounded_def 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 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 Int_ZF_2_3_L6 by auto

let N = GreaterOf(IntegerOrder,N1,N2)

from I III have N1 \leq N N2 \leq N using PositiveSet_def 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 Int_ZF_2_L1A PositiveSet_def Int_ZF_2_1_L12A by auto \}

then have False using Int_ZF_1_5_L8 by simp

thus thesis by auto

qed

ultimately show $(f \in S_+) \lor (-f) \in S_+$

using xor_def by simp

qed

The sum of positive slopes is a positive slope.

theorem (in int1) sum_of_pos_sls_is_pos_sl:
assumes A1: $f \in S_+$ $g \in S_+$
shows $f \div g \in S_+$
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 Int_ZF_2_3_L5 by simp

582
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 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: $L \in \mathbb{Z}_+$ $\exists M \in \mathbb{Z}_+ \subseteq \mathbb{Z}_+$
using GreaterOf_def PositiveSet_def by auto
moreover from A1 V have (f+g)(L) = f(L) + g(L)
using Int_ZF_2_1_L12B by auto
moreover from I II III IV have $K \leq f(L) + g(L)$
using PositiveSet_def Int_ZF_1_3_L18 Int_ZF_2_L15F by simp
ultimately have $L \in \mathbb{Z}_+$ $K \leq (f+g)(L)$
by auto
then have $\exists n \in \mathbb{Z}_+. K \leq (f+g)(n)$
by auto
} with A1 show $f+g \in S_+$
using Int_ZF_2_1_L12C Int_ZF_2_3_L7 by simp
qed

The composition of positive slopes is a positive slope.

theorem (in int1) 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 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 PositiveSet_def 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 PositiveSet_def Int_ZF_2_1_L10 by auto
} with A1 show $f \circ g \in S_+$
using Int_ZF_2_1_L11 Int_ZF_2_3_L7 by simp
qed

A slope equivalent to a positive one is positive.

lemma (in int1) Int_ZF_2_3_L9:
assumes A1: $f \in S_+$ and A2: $(f, g) \in \text{AlEqRel}$ shows $g \in S_+$
proof -
from A2 have T: $g \in S$ and $\exists L \in \mathbb{Z}. \forall m \in \mathbb{Z}. \text{abs}(f(m)-g(m)) \leq L$
583
using \texttt{IntZF.2.1.L9A} by auto then obtain \( L \) where
\[
I: L \in \mathbb{Z} \quad \text{and} \quad II: \forall m \in \mathbb{Z}. \quad \text{abs}(f(m)-g(m)) \leq L
\]
by auto
\[
\{ \text{fix } K \quad \text{assume } A3: K \in \mathbb{Z} \\
\quad \text{with } I \text{ have } K+L \in \mathbb{Z} \\
\quad \text{using } \texttt{IntZF.1.1.L5} \text{ by simp} \}
\]
with A1 obtain \( M \) where III: \( M \in \mathbb{Z}_+ \) and IV: \( K+L \leq f(M) \)
\[
\text{using } \texttt{IntZF.2.3.L6A} \text{ by auto} \quad \text{by simp}
\]
moreover from A1 II III have 
\[
f(M)-L \leq g(M)
\]
\[
\text{using } \texttt{PositiveSet_def IntZF.2.1.L2B IntZF.2.L9B} \text{ by simp}
\]
ultimately have \( K \leq g(M) \)
\[
\text{by (rule } \texttt{Int_order_transitive}) \quad \text{by auto}
\]
\[
\} \text{ with } T \text{ show } g \in S_+ \\
\quad \text{using } \texttt{IntZF.2.3.L7} \text{ by simp} \]
qed

The set of positive slopes is saturated with respect to the relation of equivalence of slopes.

\textbf{lemma (in int1) pos_slopes_saturated: shows IsSaturated(AlEqRel, S_+)}

\textit{proof -}

have 
\[
equiv(S, \texttt{AlEqRel})
\]
\[
\texttt{AlEqRel} \subseteq S \times S
\]
using \texttt{IntZF.2.1.L9B} by auto
moreover have \( S_+ \subseteq S \) by auto
moreover have \( \forall f \in S_+. \forall g \in S. \ (f,g) \in \texttt{AlEqRel} \rightarrow g \in S_+ \)
using \texttt{IntZF.2.3.L9} by blast
ultimately show \( \text{IsSaturated(AlEqRel, S_+)} \)
\[
\text{by (rule } \texttt{EquivClass.3.L3}) \quad \text{by simp}
\]
qed

A technical lemma involving a projection of the set of positive slopes and a
logical expression with exclusive or.

\textbf{lemma (in int1) IntZF.2.3.L10:}
assumes A1: \( f \in S \) \( g \in S \)
and A2: \( R = \{ \texttt{AlEqRel}\{a\}. \ a \in S_+ \} \)
and A3: \( (f \in S_+) \text{ Xor } (g \in S_+) \)
shows \( \{\texttt{AlEqRel}\{f\} \in R\} \text{ Xor } \{\texttt{AlEqRel}\{g\} \in R\} \)
\textit{proof -}

from A1 A2 A3 have 
\[
equiv(S, \texttt{AlEqRel})
\]

584
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(Z) ∈ S_
proof -
  let f = id(Z)
  { fix K assume K ∈ Z
    then obtain n where T: n ∈ 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 ∈ Z_+ and K ≤ f(n)
      by auto
    then have ∃ n ∈ 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(Z, Z)
shows ¬(id(Z) ∼ f)
proof -
  { from A1 have id(Z) ∈ S_
    using Int_ZF_2_3_L11 by simp
    moreover assume ⟨id(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(Z) ~ f) by auto
qed

49.2 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 o 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 ∈ 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 \( p \in \mathbb{Z}_+ \)
says \( f^{-1}(p) \in \mathbb{Z}_+ \)
\( p \leq f(f^{-1}(p)) \)
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 \( m \in \mathbb{Z}_+ \) and \( p \in \mathbb{Z}_+ \) and \( m \leq f(p) \)
says \( f^{-1}(m) \leq p \)
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 \( f(f^{-1}(m)-1) \leq m \) and \( f(f^{-1}(m)-1) \neq m \)
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: \( \mathbb{Z} \rightarrow \mathbb{Z} \) 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 \( f^{-1}(m) \leq f^{-1}(m)-1 \)\}
by (rule Int_ZF_2_4_L3)
with T have False using Int_ZF_1_2_L3AA
by simp
} then have I: ¬(m ≤ f(f⁻¹(m)-1)) by auto
with T1 show f(f⁻¹(m)-1) ≤ m
by (rule Int_ZF_2_2_L19)
from T1 I show f(f⁻¹(m)-1) ≠ m
by (rule Int_ZF_2_2_L19)

qed

The (candidate for) the inverse of a positive slope is nondecreasing.

lemma (in int1) Int_ZF_2_4_L5:
assumes A1: f ∈ S_+ and A2: m∈\mathbb{Z}_+ and A3: m≤n
shows f⁻¹(m) ≤ f⁻¹(n)
proof -
from A2 A3 have T: n ∈ \mathbb{Z}_+ 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_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 ∈ S_+ and A2: \( \forall m∈\mathbb{Z}_+ \). f⁻¹(m)-1 ∈ \mathbb{Z}_+
shows \( \exists U∈\mathbb{Z}_+. \forall m∈\mathbb{Z}_+. \forall n∈\mathbb{Z}_+. f(f⁻¹(m+n)-f⁻¹(m)-f⁻¹(n)) ≤ U \)
\( \exists U∈\mathbb{Z}_+. \forall m∈\mathbb{Z}_+. \forall n∈\mathbb{Z}_+. N ≤ f(f⁻¹(m+n)-f⁻¹(m)-f⁻¹(n)) \)
proof -
from A1 have \( \exists L∈\mathbb{Z} . \forall r∈\mathbb{Z} . f(r) ≤ f(r-1) + L \)
using Int_ZF_2_1_L28 by simp
then obtain L where
  I: L ∈ ℤ and II: ∀ r ∈ ℤ. f(r) ≤ f(r-1) + L
  by auto
from A1 have
  ∃ H ∈ ℤ. ∀ r ∈ ℤ. ∀ p ∈ ℤ. ∀ q ∈ ℤ. f(r-p-q) ≤ f(r)-f(p)-f(q)+M
  ∃ K ∈ ℤ. ∀ r ∈ ℤ. ∀ p ∈ ℤ. ∀ q ∈ ℤ. f(r)-f(p)-f(q)+K ≤ f(r-p-q)
  using Int_ZF_2_1_L30 by auto
then obtain M K where
  III: M ∈ ℤ and
  IV: ∀ r ∈ ℤ. ∀ p ∈ ℤ. ∀ q ∈ ℤ. f(r-p-q) ≤ f(r)-f(p)-f(q)+M
  and
  V: K ∈ ℤ and VI: ∀ r ∈ ℤ. ∀ p ∈ ℤ. ∀ q ∈ ℤ. f(r)-f(p)-f(q)+K ≤ f(r-p-q)
  by auto
from I III V have
  L+M ∈ ℤ (-L) - L + K ∈ ℤ
  using Int_ZF_1_1_L4 Int_ZF_1_1_L5 by auto
moreover
  { fix m n
    assume A3: m ∈ ℤ+ n ∈ ℤ+
    have f(f⁻¹(m+n)-f⁻¹(m)-f⁻¹(n)) ≤ L+M ∧
    (-L) - L + K ≤ f(f⁻¹(m+n)-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 ∈ ℤ+ q ∈ ℤ+ r ∈ ℤ+
  using Int_ZF_2_4_L2 pos_int_closed_add_unfolded by auto
with A3 have T2:
  m ∈ ℤ n ∈ ℤ p ∈ ℤ q ∈ ℤ r ∈ ℤ
  using PositiveSet_def by auto
from A2 A3 have T3:
  r-1 ∈ ℤ+ p-1 ∈ ℤ+ q-1 ∈ ℤ+
  using pos_int_closed_add_unfolded by auto
from A1 A3 have VII:
  m+n ≤ f(r)
m ≤ f(p)n ≤ f(q)
  using Int_ZF_2_4_L2 pos_int_closed_add_unfolded by auto
from A1 A3 T3 have VIII:
  f(r-1) ≤ m+n
  f(p-1) ≤ m
  f(q-1) ≤ n
  using pos_int_closed_add_unfolded Int_ZF_2_4_L4 by auto
have f(r-p-q) ≤ L+M
  proof -
  from IV T2 have f(r-p-q) ≤ f(r)-f(p)-f(q)+M
    by simp
  moreover

588
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 VII 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_ZF_1_2_L3 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 I 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
  \( f(r-p-q) \leq L+M \)
  \( (-L)-L+K \leq f(f^{-1}(m+n)-f^{-1}(m)-f^{-1}(n)) \)
proof -

qed

ultimately show
\[ \exists U \in \mathbb{Z}. \forall m \in \mathbb{Z}_+. \forall n \in \mathbb{Z}_+. \ f(f^{-1}(m+n)-f^{-1}(m)-f^{-1}(n)) \leq U \]
\[ \exists N \in \mathbb{Z}. \forall m \in \mathbb{Z}_+. \forall n \in \mathbb{Z}_+. \ N \leq f(f^{-1}(m+n)-f^{-1}(m)-f^{-1}(n)) \]
by auto

qed

The expression \( f^{-1}(m+n) - f^{-1}(m) - f^{-1}(n) \) is uniformly bounded for all pairs \((m,n) \in \mathbb{Z}_+ \times \mathbb{Z}_+\). Recall that in the int1 context \( \varepsilon(f,x) \) is defined so that \( \varepsilon(f,(m,n)) = f^{-1}(m+n) - f^{-1}(m) - f^{-1}(n) \).

**lemma (in int1) Int_ZF_2_4_L8:** assumes \( A1: f \in S_+ \) and
\[ A2: \forall m \in \mathbb{Z}_+. \ f^{-1}(m)-1 \in \mathbb{Z}_+ \]
shows \( \exists M. \forall x \in \mathbb{Z}_+ \times \mathbb{Z}_+. \abs{\varepsilon(f,x)} \leq M \)
proof -

from \( A1 \) \( A2 \) have
\[ \exists U \in \mathbb{Z}. \forall m \in \mathbb{Z}_+. \forall n \in \mathbb{Z}_+. \ f(f^{-1}(m+n)-f^{-1}(m)-f^{-1}(n)) \leq U \]
\[ \exists N \in \mathbb{Z}. \forall m \in \mathbb{Z}_+. \forall n \in \mathbb{Z}_+. \ N \leq f(f^{-1}(m+n)-f^{-1}(m)-f^{-1}(n)) \]
by auto

have \( \mathbb{Z}_+ \times \mathbb{Z}_+ \neq \emptyset \) 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
\[ \forall x \in \mathbb{Z}_+ \times \mathbb{Z}_+. \ \varepsilon(f,x) \in \mathbb{Z} \land f(\varepsilon(f,x)) \leq U \land N \leq f(\varepsilon(f,x)) \]
proof -

\{ fix x assume \( A3: x \in \mathbb{Z}_+ \times \mathbb{Z}_+ \)
\[ \text{let } m = \text{fst}(x) \]
\[ \text{let } n = \text{snd}(x) \]
from \( A3 \) have \( T : m \in \mathbb{Z}_+ \ n \in \mathbb{Z}_+ \ m+n \in \mathbb{Z}_+ \)
using pos_int_closed_add_unfolded by auto
with \( A1 \) have
\[ f^{-1}(m+n) \in \mathbb{Z}. \ f^{-1}(m) \in \mathbb{Z}. \ f^{-1}(n) \in \mathbb{Z} \]
using Int_ZF_2_4_L2 PositiveSet_def by auto
with \( I \) \( T \) have
\[ \varepsilon(f,x) \in \mathbb{Z}. \land f(\varepsilon(f,x)) \leq U \land N \leq f(\varepsilon(f,x)) \]
using Int_ZF_1_1_L5 by auto
\} thus thesis by simp
qed

590
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: $g = \{ (p,f^{-1}(p)) \mid p \in \mathbb{Z}_+ \}$
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
$g : \mathbb{Z}_+ \rightarrow \mathbb{Z}_+$
usingZF_fun_from_total by auto

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: $g = \{ (p,f^{-1}(p)) \mid p \in \mathbb{Z}_+ \}$ and A3: $p \in \mathbb{Z}_+$
shows $g(p) = f^{-1}(p)$
proof -
{fix $m$ $n$
assume A4: $m \in \mathbb{Z}_+$ and $n \in \mathbb{Z}_+$
then have $\langle m,n \rangle \in \mathbb{Z}_+ \times \mathbb{Z}_+$ by simp
with A1 have $\text{abs}(\varepsilon(f,\langle m,n \rangle)) \leq L$ by simp
moreover have $\varepsilon(f,\langle m,n \rangle) = f^{-1}(mn) - f^{-1}(m) - f^{-1}(n)$
}

591
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 \( \text{thesis} \) 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 I 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 \( m \leq (f \circ h)(m) \wedge (f \circ h)(m) \leq m+M \)
   by simp \}
ultimately show $f \circ h \sim \text{id}(Z)$ using Int_ZF_2_1_L32
by simp
qed
ultimately show $\exists h \in S. f \circ h \sim \text{id}(Z)$
by auto
qed

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 $\langle m, n \rangle$ 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 $f: Z \rightarrow 2 \in Z$
using AlmostHoms_def int_zero_one_are_int int_two_three_are_int
by auto
moreover have $\forall a \in Z. \exists b \in Z_+. \forall x. b \leq x 
\longrightarrow a \leq f(x)$
using Int_ZF_2_1_L33
by simp
ultimately have $\exists c \in Z_+. 2 \leq \text{Minimum}(\text{IntegerOrder},\{n \in Z_+. 1 \leq f(n)+c\})$
by (rule Int_ZF_1_6_L7)
then obtain $c$ where
I: $c \in Z_+$ and
II: $2 \leq \text{Minimum}(\text{IntegerOrder},\{n \in Z_+. 1 \leq f(n)+c\})$
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 $\langle f,g \rangle \in \text{AlEqRel}$ 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 have $\forall m \in Z_+. g^{-1}(1) \leq g^{-1}(m)$
using Int_ZF_1_5_L3 Int_ZF_2_4_L5
by simp
ultimately have $2 \leq g^{-1}(m)$
by (rule Int_order_transitive)
than 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


then show \( g^{-1}(m)-1 \in \mathbb{Z}_+ \)
using int_zero_one_are_int Int_ZF_1_2_L3 Int_ZF_1_5_L3
by simp
qed
ultimately have \( \exists h \in S. \ g \circ h \sim id(Z) \)
by (rule Int_ZF_2_4_L12)
with III IV show thesis by auto
qed

49.3 Completeness

In this section we consider properties of slopes that are needed for the proof
of completeness of real numbers constructed in Real_ZF_1.thy. 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) 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 Int_ZF_1_1_L5 ZF_fun_from_total
by simp
then show II: \( \forall n \in \mathbb{Z}. \ (m^S)(n) = m \cdot n \) using 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 Int_ZF_1_1_L5 by auto
from A1 A2 II T have \( \delta(m^S,n,k) = m \cdot k - m \cdot k \)
using Int_ZF_1_1_L5 Int_ZF_1_1_L1 Int_ZF_1_2_L3
by simp
also from T have \( \ldots = 0 \) using Int_ZF_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 Int_ZF_2_L18 int_zero_one_are_int int_ord_is_refl 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 Int_ZF_2_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 Real_ZF_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 A1: \( f \in S \)
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 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 +m \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_1_L19C by simp
    moreover from III A2 have abs(f(n)) \leq m \cdot \text{abs}(n)+k using PositiveSet_def by simp
    with A2 have 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 orginal embedding.

lemma (in int1) Int_ZF_2_5_L3: assumes A1: \( m \in \mathbb{Z} \)
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
fix n assume A2: n∈\mathbb{Z}
with A1 have 
(\langle-m\rangle_S)(n) = -(m\cdot n)
(-m\langle s\rangle)(n) = -(m\cdot n)
using Int_ZF_1_1_L4 Int_ZF_2_5_L1 Int_ZF_2_1_L12A
by auto
with A1 A2 show \langle-m\rangle_S(n) = -(m\langle s\rangle)(n)
using Int_ZF_2_1_L5 by simp
qed
ultimately show \langle-m\rangle_S = -(m\langle s\rangle)
using fun_extension_iff
by simp
qed
The sum of embeddings is the embedding of the sum.

**lemma (in int1) Int_ZF_2_5_L3A:** assumes A1: m∈\mathbb{Z}. k∈\mathbb{Z}
shows \langle m\rangle_S + \langle k\rangle_S = \langle m+k \rangle_S

**proof**
from A1 have T1: m+k ∈ \mathbb{Z} using Int_ZF_1_1_L5
by simp
with A1 have T2:
\langle m\rangle_S ∈ S  \langle k\rangle_S ∈ S
\langle m+k \rangle_S ∈ S
using Int_ZF_2_5_L1 Int_ZF_2_1_L12C by auto
then have 
\langle m\rangle_S + \langle k\rangle_S : \mathbb{Z} \rightarrow \mathbb{Z}
\langle m+k \rangle_S : \mathbb{Z} \rightarrow \mathbb{Z}
using AlmostHoms_def by auto
moreover have \forall n∈\mathbb{Z}. \langle m\rangle_S + \langle k\rangle_S)(n) = \langle m+k \rangle_S(n)
**proof**
fix n assume A2: n∈\mathbb{Z}
with A1 T1 T2 have \langle m\rangle_S + \langle k\rangle_S)(n) = (m+k)\cdot n
using Int_ZF_2_1_L12B Int_ZF_2_5_L1 Int_ZF_1_1_L1
by simp
also from T1 A2 have ... = \langle m+k \rangle_S(n)
using Int_ZF_2_5_L1 by simp
finally show \langle m\rangle_S + \langle k\rangle_S)(n) = \langle m+k \rangle_S(n)
by simp
qed
ultimately show \langle m\rangle_S + \langle k\rangle_S = \langle m+k \rangle_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∈\mathbb{Z}. k∈\mathbb{Z}
shows \langle m\rangle_S \circ \langle k\rangle_S = \langle (m\cdot k) \rangle_S

**proof**
from A1 have T1: m·k ∈ \mathbb{Z} using Int_ZF_1_1_L5
by simp
with A1 have T2:

(\langle m^S \rangle) \in S \quad (\langle k^S \rangle) \in S

(\langle m \cdot k \rangle)^S \in S

(\langle m^S \rangle \circ (k^S)) \in S

using Int_ZF_2_5_L1 Int_ZF_2_1_L11 by auto

then have

(\langle m^S \rangle \circ (k^S)) : \mathbb{Z} \rightarrow \mathbb{Z}

(\langle m \cdot k \rangle)^S : \mathbb{Z} \rightarrow \mathbb{Z}

using AlmostHoms_def by auto

moreover have \forall n \in \mathbb{Z}. \ ((\langle m^S \rangle \circ (k^S))(n) = ((\langle m \cdot k \rangle)^S)(n)

proof

fix n assume A2: n \in \mathbb{Z}

with A1 T2 have

(\langle m^S \rangle \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 (\langle m^S \rangle \circ (k^S))(n) = m \cdot k \cdot n

by simp

also from T1 A2 have m \cdot k \cdot n = (\langle m \cdot k \rangle)^S(n)

using Int_ZF_2_5_L1 by simp

finally show (\langle m^S \rangle \circ (k^S))(n) = ((\langle m \cdot k \rangle)^S)(n)

by simp

qed

ultimately show (\langle m^S \rangle \circ (k^S)) = ((\langle m \cdot k \rangle)^S)

using fun_extension_iff by simp

qed

Embedding integers in slopes preserves order.

lemma (in int1) Int_ZF_2_5_L4: assumes m \leq n

shows (\langle m^S \rangle) \sim (n^S) \lor (n^S)^+ + (\langle m^S \rangle)^+ \in S^+

proof -

from A1 have m^S \in S and n^S \in S

using Int_ZF_2_5_L1A Int_ZF_2_5_L1 by auto

moreover from A1 have \forall k \in \mathbb{Z}^+. \ (m^S)(k) \leq (n^S)(k)

using Int_ZF_1_3_L13B Int_ZF_2_5_L1A 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 FinRangeFunctions(\mathbb{Z}, \mathbb{Z})
Embeddings of two integers are almost equal only if the integers are equal.

**Lemma (in int1) IntZF_2_5_L6:**

Assumes: \( A1: \ m \in \mathbb{Z} \) \( k \in \mathbb{Z} \) and \( A2: \ (m^S)^S \sim (k^S)^S \)

Shows: \( m = k \)

**Proof:**

From \( A1 \) have \( T: m-k \in \mathbb{Z} \) using \( \text{IntZF}_1_1L5 \) by \( \text{simp} \)

From \( A1 \) have \( -(k^S) = ((-k)^S) \)

Then have \( m^S + (-k^S) = (m^S)^S + ((-k)^S) \)

By \( \text{simp} \)

With \( A1 \) have \( m^S + (-k^S) = ((m-k)^S) \)

Moreover from \( A1 \) \( A2 \) have \( m^S + (-k^S) \in \text{FinRangeFunctions}(\mathbb{Z}, \mathbb{Z}) \)

Using \( \text{IntZF}_2_5L3A \) by \( \text{simp} \)

Ultimately have \( (m-k)^S \in \text{FinRangeFunctions}(\mathbb{Z}, \mathbb{Z}) \)

By \( \text{simp} \)

With \( T \) have \( m-k = 0 \) using \( \text{IntZF}_2_5L5 \)

By \( \text{simp} \)

With \( A1 \) show \( m=k \) by (rule IntZF_1_1L15)

**QED**

Embedding of 1 is the identity slope and embedding of zero is a finite range function.

**Lemma (in int1) IntZF_2_5_L7:**

Shows:

\[ 1^S = \text{id}(\mathbb{Z}) \]

\[ 0^S \in \text{FinRangeFunctions}(\mathbb{Z}, \mathbb{Z}) \]

**Proof:**

Have \( \text{id}(\mathbb{Z}) = \{(x,x). \ x \in \mathbb{Z}\} \)

Using \( \text{id}_\mathbb{Z} \) by \( \text{blast} \)

Then show \( 1^S = \text{id}(\mathbb{Z}) \) using \( \text{IntZF}_1_1L4 \) by \( \text{simp} \)

Have \( \{0^S(n). \ n \in \mathbb{Z}\} = \{0.n. \ n \in \mathbb{Z}\} \)

Using \( \text{intZeroOneAreInt} \) \( \text{IntZF}_2_5L1 \) by \( \text{simp} \)

Also have \( \ldots = \{0\} \) using \( \text{IntZF}_1_1L4 \) \( \text{IntNotEmpty} \)

By \( \text{simp} \)

Finally have \( \{0^S(n). \ n \in \mathbb{Z}\} = \{0\} \) by \( \text{simp} \)

Then have \( \{0^S(n). \ n \in \mathbb{Z}\} \in \text{Fin}(\mathbb{Z}) \)

Using \( \text{IntZeroOneAreInt} \) \( \text{Finite1L16} \)

By \( \text{simp} \)

Moreover have \( 0^S: \mathbb{Z} \rightarrow \mathbb{Z} \)

Using \( \text{IntZeroOneAreInt} \) \( \text{IntZF}_2_5L1 \) \( \text{AlmostHomsDef} \)

By \( \text{simp} \)

Ultimately show \( 0^S \in \text{FinRangeFunctions}(\mathbb{Z}, \mathbb{Z}) \)

Using \( \text{Finite1L19} \) by \( \text{simp} \)

**QED**

A somewhat technical condition for an embedding of an integer to be "less or
equal” (in the sense apriopriate for slopes) than the composition of a slope and another integer (embedding).

lemma (in int1) Int_ZF_2_5_L8:
assumes A1: f ∈ S and A2: N ∈ ℤ M ∈ ℤ and
A3: ∀ n ∈ ℤ⁺. M·n ≤ f(N·n)
shows M⁵ = f°(N⁵) ∨ (f°(N⁵)) + (-(M⁵)) ∈ S⁺
proof -
from A1 A2 have M⁵ ∈ S f°(N⁵) ∈ S
using Int_ZF_2_5_L1 Int_ZF_2_1_L11 by auto
moreover from A1 A2 A3 have ∀ n ∈ ℤ⁺. (M⁵)(n) ≤ (f°(N⁵))(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 apriopriate for slopes) than embedding of another integer.

lemma (in int1) Int_ZF_2_5_L9:
assumes A1: f ∈ S and A2: N ∈ ℤ M ∈ ℤ and
A3: ∀ n ∈ ℤ⁺. f(N·n) ≤ M·n
shows f°(N⁵) ∼ (M⁵) ∨ (M⁵) + (-(f°(N⁵))) ∈ S⁺
proof -
from A1 A2 have f°(N⁵) ∈ S M⁵ ∈ S
using Int_ZF_2_5_L1 Int_ZF_2_1_L11 by auto
moreover from A1 A2 A3 have ∀ n ∈ ℤ⁺. (f°(N⁵))(n) ≤ (M⁵)(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

50 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.

50.1 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 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 Group_ZF_2.

The set of slopes is defined as the set of almost homomorphisms on the additive group of integers.

\textbf{definition}  
\texttt{Slopes} \equiv \texttt{AlmostHoms(int,IntegerAddition)}

The first operation on slopes (pointwise addition) is a special case of the first operation on almost homomorphisms.

\textbf{definition}  
\texttt{SlopeOp1} \equiv \texttt{AlHomOp1(int,IntegerAddition)}

The second operation on slopes (composition) is a special case of the second operation on almost homomorphisms.

\textbf{definition}  
\texttt{SlopeOp2} \equiv \texttt{AlHomOp2(int,IntegerAddition)}

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.

\textbf{definition}  
\texttt{BoundedIntMaps} \equiv \texttt{FinRangeFunctions(int,int)}

Bounded integer maps form a normal subgroup of slopes. The equivalence relation on slopes is the (group) quotient relation defined by this subgroup.

\textbf{definition}  
\texttt{SlopeEquivalenceRel} \equiv \texttt{QuotientGroupRel(Slopes,SlopeOp1,BoundedIntMaps)}

The set of real numbers is the set of equivalence classes of slopes.
RealNumbers ≡ Slopes//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.

definition
RealAddition ≡ 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.

definition
RealMultiplication ≡ ProjFun2(Slopes,SlopeEquivalenceRel,SlopeOp2)

We first show that we can use theorems proven in some proof contexts (locales). The locale group1 requires assumption that we deal with an abelian group. The next lemma allows to use all theorems proven in the context called group1.

lemma Real_ZF_1_L1: shows group1(int,IntegerAddition)
  using group1_axioms.intro group1_def Int_ZF_1_T2 by simp

Real numbers form a ring. This is a special case of the theorem proven in Ring_ZF_1.thy, where we show the same in general for almost homomorphisms rather than slopes.

theorem Real_ZF_1_T1: shows IsAring(RealNumbers,RealAddition,RealMultiplication)
proof -
  let AH = AlmostHoms(int,IntegerAddition)
  let Op1 = AlHomOp1(int,IntegerAddition)
  let FR = FinRangeFunctions(int,int)
  let Op2 = AlHomOp2(int,IntegerAddition)
  let R = QuotientGroupRel(AH,Op1,FR)
  let A = ProjFun2(AH,R,Op1)
  let M = ProjFun2(AH,R,Op2)
  have IsAring(AH//R,A,M) using Real_ZF_1_L1 group1.Ring_ZF_1_1_T1 by simp
  then show thesis using Slopes_def SlopeOp1_def SlopeOp2_def BoundedIntMaps_def SlopeEquivalenceRel_def RealNumbers_def RealAddition_def RealMultiplication_def by simp
qed

We can use theorems proven in group0 and group1 contexts applied to the group of real numbers.

lemma Real_ZF_1_L2: shows group0(RealNumbers,RealAddition)
  RealAddition (is commutative on) RealNumbers
  group1(RealNumbers,RealAddition)
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_L1 ring0_def ring0.Ring_ZF_1_L1 by auto

Lets 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
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
  \( r = \text{SlopeEquivalenceRel} \)
  \( e = 0 \)
  \( FR = \text{BoundedIntMaps} \)
  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 \( \text{SlopeEquivalenceRel} \) 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
  \( \text{Congruent2}(\text{SlopeEquivalenceRel},\text{SlopeOp1}) \)
  \( \text{Congruent2}(\text{SlopeEquivalenceRel},\text{SlopeOp2}) \)
  \( \text{SlopeEquivalenceRel} \subseteq \text{Slopes} \times \text{Slopes} \)
  \( \text{equiv}(\text{Slopes}, \text{SlopeEquivalenceRel}) \)
  \( \text{SlopeEquivalenceRel}(\text{id(int)}) = \text{TheNeutralElement}(\text{RealNumbers},\text{RealMultiplication}) \)
  \( \text{BoundedIntMaps} \subseteq \text{Slopes} \)
**proof** -
  let \( G = \text{int} \)
  let \( f = \text{IntegerAddition} \)
  let \( AH = \text{AlmostHoms}(\text{int},\text{IntegerAddition}) \)
  let \( Op1 = \text{AlHomOp1}(\text{int},\text{IntegerAddition}) \)
  let \( Op2 = \text{AlHomOp2}(\text{int},\text{IntegerAddition}) \)
  let \( FR = \text{FinRangeFunctions}(\text{int},\text{int}) \)
  let \( R = \text{QuotientGroupRel}(\text{int},\text{int}) \)
  have
    \( \text{Congruent2}(R,Op1) \)
    \( \text{Congruent2}(R,Op2) \)
    using Real_ZF_1_L1 group1.Group_ZF_3_4_L13A group1.Group_ZF_3_3_L5 by auto
then show
  \( \text{Congruent2}(\text{SlopeEquivalenceRel},\text{SlopeOp1}) \)
Congruent2(SlopeEquivalenceRel, SlopeOp2)
using SlopeEquivalenceRel_def SlopeOp1_def Slopes_def
BoundedIntMaps_def SlopeOp2_def by auto
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
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_L11 Real_ZF_1_L10
  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.

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

51 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 succesful conclusion. We put here those parts of the construc-
tion that can not be done in the general settings of abelian groups and require integers.

## 51.1 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(PositiveIntegers) ∩ PositiveIntegers /∈ 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 ≡ \{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 \( mR \). For a real number \( a \) notation \(|a|\) means the largest integer \( m \) such that the real version of it (that is, \( mR \)) is not greater than \( a \). For an integer \( m \) and a subset of reals \( S \) the expression \( \Gamma(S,m) \) is defined as \( \max\{\lfloor pR \cdot x \rfloor : x \in S\} \). This is plays a role in the proof of completeness of real numbers. We also reuse some notation defined in the int0 context, like \( Z_+ \) (the set of positive integers) and 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 AlEq (infix ~ 68)

defines AlEq_def[simp]: \( s \sim r \equiv (s,r) ∈ SlopeEquivalenceRel \)

fixes slope_add (infix + 70)

defines slope_add_def[simp]: \( s + r \equiv SlopeOp1(s,r) \)
fixes slope_comp (infix ◦ 71)
defines slope_comp_def[simp]: s ◦ r ≡ SlopeOp2(s,r)

fixes slopes (S)
defines slopes_def[simp]: S ≡ AlmostHoms(int,IntegerAddition)

fixes posslopes (S+)
defines posslopes_def[simp]: S+ ≡ PositiveSlopes

fixes slope_class (f)
defines slope_class_def[simp]: f ≡ SlopeEquivalenceRel{f}

fixes slope_neg (-_ 90 91)
defines slope_neg_def[simp]: -s ≡ GroupInv(int,IntegerAddition) O s

fixes lesseqr (infix ≤ 60)
defines lesseqr_def[simp]: a ≤ b ≡ ⟨a,b⟩ ∈ OrderOnReals

fixes sless (infix < 60)
defines sless_def[simp]: a < b ≡ a ≤ b ∧ a ≠ b

fixes positivereals (R+)
defines positivereals_def[simp]: R+ ≡ PositiveSet(R,RealAddition,OrderOnReals)

fixes intembed (_R 90 91)
defines intembed_def[simp]: mR ≡ [{⟨n,IntegerMultiplication(m,n)⟩. n ∈ int}]

fixes floor (⌊ _ ⌋)
defines floor_def[simp]: ⌊a⌋ ≡ Maximum(IntegerOrder,{m ∈ int. mR ≤ a})

fixes Γ
defines Γ_def[simp]: Γ(S,p) ≡ Maximum(IntegerOrder,{|pR|x|. x∈S})

fixes ia (infixl + 69)
defines ia_def[simp]: a+b ≡ IntegerAddition( a,b)

fixes iminus (- _ 72)
defines iminus_def[simp]: -a ≡ GroupInv(int,IntegerAddition)(a)

fixes isub (infixl - 69)
defines isub_def[simp]: a-b ≡ a+ (~ b)

fixes intpositives (Z+)
defines intpositives_def[simp]: Z+ ≡ PositiveSet(int,IntegerAddition,IntegerOrder)
fixes \( \text{zlesseq} \) (infix \( \leq \))
defines \( \text{lesseq_def}[\text{simp}] \): \( m \leq n \equiv (m,n) \in \text{IntegerOrder} \)

fixes \( \text{imult} \) (infixl \( \cdot \))
defines \( \text{imult_def}[\text{simp}] \): \( a \cdot b \equiv \text{IntegerMultiplication}(a,b) \)

fixes \( \text{izero} \) (\( 0_Z \))
defines \( \text{izero_def}[\text{simp}] \): \( 0_Z \equiv \text{TheNeutralElement}(\text{int},\text{IntegerAddition}) \)

fixes \( \text{ione} \) (\( 1_Z \))
defines \( \text{ione_def}[\text{simp}] \): \( 1_Z \equiv \text{TheNeutralElement}(\text{int},\text{IntegerMultiplication}) \)

fixes \( \text{itwo} \) (\( 2_Z \))
defines \( \text{itwo_def}[\text{simp}] \): \( 2_Z \equiv 1_Z + 1_Z \)

fixes \( \text{abs} \)
defines \( \text{abs_def}[\text{simp}] \):
\( \text{abs}(m) \equiv \text{AbsoluteValue(\text{int},\text{IntegerAddition},\text{IntegerOrder})(m)} \)

fixes \( \delta \)
defines \( \delta_{\text{def}}[\text{simp}] \): \( \delta(s,m,n) \equiv s(m+n)-s(m)-s(n) \)

51.2 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.

\text{lemma (in real1) RealZF_1_1_L2:} assumes \( A1: f \in S \ g \in S \)
\text{shows} \( \text{[f \circ g] = [g \circ f]} \)
\text{proof -}
\text{ from A1 have \( f \circ g \sim g \circ f \)}
\text{ using Slopes_def int1.Arthan_Th_9 SlopeOp1_def BoundedIntMaps_def}
\text{ SlopeEquivalenceRel_def SlopeOp2_def by simp}
\text{ then show thesis using RealZF_1_L11 equiv_class_eq by simp}
\text{qed}

Classes of slopes are real numbers.

\text{lemma (in real1) RealZF_1_1_L3:} assumes \( A1: f \in S \)
\text{shows} \( \text{[f] \in \mathbb{R}} \)
\text{proof -}

610
from A1 have \([f] \in \text{Slopes}/\text{SlopeEquivalenceRel}\) 
using \text{Slopes_def quotientI} by simp
then show \([f] \in R\) using \text{RealNumbers_def} by simp
qed

Each real number is a class of a slope.

lemma (in real1) Real_ZF_1_1_L3A: assumes A1: \(a \in \mathcal{S}/\text{SlopeEquivalenceRel}\)
shows \(\exists f \in S . \ a = [f]\)
proof - 
  from A1 have a \(\in \mathcal{S}/\text{SlopeEquivalenceRel}\)
  using \text{RealNumbers_def Slopes_def} by simp
  then show thesis using \text{quotient_def} by simp
qed

It is useful to have the definition of addition and multiplication in the 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 \([f] \cdot [g] = \text{ProjFun2}(S,r,\text{SlopeOp2})([f],[g])\)
    using \text{RealMultiplication_def Slopes_def} by simp
  also from A1 have \(... = [f \circ g]\)
    using \text{Real_ZF_1_L11 EquivClass_1_L10 Slopes_def} by simp
  finally show \([f] \cdot [g] = [f \circ g]\) by simp
  have \([f] + [g] = \text{ProjFun2}(S,r,\text{SlopeOp1})([f],[g])\)
    using \text{RealAddition_def Slopes_def} by simp
  also from A1 have \(... = [f+g]\)
    using \text{Real_ZF_1_L11 EquivClass_1_L10 Slopes_def} by simp
  finally show \([f] + [g] = [f+g]\) by simp
qed

The next lemma is essentially the same as Real_ZF_1_1_L12, but written in the notation defined in the 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 SlopeEquivalenceRel_def Real_ZF_1_1_L12} by simp

Subtracting real numbers corresponds 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: \(a \in \mathbb{R} \quad b \in \mathbb{R}\)
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 \(f \quad g\) where \(f \in S \quad g \in S \quad a = [f] \quad b = [g]\)
  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 \(\text{RealMultiplication} \text{ (is commutative on)} \text{ RealNumbers}\)
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 \([\text{id(int)}] = 1\)
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 \(f \in \text{BoundedIntMaps}\) 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: assumes $f \sim g$
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: shows $\text{id}(\text{int}) \in S$
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 $\neg(\text{id}(\text{int}) \sim f)$
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: shows $1 \neq 0$
proof -
  { assume $A1: 1=0$
    have $\exists f \in S\ . \ 0 = [f]$
      using Real_ZF_1_L4 Real_ZF_1_1_L3A by simp
    with $A1$ have $\exists f \in S\ . \ [\text{id}(\text{int})] = [f] \land [f] = 0$
      using real_one_cl_identity by auto
    then have False using Real_ZF_1_1_L5 Slopes_def
    Real_ZF_1_L10 Real_ZF_1_1_L7 id_on_int_is_slope
    by auto
  } then show $1 \neq 0$ by auto
qed

Negative of a real number is a real number. Property of groups.

lemma (in real1) Real_ZF_1_1_L8: assumes $a \in \mathbb{R}$ shows $(-a) \in \mathbb{R}$
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: assumes $a \in \mathbb{R}$, $b \in \mathbb{R}$, $c \in \mathbb{R}$
shows a · (b · c) = a · c · b
using assms real_mult_commutative Real_ZF_1_L3 ring0.Ring_ZF_2_L4
by simp

51.3 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
PositiveSlopes ⊆ Slopes
PositiveReals ⊆ RealNumbers

proof -
  have PositiveSlopes = {s ∈ Slopes. s(PositiveIntegers) ∩ PositiveIntegers ⊆ Fin(int)}
using PositiveSlopes_def by simp
  then show PositiveSlopes ⊆ Slopes by (rule subset_with_property)
  then have {SlopeEquivalenceRel{s}. s ∈ PositiveSlopes } ⊆ Slopes//SlopeEquivalenceRel
using EquivClass_1_L1A by simp
  then show PositiveReals ⊆ 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 ∈ PositiveReals ↔ (∃ f∈S+. a = [f])

proof
  assume a ∈ PositiveReals
  then have a ∈ {(s). s ∈ S+.} using PositiveReals_def
  by simp
  then show ∃ f∈S+. a = [f] by auto
next assume ∃ f∈S+. a = [f]
  then have a ∈ {(s). s ∈ S+.} by auto
  then show a ∈ 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∈S+. g∈S+
shows f+g ∈ S+

\( f \circ g \in S_+ \)

using assms \texttt{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) \texttt{Real_ZF_1_2_L5}: 
assumes \( f \in \text{BoundedIntMaps} \)
shows \( f / \notin S_+ \)

using assms \texttt{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) \texttt{Real_ZF_1_2_L6}: shows PositiveReals \{is closed under\} \texttt{RealAddition} PositiveReals \{is closed under\} \texttt{RealMultiplication} \( 0 \notin \text{PositiveReals} \)

proof -

\{ fix a b \\
  assume a \in \text{PositiveReals} and b \in \text{PositiveReals} \\
  then obtain f g where \\
  \text{I: } f \in S_+ g \in S_+ and \\
  \text{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 \texttt{Slopes_def} by auto \\
  with I II have \\
  a+b \in \text{PositiveReals} \land 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 PositiveReals \{is closed under\} \texttt{RealAddition} PositiveReals \{is closed under\} \texttt{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 \texttt{Slopes_def} Real_ZF_1_1_L10 Real_ZF_1_2_L5 by auto \\
  \} then show 0 \notin \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 \texttt{Int_ZF_2.thy}.

lemma (in real1) \texttt{Real_ZF_1_2_L7}:
assumes $A1: f \in S$ and $A2: [f] \neq 0$
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 $A1: f \in S$, $g \in S$
and $A2: (f \in S^+) \text{ Xor } (g \in S^+)$
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_L10 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 $A1: a \in R$ and $A2: a \neq 0$
shows $(a \in \text{PositiveReals}) \text{ Xor } ((-a) \in \text{PositiveReals})$
proof -
from $A1$ obtain $f$ where $I: f \in S$, $a = [f]$
using Real_ZF_1_1_L3A by auto
with $A2$ have $([f] \in \text{PositiveReals}) \text{ Xor } ([f] \in \text{PositiveReals})$
using Slopes_def BoundedIntMaps_def int1.Int_ZF_2_1_L12
Real_ZF_1_2_L7 Real_ZF_1_2_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 real_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($R$, $A$, $M$)
  $M$ (is commutative on) $R$
\( P \subseteq R \)

\( P \) is closed under \( A \)

\( \forall a \in R. \ a \neq z \implies (a \in P) \lor (\text{GroupInv}(R, A)(a) \in P) \)

\( P \) is closed under \( M \)

\( r = \text{OrderFromPosSet}(R, A, P) \)

\( \text{using real0.Real_ZF_1_L3 real_mult_commutative Real_ZF_1_2_L1 real1.Real_ZF_1_2_L6 real1.Real_ZF_1_2_L9 OrderOnReals_def} \)

\( \text{by auto} \)

then show \( \text{IsAnOrdRing}(R, A, M, r) \)

\( \text{by (rule ring0.ring_ord_by_positive_set)} \)

from I show \( r \) is total on \( R \)

\( \text{by (rule ring0.ring_ord_by_positive_set)} \)

from I show \( \text{PositiveSet}(R, A, r) = P \)

\( \text{by (rule ring0.ring_ord_by_positive_set)} \)

from I show \( \text{HasNoZeroDivs}(R, A, M) \)

\( \text{by (rule ring0.ring_ord_by_positive_set)} \)

qed

All theorems proven in the \textit{ring1} (about ordered rings), \textit{group3} (about ordered groups) and \textit{group1} (about groups) contexts are valid as applied to ordered real numbers with addition and (real) order.

\textbf{lemma RealZF_1_2_L10:} shows

\( \text{IsAnOrdGroup}(\text{RealNumbers}, \text{RealAddition}, \text{RealMultiplication}, \text{OrderOnReals}) \)

\( \text{group3}(\text{RealNumbers}, \text{RealAddition}, \text{OrderOnReals}) \)

\( \text{OrderOnReals} \) is total on \( \text{RealNumbers} \)

\textbf{proof -}

\( \text{using \textit{reals_are_ord_ring OrdRingZF_1_L2 by simp}} \)

then show

\( \text{IsAnOrdGroup}(\text{RealNumbers}, \text{RealAddition}, \text{OrderOnReals}) \)

\( \text{group3}(\text{RealNumbers}, \text{RealAddition}, \text{OrderOnReals}) \)

\( \text{OrderOnReals} \) is total on \( \text{RealNumbers} \)

\( \text{using \textit{ring1.OrdRingZF_1_L4 by auto}} \)

qed

If \( a = b \) or \( b - a \) is positive, then \( a \) is less or equal \( b \).

\textbf{lemma (in real1) RealZF_1_2_L11:} assumes \( A1: \ a \in R \quad b \in R \) and

\( A3: \ a = b \lor b-a \in \text{PositiveReals} \)

shows \( a \leq b \)

\( \text{using \textit{assms real\_are\_ord\_ring RealZF_1_2_L10 group3.OrderedGroupZF_1_L30 by simp}} \)

A sufficient condition for two classes to be in the real order.

\textbf{lemma (in real1) RealZF_1_2_L12:} assumes \( A1: \ f \in S \quad g \in S \) and

\( A2: \ f \sim g \lor (g + (-f)) \in S_+ \)

shows \( [f] \leq [g] \)
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: \(a \in \langle\) 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_antisym:
  assumes A1: \(a \leq b\) \(b \leq a\)
  shows \(a=b\)
proof -
  from A1 have \(# a,b \rangle \in \text{OrderOnReals} \langle b,a \rangle \in \text{OrderOnReals}\)
  using Real_ZF_1_2_L10 by auto
  then have \(# a,c \rangle \in \text{OrderOnReals}\)
  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 \(# a,b \rangle \in \text{OrderOnReals} \langle b,c \rangle \in \text{OrderOnReals}\)
  using Real_ZF_1_2_L10 by auto
  then have \(# a,c \rangle \in \text{OrderOnReals}\)
  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 A1: \(a \leq b\) and \(0 \leq c\)
  shows \(a \cdot c \leq b \cdot c\)
  \(\text{\(c \cdot a \leq c \cdot b\)}\)
  using assms Real_ZF_1_2_L10 ring1.OrdRing_ZF_1_L9
  by auto

618
A special case of Real_ZF_1_2_L14: we can multiply an inequality by a real number.

**lemma** (in real1) Real_ZF_1_2_L14A:

assumes A1: a ≤ b and A2: c ∈ R

shows c·a ≤ c·b

using assms Real_ZF_1_2_L10 ring1.OrdRing_ZF_1_L9A
by simp

In the real1 context notation $a ≤ b$ implies that $a$ and $b$ are real numbers.

**lemma** (in real1) Real_ZF_1_2_L15: assumes a ≤ b shows a, b ∈ R

by auto

$a ≤ b$ implies that $0 ≤ b - a$.

**lemma** (in real1) Real_ZF_1_2_L16: assumes a ≤ b

shows 0 ≤ b - a

using assms Real_ZF_1_2_L10 group3.OrderedGroup_ZF_1_L12A
by simp

A sum of nonnegative elements is nonnegative.

**lemma** (in real1) Real_ZF_1_2_L17: assumes 0 ≤ a 0 ≤ b

shows 0 ≤ a + b

using assms Real_ZF_1_2_L10 group3.OrderedGroup_ZF_1_L12
by simp

We can add sides of two inequalities

**lemma** (in real1) Real_ZF_1_2_L18: assumes a ≤ b  c ≤ d

shows a + c ≤ b + d

using assms Real_ZF_1_2_L10 group3.OrderedGroup_ZF_1_L5B
by simp

The order on real is reflexive.

**lemma** (in real1) real_ord_refl: assumes a ∈ R shows a ≤ a

using assms Real_ZF_1_2_L10 group3.OrderedGroup_ZF_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 ≤ b and c ∈ R

shows a + c ≤ b + c

using assms Real_ZF_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) Real_ZF_1_2_L19:

assumes a ∈ R  b ∈ R and c ≤ a + b

shows c - b ≤ a

using assms Real_ZF_1_2_L10 group3.OrderedGroup_ZF_1_L9C
by simp
What happens when one real number is not greater or equal than another?

**lemma (in real1) Real_ZF_1_2_L20:** assumes \( a \in R \) \( b \in R \) and \( \neg (a \leq b) \)

shows \( b < a \)

**proof** -

from assms have I:

- group3(R,RealAddition,OrderOnReals)
- OrderOnReals {is total on} \( R \)
- \( a \in R \) \( b \in R \) \( \neg (a,b) \in \text{OrderOnReals} \)
- 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 \( a \in R \) \( b \in R \) and \( c \leq a-b \)

shows \( c+b \leq a \)

using assms 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 \( \subseteq R \times R \)

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,OrderOnReals)

shows \( A \subseteq R \)

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: \( a \in R \) \( b \in R \) \( c \in R \)

shows

\[ \text{Maximum(OrderOnReals,\{a,b,c\})} \in \{a,b,c\} \]

\[ \text{Maximum(OrderOnReals,\{a,b,c\})} \in R \]

\( a \leq \text{Maximum(OrderOnReals,\{a,b,c\})} \)

\( b \leq \text{Maximum(OrderOnReals,\{a,b,c\})} \)

\( c \leq \text{Maximum(OrderOnReals,\{a,b,c\})} \)

**proof** -

have IsLinOrder(R,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}) ∈ 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)
⟨a,b⟩ ∈ OrderOnReals ⟨b,c⟩ ∈ OrderOnReals ∧ b ≠ c
using Real_ZF_1_2_L10 by auto
then have ⟨a,c⟩ ∈ 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 ∈ ′+ 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 ≤ and ≤ 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 ≤ b  b ≤ c

shows a ≤ c

**proof** -

from A1 have

⟨a,b⟩ ∈ IntegerOrder and ⟨b,c⟩ ∈ IntegerOrder

by auto

then have ⟨a,c⟩ ∈ IntegerOrder

by (rule Int_ZF_2_L5)

then show a ≤ 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: A ⊆ ℝ and ¬ HasAmaximum(OrderOnReals,A) and x ∈ A

shows ∃ y ∈ 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 ∈ ℝ  b ∈ ℝ and ¬ (a ≤ b)

shows b < a

**proof** -

from assms have

group3(R,RealAddition,OrderOnReals)

OrderOnReals {is total on} ℝ

a ∈ ℝ  b ∈ ℝ  ⟨a,b⟩ ∉ OrderOnReals

using Real_ZF_1_2_L10 by auto

then have ⟨b,a⟩ ∈ OrderOnReals  ∧ b ≠ 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 ¬ (b < a)

**proof** -

from assms have
51.4 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))

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
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: \( \exists h \in S. \ g \circ h \sim \text{id}(\text{int}) \)
by auto
from V obtain \( h \) where VII: \( h \in S \) and VIII: \( g \circ h \sim \text{id}(\text{int}) \)
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.

\textbf{Lemma (real1) Real\_ZF\_1\_3\_L1:}
\textbf{assumes} \( a \in \mathbb{R}_+ \)
\textbf{shows} \( a^{-1} \in \mathbb{R}_+ \) \( a^{-1} \in \mathbb{R} \)
\textbf{using} assms field_cntxts_ok field1.OrdField_ZF_1_1_L8 PositiveSet_def
\textbf{by auto}

A technical fact about multiplying strict inequality by the inverse of one of the sides.

\textbf{Lemma (in real1) Real\_ZF\_1\_3\_L2:}
\textbf{assumes} \( a \in \mathbb{R}_+ \) and \( a^{-1} < b \)
\textbf{shows} \( 1 < b \cdot a \)
\textbf{using} assms field_cntxts_ok field1.OrdField_ZF_2_L2
\textbf{by simp}

If \( a \) is smaller than \( b \), then \( (b-a)^{-1} \) is positive.

\textbf{Lemma (in real1) Real\_ZF\_1\_3\_L3:}
\textbf{assumes} \( a < b \)
\textbf{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:**

assumes \( A1: \ a \in \mathbb{R} \quad b \in \mathbb{R}_+ \quad \text{and} \quad A2: \ a \cdot b < c \)

shows \( a < c \cdot b^{-1} \)

using \( \text{assms field_cntxts_ok field1.OrdField_ZF_2_L6} \)
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:**

assumes \( A1: \ b \in \mathbb{R} \quad c \in \mathbb{R}_+ \quad \text{and} \quad A2: \ a < b \cdot c \)

shows \( a \cdot c^{-1} < b \)

using \( \text{assms field_cntxts_ok field1.OrdField_ZF_2_L6A} \)
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:**

assumes \( A1: \ b \in \mathbb{R} \quad c \in \mathbb{R}_+ \quad \text{and} \quad A2: \ a \leq b \cdot c \)

shows \( a \cdot c^{-1} \leq b \)

using \( \text{assms field_cntxts_ok field1.OrdField_ZF_2_L5A} \)
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:**

assumes \( A1: \ a \in \mathbb{R} \quad b \in \mathbb{R}_+ \quad \text{and} \quad A2: \ a \cdot b \leq c \)

shows \( a \leq c \cdot b^{-1} \)

using \( \text{assms field_cntxts_ok field1.OrdField_ZF_2_L5} \)
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:**

assumes \( a < b \quad \text{and} \quad (b-a)^{-1} < c \)

shows \( 1 + a \cdot c < b \cdot c \)

using \( \text{assms field_cntxts_ok field1.OrdField_ZF_2_L9} \)
by simp

We can multiply an inequality by the inverse of a positive number.

**Lemma (in real1) Real_ZF_1_3_L6:**

assumes \( a \leq b \quad \text{and} \quad c \in \mathbb{R}_+ \quad \text{shows} \quad a \cdot c^{-1} \leq b \cdot c^{-1} \)

using \( \text{assms field_cntxts_ok field1.OrdField_ZF_2_L3} \)

625
We can multiply a strict inequality by a positive number or its inverse.

**Lemma (in real1) Real_ZF_1_3_L7:**

assumes \( a < b \) and \( c \in \mathbb{R}_+ \)

shows \( a \cdot c < b \cdot c \)

\( c \cdot a < c \cdot b \)

\( a \cdot c^{-1} < b \cdot c^{-1} \)

using assms field_cntxs_ok field1.OrdField_ZF_2_L4

by auto

An identity with three real numbers, inverse and cancelling.

**Lemma (in real1) Real_ZF_1_3_L8:**

assumes \( a \in \mathbb{R} \)

\( b \in \mathbb{R} \) \( b \neq 0 \)

\( c \in \mathbb{R} \)

shows \( a \cdot b \cdot (c \cdot b^{-1}) = a \cdot c \)

using assms field_cntxs_ok field0.Field_ZF_2_L6

by simp

### 51.5 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:**

assumes \( m \in \mathbb{Z} \)

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:**

assumes \( m \in \mathbb{Z} \)

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:**

assumes \( m \in \mathbb{Z} \) \( k \in \mathbb{Z} \)

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:**

assumes \( A1: m \in \mathbb{Z} \) \( k \in \mathbb{Z} \)

shows \( m^R - k^R = ((m-k)^R) \)

proof -

\( \text{from A1 have } (-k) \in \mathbb{Z} \) using int0.Int_ZF_1_1_L4
by simp
with A1 have \((m-k)R = mR + (-k)R\)
    using Real_ZF_1_4_L1A by simp
with A1 show \(mR - kR = (m-k)R\)
    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 ∈ int  \(k \in int\)
  shows \(mR \cdot kR = (m \cdot k)R\)
proof -
  from A1 obtain f where I: \(f \in S\) and II: \(a = [f]\)
    using Real_ZF_1_1_L3A by auto
  then have \(\exists m \in int. \exists g \in S.\)
    \(\{\langle n,m \cdot n \rangle. n \in \text{int}\} \sim g \wedge (f \sim g \vee (g + (-f)) \in S_+\)\)
    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: \(m \in int\) and IV: \(g \in S\) and
    \(\{\langle n,m \cdot n \rangle. n \in \text{int}\} \sim g \wedge (f \sim g \vee (g + (-f)) \in S_+)\)
    by auto
  then have \(mR = [g]\) and f \sim g \vee (g + (-f)) \in S_+
    using Real_ZF_1_4_L5A by auto
  with I II IV have a \(\leq mR\) using Real_ZF_1_2_L12
    by simp
  with III show \(\exists m \in int. a \leq m^R\) by auto
qed

For any real numbers there is an integer whose real version is greater or equal.

lemma (in real1) Real_ZF_1_4_L2: assumes A1: a∈R
  shows \(\exists m \in \text{int. a} \leq m^R\)
proof -
  from A1 obtain f where I: \(f \in S\) and II: \(a = [f]\)
    using Real_ZF_1_1_L3A by auto
  let k = GroupInv(int,IntegerAddition)(m)
  from A1 I II have k \(\in \text{int}\) and k^R \(\leq a\)
    using Real_ZF_1_2_L13 Real_ZF_1_4_L1 int0.Int_ZF_1_1_L4
    by auto
  have \(a \geq k^R\) using Real_ZF_1_2_L14 Real_ZF_1_4_L1
    by auto
  with A1 show \(\exists m \in \text{int. a} \leq m^R\)
    using Real_ZF_1_4_L1A by simp
qed
Embeddings of two integers are equal only if the integers are equal.

**lemma (in real1) Real_ZF_1_4_L4:**

assumes A1: \( m \in \text{int} \) \( k \in \text{int} \) and A2: \( m^R = k^R \)

shows \( m=k \)

**proof** -

let \( r = \{\langle n, \text{IntegerMultiplication} \langle m, n \rangle \rangle . n \in \text{int}\} \)

let \( s = \{\langle n, \text{IntegerMultiplication} \langle k, n \rangle \rangle . n \in \text{int}\} \)

from A1 A2 have \( r \sim s \) using int1.Int_ZF_2_5_L1 AlmostHoms_def Real_ZF_1_1_L5

by simp

with A1 have \( m \in \text{int} \) \( k \in \text{int} \)

\( \langle r,s \rangle \in \text{QuotientGroupRel}(\text{AlmostHoms}(\text{int}, \text{IntegerAddition}), \text{AlHomOp1}(\text{int}, \text{IntegerAddition}), \text{FinRangeFunctions}(\text{int}, \text{int})) \)

using SlopeEquivalenceRel_def 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: \( m \leq k \)

shows \( m^R \leq k^R \)

**proof** -

let \( r = \{\langle n, m \cdot n \rangle . n \in \text{int}\} \)

let \( s = \{\langle n, k \cdot n \rangle . n \in \text{int}\} \)

from A1 have \( r \in S \) \( s \in S \)

using int0.Int_ZF_2_L1A int1.Int_ZF_2_5_L1 by auto

moreover from A1 have \( r \sim s \lor s + (-r) \in S_+ \)

using Slopes_def SlopeOp1_def BoundedIntMaps_def SlopeEquivalenceRel_def PositiveIntegers_def PositiveSlopes_def int1.Int_ZF_2_5_L4 by simp

ultimately show \( m^R \leq k^R \) 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: \( m \leq k \) \( m \neq k \)

shows \( m^R < k^R \)

**proof** -

from A1 have \( m^R \leq k^R \) using Real_ZF_1_4_L5 by simp

moreover from A1 have T: \( m \in \text{int} \) \( k \in \text{int} \)

using int0.Int_ZF_2_L1A by auto

with A1 have \( m^R \neq k^R \) using Real_ZF_1_4_L4
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: \( a \in \mathbb{R} \) shows \( \exists n \in \mathbb{Z}_+. \ a < n^R \)
proof -
  from A1 obtain \( m \) where I: \( m \in \mathbb{R} \) and II: \( a \leq m^R \)
    using Real_ZF_1_4_L2 by auto
  let \( n = \text{GreaterOf}(\mathbb{Z}, m) + 1 \) \( \mathbb{Z} \)
  from I have T: \( n \in \mathbb{Z}_+ \) and \( m \leq n \ m \neq n \)
    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: \( k \in \mathbb{Z} \) \( m \in \mathbb{Z} \) and A2: \( m \leq k^R \)
shows \( m \leq k \)
proof -
  { assume A3: \( \langle m,k \rangle \notin \mathbb{Z} \)
    with A1 have \( \langle k,m \rangle \in \mathbb{Z} \)
      by (rule int0.Int_ZF_2_L19)
    then have \( k^R \leq m^R \) 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 \( k \neq m \) by (rule int0.Int_ZF_2_L19)
    ultimately have False by simp
  } then show \( m \leq k \) by auto
qed

The floor function is well defined and has expected properties.

lemma (in real1) Real_ZF_1_4_L7: assumes A1: \( a \in \mathbb{R} \)
shows IsBoundedAbove({\( m \in \mathbb{Z}. \ m^R \leq a \}), \mathbb{R}^+)
proof -
  let \( A = \{ m \in \mathbb{Z}. \ m^R \leq a \} \)
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** -

let $A = \{m \in \text{int}. \, m^R \leq a\}$

from $A2$ have $\text{IsBoundedAbove}(A, \text{IntegerOrder})$ and $A \neq 0$

using $\text{Real_ZF}_1.2.L15$ $\text{Real_ZF}_1.4.L7$ by auto

then have $\forall x \in A. \, (x, \text{Maximum}(\text{IntegerOrder}, A)) \in \text{IntegerOrder}$

by $\text{rule int0.int_bounded_above_has_max}$

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 $0^R = 0$ \quad $1^R = 1$

using $\text{int1.Int_ZF_2.5.L7}$ $\text{BoundedIntMaps_def}$ $\text{real_one_cl_identity}$ $\text{real_zero_cl_bounded_map}$

by auto

Integer two embeds as the real two.

**lemma (in real1) int_two_is_real_two:**

shows $2^R = 2$

**proof** -

have $2^R = 1^R + 1^R$

using $\text{int0.int_zero_one_are_int}$ $\text{Real_ZF}_1.4.L1A$

by simp

also have $\ldots = 2$ using $\text{int_0_1_are_real_zero_one}$

by simp

630
finally show $2_R^R = 2$ by simp

qed

A positive integer embeds as a positive (hence nonnegative) real.

lemma (in real1) int_pos_is_real_pos: assumes A1: $p \in \mathbb{Z}_+$ shows $p^R \in \mathbb{R} \quad 0 \leq p^R \quad p^R \in \mathbb{R}_+$

proof -
  from A1 have I: $p \in \text{int} \quad 0_\mathbb{Z} \leq p \quad 0_\mathbb{Z} \neq p$
    using PositiveSet_def by auto
  then have $p^R \in \mathbb{R} \quad 0_\mathbb{Z}^R \leq p^R$
    using real_int_is_real Real_ZF_1_4_L5 by auto
  then show $p^R \in \mathbb{R}_+$
    using int_0_1_are_real_zero_one by auto
moreover have $0 \neq p^R$
  proof -
    { assume $0 = p^R$
      with I have False using int_0_1_are_real_zero_one
      int0.int_zero_one_are_int Real_ZF_1_4_L4 by auto
    } then show $0 \neq p^R$ by auto
  qed
ultimately show $p^R \in \mathbb{R}_+$ using PositiveSet_def
  by simp

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].

lemma (in real1) Arthan_Lemma14ii: assumes A1: $x \in \mathbb{R} \quad y \in \mathbb{R}_+$
  shows $\exists M \in \mathbb{Z}_+. \ x < M^R y$

proof -
  from A1 have I: $\exists C \in \mathbb{Z}_+. \ x < C^R$ and $\exists D \in \mathbb{Z}_+. \ y^{-1} < D^R$
    using Real_ZF_1_3_L1 Arthan_Lemma14i by auto
  then obtain $C \ D$ where
    I: $C \in \mathbb{Z}_+$ and II: $x < C^R$ and
    III: $D \in \mathbb{Z}_+$ and IV: $y^{-1} < D^R$
    by auto
  let $M = C \ D$
from I III have
  $T: \ M \in \mathbb{Z}_+ \quad C^R \in \mathbb{R} \quad D^R \in \mathbb{R}$
    using int0.pos_int_closed_mul_unfold PositiveSet_def real_int_is_real
    by auto
with A1 I III have $C^R \cdot (D^R \cdot y) = M^R y$
  using PositiveSet_def Real_ZF_1_6A Real_ZF_1_4_L1C by simp
moreover from A1 I II IV have
ultimately have $x < M$
by auto

with $T$ show thesis by auto
qed

Taking the floor function preserves the order.

**Lemma (in real1) Real_ZF_1_4_L9:** assumes $A1: a \leq b$
shows $\lfloor a \rfloor \leq \lfloor b \rfloor$
proof -
  from $A1$ have $T: a \in \mathbb{R}$ using Real_ZF_1_2_L15
  by simp
  with $A1$ have $\lfloor a \rfloor \leq a$ and $a \leq b$
  using Real_ZF_1_4_L7 by auto
  then have $\lfloor a \rfloor \leq b$ by (rule real_ord_transitive)
  moreover from $T$ have $\lfloor a \rfloor \in \mathbb{Z}$
  using Real_ZF_1_4_L7 by simp
  ultimately show $\lfloor a \rfloor \leq \lfloor b \rfloor$ using Real_ZF_1_4_L8
  by simp
qed

If $S$ is bounded above and $p$ is a positive integer, then $\Gamma(S,p)$ is well defined.

**Lemma (in real1) Real_ZF_1_4_L10:**
assumes $A1: \text{IsBoundedAbove}(S,\text{OrderOnReals}) \ S \neq \emptyset$ and $A2: p \in \mathbb{Z}_+$
shows $\Gamma(S,p) \in \{\lfloor p^R \cdot x \rfloor | \ x \in S\}$
$\Gamma(S,p) \in \mathbb{Z}$
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 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 int_pos_is_real_pos
      by simp
    ultimately have $p^R \cdot x \leq p^R \cdot X$ using Real_ZF_1_2_L14
      by simp
    with $II$ have $m \leq \lfloor p^R \cdot X \rfloor$ using Real_ZF_1_4_L9
      by simp
  }
  then have $\forall m \in A. \ \langle m, \lfloor p^R \cdot X \rfloor \rangle \in \text{IntegerOrder}$
  by auto
  then show $II: \text{IsBoundedAbove}(A,\text{IntegerOrder})$
    by (rule Order_ZF_3_L10)
moreover from \( A_1 \) 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 \( \Gamma(S,p) \in \{p^R \cdot x . x \in S \} \) 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 \( \Gamma(S,p) \).

\text{lemma (in real1) RealZF_1_4_L11:}
assumes \( A_1: \text{IsBoundedAbove}(S,\text{OrderOnReals}) \) and \( A_2: x \in \text{S} \) and \( A_3: p \in \mathbb{Z}_+ \)
shows \( |p^R \cdot x| \leq \Gamma(S,p) \)
proof -
let \( A = \{[p^R \cdot x] . x \in S \} \)
from \( A_2 \) have \( S \neq \emptyset \) by auto
with \( A_1 \) \( A_3 \) have \( \text{IsBoundedAbove}(A,\text{IntegerOrder}) \) \( A \neq 0 \)
using RealZF_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 \( A_2 \) show \( |p^R \cdot x| \leq \Gamma(S,p) \) by simp
qed

The candidate for supremum is an integer mapping with values given by \( \Gamma \).

\text{lemma (in real1) RealZF_1_4_L12:}
assumes \( A_1: \text{IsBoundedAbove}(S,\text{OrderOnReals}) \) \( S \neq 0 \) and \( A_2: g = \{[p,\Gamma(S,p)). p \in \mathbb{Z}_+ \} \)
shows \( g : \mathbb{Z}_+ \rightarrow \text{int} \)
\( \forall n \in \mathbb{Z}_+. g(n) = \Gamma(S,n) \)
proof -
from \( A_1 \) have \( \forall n \in \mathbb{Z}_+. \Gamma(S,n) \in \text{int} \) using RealZF_1_4_L10
by simp
with \( A_2 \) show I: \( g : \mathbb{Z}_+ \rightarrow \text{int} \) using ZF_fun_from_total by simp
\{ fix n assume n \in \mathbb{Z}_+
with \( A_2 \) 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.

\text{lemma (in real1) RealZF_1_4_L14: assumes A1: \( m \in \text{int} \)
shows \( |m^R| = m \)
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 \( A_1 \) have \( m \in A \)
Floor of (real) zero is (integer) zero.

lemma (in real1) floor_01_is_zero_one: shows \( \lfloor 0 \rfloor = 0 \) and \( \lfloor 1 \rfloor = 1 \)
proof -
  have \( \lfloor (0Z)^R \rfloor = 0Z \) and \( \lfloor (1Z)^R \rfloor = 1Z \)
  using int0.int_zero_one_are_int Real_ZF_1_4_L14
  by auto
  then show \( \lfloor 0 \rfloor = 0Z \) and \( \lfloor 1 \rfloor = 1Z \)
  using int_0_1_are_real_zero_one
  by auto
qed

Floor of (real) two is (integer) two.

lemma (in real1) floor_2_is_two: shows \( \lfloor 2 \rfloor = 2 \)
proof -
  have \( \lfloor (2Z)^R \rfloor = 2Z \)
  using int0.int_two_three_are_int Real_ZF_1_4_L14
  by simp
  then show \( \lfloor 2 \rfloor = 2Z \)
  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: \( m \in \text{int} \) \( k \in \text{int} \)
shows \( \lfloor m^R \cdot k^R \rfloor = m \cdot k \)
proof -
  from A1 have T: \( m \cdot k \in \text{int} \)
  using int0.Int_ZF_1_1_L5
  by auto
  from A1 have \( \lfloor (m^R \cdot k^R) \rfloor = (m \cdot k)^R \)
  using Real_ZF_1_4_L1C
  by simp
  with T show \( \lfloor m^R \cdot k^R \rfloor = m \cdot k \)
  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: \( x \in \mathbb{R} \) and A2: \( p \in \text{int} \)
shows \( \lfloor x + p^R \rfloor = \lfloor x \rfloor + p \)
proof -
  let \( A = \{ n \in \text{int}. n^R \leq x + p^R \} \)
have \text{antisym}(\text{IntegerOrder}) \text{ using } \text{int0.IntZF_2_L4} \text{ by simp}

moreover have $|x| + p \in A$

proof -
from A1 A2 have $|x|^R \leq x$ and $p^R \in \mathbb{R}$
  using RealZF_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 RealZF_1_4_L7 RealZF_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 RealZF_1_4_L7 int0.IntZF_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 RealZF_1_2_L19 by simp
with A2 I have $(n-p)^R \leq |x|$
  using RealZF_1_4_L1B RealZF_1_4_L9 by simp
moreover from A2 I have $n-p \in \text{int}$
  using int0.IntZF_1_1_L5 by simp
then have $|(n-p)^R| = n-p$
  using RealZF_1_4_L14 by simp
ultimately have $n-p \leq |x|$\n  by simp
with A2 I show $n \leq |x| + p$
  using int0.IntZF_2_L9C by simp
ultimately show $|x + p|^R = |x| + p$
  using OrderZF_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) RealZF_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 RealZF_1_4_L1 by simp
with A1 A2 show $|x - p^R| = |x| - p$

635
The floor of sum of embeddings is the sum of the integers.

**Lemma (in real1) Real_ZF_1_4_L17:** assumes \( m \in \text{int} \) \( n \in \text{int} \)
shows \( \lfloor (m^R) + n^R \rfloor = m + n \)
using assms real_int_is_real Real_ZF_1_4_L15 Real_ZF_1_4_L14 by simp

A lemma about adding one to floor.

**Lemma (in real1) Real_ZF_1_4_L17A:** assumes \( A1: a \in \text{'} \)
shows \( 1 + \lfloor a \rfloor^R = (1^R + \lfloor a \rfloor)^R \)
proof -
  have \( 1 + \lfloor a \rfloor^R = 1^R + \lfloor a \rfloor^R \)
    using int_0_1_are_real_zero_one by simp
  with \( A1 \) show \( 1 + \lfloor a \rfloor^R = (1^R + \lfloor a \rfloor)^R \)
    using int0.int_zero_one_are_int Real_ZF_1_4_L7 Real_ZF_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) Real_ZF_1_4_L17B:** assumes \( A1: a \in \text{'} \)
shows \( a - \lfloor a \rfloor^R < 1 \)
\( a < (1^R + \lfloor a \rfloor)^R \)
proof -
  from \( A1 \) have \( T1: \lfloor a \rfloor^R \in \text{int} \) \( \lfloor a \rfloor^R \in \text{'} \)
    using Real_ZF_1_4_L7 real_int_is_real Real_ZF_1_4_L6 Real_ZF_1_4_L4 by auto
  \{ assume \( 1 \leq a - \lfloor a \rfloor^R \)
    with \( A1 \) \( T1 \) have \( 1^R + \lfloor a \rfloor^R \leq \lfloor a \rfloor \)
      using Real_ZF_1_2_L21 Real_ZF_1_4_L9 int_0_1_are_real_zero_one by simp
    with \( T1 \) have False
      using int0.int_zero_one_are_int Real_ZF_1_4_L17 Real_ZF_1_4_L1A by simp
  } then have \( I: \neg (1 \leq a - \lfloor a \rfloor^R) \)
    by auto
  with \( T2 \) show \( II: a - \lfloor a \rfloor^R < 1 \)
    by (rule Real_ZF_1_2_L20)
  with \( A1 \) \( T1 \) \( I \) \( II \) have \( a < 1 + \lfloor a \rfloor^R \)
    using Real_ZF_1_2_L26 by simp
  with \( A1 \) show \( a < (1^R + \lfloor a \rfloor)^R \)
    using Real_ZF_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 ∃M∈int. ∃N∈Z+. x·N < M ≤ y·N proof -
  from A1 have (y-x)^{-1} ∈ R+ using Real_ZF_1_3_L3
    by simp
  then have ∃N∈Z+. (y-x)^{-1} < N
    using Arthan_Lemma14i PositiveSet_def by simp
  then obtain N where I: N∈š+ and II: (y-x)^{-1} < N
    by auto
  let M = 1 + ⌊x·N⌋ from A1 I have
    III: x ∈ ‹N〉 and II: (y-x)^{-1} < N
    using Real_ZF_1_2_L15 PositiveSet_def real_int_is_real
    Real_ZF_1_L6 int_pos_is_real_pos by auto
  then have M ∈ int using
    int0.int_zero_one_are_int Real_ZF_1_4_L7 int0.Int_ZF_1_1_L5
    by simp
  from T1 have III: x·N < M
    using Real_ZF_1_4_L17B by simp
  from T1 have (1 + ⌊x·N⌋R) ≤ (1 + x·N)
    using Real_ZF_1_4_L7 Real_ZF_1_4_L17A by simp
  moreover from A1 II have (1 + x·N) < y·N
    using Real_ZF_1_3_L5 by simp
  ultimately have M < y·N
    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∈R y∈R
  shows abs(⌊x+y⌋ - ⌊x⌋ - ⌊y⌋) ≤ 2
proof -
  from A1 have T:
    |x| ≤ |x| + (|y|)
    x+y - (|x|) ∈ R
    using Real_ZF_1_4_L17B Real_ZF_1_4_L17B by auto
  from A1 have
    0 ≤ x - (|x|) + (y - (|y|))
    x - (|x|) + (y - (|y|)) ≤ 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
\[ x - (\lfloor x \rfloor) + (y - (\lfloor y \rfloor)) = x+y - (\lfloor x \rfloor) - (\lfloor y \rfloor) \]

using **Real_ZF_1_L7A** by simp

ultimately have

\[ 0 \leq x+y - (\lfloor x \rfloor) - (\lfloor y \rfloor) \]

\[ x+y - (\lfloor x \rfloor) - (\lfloor y \rfloor) \leq 2 \]

by auto

then have

\[ [0] \leq \lfloor x+y - (\lfloor x \rfloor) - (\lfloor y \rfloor) \rfloor \]

\[ \lfloor x+y - (\lfloor x \rfloor) - (\lfloor y \rfloor) \rfloor \leq [2] \]

using **Real_ZF_1_4_L9** by auto

then have

\[ 0 \leq \lfloor x+y - (\lfloor x \rfloor) - (\lfloor y \rfloor) \rfloor \]

\[ \lfloor x+y - (\lfloor x \rfloor) - (\lfloor y \rfloor) \rfloor \leq 2 \]

by auto

then show \( \abs{(\lfloor x+y - (\lfloor x \rfloor) - (\lfloor y \rfloor) \rfloor)} \leq 2 \)

using **floor_01_is_zero_one floor_2_is_two** by auto

qed

Suppose \( S \neq \emptyset \) is bounded above and \( \Gamma(S, m) = \lfloor m^R \cdot x \rfloor \) for some positive integer \( m \) and \( x \in S \). Then if \( y \in S \), \( x \leq y \) we also have \( \Gamma(S, m) = \lfloor m^R \cdot y \rfloor \).

**lemma (in real1) Real_ZF_1_4_L20:**

assumes \( A1: \text{IsBoundedAbove}(S,\text{OrderOnReals}) \), \( S \neq 0 \) and

\( A2: n \in \mathbb{Z}^+ \), \( x \in S \) and

\( A3: \Gamma(S, n) = \lfloor n^R \cdot x \rfloor \) and

\( A4: y \in S \), \( x \leq y \)

shows \( \Gamma(S, n) = \lfloor n^R \cdot y \rfloor \)

**proof** -

from \( A2\ A4 \) have \( \lfloor n^R \cdot x \rfloor \leq \lfloor (n^R) \cdot y \rfloor \)

using **int_pos_is_real_pos Real_ZF_1_2_L14 Real_ZF_1_4_L9** by simp

with \( A3 \) have \( (\Gamma(S, n), \lfloor (n^R) \cdot y \rfloor) \in \text{IntegerOrder} \)

by simp

moreover from \( A1\ A2\ A4\) have \( (\lfloor n^R \cdot y \rfloor, \Gamma(S, n)) \in \text{IntegerOrder} \)

using **Real_ZF_1_4_L11** by simp

ultimately show \( \Gamma(S, n) = \lfloor n^R \cdot y \rfloor \)

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: IsBoundedAbove(S,OrderOnReals) \( S \neq 0 \) and
A2: \( m \in \mathbb{Z}_+ \), \( n \in \mathbb{Z}_+ \)
shows \( \left| \Gamma(S,m+n) - \Gamma(S,m) - \Gamma(S,n) \right| \leq 2Z \)

proof -
from A2 have T: \( m+n \in \mathbb{Z}_+ \) using int0.pos_int_closed_add_unfolded
by simp
with A1 A2 have
\( \Gamma(S,m) \in \{ [m^R \cdot x]. x \in S \} \) and
\( \Gamma(S,n) \in \{ [n^R \cdot x]. x \in S \} \) and
\( \Gamma(S,m+n) \in \{ [(m+n)^R \cdot x]. x \in S \} \)
using Real_ZF_1_4_L10 by auto
then obtain a b c where I: \( a \in S \), \( b \in S \), \( c \in S \)
and II:
\( \Gamma(S,m) = [m^R \cdot a] \)
\( \Gamma(S,n) = [n^R \cdot b] \)
\( \Gamma(S,m+n) = [(m+n)^R \cdot c] \)
by auto
let d = Maximum(OrderOnReals,\{a,b,c\})
from A1 I have acR bR cR cR
using Real_ZF_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 Real_ZF_1_2_L24 by auto
with I have V: \( d \in S \) by auto
from A1 T I II IV V have \( \Gamma(S,m+n) = [(m+n)^R \cdot d] \)
using Real_ZF_1_4_L20 by blast
also from A2 have \( \ldots = [(m^R \cdot d + n^R \cdot d)] \)
using Real_ZF_1_4_L1A PositiveSet_def by simp
also from A2 IV have \( \ldots = [(m^R \cdot d + n^R \cdot d)] \)
using PositiveSet_def real_int_is_real Real_ZF_1_L7 by simp
finally have \( \Gamma(S,m+n) = [(m^R \cdot d + n^R \cdot d)] \)
by simp
moreover from A1 A2 I II IV V have \( \Gamma(S,m) = [m^R \cdot d] \)
using Real_ZF_1_4_L20 by blast
moreover from A1 A2 I II IV V have \( \Gamma(S,n) = [n^R \cdot d] \)
using Real_ZF_1_4_L20 by blast
moreover from A1 T I II IV V have \( \Gamma(S,m+n) = [(m+n)^R \cdot d] \)
using Real_ZF_1_4_L20 by blast
ultimately have \( \left| \Gamma(S,m+n) - \Gamma(S,m) - \Gamma(S,n) \right| \leq 2Z \)
by simp
with A2 IV show
\( \left| \Gamma(S,m+n) - \Gamma(S,m) - \Gamma(S,n) \right| \leq 2Z \)
using PositiveSet_def real_int_is_real Real_ZF_1_L6

639
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 $\text{OddExtension}$ 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:
- $f: \mathbb{Z}_+ \to \mathbb{Z}$. $\forall a \in \mathbb{Z}_+. \forall b \in \mathbb{Z}_+. \ abs(\delta(f,a,b)) \leq L$

Shows: $\text{OddExtension}(\mathbb{Z}, \text{IntegerAddition}, \text{IntegerOrder}, f) \in S$

Using: a1 int1.Int_ZF_2_1_L24 by auto

The candidate for (a representant of) the supremum of a nonempty bounded above set is a slope.

**Lemma (in real1) Real_ZF_1_4_L22:**

Assumes:
- $\text{IsBoundedAbove}(S, \text{OrderOnReals}) S \neq 0$ and
- $g = \{\langle p, \Gamma(S, p) \rangle. \ p \in \mathbb{Z}_+\}$

Shows: $\text{OddExtension}(\mathbb{Z}, \text{IntegerAddition}, \text{IntegerOrder}, g) \in S$

Proof -
- from A1 A2 have $g: \mathbb{Z}_+ \to \mathbb{Z}$ by (rule Real_ZF_1_4_L12)
- moreover have $\forall m \in \mathbb{Z}_+. \forall n \in \mathbb{Z}_+. \ abs(\delta(g,m,n)) \leq 2Z$

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
- $abs(\Gamma(S, m+n) - \Gamma(S, m) - \Gamma(S, n)) \leq 2Z$
- by (rule Real_ZF_1_4_L21)
- ultimately have $abs(\delta(g,m,n)) \leq 2Z$
- by simp
- } then show $\forall m \in \mathbb{Z}_+. \forall n \in \mathbb{Z}_+. \ abs(\delta(g,m,n)) \leq 2Z$
- 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:**

640
assumes $A_1: f \in S$ and $A_2: N \in \text{int} \ M \in \text{int}$ and
$A_3: \forall n \in \mathbb{Z}^+. \ Mn \leq f(Nn)$
shows $M^R \leq [f] \cdot (N^R)$

proof -

let $M_S = \{(n, M \cdot n) . n \in \text{int}\}$
let $N_S = \{(n, N \cdot n) . n \in \text{int}\}$

from $A_1 A_2$ 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 $A_1 A_2 A_3$ 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 $A_1 T$ show $M^R \leq [f] \cdot (N^R)$ 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 $A_1: \text{IsBoundedAbove}(S, \text{OrderOnReals})$ and
$A_2: x<y$ $y \in S$ and
$A_4: N \in \mathbb{Z}^+$ $M \in \text{int}$ and
$A_5: M^R < y \cdot N^R$ and $A_6: p \in \mathbb{Z}^+$
shows \( p \cdot M \leq \Gamma(S, p \cdot N) \)

proof - 
from A2 A4 A6 have T1:
\[ N^R \in \mathbb{R}^+ \quad y \in \mathbb{R} \quad p^R \in \mathbb{R}^+ \]
\[ p^N \in \mathbb{Z}^+ \quad (p \cdot N)^R \in \mathbb{R}^+ \]
using int_pos_is_real_pos Real_ZF_1_2_L15
int0.pos_int_closed_mul_unfold by auto
with A4 A6 have T2:
\[ p \in \mathbb{R} \quad N^R \in \mathbb{R} \quad N^R \neq 0 \quad M^R \in \mathbb{R} \]
using real_int_is_real PositiveSet_def by auto
from T1 A5 have I:
\[ \lfloor (p \cdot N)^R \cdot (M^R \cdot (N^R)^{-1}) \rfloor \leq \lfloor (p \cdot N)^R \cdot y \rfloor \]
using Real_ZF_1_3_L4A Real_ZF_1_3_L7 Real_ZF_1_4_L9 by simp
moreover from A1 A2 T1 have I:
\[ \lfloor (p \cdot N)^R \cdot (M^R \cdot (N^R)^{-1}) \rfloor \leq \Gamma(S, p \cdot N) \]
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)

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) \( S \neq 0 \) and A2: \( p \in \mathbb{Z}^+ \)
and A3: \( h = \text{OddExtension}(\mathbb{I}, \mathbb{A}, \mathbb{O}, \{(p, \Gamma(S, p)) \text{. } p \in \mathbb{Z}^+\}) \)
shows \( x \leq [h] \)
proof -
from A1 A2 A3 have I:
\[ g = \{(p, \Gamma(S, p)) \text{. } p \in \mathbb{Z}^+\} \]
from A1 have I:
\[ g : \mathbb{Z}^+ \rightarrow \mathbb{I} \]
using Real_ZF_1_4_L12 by blast
with A2 A3 show h(p) = \( \Gamma(S, p) \)
using Real_ZF_1_4_L14A 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:
\[ h = \text{OddExtension}(\mathbb{I}, \mathbb{A}, \mathbb{O}, \{(p, \Gamma(S, p)) \text{. } p \in \mathbb{Z}^+\}) \]
shows \( h(p) = \Gamma(S, p) \)
proof -
let g = \( \{(p, \Gamma(S, p)) \text{. } p \in \mathbb{Z}^+\} \)
from A1 have I:
\[ g : \mathbb{Z}^+ \rightarrow \mathbb{I} \]
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
\text{The essential condition to claim that the candidate for supremum of } S \text{ 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: $N \in \mathbb{Z}_+$ $M \in \mathbb{Z}$ and
  A5: $y \cdot N < M$ and A6: $p \in \mathbb{Z}_+$
  shows $\lfloor (N \cdot p) \cdot x \rfloor \leq M \cdot p$
proof -
  from A2 A4 A6 have T:
    $p \cdot N \in \mathbb{Z}_+$ $p \in \mathbb{Z}$ $N \in \mathbb{Z}$
    $p^R \in \mathbb{R}_+$ $p^R \in \mathbb{R}$ $N^R \in \mathbb{R}$ $x \in \mathbb{R}$ $y \in \mathbb{R}$
    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$^R$.$(y \cdot N^R) < (p \cdot M)^R$
    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 $p \cdot M \in \mathbb{Z}$ using int0.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
  $h =$ OddExtension($\mathbb{Z}$,$\mathbb{Z}_+$,$\mathbb{Z}_+$,$\{p,\Gamma(S,p)\}$. $p \in \mathbb{Z}_+$)
  and $p \in \mathbb{Z}_+$
show \( \exists x \in S. \ h(p) = \lfloor pR \cdot x \rfloor \)

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:
  \( h = \text{OddExtension}(\text{int},\text{IntegerAddition},\text{IntegerOrder},\{\langle p,\Gamma(S,p)\rangle. \ p \in \mathbb{Z}_+\}) \)
  shows \( [h] \leq y \)

proof -
  from A1 obtain a where a:S by auto
  with A1 A2 A3 have T: \( y \in S \) \( [h] \in \mathbb{R} \)
    by auto
  \{ assume \( \neg([h] \leq y) \)
    with T have y < [h] using Real_ZF_1_2_L28
      by blast
    then have \( \exists M \in \text{int}. \ \exists N \in \mathbb{Z}_+. \ yN^R < M^R \land M^R < [h]N^R \)
      using Arthan_Lemma14iii by simp
    then obtain M N where I: M \( \in \text{int} \) \( N \in \mathbb{Z}_+ \) and
      II: \( yN^R < M^R \) \( M^R < [h]N^R \)
      by auto
    from I have III: \( N^R \in \mathbb{R}_+ \) 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 \( \exists x \in S. \ h(N \cdot p) = \lfloor (N \cdot p)^R \cdot x \rfloor \)
          using int0.pos_int_closed_mul_unfold Real_ZF_1_4_L27
        by simp
        with A1 A2 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 I have \( [h] \leq M \cdot (N^R)^{-1} \)
      using Real_ZF_1_3_L4C by simp
    moreover from T II III have \( M \cdot (N^R)^{-1} < [h] \)
      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) \ not S = 0
shows HasAminimum(OrderOnReals,\bigcap_{a \in S} OrderOnReals\{a\})
proof
{ assume HasAmaximum(OrderOnReals,S) 
  with A1 have HasAminimum(OrderOnReals,\bigcap_{a \in S} OrderOnReals\{a\}) 
  using Real_ZF_1_2_L10 IsAnOrdGroup_def IsPartOrder_def Order_ZF_S_L6 by simp 
} moreover 
{ assume A2: \neg HasAmaximum(OrderOnReals,S) 
  let h = OddExtension(int,IntegerAddition,IntegerOrder,\langle p,\Gamma(S,p) \rangle. p \in \mathbb{Z}^+ ) 
  let r = OrderOnReals 
  from A1 have antisym(OrderOnReals) S \neq 0 
  using Real_ZF_1_2_L10 IsAnOrdGroup_def IsPartOrder_def by auto 
  moreover from A1 A2 have \forall x \in S. \langle x,\{h\} \rangle \in r 
  using Real_ZF_1_4_L25 by simp 
  moreover from A1 have \forall y. (\forall x \in S. \langle x,y \rangle \in r) \implies \langle \{h\},y \rangle \in r 
  using Real_ZF_1_4_L28 by simp 
  ultimately have HasAminimum(OrderOnReals,\bigcap_{a \in S} OrderOnReals\{a\}) 
  by (rule Order_ZF_S_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

52  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.

52.1  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\}) \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 \text{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 \text{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 \text{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 \text{Boundary}(A,T) \equiv \text{Closure}(A,T) \cap \text{Closure}(\bigcup T - A,T)
```

647
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
IsCompact (infixl \{is compact in\} 90) where
$K$ {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 Pow_is_top: shows $\text{Pow}(X)$ {is a topology}
proof -
have $\forall A \in \text{Pow}(\text{Pow}(X)). \ \bigcup A \in \text{Pow}(X)$ 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 IsATopology_def by auto
qed

Empty set is open.

lemma empty_open: assumes $T$ {is a topology} shows $0 \in T$
proof -
have $0 \in \text{Pow}(T)$ by simp
with assms have $\bigcup 0 \in T$ using IsATopology_def by blast
thus $0 \in T$ by simp
qed

The carrier is open.

lemma carr_open: assumes $T$ {is a topology} shows $(\bigcup T) \in T$
using assms IsATopology_def by auto

Union of a collection of open sets is open.

lemma union_open: assumes $T$ {is a topology} and $\forall A \in A. \ A \in T$
sshows $(\bigcup A) \in T$ using assms IsATopology_def by auto

Union of a indexed family of open sets is open.

lemma union_indexed_open: assumes $A1$: $T$ {is a topology} and $A2$: $\forall i \in I. \ P(i) \in T$
sshows $(\bigcup i \in I. \ P(i)) \in T$ using assms union_open by simp

The intersection of any nonempty collection of topologies on a set $X$ is a topology.

lemma Inter_tops_is_top: assumes $A1$: $\mathcal{M} \neq 0$ and $A2$: $\forall T \in \mathcal{M}. \ T$ {is a topology}
sshows $(\bigcap \mathcal{M})$ {is a topology}
proof -
{ fix $A$ assume $A \in \text{Pow}(\bigcap \mathcal{M})$}
with \( A_1 \) have \( \forall T \in \mathcal{M}. A \in \text{Pow}(T) \) by \texttt{auto}  
with \( A_1 \) \( A_2 \) have \( \bigcup A \in \bigcap \mathcal{M} \) using \texttt{IsATopology_def} by \texttt{auto}  
\{ 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 \texttt{auto}  
with \( A_1 \) \( A_2 \) have \( \forall T \in \mathcal{M}. U \cap V \in T \) using \texttt{IsATopology_def} by \texttt{simp}  
\} then have \( \forall U \in \bigcap \mathcal{M}. V \in \bigcap \mathcal{M} \)  
by \texttt{auto}  
ultimately show \( (\bigcap \mathcal{M}) \) \{is a topology\}  
using \texttt{IsATopology_def} by \texttt{simp}  
\qed

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 \texttt{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 \( \partial A \).

\texttt{locale \texttt{topology0} =}  
\texttt{fixes \( T \) \assumes \texttt{topSpaceAssum}: \( T \) \{is a topology\}}  
\texttt{fixes \texttt{int}} \texttt{defines \texttt{int_def} \{simp\}: \( \text{int}(A) \equiv \text{Interior}(A,T) \)}  
\texttt{fixes \texttt{cl}} \texttt{defines \texttt{cl_def} \{simp\}: \( \text{cl}(A) \equiv \text{Closure}(A,T) \)}  
\texttt{fixes \texttt{boundary} (\( \partial \_ \{91\} \_92\)) \texttt{defines \texttt{boundary_def} \{simp\}: \( \partial A \equiv \text{Boundary}(A,T) \)}}

Intersection of a finite nonempty collection of open sets is open.

\texttt{lemma (in topology0) \texttt{fin_inter_open_open}: \assumes \( N \neq 0 \) \( N \in \text{FinPow}(T) \) \shows \( \bigcap N \in T \) \texttt{using \texttt{topSpaceAssum} \assms \texttt{IsATopology_def} \texttt{inter_two_inter_fin} by \texttt{simp}}}  

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 \texttt{ZF1.thy}.

\texttt{lemma (in topology0) \texttt{Top_1_L4}:} \texttt{shows} \( (T \{\text{restricted to}\} X) \) \{is a topology\}  
\texttt{proof} -  
\texttt{let \( S \equiv T \{\text{restricted to}\} X \)}
have \( \forall A \in \text{Pow}(S). \bigcup A \in S \)

proof
fix \( A \) assume \( A \in \text{Pow}(S) \)
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 \( \bigcup M \in T \)
  using IsATopology_def by simp
  thus thesis by simp
  qed
  hence \( \{ \bigcup \{ U \in T. V = U \cap X \}. V \in A \} \subseteq T \) by auto
  with topSpaceAssum have \( \bigcup \{ U \in T. V = U \cap X \} \in T \)
  using RestrictedTo_def by auto
  moreover from \( A \) have \( \forall V \in A. \exists U \in T. V = U \cap X \)
  using RestrictedTo_def by auto
  hence \( \bigcup \{ U \in T. V = U \cap X \} \cap X = \bigcup A \) by blast
  ultimately show \( \bigcup A \in S \)
  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
    \( U_1 \in T \) \( U = U_1 \cap X \) and \( V_1 \in T \) \( V = V_1 \cap X \)
    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 \( S \) \{is a topology\} using IsATopology_def by simp
qed

52.2 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 Interior_def by auto

Interior is open.

lemma (in topology0) Top_2_L2: shows \( \text{int}(A) \in T \)
proof -
A set is open iff it is equal to its interior.

**Lemma (in topology0) Top_2_L3:** shows $U \in T \iff \text{int}(U) = U$

**Proof**
1. Assume $U \in T$
   - Then show $\text{int}(U) = U$
     - Using $\text{Interior_def}$ by auto
2. Next assume $A1: \text{int}(U) = U$
   - Have $\text{int}(U) \in T$
     - Using $\text{Top_2_L2}$ by simp
   - Show $U \in T$
     - 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 $U \in T$
    - 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 $\forall U \in T. (U \subseteq A \rightarrow U \subseteq B)$ 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 $U \subseteq \text{int}(A)$

**Proof**
- Using $\text{assms $\text{Interior_def}$ by auto}$

If a point of a set has an open neighborhood 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 $x \in \text{int}(A)$

**Proof**
- Using $\text{assms $\text{Interior_def}$ by auto}$

A set is open iff its every point has a an open neighborhood 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 a an open neighborhood 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 a 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 a 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

52.3 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

652
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 $A \subseteq \bigcup T$ shows $\text{ClosedCovers}(A,T) \neq 0$

**proof** -

from A1 have $\bigcup T \in \text{ClosedCovers}(A,T)$ using $\text{ClosedCovers_def}$ $\text{Top_3_L1}$

by auto

thus thesis by auto

qed

Intersection of a nonempty family of closed sets is closed.

**lemma (in topology0) Top_3_L4:**

assumes $K \neq 0$ and $\forall D \in K. \ D \text{ is closed in } T$

shows $(\bigcap K) \text{ is closed in } T$

**proof** -

from A2 have $I: \forall D \in K. \ (D \subseteq \bigcup T \land (\bigcup T - D) \in T)$

using $\text{IsClosed_def}$ by simp

then have $\{ \bigcup T - D. \ D \in K \} \subseteq T$ by auto

with $\text{topSpaceAssum}$ have $(\bigcup \{ \bigcup T - D. \ D \in K \}) \in T$

using $\text{IsATopology_def}$ by auto

moreover from A1 have $\bigcup \{ \bigcup T - D. \ D \in K \} = \bigcup T - \bigcap K$ by fast

moreover from A1 I have $\bigcap K \subseteq \bigcup T$ by blast

ultimately show $(\bigcap K) \text{ is closed in } T$ using $\text{IsClosed_def}$ by simp

qed

The union and intersection of two closed sets are closed.

**lemma (in topology0) Top_3_L5:**

assumes $D_1 \{\text{is closed in}\} T \ D_2 \{\text{is closed in}\} T$

shows $(D_1 \cap D_2) \{\text{is closed in}\} T$

$(D_1 \cup D_2) \{\text{is closed in}\} T$

**proof** -

have $\{D_1,D_2\} \neq 0$ by simp

with A1 have $(\bigcap \{D_1,D_2\}) \{\text{is closed in}\} T$ using $\text{Top_3_L4}$

by fast

thus $(D_1 \cap D_2) \{\text{is closed in}\} T$ by simp

from $\text{topSpaceAssum}$ A1 have $(\bigcup T - D_1) \cap (\bigcup T - D_2) \in T$

using $\text{IsClosed_def}$ $\text{IsATopology_def}$ by simp

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 $\text{IsClosed_def}$

by auto

ultimately show $(D_1 \cup D_2) \{\text{is closed in}\} T$ using $\text{IsClosed_def}$

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.

**Lemma (in topology0) cl_is_closed:**

assumes $A \subseteq \bigcup T$

shows $\text{cl}(A) \ {\text{is closed in}} \ T$

using assms Closure_def Top_3_L3 ClosedCovers_def Top_3_L4

by simp

Closure of a bigger sets is bigger.

**Lemma (in topology0) top_closure_mono:**

assumes $A1: A \subseteq \bigcup T \ B \subseteq \bigcup T$ and $A2: A \subseteq B$

shows $\text{cl}(A) \subseteq \text{cl}(B)$

**Proof** -

from $A2$ have $\text{ClosedCovers}(B,T) \subseteq \text{ClosedCovers}(A,T)$

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**

A set is closed iff it is equal to its closure.

**Lemma (in topology0) Top_3_L8:**

assumes $A1: A \subseteq \bigcup T$

shows $A \ {\text{is closed in}} \ T \iff \text{cl}(A) = A$

**Proof**

assume $A \ {\text{is closed in}} \ T$

with $A1$ show $\text{cl}(A) = A$

using Closure_def ClosedCovers_def by auto

next assume $\text{cl}(A) = A$

654
then have $\bigcup T - A = \bigcup T - \operatorname{cl}(A)$ by simp

with A1 show $A$ {is closed in} $T$ using cl_is_closed IsClosed_def by simp

qed

Complement of an open set is closed.

**Lemma (in topology0) Top_3_L9:**

assumes A1: $A \in T$

shows $(\bigcup T - A)$ {is closed in} $T$

proof -

from topSpaceAssum A1 have $\bigcup T - (\bigcup T - A) = A$ and $\bigcup T - A \subseteq \bigcup T$

using IsATopology_def by auto

with A1 show $(\bigcup T - A)$ {is closed in} $T$ using IsClosed_def by simp

qed

A set is contained in its closure.

**Lemma (in topology0) cl_contains_set:**

assumes $A \subseteq \bigcup T$

shows $A \subseteq \operatorname{cl}(A)$

using assms Top_3_L1 ClosedCovers_def Top_3_L3 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 A1: $A \subseteq \bigcup T$

shows $\operatorname{cl}(A) \subseteq \bigcup T$

$\operatorname{cl}(\bigcup T - A) = \bigcup T - \operatorname{int}(A)$

proof -

from A1 show $\operatorname{cl}(A) \subseteq \bigcup T$ using Top_3_L1 Closure_def ClosedCovers_def by auto

from A1 have $\bigcup T - A \subseteq \bigcup T - \operatorname{int}(A)$ using Top_2_L1 by auto

moreover have I: $\bigcup T - \operatorname{int}(A) \subseteq \bigcup T$ $\bigcup T - A \subseteq \bigcup T$ by auto

ultimately have $\operatorname{cl}(\bigcup T - A) \subseteq \operatorname{cl}(\bigcup T - \operatorname{int}(A))$

using top_closure_mono by simp

moreover

from I have $(\bigcup T - \operatorname{int}(A))$ {is closed in} $T$

using Top_2_L2 Top_3_L9 by simp

with I have $\operatorname{cl}((\bigcup T) - \operatorname{int}(A)) = \bigcup T - \operatorname{int}(A)$

using Top_3_L8 by simp

ultimately have $\operatorname{cl}(\bigcup T - A) \subseteq \bigcup T - \operatorname{int}(A)$ by simp

moreover

from I have $\bigcup T - A \subseteq \operatorname{cl}(\bigcup T - A)$ using cl_contains_set by simp

hence $\bigcup T - \operatorname{cl}(\bigcup T - A) \subseteq A$ and $\bigcup T - A \subseteq \bigcup T$ by auto

then have $\bigcup T - \operatorname{cl}(\bigcup T - A) \subseteq \operatorname{int}(A)$

using cl_is_closed IsClosed_def Top_2_L5 by simp

hence $\bigcup T - \operatorname{int}(A) \subseteq \operatorname{cl}(\bigcup T - A)$ by auto

ultimately show $\operatorname{cl}(\bigcup T - A) = \bigcup T - \operatorname{int}(A)$ 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 ⊆ ∪T
  shows ∂A = cl(A) - int(A)
proof -
  from A1 have ∂A = cl(A) ∩ (∪T - int(A))
    using Boundary_def Top_3_L11 by simp
  moreover from A1 have
    cl(A) ∩ (∪T - int(A)) = cl(A) - int(A)
    using Top_3_L11 by blast
  ultimately show ∂A = cl(A) - 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 {is closed in} T A ⊆ B
  shows cl(A) ⊆ B
proof -
  from A1 have B ⊆ ∪T using IsClosed_def by simp
  with A1 show cl(A) ⊆ 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 ⊆ ∪T V ∈ T and A2: A ∩ V = 0
  shows cl(A) ∩ V = 0
using assms disj_open_cl_disj by auto

A reverse 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 A1: A ⊆ ∪T and U ∈ T and x ∈ cl(A) ∩ U
  shows A ∩ U ≠ 0 using assms disj_open_cl_disj by auto

A reformulation of cl_inter_neigh: if every open neighborhood 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 ≠ Ø
  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 = Ø
      unfolding IsClosed_def by auto
    with A3 have False by auto
  } thus thesis by auto
qed

53 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.

53.1 Separation axioms.

Topological spaces cas be classified according to certain properties called ”separation axioms”. In this section we define what it means that a topological space is T₀, T₁ or T₂.

A topology on X is T₀ if for every pair of distinct points of X there is an open set that contains only one of them.

definition
  isT0 (_ {is T₀} [90] 91) where
  T {is T₀} ≡ ∀ x y. ((x ∈ ∪T ∧ y ∈ ∪T ∧ x ≠ y) →
                               (∃ U ∈ T. (x ∈ U ∧ y ∉ U) ∨ (y ∈ U ∧ x ∉ U)))

A topology is T₁ 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)))
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.

\[
\text{definition} \quad \text{is}T_2 \quad \text{where} \quad T \text{ is } T_2 \iff \forall \ldotp (x \in T \land y \in T \land x \neq y \to (\exists U \in T. \exists V \in T. x \in U \land y \in V \land U \cap V = \emptyset))
\]

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 is$T_0$ condition and is$T_1$.

**Lemma T1_is_T0:** assumes $A1: T \text{ is } T_1$ shows $T \text{ is } T_0$

**proof**
- from $A1$ have $\forall \ldotp x \in T \land y \in T \land x \neq y \to (\exists U \in T. \exists V \in T. x \in U \land y \notin U)$
  using isT1_def by simp
- then have $\forall \ldotp x \in T \land y \in T \land x \neq y \to (\exists U \in T. \exists V \in T. x \in U \land y \notin U \lor y \in U \land x \notin U)$
  by auto
- then show $T \text{ is } T_0$ using isT0_def by simp
qed

If a topology is $T_2$ then it is $T_1$.

**Lemma T2_is_T1:** assumes $A1: T \text{ is } T_2$ shows $T \text{ is } T_1$

**proof**
- \{ fix \ldotp \choose x \ y \ assumes \ x \in T \land y \in T \land x \neq y \\
  \quad \text{with} \ A1 \ \text{have} \ \exists U \in T. \exists V \in T. x \in U \land y \notin U \land U \cap V = \emptyset \\
  \quad \text{using} \ \text{isT2_def by simp} \\
  \quad \text{then have} \ \exists U \in T. x \in U \land y \notin U \land y \in U \land x \notin U \\
  \quad \text{by auto} \\
  \} \ \text{then have} \ \forall \ldotp \choose x \ y \ assumes \ x \in T \land y \in T \land x \neq y \\
  \quad \text{to} \ \exists U \in T. \exists V \in T. x \in U \land y \notin U \\
  \quad \text{by simp} \\
  \text{then show} \ T \text{ is } T_1 \text{ 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 $A2: x \in T \land y \in T$ and $A3: \forall U \in T. (x \in U \iff y \in U)$

shows $x=y$

**proof**
- \{ assume \ldotp \choose x \neq y \\
  \quad \text{with} \ A1 \ A2 \ \text{have} \ \exists U \in T. x \in U \land y \notin U \lor y \in U \land x \notin U \\
  \quad \text{using} \ \text{isT0_def by simp} \\
  \quad \text{with} \ A3 \ \text{have} \ \text{False by auto} \\
  \} \ \text{then show} \ x=y \ \text{by auto} \\
qed
53.2 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**

\[ \text{IsAbaseFor (infixl \{is a base for\} 65) where} \]
\[ B \{\text{is a base for}\} T \equiv B \subseteq T \land T = \bigcup A. A \in \text{Pow}(B) \]

A subbase is a collection of open sets such that finite intersection of those sets form a base.

**definition**

\[ \text{IsAsubBaseFor (infixl \{is a subbase for\} 65) where} \]
\[ B \{\text{is a subbase for}\} T \equiv B \subseteq T \land \{\bigcap A. A \in \text{FinPow}(B)\} \{\text{is a base for}\} T \]

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 neighboorhood from \( B \) contained in \( U \cap V \).

**definition**

\[ \text{SatisfiesBaseCondition (_ \{satisfies the base condition\} [50] 50) where} \]
\[ B \{\text{satisfies the base condition}\} \equiv \forall U V. ((U \in B \land V \in B) \rightarrow (\forall x \in U \cap V. \exists W \in B. x \in W \land W \subseteq U \cap V)) \]

A collection that is closed with respect to intersection satisfies the base condition.

**lemma** \text{inter\_closed\_base}: assumes \( \forall U \in B. (\forall V \in B. U \cap V \in B) \)
shows \( B \{\text{satisfies the base condition}\} \)

**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** \text{Top\_1\_2\_L1}: assumes \( B \{\text{is a base for}\} T \) 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** \text{base\_sets\_open}:

assumes \( B \{\text{is a base for}\} T \) and \( U \in B \)
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 ∈ T and A3: x ∈ U

shows ∃ V ∈ B. V ⊆ U ∧ x ∈ V

proof -

from A1 A2 obtain A where A ∈ Pow(B) and U = ∪ A

using Top_1_2_L1 by blast

with A3 obtain V where V ∈ A and x ∈ V by auto

with <A ∈ Pow(B)> <U = ∪A> 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 ⊆ T and A3: ∀ V ∈ T. ∃ A ∈ Pow(B). V = ∪ A

shows B {is a base for} T

proof -

from A3 have T ⊆ {∪ A. A ∈ Pow(B)} by auto

moreover have {∪ A. A ∈ Pow(B)} ⊆ T

proof

fix U assume U ∈ {∪ A. A ∈ Pow(B)}

then obtain A where A ∈ Pow(B) and U = ∪ A

by auto

with <B ⊆ T> have A ∈ Pow(T) by auto

with A1 <U = ∪A> show U ∈ T

unfolding IsATopology_def by simp

qed

ultimately have T = {∪ A. A ∈ 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: \( \exists T. T \text{ is a topology} \) \( \land \) B \{ is a base for \} T
and A2: \( V \in B \) \( \land \) W \in B
shows \( \forall x \in V \cap W. \exists U \in B. x \in U \land U \subseteq V \cap W \)
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 \( \forall x \in V \cap W. \exists U \in B. (x \in U \land U \subseteq V \cap W) \)
  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: \( \forall x \in V \cap W. \exists U \in B. x \in U \land U \subseteq V \cap W \)
shows V \cap W \in \{ \bigcup A. A \in \text{Pow}(B) \}
proof
  let A = \bigcup x \in V \cap W. \{ U \cap V. V \in A \}
  from A1 show A \subseteq \{ \bigcup A. A \in \text{Pow}(B) \}
  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 \}
s 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
    by auto
  let C = \bigcup \{ U \in A_1. V \in A_2 \}
  from D1 have \( \forall U \in A_1. U \in B \) \( \land \) \( \forall V \in A_2. V \in B \) by auto
  with A2 have C \subseteq \{ \bigcup A. A \in \text{Pow}(B) \}
    using Top_1_2_L3 SatisfiesBaseCondition_def by auto
  moreover from D1 have \( U_1 \cap U_2 = \bigcup C \)
  by auto
  ultimately show thesis by auto
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:**

**Assumptions:**
- $A1$: $B$ satisfies the base condition
- $A2$: $T = \{\bigcup A. A \in \text{Pow}(B)\}$

**Shows:**
- $T$ is a topology
- $B$ is a base for $T$

**Proof:**

1. **Show $T$ is a topology**
   - **Proof:**
     - **Have:** $\forall C \in \text{Pow}(T). \bigcup C \in T$
       - **Proof:**
         - **Fix $C.$ Assume $A3$: $C \in \text{Pow}(T)$
           - Let $Q = \bigcup \{\bigcup \{A \in \text{Pow}(B). U = \bigcup A\}. U \in C\}$
           - From $A2$ $A3$ have $\forall U \in C. \exists A \in \text{Pow}(B). U = \bigcup A$ 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:** $\forall U \in T. \forall V \in T. U \cap V \in T$
     - **Proof:**
       - **Fix $U, V.$ Assume $U \in T. V \in T$ 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 $C \subseteq T$ and $U \cap V = \bigcup C$ by auto
         - With $I$ have $U \cap V \in T$ by simp
     - Then show $\forall U \in T. \forall V \in T. U \cap V \in T$ by simp
     - **Qed**

   - **Ultimately show $T$ is a topology** using $\text{IsATopology_def}$ by simp
     - **Qed**

   - From $A2$ have $B \subseteq T$ by auto
     - With $A2$ show $B$ is a base for $T$ using $\text{IsAbaseFor_def}$ by simp
     - **Qed**

The carrier of the base and topology are the same.

**Lemma Top_1_2_L5:**

**Assume:** $B$ is a base for $T$

**Shows:** $\bigcup T = \bigcup B$

**Using:** $\text{assms IsAbaseFor_def}$ by auto

If $B$ is a base for $T$, then $T$ is the smallest topology containing $B$.

**Lemma base_smallest_top:**

**Assume:** $A1$: $B$ is a base for $T$ and $A2$: $S$ is a topology and $A3$: $B \subseteq S$

**Shows:** $T \subseteq S$
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 <U = ∪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

53.3 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.

definition
  ProductTopology(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 ∀x∈(A∩B). ∃W∈ProductCollection(T,S). (x∈W ∧ W ⊆ A∩B)
proof
  fix x assume A3: x ∈ A∩B
  from A2 obtain U_1 V_1 U_2 V_2 where
    D1: U_1∈T V_1∈S A=U_1×V_1 U_2∈T V_2∈S B=U_2×V_2
    using ProductCollection_def by auto
  let W = (U_1∩U_2) × (V_1∩V_2)
  from A1 D1 have U_1∩U_2 ∈ T and V_1∩V_2 ∈ S
    using IsATopology_def by auto
  then have W ∈ ProductCollection(T,S) using ProductCollection_def
    by auto
  moreover from A3 D1 have x∈W and W ⊆ A∩B by auto
  ultimately have ∃W. (W ∈ ProductCollection(T,S) ∧ x∈W ∧ W ⊆ A∩B)
    by auto
  thus ∃W∈ProductCollection(T,S). (x∈W ∧ W ⊆ A∩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}
ProductTopology(T,S) {is a topology}
ProductCollection(T,S) {is a base for} ProductTopology(T,S)
∪ ProductTopology(T,S) = ∪ T × ∪ S

proof -
from A1 show
  ProductTopology(T,S) {is a topology}
  ProductCollection(T,S) {is a base for} ProductTopology(T,S)
  using Top_1_4_L1 ProductCollection_def
  SatisfiesBaseCondition_def ProductTopology_def Top_1_2_T1
  by auto
then show ∪ ProductTopology(T,S) = ∪ T × ∪ 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 ∈ ProductTopology(T,S) and A3: x ∈ U
  shows ∃ V W. V ∈ T ∧ W ∈ S ∧ V × W ⊆ U ∧ x ∈ V × W
proof -
from A1 have
  ProductCollection(T,S) {is a base for} ProductTopology(T,S)
  using Top_1_4_T1 by simp
moreover from A2 A3 obtain Z where
  Z ∈ ProductCollection(T,S) and Z ⊆ U ∧ x ∈ Z
  using point_open_base_neigh by blast
then obtain V W where V ∈ T and W ∈ S and V × W ⊆ U ∧ x ∈ V × W
  using ProductCollection_def by auto
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 ⊆ ∪ T × ∪ S
proof -
  from A2 have V ⊆ ∪ ProductTopology(T,S) by auto
  with A1 show thesis using Top_1_4_T1 by simp
qed

Suppose we have subsets A ⊆ X, B ⊆ Y, where X,Y are topological spaces with topologies T, S. We can the 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 × V : U ∈ T_A, V ∈ S_B\}). The next lemma states that this collection is a base of the product topology on X × Y restricted to the product A × B.

lemma prod_restr_base_restr: assumes A1: T {is a topology} S {is a topology} shows ProductCollection(T {restricted to} A, S {restricted to} B) {is a base for} (ProductTopology(T,S) {restricted to} A × B)
proof -
let B = ProductCollection(T {restricted to} A, S {restricted to} B)
let τ = ProductTopology(T,S)
from A1 have (τ {restricted to} A × B) {is a topology}
  using Top_1_4_T1 topology0_def topology0.Top_1_L4 by simp
moreover have B ⊆ (τ {restricted to} A × B)
proof
  fix U assume U ∈ B
  then obtain U_A U_B where U = U_A × U_B and
    U_A ∈ (T {restricted to} A) and U_B ∈ (S {restricted to} B)
    using ProductCollection_def by auto
  then obtain W_A W_B where
    W_A ∈ T U_A = W_A ∩ A and W_B ∈ S U_B = W_B ∩ B
    using RestrictedTo_def by auto
  with <U = U_A × U_B> have U = W_A × W_B ⊆ (A × B) by auto
  moreover from A1 <W_A ∈ T> and <W_B ∈ S> have W_A × W_B ∈ τ
    using prod_open_open_prod by simp
  ultimately show U ∈ τ {restricted to} A × B
    using RestrictedTo_def by auto
qed

moreover have ∀ U ∈ τ {restricted to} A × B.
  ∃ C ∈ Pow(B), U = ∪ C
proof
  fix U assume U ∈ τ {restricted to} A × B
  then obtain W where W ∈ τ and U = W ∩ (A × B)
    using RestrictedTo_def by auto

665
from A1 \( W \in \tau \) obtain \( A_W \) where 
\( A_W \in \text{Pow(ProductCollection}(T,S)) \) 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 \( C \in \text{Pow}(B) \) 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 \( <A_W \in \text{Pow(ProductCollection}(T,S))> \) obtain \( V_T \) \( V_S \) where 
\( V_T \in T \) and \( V_S \in S \) and \( V = V_T \times V_S \) 
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 \( <U = W \cap (A \times B)> \) and \( <W = \bigcup A_W> \) 
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: \( T \) \( \{ \text{is a topology} \} \) \( S \) \( \{ \text{is a topology} \} \) 
shows 
ProductTopology(T \{restricted to\} A,S \{restricted to\} B) = 
ProductTopology(T,S) \{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\} ProductTopology(T \{restricted to\} A,S \{restricted to\} B) 
using topology0_def topology0.Top_1_L4 Top_1_4_T1 by simp 
moreover from A1 have 
\( B \) \{is a base for\} ProductTopology(T,S) \{restricted to\} (A \times B) 
using prod_restr_base_restr by simp 
ultimately show thesis by (rule same_base_same_top) 
qed 

Projection of a section of an open set is open.

lemma prod_sec_open1: assumes A1: \( T \) \( \{ \text{is a topology} \} \) \( S \) \( \{ \text{is a topology} \} \) and 
A2: \( \forall V \in \text{ProductTopology}(T,S) \) and A3: \( x \in \bigcup T \) 
shows \( \{ y \in \bigcup S. (x,y) \in V \} \in S \) 
proof -

666
let \( A = \{ y \in \bigcup S. \langle x, y \rangle \in V \} \)

from A1 have toplogy0(S) using topology0_def by simp

moreover have \( \forall y \in A. \exists W \in S. (y \in W \land W \subseteq A) \)

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 \( \langle x, y \rangle \in V \) have \( \exists U W. U \in T \land W \in S \land U \times W \subseteq V \land \langle x, y \rangle \in U \times W \) by (rule prod_top_point_neighb)

then obtain \( U W \) where \( U \in T \) W \( \in S \) U \( \times W \subseteq V \land \langle x, y \rangle \in U \times W \) by auto

with A1 A2 show \( \exists W \in S. (y \in W \land W \subseteq A) \) 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 \( \{ x \in \bigcup T. \langle x, y \rangle \in V \} \in T \)

**proof**

- let \( A = \{ x \in \bigcup T. \langle x, y \rangle \in V \} \)

from A1 have toplogy0(T) using topology0_def by simp

moreover have \( \forall x \in A. \exists W \in T. (x \in W \land W \subseteq A) \)

proof

fix \( x \) assume \( x \in A \)

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 \( \langle x, y \rangle \in V \) have \( \exists U W. U \in T \land W \in S \land U \times W \subseteq V \land \langle x, y \rangle \in U \times W \) by (rule prod_top_point_neighb)

then obtain \( U W \) where \( U \in T \) W \( \in S \) U \( \times W \subseteq V \land \langle x, y \rangle \in U \times W \) by auto

with A1 A2 show \( \exists W \in T. (x \in W \land W \subseteq A) \) using prod_open_type section_proj by auto

qed

ultimately show thesis by (rule topology0.open_neigh_open)

qed

end
A metric space is a set on which a distance between points is defined as a function \( d : X \times X \rightarrow [0, \infty) \). With this definition each metric space is a topological space which is paracompact and Hausdorff (\( T_2 \)), hence normal (in fact even perfectly normal).

### 54.1 Pseudometric - definition and basic properties

A metric on \( X \) is usually defined as a function \( d : X \times X \rightarrow [0, \infty) \) 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),
- \( d(x, y) \leq d(x, z) + d(z, y) \) (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 \( G \).

**definition**

\[
\text{IsApseudoMetric}(d, X, G, A, r) \equiv d : X \times X \rightarrow \text{Nonnegative}(G, A, r) \\
\wedge (\forall x \in X. d(x, x) = \text{TheNeutralElement}(G, A)) \\
\wedge (\forall x \in X. \forall y \in X. d(x, y) = d(y, x)) \\
\wedge (\forall x \in X. \forall y \in X. \forall z \in X. (d(x, z), A(d(x, y), d(y, z))) \in r)
\]

We add the full axiom of identity of indiscernibles to the definition of a pseudometric to get the definition of a metric.

**definition**

\[
\text{IsAmetric}(d, X, G, A, r) \equiv \\
\text{IsApseudoMetric}(d, X, G, A, r) \wedge (\forall x \in X. \forall y \in X. d(x, y) = \text{TheNeutralElement}(G, A)) \\
\Rightarrow x = y
\]

A disk is defined as set of points located less than the radius from the center.

**definition**

\[
\text{Disk}(X, d, r, c, R) \equiv \{ x \in X. (d(c, x), R) \in \text{StrictVersion}(r) \}
\]

Next we define notation for metric spaces. We will reuse the additive notation defined in the \textit{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.
locale pmetric_space = loop1 + 
  fixes d and X

assumes pmetricAssum: IsApseudoMetric(d,X,L,A,r)

fixes disk
defines disk_def [simp]: disk(c,R) ≡ Disk(X,d,r,c,R)

The next lemma shows the definition of the pseudometric in the notation used in the metric_space context.

lemma (in pmetric_space) pmetric_properties: shows 
d: X×X → L+
∀x∈X. d(x,x) = 0
∀x∈X.∀y∈X. d(x,y) = d(y,x)
∀x∈X.∀y∈X.∀z∈X. d(x,z) ≤ d(x,y) + d(y,z)
using pmetricAssum unfolding IsApseudoMetric_def by auto

The definition of the disk in the notation used in the pmetric_space context:

lemma (in pmetric_space) disk_definition: shows disk(c,R) = {x∈X. d(c,x) < R}
proof -
  have disk(c,R) = Disk(X,d,r,c,R) by simp
  then have disk(c,R) = {x∈X. (d(c,x),R) ∈ StrictVersion(r)} unfolding Disk_def by simp
  moreover have ∀x∈X. (d(c,x),R) ∈ StrictVersion(r) ←→ d(c,x) < R
    using 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∈X and R∈L⁺ shows c ∈ disk(c,R)
  using 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∈X and x ∈ disk(c,R)
  shows R∈L 0<R R∈L⁺ (-d(c,x) + R) ∈ L⁺
proof -
  from assms(2) have x∈X and d(c,x) < R using disk_definition by auto
  with assms(1) show 0<R using pmetric_properties(1) apply_funtype nonneg_def loop_strict_ord_trans by blast
  then show R∈L and R∈L⁺ using posset_definition PositiveSet_def by auto
  from <d(c,x) < R> show (-d(c,x) + R) ∈ L⁺
    using ls_other_side(2) by simp
qed
If a point $x$ is inside a disk $B$ and $m \leq R - d(c,x)$ 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 assms(1,2) $<y \in \text{disk}(x,m)>$ have $R \in L \ x \in X \ y \in X$
using $\text{radius_in_loop}(1)$ $\text{disk_definition}$ by auto
with assms(1) have $d(c,y) \leq d(c,x) + d(x,y)$ using $\text{pmetric_properties}(4)$ by simp
from assms(1) $<x \in X>$ have $d(c,x) \in L$
using $\text{pmetric_properties}(1)$ $\text{apply_funtype}$ $\text{nonneg_subset}$ by auto
with $<d(x,y) < m>$ assms(3) have $d(c,x) + d(x,y) < d(c,x) + (-d(c,x) + R)$
using $\text{loop_strict_ord_trans1}$ $\text{strict_ord_trans_inv}(2)$ by blast
with $<d(c,x) \in L \ <R \in L > <d(c,y) \leq d(c,x) + d(x,y)> <y \in X>$ show $y \in \text{disk}(c,R)$
using $\text{lrdiv_props}(6)$ $\text{loop_strict_ord_trans}$ $\text{disk_definition}$ 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.

**Lemma (in pmetric_space) disks_form_base:**

assumes IsMeetSemilattice($L_+, r \cap L_+ \times L_+$)

defines $B \equiv \bigcup c \in X. \ (\text{disk}(c,R)). \ R \in L_+$

shows $B$ (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 assms(2) $<U \in B > <V \in B >$ 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 $<x \in U \cap V>$ 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 $<c_U \in X > <x \in \text{disk}(c_U , R_U)> <c_V \in X > <x \in \text{disk}(c_V , R_V)>$ have $m_U \in L_+$

and $m_V \in L_+$

using $\text{radius_in_loop}(4)$ by auto

let $m = \text{Meet}(L_+, r \cap L_+ \times L_+) (m_U , m_V)$

670
let \( W = \text{disk}(x,m) \)
from \( \text{assms}(1) <m_U \in L_+> <m_V \in L_+> \) have \( \langle m,m_U \rangle \in r \cap L_+ \times L_+ \)
using \text{meet_val}(3) by \text{blast}
moreover from \( \text{assms}(1) <m_U \in L_+> <m_V \in L_+> \) have \( m \in L_+ \)
using \text{meet_val}(1) by \text{simp}
ultimately have \( m \in L_+ \)
moreover from \( \text{assms}(1) <m_U \in L_+> <m_V \in L_+> \) have \( m \leq m_U \)
moreover from \( \text{assms}(1) <m_U \in L_+> <m_V \in L_+> \) have \( m \leq m_V \)
by \text{auto}
with \( <c_U \in X> <x \in \text{disk}(c_U,R_U)> <c_V \in X> <x \in \text{disk}(c_V,R_V)> <U = \text{disk}(c_U,R_U)> <V = \text{disk}(c_V,R_V)> \)
have \( W \subseteq U \cap V \) using \text{disk_in_disk} by \text{blast}
moreover from \( \text{assms}(2) <x \in X> <m \in L_+> \) have \( W \in B \) and \( x \in W \)
using \text{center_in_disk} by \text{auto}
ultimately show thesis by auto
qed
}
then show thesis unfolding \text{SatisfiesBaseCondition_def} by auto
qed

Unions of disks form a topology, hence (pseudo)metric spaces are topological spaces. We have to add the assumption that the positive set is not empty. This is necessary to show that we can cover the space with disks and it does not look like it follows from anything we have assumed so far.

\textbf{theorem \textit{in} pmetric_space \textit{pmetric_is_top}}:
assumes IsMeetSemilattice\((L_+,r \cap L_+ \times L_+)\) \( L_+ \not= 0 \)
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 \( T \) \{ \text{is a topology} \} \ B \{ \text{is a base for} \} \ T \quad \bigcup T = X \)
proof -
from \( \text{assms}(1,3,4) \) show \( T \) \{ \text{is a topology} \} \ B \{ \text{is a base for} \} \ T
using \text{disks_form_base Top_1_2_T1} by auto
then have \( \bigcup T = \bigcup B \) using \text{Top_1_2_L5} by \text{simp}
moreover have \( \bigcup B = X \)
proof
from \( \text{assms}(3) \) show \( \bigcup B \subseteq X \) using \text{disk_definition} by \text{auto}
\{ fix \( x \) assume \( x \in X \)
from \( \text{assms}(2) \) obtain \( R \) where \( R \in L_+ \) by \text{auto}
with \( \text{assms}(3) \) \( <x \in X> \) have \( x \in \bigcup B \) using \text{center_in_disk} by \text{auto}
\} thus \( X \subseteq \bigcup B \) by \text{auto}
qed
ultimately show \( \bigcup T = X \) by \text{simp}
qed

end

671
55 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.

55.1 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 (R) and Add and Mul and ROrd  
  assumes R_are_reals: IsAmodelOfReals (R, Add, Mul, ROrd)

  fixes zero (0)  
  defines zero_def [simp]: 0 ≡ TheNeutralElement (R, Add)

  fixes one (1)  
  defines one_def [simp]: 1 ≡ TheNeutralElement (R, Mul)

  fixesrealmul (infixl · 71)  
  defines realmul_def [simp]: x · y ≡ Mul (x, y)

  fixesrealadd (infixl + 69)  
  defines realadd_def [simp]: x + y ≡ Add (x, y)

  fixesrealminus(- _) [89]  
  defines realminus_def [simp]: (-x) ≡ GroupInv (R, Add) (x)

  fixesrealsub (infixl - 90)  
  defines realsub_def [simp]: x - y ≡ x + (-y)

  fixeslesseq (infix ≤ 68)  
  defines lesseq_def [simp]: x ≤ y ≡ (x, y) ∈ ROrd

  fixessless (infix < 68)  
  defines sless_def [simp]: x < y ≡ x ≤ y ∧ x ≠ y

  fixesnonnegative (R⁺)  
  defines nonnegative_def [simp]: R⁺ ≡ Nonnegative (R, Add, ROrd)
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 ⊆ R × R and IsLinOrder(R, ROrd) and ∀x y. ∀z∈R. (x, y) ∈ ROrd → (Add(x, z), Add(y, z)) ∈ ROrd

and Nonnegative(R, Add, ROrd) {is closed under} Mul and TheNeutralElement(R, Add) ≠ TheNeutralElement(R, Mul)

and ∀x∈R. x ≠ TheNeutralElement(R, Add) → (∃y∈R. Mul(x, y) = TheNeutralElement(R, 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 realsubrealmul
    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} R and Add (is commutative on) R
    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×R → R⁺
    ∀x∈R.∀y∈R. dist⟨x,y⟩ = |x - y|
    ∀x∈R.∀y∈R. dist⟨y,x⟩ = dist⟨x,y⟩
    ∀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,x⟩ = 0 −→ x=y
    IsApseudoMetric(dist,R,R,Add,ROrd)
    IsAmetric(dist,R,R,Add,ROrd)

proof -
    show I: dist : R×R → R⁺ using group_op_closed inverse_in_group OrdRing_ZF_1_L4

    OrderedGroup_ZF_3_L3B ZF_fun_from_total by simp
    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 group0_2_L6 OrderedGroup_ZF_3_L2A
    by simp

    { fix x y
      assume x∈R y∈R
      then have ⟨-x⟩ = y-x using group0_2_L12 by simp
      moreover from x∈R y∈R have ⟨x-y⟩ = |x-y|
      using group_op_closed inverse_in_group basic_props(1) OrderedGroup_ZF_3_L7A

    674
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 \(x\in\mathbb{R}\) \(y\in\mathbb{R}\) \(\text{dist}(x,y) = 0\)
with II have \(|x-y| = 0\) by simp
with \(<x\in\mathbb{R}> <y\in\mathbb{R}>\) have \(x-y = 0\)
using group_op_closed inverse_in_group OrderedGroup_ZF_3_L3D by auto
\}
\{ fix \(x\) \(y\) \(z\)
assume \(x\in\mathbb{R}\) \(y\in\mathbb{R}\) \(z\in\mathbb{R}\)
then have \(|x-z| = |(x-y)+(y-z)|\) using cancel_middle(5) by simp
with \(<x\in\mathbb{R}> <y\in\mathbb{R}> <z\in\mathbb{R}>\) have \(|x-z| \leq |x-y| + |y-z|\)
using group_op_closed inverse_in_group OrdRing_ZF_1_L4(2,3) OrdGroup_triangle_ineq by simp
} thus \(\forall x\in\mathbb{R}.\forall y\in\mathbb{R}.\forall z\in\mathbb{R}.\ \text{dist}(x,z) \leq \text{dist}(x,y) + \text{dist}(y,z)\) by auto
with I III IV V show IsApsedoMetric(\(\text{dist}\),\(\mathbb{R}\),Add,ROrd) and IsAmetric(\(\text{dist}\),\(\mathbb{R}\),Add,ROrd)
unfolding IsApsedoMetric_def IsAmetric_def by auto
qed

Real numbers form an ordered loop.

lemma (in reals) reals_loop: shows IsAnOrdLoop(\(\mathbb{R}\),Add,ROrd)
proof -
\quad have IsAloop(\(\mathbb{R}\),Add) using group_is_loop by simp
moreover from R_are_reals have ROrd \(\subseteq\) \(\mathbb{R}\) × \(\mathbb{R}\) and IsPartOrder(\(\mathbb{R}\),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 \(x\leq y \iff x+z \leq y+z\)
using ord_transl_inv ineq_cancel_right by blast
moreover from A have \(x\leq y \iff z+x \leq z+y\)
using ord_transl_inv OrderedGroup_ZF_1_L5AE by blast
ultimately have \((x\leq y \iff x+z \leq y+z) \land (x\leq y \iff z+x \leq z+y)\)
by simp
\}
ultimately show IsAnOrdLoop(\(\mathbb{R}\),Add,ROrd) unfolding IsAnOrdLoop_def by auto
qed

The assumptions of the pmetric_space locale hold in the reals locale.

lemma (in reals) pmetric_space_valid: shows pmetric_space(\(\mathbb{R}\),Add,ROrd,dist,\(\mathbb{R}\))
unfolding pmetric_space_def pmetric_space_axioms_def loop1_def
using reals_loop dist_is_metric(h)
by blast

Some properties of the order relation on reals:

lemma (in reals) pos_is_lattice: shows
IsLinOrder(R,ROrd)
IsLinOrder(R+,ROrd ∩ R+×R+)
(ROrd ∩ 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 (ROrd ∩ R+×R+) ⊆ R+×R+ by auto
ultimately show (ROrd ∩ R+×R+) {is a lattice on} R+ using lin_is_latt
by simp
qed

We define the topology on reals as one consisting of the unions of open disks.

definition (in reals) RealTopology (τ_R)
where τ_R ≡ (⋃ c ∈ R. {disk(c,r). r ∈ R+})

Real numbers form a topological space with topology generated by open
disks.

theorem (in reals) reals_is_top: shows τ_R {is a topology} and ⋃ τ_R = R

proof -
  let B = ⋃ c ∈ R. {disk(c,r). r ∈ R+}
  have pmetric_space(R,Add, ROrd,dist,R) using pmetric_space_valid by simp
moreover have IsMeetSemilattice(R+,ROrd ∩ R+×R+)
  using pos_is_lattice(3) unfolding IsAlattice_def by simp
moreover from R_are_reals have R+≠0
  unfolding IsAmodelOfReals_def IsAnOrdField_def using ordring_one_is_pos
by auto
moreover have B = (⋃ c ∈ R. {disk(c,r). r ∈ R+}) by simp
moreover have τ_R = (⋃ A. A ∈ Pow(B)) unfolding RealTopology_def by simp
ultimately show τ_R {is a topology} and ⋃ τ_R = R using pmetric_space.pmetric_is_top
by auto
qed

56 Complex numbers

theory Complex_ZF imports func_ZF_1 OrderedField_ZF
The goal of this theory is to define complex numbers and prove that the Metamath complex numbers axioms hold.

56.1 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 $\mathbb{R} = R \times 0$. 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 $\{\langle x, 0 \rangle : x \in R \} \subseteq R \times R$. We call the set $\{\langle x, 0 \rangle : x \in R \}$ $\text{ComplexReals}(R, A)$ 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 $f : R \times \{0\} \rightarrow R, f(x, 0) = x$ which is defined under a name of $\text{SliceProjection}$ in $\text{func_ZF.thy}$. This defines a relation $r_1$ (called $\text{InducedRelation}(f, r)$, see $\text{func_ZF}$) on $R \times \{0\}$ such that $\langle x, 0 \rangle \langle y, 0 \rangle \in r_1$ iff $\langle x, y \rangle \in r$. This way we get what we call $\text{CplxROrder}(R, A, r)$. 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 $\text{complex0}$ 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(\text{fst}(a), \text{fst}(b))
An expression representing the imaginary part of the sum of two complex numbers.

**definition**

\[ \text{ImCxAdd}(R, A, a, b) \equiv A(\text{snd}(a), \text{snd}(b)) \]

The set (function) that is the binary operation that adds complex numbers.

**definition**

\[ \text{CplxAdd}(R, A) \equiv \{ (p, (\text{ReCxAdd}(R, A, \text{fst}(p), \text{snd}(p)), \text{ImCxAdd}(R, A, \text{fst}(p), \text{snd}(p))) \} \}\] \[ p \in (R \times R) \times (R \times R) \]

The expression representing the imaginary part of the product of complex numbers.

**definition**

\[ \text{ImCxMul}(R, A, M, a, b) \equiv A(M(\text{fst}(a), \text{snd}(b)), M(\text{snd}(a), \text{fst}(b))) \]

The expression representing the real part of the product of complex numbers.

**definition**

\[ \text{ReCxMul}(R, A, M, a, b) \equiv A(M(\text{fst}(a), \text{fst}(b)), \text{GroupInv}(R, A)(M(\text{snd}(a), \text{snd}(b)))) \]

The function (set) that represents the binary operation of multiplication of complex numbers.

**definition**

\[ \text{CplxMul}(R, A, M) \equiv \{ (p, (\text{ReCxMul}(R, A, M, \text{fst}(p), \text{snd}(p)), \text{ImCxMul}(R, A, M, \text{fst}(p), \text{snd}(p))) \} \}\] \[ p \in (R \times R) \times (R \times R) \]

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**

\[ \text{CplxROrder}(R, A, r) \equiv \text{InducedRelation}((\text{SliceProjection}(\text{ComplexReals}(R, A)), r)) \]

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 \equiv R \times R
fixes rone ($1_R$)
defines rone_def[simp]: $1_R \equiv \text{TheNeutralElement}(R,M)$

fixes rzero ($0_R$)
defines rzero_def[simp]: $0_R \equiv \text{TheNeutralElement}(R,A)$

fixes one (1)
defines one_def[simp]: $1 \equiv \langle 1_R, 0_R \rangle$

fixes zero (0)
defines zero_def[simp]: $0 \equiv \langle 0_R, 0_R \rangle$

fixes iunit (i)
defines iunit_def[simp]: $i \equiv \langle 0_R, 1_R \rangle$

fixes creal (R)
defines creal_def[simp]: $R \equiv \{ \langle r, 0_R \rangle. r \in R \}$

fixes rmul (infixl · 71)
defines rmul_def[simp]: $a \cdot b \equiv M(a,b)$

fixes radd (infixl + 69)
defines radd_def[simp]: $a + b \equiv A(a,b)$

fixes rneg (− _ 70)
defines rneg_def[simp]: $- a \equiv \text{GroupInv}(R,A)(a)$

fixes ca (infixl + 69)
defines ca_def[simp]: $a + b \equiv \text{CplxAdd}(R,A)(a,b)$

fixes cm (infixl · 71)
defines cm_def[simp]: $a \cdot b \equiv \text{CplxMul}(R,A,M)(a,b)$

fixes cdiv (infixl / 70)
defines cdiv_def[simp]: $a / b \equiv \bigcup \{ x \in C. b \cdot x = a \}$

fixes sub (infixl − 69)
defines sub_def[simp]: $a - b \equiv \bigcup \{ x \in C. b + x = a \}$

fixes cneg (− _ 95)
defines cneg_def[simp]: $- a \equiv 0 - a$

fixes lessr (infix < _ 68)
defines lessr_def[simp]: $a <_R b \equiv (a,b) \in \text{StrictVersion}(\text{CplxROrder}(R,A,r))$

fixes cpnf (+∞)
defines cpnf_def[simp]: $+\infty \equiv C$
fixes cmnf \((\infty)\)
defines cmnf_def[simp]: \(-\infty \equiv \{\mathbb{C}\}\)

fixes cxr \((R^*)\)
defines cxr_def[simp]: \(R^* \equiv R \cup \{+\infty, -\infty\}\)

fixes cxn \((N)\)
defines cxn_def[simp]:
\(N \equiv \bigcap \{n \in \text{Pow}(R). 1 \in N \land (\forall n. n \in N \implies n + 1 \in N)\}\)

fixes cltrrset \((<)\)
defines cltrrset_def[simp]:
\(< \equiv \text{StrictVersion(CplxROrder}(R,A,r)) \cap R \times R \cup \\
\{(-\infty, +\infty)\} \cup (R \times \{+\infty\}) \cup \{(-\infty) \times R\}\)

fixes cltrr \((\text{infix } < 68)\)
defines cltrr_def[simp]: \(a < b \equiv (a, b) \in <\)

fixes lsq \((\text{infix } \leq 68)\)
defines lsq_def[simp]: \(a \leq b \equiv \neg (b < a)\)

fixes two \((2)\)
defines two_def[simp]: \(2 \equiv 1 + 1\)

fixes three \((3)\)
defines three_def[simp]: \(3 \equiv 2 + 1\)

fixes four \((4)\)
defines four_def[simp]: \(4 \equiv 3 + 1\)

fixes five \((5)\)
defines five_def[simp]: \(5 \equiv 4 + 1\)

fixes six \((6)\)
defines six_def[simp]: \(6 \equiv 5 + 1\)

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]: \(9 \equiv 8 + 1\)
56.2 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,M,⟨a,b⟩,⟨c,d⟩) = a + c
- ImCxAdd(R,A,M,⟨a,b⟩,⟨c,d⟩) = b + d
- ImCxMul(R,A,M,⟨a,b⟩,⟨c,d⟩) = a·d + b·c
- ReCxMul(R,A,M,⟨a,b⟩,⟨c,d⟩) = a·c + (-b·d)

**proof**
  let z₁ = ⟨a,b⟩
  let z₂ = ⟨c,d⟩
  have ReCxAdd(R,A,z₁,z₂) ≡ A(fst(z₁),fst(z₂))
    by (rule ReCxAdd_def)
  moreover have ImCxAdd(R,A,z₁,z₂) ≡ A(snd(z₁),snd(z₂))
    by (rule ImCxAdd_def)
  moreover have

681
\[ \text{ImCxMul}(R,A,M,z_1,z_2) \equiv A(M<\text{fst}(z_1),\text{snd}(z_2)>,M<\text{snd}(z_1),\text{fst}(z_2)>) \]

by (rule ImCxMul_def)

moreover have
\[ \text{ReCxMul}(R,A,M,z_1,z_2) \equiv A(M<\text{fst}(z_1),\text{fst}(z_2)>,\text{GroupInv}(R,A)(M<\text{snd}(z_1),\text{snd}(z_2)>)) \]

by (rule ReCxMul_def)

ultimately show
\[ \text{ReCxAdd}(R,A,z_1,z_2) = a + c \]
\[ \text{ImCxAdd}(R,A,z_1,z_2) = b + d \]
\[ \text{ImCxMul}(R,A,M,z_1,z_2) = a \cdot d + b \cdot c \]
\[ \text{ReCxMul}(R,A,M,z_1,z_2) = a \cdot c + (-b \cdot d) \]

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
\[ \text{ReCxAdd}(R,A,z_1,z_2) \in R \]
\[ \text{ImCxAdd}(R,A,z_1,z_2) \in R \]
\[ \text{ImCxMul}(R,A,M,z_1,z_2) \in R \]
\[ \text{ReCxMul}(R,A,M,z_1,z_2) \in R \]

using cplx_mul_add_defs by auto

qed

Complex reals are complex. Recall the definition of \( \mathbb{R} \) in the complex0 locale.

**lemma** (in complex0) axresscn: shows \( \mathbb{R} \subseteq \mathbb{C} \)

using valid_cntxts group0.group0_2_L2 by auto

Complex 1 is not complex 0.
lemma (in complex0) axine0: shows $1 \neq 0$
proof -
  have IsAfield($R, A, M$) using valid_cntxts field0.Field_ZF_1_L1
    by simp
  then show $1 \neq 0$ using IsAfield_def by auto
qed

Complex addition is a complex valued binary operation on complex numbers.

lemma (in complex0) axaddopr: shows CplxAdd($R, A$): $\mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C}$
proof -
  have $\forall p \in \mathcal{C} \times \mathcal{C}$.
    $\langle \text{ReCxAdd}(R, A, \text{fst}(p), \text{snd}(p)), \text{ImCxAdd}(R, A, \text{fst}(p), \text{snd}(p)) \rangle \in \mathcal{C}$
    using cplx_mul_add_types by simp
  then have
    $\{ \langle p, \langle \text{ReCxAdd}(R, A, \text{fst}(p), \text{snd}(p)), \text{ImCxAdd}(R, A, \text{fst}(p), \text{snd}(p)) \rangle \rangle.
      p \in \mathcal{C} \times \mathcal{C} \rangle: \mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C}$
    by (rule ZF_fun_from_total)
  then show CplxAdd($R, A$): $\mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C}$ using CplxAdd_def by simp
qed

Complex multiplication is a complex valued binary operation on complex numbers.

lemma (in complex0) axmulopr: shows CplxMul($R, A, M$): $\mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C}$
proof -
  have $\forall p \in \mathcal{C} \times \mathcal{C}$.
    $\langle \text{ReCxMul}(R, A, M, \text{fst}(p), \text{snd}(p)), \text{ImCxMul}(R, A, M, \text{fst}(p), \text{snd}(p)) \rangle \in \mathcal{C}$
    using cplx_mul_add_types by simp
  then have
    $\{ \langle p, \langle \text{ReCxMul}(R, A, M, \text{fst}(p), \text{snd}(p)), \text{ImCxMul}(R, A, M, \text{fst}(p), \text{snd}(p)) \rangle \rangle.$
      $p \in \mathcal{C} \times \mathcal{C} \rangle: \mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C}$ by (rule ZF_fun_from_total)
  then show CplxMul($R, A, M$): $\mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C}$ 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: $a \in R$ $b \in R$ $c \in R$ $d \in R$
  shows
    $\langle a, b \rangle + \langle c, d \rangle = \langle a + c, b + d \rangle$
    $\langle a, b \rangle \cdot \langle c, d \rangle = \langle a \cdot c + (-b) \cdot d, a \cdot d + b \cdot c \rangle$
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 $S: \mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C}$ and $p \in \mathcal{C} \times \mathcal{C}$
    using axaddopr by auto
  moreover have
    $S = \{ \langle p, \langle \text{ReCxAdd}(R, A, \text{fst}(p), \text{snd}(p)), \text{ImCxAdd}(R, A, \text{fst}(p), \text{snd}(p)) \rangle \rangle.$
    $p \in \mathcal{C} \times \mathcal{C} \rangle$.
\( p \in C \times C \) using CplxAdd_def by simp
ultimately have \( S(p) = (\text{ReCxAdd}(R,A,\text{fst}(p),\text{snd}(p)), \text{ImCxAdd}(R,A,\text{fst}(p),\text{snd}(p))) \)
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 \( P : C \times C \rightarrow C \) and \( p \in C \times C \)
moreover have
\[ P = \{ (p, (\text{ReCxMul}(R,A,M,\text{fst}(p),\text{snd}(p)), \text{ImCxMul}(R,A,M,\text{fst}(p),\text{snd}(p))) \} \} \]
\( p \in C \times C \) using CplxMul_def by simp
ultimately have \( P(p) = (\text{ReCxMul}(R,A,M,\text{fst}(p),\text{snd}(p)), \text{ImCxMul}(R,A,M,\text{fst}(p),\text{snd}(p))) \)
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: \( a \in C \) \( b \in C \) \( c \in C \)
shows \( a \cdot (b + c) = a \cdot b + a \cdot c \)
proof -
  let \( a_r = \text{fst}(a) \)
  let \( a_i = \text{snd}(a) \)
  let \( b_r = \text{fst}(b) \)
  let \( b_i = \text{snd}(b) \)
  let \( c_r = \text{fst}(c) \)
  let \( c_i = \text{snd}(c) \)
from A1 have T:
\[a_r \in R \quad a_i \in R \quad b_r \in R \quad b_i \in R \quad c_r \in R \quad c_i \in R\]
\[b_r + c_r \in R \quad b_i + c_i \in R\]
\[a_r + b_r, (-a_i, b_i) \in R\]
\[a_r - b_r, (a_r, c_i) \in R\]
\[a_r - b_r, a_i \in R\]
\[a_r - b_r, (a_r, c_i) \in R\]
\[r \quad \text{using valid_cntxts ring0.Ring_ZF_1_L4 by auto}\]
\[\text{with A1 have } a \cdot (b + c) = \langle a_r \cdot (b_r + c_r), a_i \cdot (b_i + c_i) \rangle\]
\[\text{using cplx_mul_add_vals by auto}\]
\[\text{moreover from T have}\]
\[\langle a_r \cdot (b_r + c_r), a_i \cdot (b_i + c_i) \rangle = a_r \cdot b_r + (a_r - b_r, (a_r, c_i) + a_i \cdot c_r)\]
\[\text{using valid_cntxts ring0.Ring_ZF_2_L6 by auto}\]
\[\text{moreover from A1 T have}\]
\[\langle a_r \cdot b_r + (-a_i, b_i) + (a_r, c_i) + (-a_i, c_i)\rangle,\]
\[a_r \cdot b_i + a_i \cdot b_r + (a_r, c_i + a_i \cdot c_r) = a \cdot b + a \cdot c\]
\[\text{using cplx_mul_add_vals by auto}\]
\[\text{ultimately show } a \cdot (b + c) = a \cdot b + a \cdot c\]
\[\text{by simp}\]
\[\text{qed}\]

Complex addition is commutative.

**Lemma (in complex0) axaddcom:**

assumes \(a \in C \quad b \in C\)

shows \(a + b = b + a\)

**proof** -

\[
\text{let } a_r = \text{fst}(a) \\
\text{let } a_i = \text{snd}(a) \\
\text{let } b_r = \text{fst}(b) \\
\text{let } b_i = \text{snd}(b) \\
\text{let } c_r = \text{fst}(c) \\
\text{let } c_i = \text{snd}(c) \\
\text{from A1 have T:}\]
\[
\text{a_r \in R \quad a_i \in R \quad b_r \in R \quad b_i \in R \quad c_r \in R \quad c_i \in R} \\
\text{a_r + b_r \in R \quad a_i + b_i \in R} \\
\text{b_r + c_r \in R \quad b_i + c_i \in R} \\
\text{using valid_cntxts ring0.Ring_ZF_1_L4 by auto}\]
\[\text{with A1 have } a + b + c = \langle a_r + b_r + c_r, a_i + b_i + c_i \rangle\]
\[\text{using cplx_mul_add_vals by auto}\]

Complex addition is associative.

**Lemma (in complex0) axaddass:**

assumes \(A1: a \in C \quad b \in C \quad c \in C\)

shows \(a + b + c = a + (b + c)\)

**proof** -

\[
\text{let } a_r = \text{fst}(a) \\
\text{let } a_i = \text{snd}(a) \\
\text{let } b_r = \text{fst}(b) \\
\text{let } b_i = \text{snd}(b) \\
\text{let } c_r = \text{fst}(c) \\
\text{let } c_i = \text{snd}(c) \\
\text{from A1 have T:}\]
\[
\text{a_r \in R \quad a_i \in R \quad b_r \in R \quad b_i \in R \quad c_r \in R \quad c_i \in R} \\
\text{a_r + b_r \in R \quad a_i + b_i \in R} \\
\text{b_r + c_r \in R \quad b_i + c_i \in R} \\
\text{using valid_cntxts ring0.Ring_ZF_1_L4 by auto}\]
\[\text{with A1 have } a + b + c = \langle a_r + b_r + c_r, a_i + b_i + c_i \rangle\]
\[\text{using cplx_mul_add_vals by auto}\]
also from A1 T have \( a + (b + c) \)
using valid_cntxts 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: \( a \in \mathbb{C} \ b \in \mathbb{C} \ c \in \mathbb{C} \)
shows \( a \cdot b \cdot c = a \cdot (b \cdot c) \)
proof -
let \( a_r = \text{fst}(a) \)
let \( a_i = \text{snd}(a) \)
let \( b_r = \text{fst}(b) \)
let \( b_i = \text{snd}(b) \)
let \( c_r = \text{fst}(c) \)
let \( c_i = \text{snd}(c) \)
from A1 have T:
\( a_r \in \mathbb{R} \ a_i \in \mathbb{R} \ b_r \in \mathbb{R} \ b_i \in \mathbb{R} \ c_r \in \mathbb{R} \ c_i \in \mathbb{R} \)
\( a_r \cdot b_r + (-a_i \cdot b_i) \in \mathbb{R} \)
\( b_r \cdot c_r + (-b_i \cdot c_i) \in \mathbb{R} \)
\( b_r \cdot c_i + b_i \cdot c_r \in \mathbb{R} \)
using valid_cntxts ring0.Ring_ZF_1_L4 by auto
with A1 have \( a \cdot b \cdot c = \langle (a_r \cdot b_r + (-a_i \cdot b_i)) \cdot c_r + (-a_r \cdot b_i + a_i \cdot b_r) \cdot c_i, \( a_r \cdot b_r + (-a_i \cdot b_i)) \cdot c_i + (a_r \cdot b_i + a_i \cdot b_r) \cdot c_r \rangle \)
using cplx_mul_add_vals by auto
moreover from A1 T have
\( \langle a_r \cdot (b_r \cdot c_r + (-b_i \cdot c_i)) + (-a_r \cdot (b_i \cdot c_r + b_i \cdot c_i)), a_r \cdot (b_i \cdot c_r + b_i \cdot c_i) + a_i \cdot (b_r \cdot c_r + (-a_i \cdot c_i)) \rangle = a \cdot (b \cdot c) \)
using cplx_mul_add_vals by auto
moreover from T have
\( a_r \cdot (b_r \cdot c_r + (-b_i \cdot c_i)) = (-a_r \cdot (b_r \cdot c_r + b_r \cdot c_i) + \langle a_r \cdot b_r + (-a_i \cdot b_i)) \cdot c_r + (-a_r \cdot b_i + a_i \cdot b_r) \cdot c_i \rangle \)
and
\( a_r \cdot (b_r \cdot c_i + b_r \cdot c_r) = a_i \cdot (b_r \cdot c_r + (-b_i \cdot c_i)) = (a_r \cdot b_r + (-a_i \cdot b_i)) \cdot c_i + (a_r \cdot b_i + a_i \cdot b_r) \cdot c_r \)
using valid_cntxts ring0.Ring_ZF_2_L6 by auto
ultimately show \( a \cdot b \cdot c = a \cdot (b \cdot c) \)
by auto
qed

Complex 1 is real. This really means that the pair \( \langle 1, 0 \rangle \) is on the real axis.

lemma (in complex0) ax1re: shows \( 1 \in \mathbb{R} \)
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 \( i \cdot i + 1 = 0 \)
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 \( a \in \mathbb{C} \)
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 \( i \in \mathbb{C} \)
using valid_cntxts ring0.Ring_ZF_1_L3 by auto

All complex numbers have additive inverses.

Lemma (in complex0) axnegex: assumes A1: \( a \in \mathbb{C} \)
shows \( \exists x \in \mathbb{C}. a + x = 0 \)
proof -
let \( a_r = \text{fst}(a) \)
let \( a_i = \text{snd}(a) \)
let \( x = (-a_r, -a_i) \)
from A1 have T:
    \( a_r \in \mathbb{R} \quad a_i \in \mathbb{R} \)
    \( (-a_r) \in \mathbb{R} \quad (-a_i) \in \mathbb{R} \)
using valid_cntxts ring0.Ring_ZF_1_L3 by auto
then have \( x \in \mathbb{C} \) 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 \( \exists x \in \mathbb{C}. a + x = 0 \)
by auto
qed

A non-zero complex number has a multiplicative inverse.

Lemma (in complex0) axrecex: assumes A1: \( a \in \mathbb{C} \) and A2: \( a \neq 0 \)
shows \( \exists x \in \mathbb{C}. a \cdot x = 1 \)
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: \( a_r \in \mathbb{R} \quad a_i \in \mathbb{R} \) by auto
moreover from A1 A2 have \( a_r \neq 0_R \quad a_i \neq 0_R \)
by auto
ultimately have \( \exists c \in \mathbb{R}. m \cdot c = 1_R \)
using valid_cntxts field1.OrdField_ZF_1_L10 by auto

687
then obtain $c$ where I: $c \in \mathbb{R}$ and II: $m \cdot c = 1_R$

by auto

let $x = (a_r \cdot c, -a_i \cdot c)$

from T1 I have T2: $a_r \cdot c \in \mathbb{R}$ $(-a_i \cdot c) \in \mathbb{R}$

using valid_cntxts ring0.Ring_ZF_1_L4 ring0.Ring_ZF_1_L3

by auto

then have $x \in C$ by auto

moreover from A1 T1 T2 I II have $a \cdot x = 1$

using cplx_mul_add_vals valid_cntxts ring0.ring_rearr_3_elemA

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) ax1id: assumes A1: $a \in C$

shows $a \cdot 1 = a$

using assms valid_cntxts ring0.Ring_ZF_1_L2 cplx_mul_add_vals

ring0.Ring_ZF_1_L3 ring0.Ring_ZF_1_L6 by auto

A formula for sum of (complex) real numbers.

lemma (in complex0) sum_of_reals: assumes $a \in \mathbb{R}$ $b \in \mathbb{R}$

shows $a + b = \langle \text{fst}(a) + \text{fst}(b), 0_R \rangle$

using assms valid_cntxts ring0.Ring_ZF_1_L2 cplx_mul_add_vals

ring0.Ring_ZF_1_L3 by auto

The sum of real numbers is real.

lemma (in complex0) axaddrcl: assumes A1: $a \in \mathbb{R}$ $b \in \mathbb{R}$

shows $a + b \in \mathbb{R}$

using assms sum_of_reals valid_cntxts ring0.Ring_ZF_1_L4 by auto

The formula for the product of (complex) real numbers.

lemma (in complex0) prod_of_reals: assumes A1: $a \in \mathbb{R}$ $b \in \mathbb{R}$

shows $a \cdot b = \langle \text{fst}(a) \cdot \text{fst}(b), 0_R \rangle$

proof -

let $a_r = \text{fst}(a)$

let $b_r = \text{fst}(b)$

from A1 have T:

$a_r \in \mathbb{R}$ $b_r \in \mathbb{R}$ $0_R \in \mathbb{R}$ $a_r \cdot b_r \in \mathbb{R}$

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 \( a \in R \) \( b \in R \)
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 \( \exists x \in R. \ a + x = 0 \)
proof -
let \( a_r = \text{fst}(a) \)
let \( x = (-a_r,0_R) \)
from \( A1 \) have \( T: \)
\( a_r \in R \) \(-a_r \in R \) \( 0_R \in R \)
using valid_cntxts ring0.Ring_ZF_1_L3 ring0.Ring_ZF_1_L2
by auto
then have \( x \in R \) 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 \( \exists x \in R. \ a + x = 0 \) by auto
qed

Each nonzero real number has a real inverse

lemma (in complex0) axrrecex:
assumes \( A1: a \in R \ a \neq 0 \)
shows \( \exists x \in R. \ a \cdot x = 1 \)
proof -
let \( R_0 = R - \{0_R\} \)
let \( a_r = \text{fst}(a) \)
let \( y = \text{GroupInv}(R_0,\text{restrict}(M,R_0 \times R_0))(a_r) \)
from \( A1 \) have \( \langle y,0_R \rangle \in R \) 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 \( \exists x \in R. \ a \cdot x = 1 \) by auto
qed

Our \( R \) symbol is the real axis on the complex plane.

lemma (in complex0) real_means_real_axis: shows \( R = \text{ComplexReals}(R,A) \)
using ComplexReals_def by auto

The \( \text{CplxROrder} \) thing is a relation on the complex reals.

lemma (in complex0) cplx_ord_on_cplx_reals:
shows \( \text{CplxROrder}(R,A,r) \subseteq R \times R \)
using ComplexReals_def slice_proj_bij real_means_real_axis
CplxROrder_def InducedRelation_def by auto

The strict version of the complex relation is a relation on complex reals.

689
lemma (in complex0) cplx_strict_ord_on_cplx_reals:  
  shows StrictVersion(CplxROrder(R,A,r)) ⊆ R×R  
  using cplx Ord on cplx_reals strict_ver_rel by simp

The CplxROrder thing is a relation on the complex reals. Here this is formulated as a statement that in complex0 context a < b implies that a, b are complex reals

lemma (in complex0) strict_cplx_ord_type: assumes a <R b  
  shows a∈R b∈R  
  using assms CplxROrder_def def_of_strict_ver InducedRelation_def  
  slice_proj_bij ComplexReals_def 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 complex0 context r denotes the (non-strict) order relation on the underlying model of real numbers.

lemma (in complex0) def_of_real_axis_order: shows ⟨x,0⟩ <R ⟨y,0⟩ ←→ ⟨x,y⟩ ∈ r ∧ x≠y
proof
  let f = SliceProjection(ComplexReals(R,A))  
  assume A1: ⟨x,0⟩ <R ⟨y,0⟩  
  then have ( f(x,0R), f(y,0R) ) ∈ r ∧ x ≠ y  
    using CplxROrder_def def_of_strict_ver def_of_ind_relA  
    by simp
  moreover from A1 have ⟨x,0⟩ ∈ R ⟨y,0⟩ ∈ R  
    using strict_cplx_ord_type by auto
  ultimately show ⟨x,y⟩ ∈ r ∧ x≠y  
    using slice_proj_bij ComplexReals_def by simp
next assume A1: ⟨x,y⟩ ∈ r ∧ x≠y
  let f = SliceProjection(ComplexReals(R,A))
  have f : R → R  
    using ComplexReals_def slice_proj_bij real_means_real_axis  
    by simp
  moreover from A1 have T: ⟨x,0⟩ ∈ R ⟨y,0⟩ ∈ R  
    using valid_cntxts ring1.OrdRing_ZF_1.L3 by auto
  moreover from A1 T have ( f(x,0R), f(y,0R) ) ∈ r  
    using slice_proj_bij ComplexReals_def by simp
  ultimately have ⟨x,0⟩, ⟨y,0⟩ ∈ InducedRelation(f,r)  
    using def_of_ind_relB by simp
  with A1 show ⟨x,0⟩ <R ⟨y,0⟩  
    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 \( f = \text{SliceProjection}(\text{ComplexReals}(R,A)) \)
  have \( f \in \text{ord_iso}(R,\text{CplxROrder}(R,A,r),R,r) \)
    using \text{ComplexReals_def\ slice_proj_bij real_means_real_axis}
               \text{bij_is_ord_iso CplxROrder_def by simp}
  moreover have \( \text{CplxROrder}(R,A,r) \subseteq R \times R \)
    using \text{cplx_ord_on_cplx_reals by simp}
  moreover have \( I: \text{antisym}(r) \ \text{r \{is total on\} R \ trans(r)} \)
    using \text{valid_cnttxts ring1.OrdRing_ZF_1_L1 IsAnOrdRing_def}
               \text{IsLinOrder_def by auto}
  ultimately show
    \( \text{antisym}(\text{CplxROrder}(R,A,r)) \)
    \( \text{trans}(\text{CplxROrder}(R,A,r)) \)
    \( \text{CplxROrder}(R,A,r) \{\text{is total on}\} R \)
    using \text{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.

\textbf{lemma (in complex0) cplx_strict_ord_trich:}
\textbf{assumes} \( a \in R \ b \in R \)
\textbf{shows} \( \text{Exactly\_1\_of\_3\_holds}(a < R b, a=b, b < R a) \)
\textbf{using} \text{assms cplx_ord_antsym_trans_tot strict_ans_tot_trich}
\textbf{by simp}

The strict order on the complex reals is kind of antisymmetric.

\textbf{lemma (in complex0) pre_axlttri: assumes} \( A1: a \in R \ b \in R \)
\textbf{shows} \( a < R b \iff \neg(a=b \lor b < R a) \)
\textbf{proof -}
  \textbf{from} \( A1 \) \textbf{have} \( \text{Exactly\_1\_of\_3\_holds}(a < R b, a=b, b < R a) \)
    \textbf{by (rule cplx_strict_ord_trich)}
  \textbf{then show} \( a < R b \iff \neg(a=b \lor b < R a) \)
    \textbf{by (rule Fol1_L8A)}
qed

The strict order on complex reals is transitive.

\textbf{lemma (in complex0) cplx_strict_ord_trans:}
\textbf{shows} \( \text{trans}(\text{StrictVersion}(\text{CplxROrder}(R,A,r))) \)
\textbf{using} \text{cplx_ord_antsym_trans_tot strict_of_transB by simp}

The strict order on complex reals is transitive - the explicit version of \text{cplx_strict_ord_trans}.

\textbf{lemma (in complex0) pre_axlttrn:}
\textbf{assumes} \( A1: a < R b \ b < R c \)
\textbf{shows} \( a < R c \)

691
proof - 
let \( s = \text{StrictVersion}(\text{CplxROrder}(R,A,r)) \) 
from A1 have 
\( \text{trans}(a) \quad (a,b) \in s \land (b,c) \in s \) 
using \( \text{cplx_strict_ord_trans} \) by auto 
then have \( (a,c) \in s \) 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: \( c \in R \)
shows \( c+a <_R c+b \)
proof - 
from A1 have T: \( a \in R \land b \in R \) using \( \text{strict_cplx_ord_type} \) 
by auto 
with A1 A2 show \( c+a <_R c+b \) 
using \( \text{def_of_real_axis_order} \) valid_cntxts 
\( \text{group3.group_strict_ord_transl_inv} \) \( \text{sum_of_reals} \) 
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 \( 0 <_R a \cdot b \)
proof - 
from A1 have T: \( a \in R \land b \in R \) using \( \text{strict_cplx_ord_type} \) 
by auto 
with A1 show \( 0 <_R a \cdot b \) 
using \( \text{def_of_real_axis_order} \) valid_cntxts \( \text{field1.pos_mul_closed} \) 
\( \text{def_of_real_axis_order} \) \( \text{prod_of_reals} \) 
by auto 
qed 

The order on complex reals is linear and complete.

lemma (in complex0) cmplx_reals_ord_lin_compl: shows 
\( \text{CplxROrder}(R,A,r) \{\text{is complete}\} \) 
\( \text{IsLinOrder}(R,\text{CplxROrder}(R,A,r)) \)
proof - 
have \( \text{SliceProjection}(R) \subseteq \text{bij}(R,R) \) 
using \( \text{slice_proj_bij} \) \( \text{ComplexReals_def} \) \( \text{real_means_real_axis} \) 
by simp 
moreover have \( r \subseteq R \times R \) using valid_cntxts \( \text{ring1.OrdRing_ZF_1_L1} \) 
\( \text{IsAnOrdRing_def} \) by simp 
moreover from \( \text{R_are_reals} \) have 
\( r \{\text{is complete}\} \) and \( \text{IsLinOrder}(R,r) \) 
using \( \text{IsAmodelOfReals_def} \) valid_cntxts \( \text{ring1.OrdRing_ZF_1_L1} \) 
\( \text{IsAnOrdRing_def} \) by auto 
ultimately show
The property of the strict order on complex reals that corresponds to completeness.

lemma (in complex0) pre_axsup: assumes A1: X ⊆ R  X ≠ 0 and A2: ∃x∈R. ∀y∈X. y <R x shows ∃x∈R. (∀y∈X. ¬(x <R y)) ∧ (∀y∈R. (y <R x → (∃z∈X. y <R z))) proof - let s = StrictVersion(CplxROrder(R,A,r)) have CplxROrder(R,A,r) ⊆ R×R IsLinOrder(R,CplxROrder(R,A,r)) CplxROrder(R,A,r) {is complete} using CplxROrder_def real_means_real_axis ind_rel_pres_compl ind_rel_pres_lin by auto moreover note A1 moreover have s = StrictVersion(CplxROrder(R,A,r)) by simp moreover from A2 have ∃u∈R. ∀y∈X. ⟨y,u⟩ ∈ s by simp ultimately have ∃x∈R. ( ∀y∈X. (x,y) /∈ s ) ∧ (∀y∈R. ⟨y,x⟩ ∈ s → (∃z∈X. ⟨y,z⟩ ∈ s)) by (rule strict_of_compl) then show (∃x∈R. ( ∀y∈X. ¬(x <R y) ) ∧ (∀y∈R. (y <R x → (∃z∈X. y <R z))))) by simp qed

end

57  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\textsubscript{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 $y \notin \overline{U_x}$. 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 $y \notin \overline{U}$" (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 $\forall \epsilon \exists \delta \cdots$ 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.

57.1 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_cl_open_sep:**

assumes $T \{\text{is } T_2\}$ and $x \in \bigcup T \land y \in \bigcup T \land x \neq y$

shows $\exists U \in T. (x \in U \land y \notin \overline{U})$

**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 $U \land V \in T \land x \in U \land y \in V \land U \cap V = 0$

by auto

then have $U \in T \land x \in U \land \forall y. y \in V \land \overline{U} \cap V = 0$

using disj_open_cl_disj by auto

thus $\exists U \in T. (x \in U \land y \notin \overline{U})$ 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 $A1: T \{\text{is } T_2\}$ and $A2: K \{\text{is compact in } T\}$

shows $K \{\text{is closed in } T\}$

**proof**

let $X = \bigcup T$

have $\forall y \in X - K. \exists U \in T. y \in U \land U \subseteq X - K$

**proof**

{ fix $y$ assume $y \in X \land y \notin K$

have $\exists U \in T. y \in U \land U \subseteq X - K$}

694
proof -
let \( B = \bigcup_{x \in K} \{ V \in T. x \in V \land y \notin \text{cl}(V) \} \)
have \( I : B \in \text{Pow}(T) \)  \( \text{FinPow}(B) \subseteq \text{Pow}(B) \)
  using \( \text{FinPow_def} \) by auto
from \( <K \{ \text{is compact in} \} T> \)  \( <y \in X> \)  \( <y \notin K> \) have
  \( \forall x \in K. x \in X \land y \in X \land x \neq y \)
  using \( \text{IsCompact_def} \) by auto
with \( <T \{ \text{is T}_2 \}> \) have \( \forall x \in K. \{ V \in T. x \in V \land y \notin \text{cl}(V) \} \neq 0 \)
  using \( \text{T2_cl_open_sep} \) by auto
hence \( K \subseteq \bigcup B \) by blast
with \( <K \{ \text{is compact in} \} T> \) I have
  \( \exists N \in \text{FinPow}(B). K \subseteq \bigcup N \)
  using \( \text{IsCompact_def} \) by auto
then obtain \( N \) where \( N \in \text{FinPow}(B) \)  \( K \subseteq \bigcup N \)
  by auto
with I have \( N \subseteq B \) by auto
hence \( \forall V \in N. V \in B \) by auto
let \( M = \{ \text{cl}(V). V \in N \} \)
let \( C = \{ D \in \text{Pow}(X). D \{ \text{is closed in} \} T \} \)
from \( <N \in \text{FinPow}(B). K \subseteq \bigcup N \) have \( \forall V \in B. \text{cl}(V) \in C \)
  \( N \in \text{FinPow}(B) \)
  using \( \text{cl_is_closed} \) \( \text{IsClosed_def} \) by auto
then have \( M \in \text{FinPow}(C) \) by (rule \( \text{fin_image_fin} \))
then have \( X - \bigcup M \in T \) using \( \text{fin_union_cl_is_cl} \) \( \text{IsClosed_def} \)
by simp
moreover from \( <y \in X> \)  \( <y \notin K> \)  \( \forall V \in N. V \subseteq B \) have
  \( y \in X - \bigcup M \) by simp
moreover have \( X - \bigcup M \subseteq X - K \)
proof -
  from \( \forall V \in N. V \subseteq B \) have \( \bigcup N \subseteq \bigcup M \) using \( \text{cl_contains_set} \) by auto
  with \( <K \subseteq \bigcup N> \) show \( X - \bigcup M \subseteq X - K \) by auto
qed
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

58 Topology 2

theory Topology_ZF_2 imports Topology_ZF_1 func1 Fol1

695
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.

58.1 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**

\[ \text{IsContinuous}(\tau_1, \tau_2, f) \equiv (\forall U \in \tau_2. f-(U) \in \tau_1) \]

A trivial example of a continuous function - identity is continuous.

**lemma** id_cont: shows IsContinuous($\tau, \tau, \text{id(}\bigcup \tau\text{)}$)

**proof** -

{ fix U assume U∈τ
  then have id($\bigcup \tau$)-(U) = U using vimage_id_same by auto
  with $\langle U\in\tau\rangle$ have id($\bigcup \tau$)-(U) ∈ τ by simp
  } then show IsContinuous($\tau, \tau, \text{id(}\bigcup \tau\text{)}$) using IsContinuous_def
  by simp

**qed**

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 $f : X_1 \to X_2$, where $X_i$ is defined as $\bigcup \tau_i$ for $i = 1, 2$. We also define notation $c_1(A)$ and $c_2(A)$ 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 $X_1 \text{ def [simp]}$: $X_1 \equiv \bigcup \tau_1$

fixes $X_2$
defines $X_2 \text{ def [simp]}$: $X_2 \equiv \bigcup \tau_2$

fixes $f$
assumes $\text{fmapAssum}: f: X_1 \rightarrow X_2$

fixes $\text{isContinuous} (\_ \{\text{is continuous}\} [50] 50)$
defines $\text{isContinuous_def [simp]}$: $g \{\text{is continuous}\} \equiv \text{IsContinuous}(\tau_1, \tau_2, g)$

fixes $\text{cl}_1$
defines $\text{cl}_1 \text{ def [simp]}$: $\text{cl}_1(A) \equiv \text{Closure}(A, \tau_1)$

fixes $\text{cl}_2$
defines $\text{cl}_2 \text{ def [simp]}$: $\text{cl}_2(A) \equiv \text{Closure}(A, \tau_2)$

First we show that theorems proven in locale $\text{topology0}$ are valid when applied to topologies $\tau_1$ and $\tau_2$.

**lemma** (in $\text{two_top_spaces0}$) $\text{topol_cntxs_valid}$:
shows $\text{topology0}(\tau_1)$ and $\text{topology0}(\tau_2)$
using $\text{tau1_is_top}$ $\text{tau2_is_top}$ $\text{topology0_def}$ by auto

For continuous functions the inverse image of a closed set is closed.

**lemma** (in $\text{two_top_spaces0}$) $\text{TopZF_2_1_L1}$:
assumes $A1: f \{\text{is continuous}\}$ and $A2: D \{\text{is closed in}\} \tau_2$
shows $f-(D) \{\text{is closed in}\} \tau_1$
proof -
from $\text{fmapAssum}$ have $f-(D) \subseteq X_1$ using $\text{func1_1_L3}$ by simp
moreover from $\text{fmapAssum}$ have $f-(X_2 - D) = X_1 - f-(D)$
using $\text{Pi_iff}$ $\text{function_vimage_Diff}$ $\text{func1_1_L4}$ by auto
ultimately have $X_1 - f-(X_2 - D) = f-(D)$ by auto
moreover from $A1$ $A2$ have $(X_1 - f-(X_2 - D)) \{\text{is closed in}\} \tau_1$
using $\text{IsClosed_def}$ $\text{IsContinuous_def}$ $\text{topol_cntxs_valid}$ $\text{topology0.Top_3_L9}$
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 $\text{two_top_spaces0}$) $\text{TopZF_2_1_L2}$:
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 $f(\text{cl}_1(A)) \subseteq \text{cl}_2(f(A))$
proof -
from $\text{fmapAssum}$ have $f(A) \subseteq \text{cl}_2(f(A))$
using $\text{func1_1_L6}$ $\text{topol_cntxs_valid}$ $\text{topology0.cl_contains_set}$
by simp

697
with fmapAssum have \( f^{-1}(f(A)) \subseteq f^{-1}(\text{cl}_2(f(A))) \)
  by auto
moreover from fmapAssum A2 have \( A \subseteq f^{-1}(f(A)) \)
  using func1_1_L9 by simp
ultimately have \( A \subseteq f^{-1}(\text{cl}_2(f(A))) \)
  by auto
with fmapAssum A1 have \( f(\text{cl}_1(A)) \subseteq f^{-1}(\text{cl}_2(f(A))) \)
  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 \( f(f^{-1}(\text{cl}_2(f(A)))) \subseteq \text{cl}_2(f(A)) \)
  using fun_is_function function_image_vimage by simp
ultimately show \( f(\text{cl}_1(A)) \subseteq \text{cl}_2(f(A)) \)
  by auto
qed

If \( f(A) \subseteq \overline{f(A)} \) (the image of the closure is contained in the closure of the image), then \( f^{-1}(B) \subseteq f^{-1} \left( \overline{B} \right) \) (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: \( \forall A. \ ( A \subseteq X_1 \implies f(\text{cl}_1(A)) \subseteq \text{cl}_2(f(A))) \)
shows \( \forall B. \ ( B \subseteq X_2 \implies \text{cl}_1(f^{-1}(B)) \subseteq f^{-1}(\text{cl}_2(B)) ) \)
proof -
\begin{align*}
\{ \ & \text{fix } B \ \text{assume } B \subseteq X_2 \\
\ & \text{from fmapAssum A1 have } f(\text{cl}_1(f^{-1}(B))) \subseteq \text{cl}_2(f(f^{-1}(B))) \\
\ & \text{using func1_1_L3 by simp} \\
\ & \text{moreover from fmapAssum have } \text{cl}_2(f(f^{-1}(B))) \subseteq \text{cl}_2(B) \\
\ & \text{using fun_is_function function_image_vimage func1_1_L6} \\
\ & \text{topol_cntxs_valid topology0.top_closure_mono} \\
\ & \text{by simp} \\
\ & \text{ultimately have } f^{-1}(f(\text{cl}_1(f^{-1}(B)))) \subseteq f^{-1}(\text{cl}_2(B)) \\
\ & \text{using fmapAssum fun_is_function by auto} \\
\ & \text{moreover from fmapAssum have } \text{cl}_1(f^{-1}(B)) \subseteq f^{-1}(f(\text{cl}_1(f^{-1}(B)))) \\
\ & \text{using func1_1_L3 func1_1_L9 IsClosed_def} \\
\ & \text{topol_cntxs_valid topology0.cl_is_closed by simp} \\
\ & \text{ultimately show } \text{cl}_1(f^{-1}(B)) \subseteq f^{-1}(\text{cl}_2(B)) \text{ by auto} \\
\} \text{ then show thesis by simp} \\
\end{align*}
qed

If \( \overline{f^{-1}(B)} \subseteq \overline{f^{-1}(B)} \) (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 Top_ZF_2_1_L1, Top_ZF_2_1_L2 and Top_ZF_2_1_L3 showing equivalence of four definitions of continuity.

lemma (in two_top_spaces0) Top_ZF_2_1_L4:
assumes A1: \( \forall B. \ ( B \subseteq X_2 \implies \text{cl}_1(f^{-1}(B)) \subseteq f^{-1}(\text{cl}_2(B)) ) \)
shows \( f \text{ is continuous} \)
proof -
\begin{align*}
\{ \ & \text{fix } U \ \text{assume } U \in \tau_2 \\
\end{align*}

698
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 \(\text{cl}_1(f-(X_2 - U)) \subseteq f-(\text{cl}_2(X_2 - U))\) by auto
ultimately have \(\text{cl}_1(f-(X_2 - U)) = f-(X_2 - U)\) using topol_cntxs_valid topology0.Top_3_L8 by simp
moreover from fmapAssum have \(f-(X_2 - U) \subseteq \text{cl}_1(f-(X_2 - U))\) using func1_1_L3 topol_cntxs_valid topology0.cl_contains_set by simp
ultimately have \(f-(X_2 - U)\) \{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 show \(f\) \{is continuous\} using IsContinuous_def by simp
qed

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 -
\{ fix \(V\) assume A3: \(V \in \tau_2\)
with A1 obtain \(A\) where \(A \subseteq B\ \ V = \bigcup A\)
using IsAbaseFor_def by auto
with A2 have \(\{f-(U). \ U \in A\} \subseteq \tau_1\) by auto
with tau1_is_top have \(\bigcup \{f-(U). \ U \in A\} \in \tau_1\)
using IsATopology_def by simp
moreover from A A2 \(\forall U \in \bigcup A\) have \(f-(V) = \bigcup \{f-(U). \ U \in A\}\)
by auto
ultimately have \(f-(V) \in \tau_1\) by simp
\}
then show \(f\) \{is continuous\} using IsContinuous_def by simp
qed

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) TopZF_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\inC
  { 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 \neq 0 by (rule func1_1_L13)
moreover from \( \langle U \in C \rangle \) obtain A where
A \in FinPow(B) and U = \bigcap A
by auto
  ultimately have \bigcap A \neq 0 by simp
  then have A \neq 0 by (rule inter_nempty_nempty)
  then have \{f^{-}(W). W \in A\} \neq 0 by simp
moreover from A2 \( \langle A \in FinPow(B) \rangle \) have \{f^{-}(W). W \in A\} \in 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 \( \langle A \in FinPow(B) \rangle \) 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 \( \langle A \neq 0 \rangle \) \( \langle U = \bigcap A \rangle \) have f^{-}(U) = \bigcap \{f^{-}(W). W \in A\}
using func1_1_L12 by simp
  ultimately have f^{-}(U) \in \tau_1 by simp }
ultimately show \( f^{-}(U) \in \tau_1 \) by blast
qed
ultimately show f {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:
  assumes A1: \( B \) {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 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 \( \langle E \in Pow(B) \rangle \) have \{f(E). E \in E\} \in 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

700
A composition of two continuous functions is continuous.

**lemma comp_cont**: assumes \( \text{IsContinuous}(T,S,f) \) and \( \text{IsContinuous}(S,R,g) \) shows \( \text{IsContinuous}(T,R,g \circ f) \) using assms IsContinuous_def vimage_comp by simp

A composition of three continuous functions is continuous.

**lemma comp_cont3**: assumes \( \text{IsContinuous}(T,S,f) \) and \( \text{IsContinuous}(S,R,g) \) and \( \text{IsContinuous}(R,P,h) \) shows \( \text{IsContinuous}(T,P,h \circ g \circ f) \) using assms IsContinuous_def vimage_comp by simp

### 58.2 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.

**definition IsAhomeomorphism(T,S,f) ≡** 
\[
f \in \text{bij}(\bigcup T, \bigcup S) \land \text{IsContinuous}(T,S,f) \land \text{IsContinuous}(S,T,\text{converse}(f))
\]

Inverse (converse) of a homeomorphism is a homeomorphism.

**lemma homeo_inv**: assumes \( \text{IsAhomeomorphism}(T,S,f) \) shows \( \text{IsAhomeomorphism}(S,T,\text{converse}(f)) \) using assms IsAhomeomorphism_def bij_converse_bij bij_converse_converse by auto

Homeomorphisms are open maps.

**lemma homeo_open**: assumes \( \text{IsAhomeomorphism}(T,S,f) \) and \( U \subseteq T \) shows \( f(U) \subseteq S \) 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 \( f \in \text{bij}(\bigcup T, \bigcup S) \) and \( \text{IsContinuous}(T,S,f) \) and \( \forall U \subseteq T. \ f(U) \subseteq S \) shows \( \text{IsAhomeomorphism}(T,S,f) \) 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 \( A1: f \in \text{bij}(X_1,X_2) \) and \( A2: B \{\text{is a base for}\} \tau_1 \) and \( A3: C \{\text{is a base for}\} \tau_2 \) and
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 I have f-(C) ∈ τ₁ using bij_def inj_vimage_image base_sets_open by auto
      } hence ∀ C ∈ {f(B). B ∈ B}. f-(C) ∈ τ₁ by auto
      with A3 show thesis by (rule Top_ZF_2_1_L5)
  qed
  moreover from A3 have ∀ B ∈ B. f(B) ∈ τ₂ using base_sets_open by auto
  with A2 have ∀ U ∈ τ₁. f(U) ∈ τ₂ 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 ⊆ T and A2: IsAhomeomorphism(T,S,f)
shows f(Interior(A,T)) = Interior(f(A),S)
proof -
  let A = {U ∈ T. U ⊆ A}
  have I: {f(U). U ⊆ A} = {V ∈ S. V ⊆ f(A)}
  proof
    from A2 show {f(U). U ⊆ A} ⊆ {V ∈ S. V ⊆ f(A)}
      using homeo_open by auto
    { fix V assume V ∈ {V ∈ S. V ⊆ f(A)}
      hence ∀ V ∈ S. V ⊆ f(A) by auto
      let U = f-(V)
      from II have U ⊆ f-(f(A)) by auto
      moreover from assms have f-(f(A)) = A
        using IsAhomeomorphism_def bij_def inj_vimage_image by auto
      moreover from A2 ∈ S- have U ∈ T
        using IsAhomeomorphism_def IsContinuous_def by simp
      moreover from ∈ S- have V ⊆ U by auto
    }
with A2 have \( V = f(U) \)

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 \text{Interior_def} by simp

also from A2 have \( = \bigcup \{ f(U). U \in A \} \)

using IsAhomeomorphism_def bij_def inj_def image_of_Union by auto

also from I have \( = \text{Interior}(f(A),S) \) unfolding \text{Interior_def} by simp

finally show thesis by simp

qed

58.3 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 
\( \{ \{ f(x). x \in U \}. U \in T \} \) \{ is a topology \} and 
\( (\bigcup \{ f(U). U \in T \}) = Y \) and 
IsAhomeomorphism(T, \{ f(U). 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 \( S = \{ f(U). U \in T \} \)
\( \{ \) fix \( M \) assume \( M \in \text{Pow}(S) \)
let \( M_T = \{ f-(V). V \in M \} \)
have \( M_T \subseteq T \)
proof
\( \) fix \( W \) assume \( W \in M_T \)
then obtain \( V \) where \( V \in M \) and \( I: W = f-(V) \) by auto
with \( \langle M \in \text{Pow}(S) \rangle \) have \( V \in S \) by auto
then obtain \( U \) where \( U \in T \) and \( V = f(U) \) by auto
with \( I \) have \( W = f-(f(U)) \) by simp
with \( \langle f \in \text{inj}(\bigcup T,Y) \rangle \) \( \langle U \in T \rangle \) have \( W = U \) using inj_vimage_image
by blast
\( \) with \( \langle U \in T \rangle \) show \( W \in T \) by simp
qed
with \( A1 \) have \( (\bigcup M_T) \in T \) using IsATopology_def by simp
hence \( f(\bigcup M_T) \in S \) by auto
moreover have \( f(\bigcup M_T) = \bigcup M \)
proof -
from \( \langle f: \bigcup T \to Y \rangle \) \( M_T \subseteq T \) have \( f(\bigcup M_T) = \bigcup \{ f(U). U \in M_T \} \)
using image_of_Union by auto
moreover have \( \{ f(U). U \in M_T \} = M \)
proof -
  from $<f: \bigcup T \to Y>$ have $\forall U \in T. \ f(U) \subseteq Y$ using func1_1_L6 by simp
  with $<M \in \text{Pow}(S)>$ have $M \subseteq \text{Pow}(Y)$ by auto
  with A2 show $\{f(U). \ U \in \mathcal{M}\} = M$ using bij_def surj_subsets by auto
  qed
ultimately show $f(\bigcup M_T) = \bigcup M$ by simp
  qed
ultimately have $\bigcup M \in S$ by auto
} then have $\forall M \in \text{Pow}(S). \ \bigcup M \in S$ by auto
moreover
{ fix $U \ V$ assume $U \in S \ V \in S$
  then obtain $U_T \ V_T$ where $U_T \in T \ \ V_T \in T$
  I: $U = f(U_T) \ \ V = f(V_T)$
  by auto
  with $A1$ have $U_T \cap V_T \in T$ using IsATopology_def by simp
  hence $f(U_T \cap V_T) \in S$ by auto
  moreover have $f(U_T \cap V_T) = U \cap V$
proof -
  from $<U_T \in T \ \ V_T \in T>$ have $U_T \subseteq \bigcup T \ \ V_T \subseteq \bigcup T$
  using bij_def by auto
  with $<f \in \text{inj}(\bigcup T, Y)> \ I$ show $f(U_T \cap V_T) = U \cap V$ using inj_image_inter
  by simp
  qed
ultimately have $U \cap V \in S$ by simp
} then have $\forall U \in S. \ \forall V \in S. \ U \cap V \in S$ by auto
ultimately show $S$ {is a topology} using IsATopology_def by simp
moreover from $<f: \bigcup T \to Y>$ have $\forall U \in T. \ f(U) = \{f(x). x \in U\}$
  using func_imagedef by blast
ultimately show $\{ \{f(x). x \in U\}. \ U \in T\}$ {is a topology} by simp
show $\bigcup S = Y$
proof
  from $<f: \bigcup T \to Y>$ have $\forall U \in T. \ f(U) \subseteq Y$ using func1_1_L6 by simp
  thus $\bigcup S \subseteq Y$ by auto
  from A1 have $f(\bigcup T) \subseteq \bigcup S$ using IsATopology_def by auto
  with A2 show $Y \subseteq \bigcup S$ using bij_def surj_range_image_domain
  by auto
  qed
show IsAhomeomorphism($T, S, f$)
proof -
  from A2 $\bigcup S = Y$ have $f \in \text{bij}(\bigcup T, \bigcup S)$ by simp
  moreover have IsContinuous($T, S, f$)
proof -
  { fix $V$ assume $V \in S$
    then obtain $U$ where $U \in T$ and $V = f(U)$ by auto
    hence $U \subseteq \bigcup T$ and $f^{-1}(V) = f^{-1}(f(U))$ by auto
    with $<f \in \text{inj}(\bigcup T, Y)> \ <U \in T> \ have \ f^{-1}(V) \in T$ using inj_vimage_image
  }
58.4 Partial functions and continuity

Suppose we have two topologies \( \tau_1, \tau_2 \) on sets \( X_1 = \bigcup \tau_i, i = 1, 2 \). 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 \( A1 : A \subseteq X_1 \) and \( A2 : f \{\text{is continuous}\} \)
- Shows \( \text{IsContinuous}(\tau_1 \{\text{restricted to} A\}, \tau_2, \text{restrict}(f,A)) \)

**Proof -**

- Let \( g = \text{restrict}(f,A) \)
  - Fix \( U \) assume \( U \in \tau_2 \)
    - With \( A2 \) have \( f-(U) \in \tau_1 \) using \( \text{IsContinuous_def} \) by simp
    - Moreover from \( A1 \) have \( g-(U) = f-(U) \cap A \)
    - Using \( \text{fmapAssum func1_2_L1} \) by simp
    - Ultimately have \( g-(U) \in (\tau_1 \{\text{restricted to} A\}) \)
    - Using \( \text{RestrictedTo_def} \) by auto
- Then show thesis using \( \text{IsContinuous_def} \) by simp

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 \( A1 : f \{\text{is continuous}\} \)
- Shows \( \text{IsContinuous}(\tau_1, \tau_2 \{\text{restricted to} f(X_1),f\}) \)

**Proof -**

- Have \( \forall U \in \tau_2 \{\text{restricted to} f(X_1)\}. f-(U) \in \tau_1 \)
- Fix \( U \) assume \( U \in \tau_2 \{\text{restricted to} f(X_1)\} \)
  - Then obtain \( V \) where \( V \in \tau_2 \) and \( U = V \cap f(X_1) \)
  - Using \( \text{RestrictedTo_def} \) by auto
  - With \( A1 \) show \( f-(U) \in \tau_1 \)
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 ⊆ X_1 and A2: f {is continuous} and
  A3: g = restrict(f,A) and
  A4: τ_3 = τ_1 {restricted to} A
  shows IsContinuous(τ_3, τ_2 {restricted to} g(A),g)
proof -
  from A1 A4 have \bigcup τ_3 = A
    using union_restrict by auto
  have two_top_spaces0(τ_3, τ_2, g)
    proof -
      from A4 have τ_3 {is a topology} and τ_2 {is a topology}
        using tau1_is_top tau2_is_top
        topology0_def topology0.Top_1_L4 by auto
      moreover from A1 A3 \bigcup τ_3 = A
        have g: \bigcup τ_3 \rightarrow \bigcup τ_2
          using fmapAssum restrict_type2 by simp
      ultimately show thesis using two_top_spaces0_def
        by simp
    qed
  moreover from assms have IsContinuous(τ_3, τ_2, g)
    using restr_cont by simp
  ultimately have IsContinuous(τ_3, τ_2 {restricted to} g(\bigcup τ_3),g)
    by (rule two_top_spaces0.restr_image_cont)
  moreover note \bigcup τ_3 = A
  ultimately show thesis by simp
qed

We need a context similar to two_top_spaces0 but without the global function f : X_1 → X_2.

locale two_top_spaces1 =

fixes τ_1
assumes tau1_is_top: τ_1 {is a topology}

fixes τ_2
assumes tau2_is_top: τ_2 {is a topology}

fixes X_1
defines X1_def [simp]: X_1 ≡ \bigcup τ_1

fixes X_2
defines X2_def [simp]: X_2 ≡ \bigcup τ_2

706
If a partial function \( g : X_1 \supseteq A \rightarrow X_2 \) is continuous with respect to \((\tau_1, \tau_2)\), then \( A \) is open (in \( \tau_1 \)) and the function is continuous in the relative topology.

**lemma** (in two_top_spaces1) partial_fun_cont:

assumes \( A1: g: A \rightarrow X_2 \) and \( A2: \text{IsContinuous}(\tau_1, \tau_2, g) \)

shows \( A \in \tau_1 \) and \( \text{IsContinuous}(\tau_1 \{\text{restricted to} \} A, \tau_2, g) \)

**proof**

from \( A2 \) have \( g-(X_2) \in \tau_1 \)

using tau2_is_top IsATopology_def IsContinuous_def by simp

with \( A1 \) show \( A \in \tau_1 \) using func1_1_L4 by simp

{ fix \( V \) assume \( V \in \tau_2 \)

with \( A2 \) have \( g-(V) \in \tau_1 \) 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 \( g-(V) \in (\tau_1 \{\text{restricted to} \} A) \)

using RestrictedTo_def by auto

} then show \( \text{IsContinuous}(\tau_1 \{\text{restricted to} \} A, \tau_2, g) \)

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 \( A1: g: A \rightarrow X_2 \) and \( A2: A \in \tau_1 \)

shows \( \text{IsContinuous}(\tau_1, \tau_2, g) \iff \text{IsContinuous}(\tau_1 \{\text{restricted to} \} A, \tau_2, g) \)

**proof**

assume \( \text{IsContinuous}(\tau_1, \tau_2, g) \)

with \( A1 \) show \( \text{IsContinuous}(\tau_1 \{\text{restricted to} \} A, \tau_2, g) \)

using partial_fun_cont by simp

next

assume \( I: \text{IsContinuous}(\tau_1 \{\text{restricted to} \} A, \tau_2, g) \)

{ fix \( V \) assume \( V \in \tau_2 \)

with \( I \) have \( g-(V) \in (\tau_1 \{\text{restricted to} \} A) \)

using IsContinuous_def by simp

then obtain \( W \) where \( W \in \tau_1 \) and \( g-(V) = A \cap W \)

using RestrictedTo_def by auto

with \( A2 \) have \( g-(V) \in \tau_1 \) using tau1_is_top IsATopology_def by simp

} then show \( \text{IsContinuous}(\tau_1, \tau_2, g) \) using IsContinuous_def by simp

qed

**58.5 Product topology and continuity**

We start with three topological spaces \((\tau_1, X_1), (\tau_2, X_2)\) and \((\tau_3, X_3)\) and a function \( f : X_1 \times X_2 \rightarrow X_3 \). We will study the properties of \( f \) with respect to the product topology \( \tau_1 \times \tau_2 \) and \( \tau_3 \). This situation is similar as in locale
two_top_spaces 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 \( \tau_1 \)
  assumes tau1_is_top: \( \tau_1 \) {is a topology}

  fixes \( \tau_2 \)
  assumes tau2_is_top: \( \tau_2 \) {is a topology}

  fixes \( \tau_3 \)
  assumes tau3_is_top: \( \tau_3 \) {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 \( X_3 \)
  defines X3_def [simp]: \( X_3 \equiv \bigcup \tau_3 \)

  fixes \( \eta \)
  defines eta_def [simp]: \( \eta \equiv \text{ProductTopology}(\tau_1, \tau_2) \)

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: \( X_1 \times X_2 \to X_3 \) and IsContinuous(\( \eta, \tau_3, f \))
  and \( x \in X_1 \)
  shows IsContinuous(\( \tau_2, \tau_3, \text{Fix1stVar}(f, x) \))
  using assms 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: \( X_1 \times X_2 \to X_3 \) and IsContinuous(\( \eta, \tau_3, f \))
  and \( y \in X_2 \)
  shows IsContinuous(\( \tau_1, \tau_3, \text{Fix2ndVar}(f, y) \))
  using assms 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: $\tau_1$ is a topology $\tau_2$ is a topology and
A2: $\eta_1$ is a topology $\eta_2$ is a topology and
A3a: $f_1: \bigcup \tau_1 \rightarrow \bigcup \eta_1$ and A3b: $f_2: \bigcup \tau_2 \rightarrow \bigcup \eta_2$ and
A4: IsContinuous($\tau_1, \eta_1, f_1$) IsContinuous($\tau_2, \eta_2, f_2$) and
A5: $g = \{(p, (f_1(fst(p)), f_2(snd(p)))) \mid p \in \bigcup \tau_1 \times \bigcup \tau_2\}$ shows IsContinuous(ProductTopology($\tau_1, \eta_1, f_1$), ProductTopology($\tau_2, \eta_2, f_2$), g)

proof -
let $\tau = \text{ProductTopology}(\tau_1, \tau_2)$
let $\eta = \text{ProductTopology}(\eta_1, \eta_2)$
let $X_1 = \bigcup \tau_1$
let $X_2 = \bigcup \tau_2$
let $Y_1 = \bigcup \eta_1$
let $Y_2 = \bigcup \eta_2$
let $B = \text{ProductCollection}(\eta_1, \eta_2)$
from A1 A2 have $\tau$ is a topology and $\eta$ is a topology
using Top_1_4_T1 by auto
moreover have $g: X_1 \times X_2 \rightarrow Y_1 \times Y_2$
proof -
{ fix $p$ assume $p \in X_1 \times X_2$ hence $\text{fst}(p) \in X_1$ and $\text{snd}(p) \in X_2$ by auto
  from A3a $\text{fst}(p) \in X_1$ have $f_1(\text{fst}(p)) \in Y_1$
  by (rule apply_funtype)
  moreover from A3b $\text{snd}(p) \in X_2$ have $f_2(\text{snd}(p)) \in Y_2$
  by (rule apply_funtype)
  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 $\forall V \in \eta_1 \ W \in \eta_2$ 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
A reformulation of the \texttt{cart\_prod\_cont} lemma above in slightly different notation.

\begin{verbatim}
theorem (in two_top_spaces0) product_cont_functions:
  assumes f:X1 \rightarrow X2 \quad g: \bigcup \tau_3 \rightarrow \bigcup \tau_4
  \quad \text{IsContinuous}(\tau_1,\tau_2,f) \quad \text{IsContinuous}(\tau_3,\tau_4,g)
  \quad \tau_4 \text{is a topology} \quad \tau_3 \text{is a topology}
  \quad \text{shows \ IsContinuous}(\text{ProductTopology}(\tau_1,\tau_3),\text{ProductTopology}(\tau_2,\tau_4),\{(x,y),(fx,gy)\}).
\end{verbatim}

\begin{verbatim}
\textbf{proof -}
  \textbf{have} \{(x,y),(fx,gy)\}. \quad \textbf{proof -}
  \textbf{have} \{(x,y) \in X_1 \times \bigcup \tau_3\} = \{(p,(f(fst(p)),g(snd(p)))) \quad \text{by force}
  \textbf{by \ simp \ thesis \ using \ \texttt{cart\_prod\_cont}}
\end{verbatim}

A special case of \texttt{cart\_prod\_cont} when the function acting on the second axis is the identity.

\begin{verbatim}
lemma cart_prod_cont1:
  assumes A1: \tau_1 \text{is a topology} \quad \text{and} \quad A1a: \tau_2 \text{is a topology}
  \quad A2: \eta_1 \text{is a topology} \quad \text{and} \quad A3: f_1: \bigcup \tau_1 \rightarrow \bigcup \eta_1 \quad \text{and} \quad A4: \text{IsContinuous}(\tau_1,\eta_1,f_1) \quad \text{and} \quad A5: g = \{(p,(f_1(fst(p)),snd(p)))\} \quad \text{proof -}
  \textbf{let} f_2 = \text{id}(\bigcup \tau_2)
  \textbf{have} \forall x \in \bigcup \tau_2. \quad f_2(x) = x \quad \text{using \ id\_conv \ by \ blast}
  \textbf{hence I:} \quad \forall p \in \bigcup \tau_1 \times \bigcup \tau_2. \quad \text{snd(p)} = f_2(snd(p)) \quad \text{by \ simp}
  \textbf{note} \quad A1 \quad A1a \quad A2 \quad A1a \quad A3
  \textbf{moreover have} \quad f_2:(\bigcup \tau_2 \rightarrow \bigcup \tau_2 \quad \text{using \ id\_type \ by \ simp}
  \textbf{moreover note} \quad A4
  \textbf{moreover have} \quad \text{IsContinuous}(\tau_2,\tau_2,f_2) \quad \text{using \ id\_cont \ by \ simp}
  \textbf{moreover have} \quad g = \{(p,(f_1(fst(p)),f_2(snd(p)))\} \quad \text{proof -}
  \textbf{from A5 I show} \quad g \subseteq \{(p,(f_1(fst(p)),f_2(snd(p))))\} \quad \text{by \ auto}
  \textbf{from A5 I show} \quad \{(p,(f_1(fst(p)),f_2(snd(p))))\} \quad \subseteq \quad \text{by \ auto}
\end{verbatim}

A special case of \texttt{cart\_prod\_cont} when the function acting on the second axis is the identity.

\begin{verbatim}

58.6 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

\end{verbatim}
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 \texttt{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 \texttt{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 continuous.

\textbf{lemma} (in \texttt{two_top_spaces0}) \texttt{restriction_continuous1}:
\begin{itemize}
  \item \texttt{assumes A1: A \subseteq X_1 and A2: restrict(f,A) \{is continuous\}}
  \item \texttt{shows A \in \tau_1}
\end{itemize}
\texttt{proof} -

- from \texttt{assms} have \texttt{two_top_spaces1(\tau_1, \tau_2) and}
  \texttt{restrict(f,A):A\rightarrow X_2 and restrict(f,A) \{is continuous\}}
  \texttt{using tau1_is_top tau2_is_top two_top_spaces1_def fmapAssum restrict_fun}
  \texttt{by auto}

then show \texttt{thesis} using \texttt{two_top_spaces1.partial_fun_cont} by \texttt{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.

\textbf{lemma} (in \texttt{two_top_spaces0}) \texttt{pasting_lemma1}:
\begin{itemize}
  \item \texttt{assumes A1: M \subseteq \tau_1 and A2: \forall U\in M. restrict(f,U) \{is continuous\}}
  \item \texttt{shows restrict(f,\bigcup M) \{is continuous\}}
\end{itemize}
\texttt{proof} -

\{ fix V assume V\in \tau_2 from A1 have \bigcup M \subseteq X_1 by auto then have restrict(f,\bigcup M)-(V) = f-(V) \cap (\bigcup M)
  \texttt{using func1_2_L1 fmapAssum by simp}
also have ... = \bigcup \{f-(V) \cap U. U\in M\} by auto finally have 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 Pow(\tau_1)
\texttt{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 restrict(f,U) \{is continuous\} by simp with \texttt{V\in \tau_2} have restrict(f,U)-(V) \in \tau_1
  \texttt{using IsContinuous_def by simp}
\}

711
moreover from \( \bigcup M \subseteq X_1 \) and \( \{ U \in M \} \)
have \( \text{restrict}(f, U) -(V) = f-(V) \cap U \) using \text{fmapAssum} \text{func1_2_L1} by blast
ultimately have \( f-(V) \cap U \in \tau_1 \) by simp
with I have \( \omega \in \tau_1 \) by simp
} then show thesis by auto
qed
then have \( \bigcup \{ f-(V) \cap U. U \in M \} \in \tau_1 \) using \text{tau1_is_top} \text{IsATopology_def} by auto
ultimately have \( \text{restrict}(f, \bigcup M)-(V) \in \tau_1 \) by simp
} then show thesis using \text{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: \( \text{restrict}(f, A) \) \{is continuous\} \( \text{restrict}(f, B) \) \{is continuous\}
shows \( \text{restrict}(f, A \cap B) \) \{is continuous\}
proof -
{ fix V assume \( V \in \tau_2 \)
with assms have
\( \text{restrict}(f, A)-(V) = f-(V) \cap A \) \( \text{restrict}(f, B)-(V) = f-(V) \cap B \) and
\( \text{restrict}(f, A)-(V) \in \tau_1 \) \( \text{and} \) \( \text{restrict}(f, B)-(V) \in \tau_1 \)
using \text{func1_2_L1} \text{fmapAssum} \text{IsContinuous_def} by auto
then have \( (\text{restrict}(f, A)-(V)) \cap (\text{restrict}(f, B)-(V)) = f-(V) \cap (A \cap B) \) by auto
moreover
from A2 \( \{ V \in \tau_2 \} \)
have \( \text{restrict}(f, A)-(V) \in \tau_1 \) \( \text{and} \) \( \text{restrict}(f, B)-(V) \in \tau_1 \)
using \text{IsContinuous_def} by auto
then have \( (\text{restrict}(f, A)-(V)) \cap (\text{restrict}(f, B)-(V)) \in \tau_1 \)
using \text{tau1_is_top} \text{IsATopology_def} by simp
moreover
from A1 have \( (A \cap B) \subseteq X_1 \) by auto
then have \( \text{restrict}(f, A \cap B)-(V) = f-(V) \cap (A \cap B) \)
using \text{func1_2_L1} \text{fmapAssum} by simp
ultimately have \( \text{restrict}(f, A \cap B)-(V) \in \tau_1 \) by simp
} then show thesis using \text{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 \( \{ U \in \tau_1. \text{restrict}(f, U) \} \) \{is a topology\}
proof -
let \( T = \{ U \in \tau_1. \text{restrict}(f, U) \} \) \{is continuous\}
have \( \forall M \in \text{Pow}(T). \bigcup M \in T \)
proof
712
fix $M$ assume $M \in \text{Pow}(T)$
then have $\text{restrict}(f, \bigcup M)$ {is continuous}
using pasting lemma 1 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 T. \ \text{restrict}(f,U) \ {is \ continuous}\}\$
  have $T$ {is a topology} by (rule pasting_theorem)
  then have $0 \in T$ by (rule empty_open)
  hence $\text{restrict}(f,0)$ {is continuous} by simp
  moreover have $\text{restrict}(f,0) = 0$ by simp
  ultimately show thesis by simp
qed

59 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 $n = \{0,1,...,n-1\}$ 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.

59.1 The base of the product topology
In this section we define the base of the product topology.

Suppose $\mathcal{X} = I \to \bigcup T$ 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 \( \mathcal{X} \) that models the cartesian product of \( W \).

definition
\[
\text{FinProd}(\mathcal{X},W) \equiv \{x \in \mathcal{X}. \ \forall i \in \text{domain}(W). \ x(i) \in W(i)\}
\]

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
\[
\text{ProductTopBase}(I,T) \equiv \bigcup N \in \text{FinPow}(I). \{\text{FinProd}(I \to T,W). \ W \in N \to T\}
\]

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 \( \mathcal{X} = I \to \bigcup T \)
  from A2 obtain \( N_1, W_1, N_2, W_2 \) where
    I: \( N_1 \in \text{FinPow}(I) \ W_1 \in N_1 \to T \ U = \text{FinProd}(\mathcal{X},W_1) \) and
    II: \( N_2 \in \text{FinPow}(I) \ W_2 \in N_2 \to T \ V = \text{FinProd}(\mathcal{X},W_2) \)
  using ProductTopBase_def by auto
  let \( N_3 = N_1 \cup N_2 \)
  let \( W_3 = \{\langle i,\text{if} \ i \in N_1-N_2 \text{ then } W_1(i) \ 
     \text{else if} \ i \in N_2-N_1 \text{ then } W_2(i) \ 
     \text{else} \ (W_1(i)) \cap (W_2(i))\}. \ i \in N_3\} \)
  from A1 I II have \( \forall i \in N_1 \cap N_2. \ (W_1(i) \cap W_2(i)) \in T \)
    using apply_functype 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_functype by auto
  ultimately have \( W_3 : N_3 \to T \) by (rule fun_union_overlap)
  with I II have \( \text{FinProd}(\mathcal{X},W_3) \in \text{ProductTopBase}(I,T) \) using union_finpow
  ProductTopBase_def by auto
  moreover have \( U \cap V = \text{FinProd}(\mathcal{X},W_3) \)
  proof
  \{ fix x assume x \in U and x \in V
    \<U = \text{FinProd}(\mathcal{X},W_1)> \ <W_1 \in N_1 \to T> \ and \ <V = \text{FinProd}(\mathcal{X},W_2)> \ <W_2 \in N_2 \to T>
    have x \in \mathcal{X} and \( \forall i \in N_1. \ x(i) \in W_1(i) \) and \( \forall i \in N_2. \ x(i) \in W_2(i) \)
      using func1_1_L1 FinProd_def by auto
    with \( W_3 : N_3 \to T \) \( x \in \mathcal{X} \) have \( x \in \text{FinProd}(\mathcal{X},W_3) \)
      using ZF_fun_from_tot_val func1_1_L1 FinProd_def by auto
  \}

714
\textbf{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.}

\textbf{Theorem prod_top_base_is_base: assumes} \( T \) \{is a topology\}
\shownotes{\text{shows} ProductTopBase(I,T) \{satisfies the base condition\}}
\shownotes{using} \text{assms prod_top_base_inter inter_closed_base} \text{by simp}

\textbf{The (sequence) product topology is indeed a topology on the space of sequences. In the proof we are using the fact that (\( \emptyset \rightarrow X \)) = \{\emptyset\}.}

\textbf{Theorem seq_prod_top_is_top: assumes} \( T \) \{is a topology\}
\shownotes{\text{shows} SeqProductTopology(I,T) \{is a topology\} and ProductTopBase(I,T) \{is a base for\} SeqProductTopology(I,T) and \( \bigcup \text{SeqProductTopology}(I,T) = (I \rightarrow \bigcup T) \)}
\textbf{proof -}
\textbf{from} \text{assms show} SeqProductTopology(I,T) \{is a topology\} and ProductTopBase(I,T) \{is a base for\} SeqProductTopology(I,T) and
\textbf{using} \text{prod_top_base_is_base SeqProductTopology_def Top_1_2_T1 by auto}
\textbf{from} I \text{have} \text{\( \bigcup \text{SeqProductTopology}(I,T) = \bigcup \text{ProductTopBase}(I,T) \)}
\textbf{using} Top_1_2_L5 \text{by simp}
\textbf{also have} \text{\( \bigcup \text{ProductTopBase}(I,T) = (I \rightarrow \bigcup T) \)}
\textbf{proof}
\textbf{show} \text{\( \bigcup \text{ProductTopBase}(I,T) \subseteq (I \rightarrow \bigcup T) \)} \text{using ProductTopBase_def FinProd_def}
\textbf{by auto}
\textbf{have} 0 \in FinPow(I) \text{using empty_in_finpow by simp}
59.2 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 $n = \{0, 1, ..., n-1\}$). 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 by auto
{
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 $A2$ $\langle N \in \text{FinPow}(n)\rangle$ $\langle W \in n \to T\rangle$ have $\forall i \in n. (if i \in N then W(i) else \bigcup T)$ using apply_funtype FinPow_def IsATopology_def by auto
moreover have $B = \text{FinProd}(n \to \bigcup T, W_n)$
proof
{fix $x$ assume $x \in B$
with $B = \text{FinProd}(n \to \bigcup T, W)$ have $x \in n \to \bigcup T$ using FinProd_def
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 $\langle W_n : n \to T\rangle$ have $i \in n$ using func1_1_L1 by simp
with $\langle x : n \to \bigcup T\rangle$ have $x(i) \in \bigcup T$ using apply_funtype by blast
with $\langle x \in B\rangle$ $\langle B = \text{FinProd}(n \to \bigcup T, W)\rangle$ $\langle W \in n \to T\rangle$ $\langle W_n : n \to T\rangle$ $\langle i \in n\rangle$

716
show \( x(i) \in W_n(i) \) using `func1_1_L1` `FinProd_def` `ZF_fun_from_tot_val`

by simp

qed

ultimately have \( x \in \text{FinProd}(n \rightarrow \bigcup T, W_n) \) using `FinProd_def` by simp

} thus \( B \subseteq \text{FinProd}(n \rightarrow \bigcup T, W_n) \) by auto

next

\{
fix \( x \)
assume \( x \in \text{FinProd}(n \rightarrow \bigcup T, W_n) \)
then have \( x \in n \rightarrow \bigcup T \) and \( \forall i \in \text{domain}(W_n). \ x(i) \in W_n(i) \)
using `FinProd_def` by auto
with \( \langle W_n : n \rightarrow T \rangle \) 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 \( \langle W_n : n \rightarrow T \rangle \) and \( \langle N \in \text{FinPow}(n) \rangle \)
have \( \forall i \in N. \ x(i) \in W_n(i) \)
using `func1_1_L1` `FinPow_def`
by auto
ultimately have \( \forall i \in N. \ x(i) \in W(i) \)
with \( \langle W \in N \rightarrow T \rangle \) \( \langle x \in n \rightarrow \bigcup T \rangle \) \( \langle B = \text{FinProd}(n \rightarrow \bigcup T, W) \rangle \)
have \( x \in B \)
using `func1_1_L1` `FinProd_def`
by simp
\}

thus \( \text{FinProd}(n \rightarrow \bigcup T, W_n) \subseteq B \) by auto

qed

ultimately have \( B \subseteq \{ \text{FinProd}(n \rightarrow \bigcup T, W). \ W \in n \rightarrow T \} \) by auto

\}

thus \( \text{ProductTopBase}(n, T) \subseteq \{ \text{FinProd}(n \rightarrow \bigcup T, W). \ W \in n \rightarrow T \} \) by auto

qed

A technical lemma providing a formula for finite product on one topological space.

lemma `single_top_prod`:
assumes \( A1: W : 1 \rightarrow \tau \)
shows \( \text{FinProd}(1 \rightarrow \bigcup \tau, W) = \{ \langle 0, y \rangle. \ y \in W(0) \} \)

proof -

have \( 1 = \{0\} \) by auto

from \( A1 \) have \( \text{domain}(W) = \{0\} \) using `func1_1_L1` by auto

then have \( \text{FinProd}(1 \rightarrow \bigcup \tau, W) = \{ x \in 1 \rightarrow \bigcup \tau. \ x(0) \in W(0) \} \)
using `FinProd_def` by simp

also have \( \{ x \in 1 \rightarrow \bigcup \tau. \ x(0) \in W(0) \} = \{ \langle 0, y \rangle. \ y \in W(0) \} \)

proof

from \( 1 = \{0\} \) show \( \{ x \in 1 \rightarrow \bigcup \tau. \ x(0) \in W(0) \} \subseteq \{ \langle 0, y \rangle. \ y \in W(0) \} \)
using `func_singleton_pair` by auto

\{
fix \( x \)
assume \( x \in \{ \langle 0, y \rangle. \ y \in W(0) \} \)
then obtain \( y \) where \( x = \{ \langle 0, y \rangle \} \) and \( II: y \in W(0) \) by auto
with \( A1 \) have \( y \in \bigcup \tau \) using `apply_funttype` by auto
with \( \langle x = \{ \langle 0, y \rangle \} \rangle \) \( 1 = \{0\} \) have \( x : 1 \rightarrow \bigcup \tau \) using `pair_func_singleton` by auto
with \( \langle x = \{ \langle 0, y \rangle \} \rangle \) \( 1 = \{0\} \) have \( x : 1 \rightarrow \bigcup \tau \) using `pair_func_singleton` by auto
\}

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

$$\text{SeqProductTopology}(1, \tau) = \{ \{ \langle 0, y \rangle \}. y \in U \}. U \in \tau \} \land \text{IsAhomeomorphism}(\tau, \text{SeqProductTopology}(1, \tau), \{ \langle y, \{ \langle 0, y \rangle \} \}. y \in U \cup \tau \})$$

**proof**

- have $\{ 0 \} = 1$ by auto
- let $b = \{ \langle y, \{ \langle 0, y \rangle \} \}. y \in U \}$
- have $b \in \text{bij}(<U \cup \tau, 1 \to \tau>)$ using list_singleton_bij by blast
- with A1 have $\{ b(U). U \in \tau \} \land \text{IsAhomeomorphism}(\tau, \{ b(U). U \in \tau \}, b)$
- using bij_induced_top by auto
- moreover have $\forall U \in \tau. b(U) = \{ \{ \langle 0, y \rangle \}. y \in U \}$
- proof
- fix $U$ assume $U \in \tau$
- from $<b \in \text{bij}(<U \cup \tau, 1 \to \tau>)$ have $b: <U \to (1 \to \tau)$ using bij_def inj_def
- by simp
- $\{ \text{fix} y \text{ assume} y \in U \}$
- with $<b: <U \to (1 \to \tau)>$ have $b(y) = \{ \langle 0, y \rangle \}$ usingZF_fun_from_tot_val
- by simp
- } hence $\forall y \in U \tau. b(y) = \{ \{ \langle 0, y \rangle \}$ by auto
- with $<U \in \tau> <b: <U \to (1 \to \tau)>$ show $b(U) = \{ \{ \langle 0, y \rangle \}. y \in U \}$
- using func_imagedef by auto
- qed
- moreover have $\text{ProductTopBase}(1, \tau) = \{ \{ \langle 0, y \rangle \}. y \in U \}. U \in \tau \}$
- proof
- $\{ \text{fix} V \text{ assume} V \in \text{ProductTopBase}(1, \tau)$
- with A1 obtain $W$ where $W: 1 \to \tau$ and $V = \text{FinProd}(1 \to \tau, W)$
- using fin_prod_def_nat by auto
- then have $V \in \{ \{ \langle 0, y \rangle \}. y \in U \}. U \in \tau \}$ using apply_functype single_top_prod
- by auto
- } thus $\text{ProductTopBase}(1, \tau) \subseteq \{ \{ \langle 0, y \rangle \}. y \in U \}. U \in \tau \}$ by auto
- $\{ \text{fix} V \text{ assume} V \in \{ \{ \langle 0, y \rangle \}. y \in U \}. U \in \tau \}$
- then obtain $U$ where $U \in \tau$ and $V = \{ \{ \langle 0, y \rangle \}. y \in U \}$ by auto
- let $W = \{ \langle 0, U \rangle \}$
- from $<U \in \tau>$ have $W: \{ \langle 0 \rangle \} \to \tau$ using pair_func_singleton by simp
- with $<\{ 0 \} = 1$ have $W: 1 \to \tau$ and $W(0) = U$ using pair_val by auto
- with $<V = \{ \{ \langle 0, y \rangle \}. y \in U \}>$ have $V = \text{FinProd}(1 \to \tau, W)$
- using single_top_prod by simp
- with A1 $<W: 1 \to \tau>$ have $V \in \text{ProductTopBase}(1, \tau)$ using fin_prod_def_nat
- by auto
- } thus $\{ \{ \langle 0, y \rangle \}. y \in U \}. U \in \tau \} \subseteq \text{ProductTopBase}(1, \tau)$ by auto
- qed

718
ultimately have I: ProductTopBase(1,\(\tau\)) \{is a topology\} and
II: IsAhomeomorphism(\(\tau\), ProductTopBase(1,\(\tau\)),b) by auto
from A1 have ProductTopBase(1,\(\tau\)) \{is a base for\} SeqProductTopology(1,\(\tau\))
using seq_prod_top_is_top by simp
with I have ProductTopBase(1,\(\tau\)) = SeqProductTopology(1,\(\tau\))
by (rule base_topology)
with <ProductTopBase(1,\(\tau\)) = \{ { \langle 0,y \rangle \}. y \in U \}. U \in \tau\}
II show 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\}) 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 IsAhomeomorphism(SeqProductTopology(1,\(\tau\)),\(\tau\),\{ \langle x,x(0) \rangle \}. x \in 1 \rightarrow \bigcup \tau\})
proof -
  have \{ \langle x,x(0) \rangle \}. x \in 1 \rightarrow \bigcup \tau\}) = converse(\{ \langle y,\{\langle 0,y \rangle \}\}. y \in \bigcup \tau\})
  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 \((\bigcup \text{ProductTopology}(T,\tau)) = (n \rightarrow \bigcup \tau) \times \bigcup \tau\)
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: \(n \in \mathbb{N}\)
and A3: \(U \in \text{ProductTopBase}(n,\tau)\) and A4: \(V \in \tau\)
shows \(\{ x \in \text{succ}(n) \rightarrow \bigcup \tau, \text{Init}(x) \in U \land x(n) \in V \} \in \text{ProductTopBase}(\text{succ}(n),\tau)\)
proof -
  let B = \{ x \in \text{succ}(n) \rightarrow \bigcup \tau, \text{Init}(x) \in U \land x(n) \in V \}
  from A1 A2 have \(\text{ProductTopBase}(n,\tau) = \{ \text{FinProd}(n \rightarrow \bigcup \tau,W) \}. W \in n \rightarrow \tau\} = \{ \text{FinProd}(n \rightarrow \bigcup \tau,W) \}. W \in n \rightarrow \tau\} = \{ \text{FinProd}(n \rightarrow \bigcup \tau,W) \}. W \in n \rightarrow \tau\} = \{ \text{FinProd}(n \rightarrow \bigcup \tau,W) \}. W \in n \rightarrow \tau\} = \{ \text{FinProd}(n \rightarrow \bigcup \tau,W) \}. W \in n \rightarrow \tau\} = \{ \text{FinProd}(n \rightarrow \bigcup \tau,W) \}. W \in n \rightarrow \tau\} = \{ \text{FinProd}(n \rightarrow \bigcup \tau,W) \}. W \in n \rightarrow \tau\} = \{ \text{FinProd}(n \rightarrow \bigcup \tau,W) \}. W \in n \rightarrow \tau\} = \{ \text{FinProd}(n \rightarrow \bigcup \tau,W) \}. W \in n \rightarrow \tau\}
  using fin_prod_def_nat by simp
  with A3 obtain \(W_U\) where \(W_U:n \rightarrow \tau\) and U =FinProd(n \rightarrow \bigcup \tau,W_U) by auto
  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) = \text{succ}(n)
      using func1_1_L1
    }
    719
by auto
moreover from A2 A4 <x∈B> <U =FinProd(n→∪τ,W) <W:U:n→τ> <x ∈ succ(n)→∪τ> have ∀i∈succ(n). x(i) ∈ W(i) using func1_1_L1 FinProd_def init_props append Props
by simp
ultimately have x ∈ FinProd(succ(n)→∪τ,W) using FinProd_def by simp
next
{ fix x assume x ∈ FinProd(succ(n)→∪τ,W)
then have x:succ(n)→τ and I: ∀i ∈ domain(W). x(i) ∈ W(i)
using FinProd_def by auto
moreover have Init(x) ∈ U
proof -
from A2 and <x:succ(n)→∪τ> have Init(x):n→∪τ using init Props
by simp
moreover have ∀i∈domain(W). Init(x)(i) ∈ W(i)
proof -
from A2 <x ∈ FinProd(succ(n)→∪τ,W) <W:succ(n)→τ> have ∀i∈n. x(i) ∈ W(i)
using FinProd_def func1_1_L1 by simp
moreover from A2 <x: succ(n)→∪τ> have ∀i∈n. Init(x)(i) = x(i)
using init Props by simp
moreover from A4 <W:U:n→τ> have ∀i∈n. W(i) = W_U(i)
using append Props by simp
ultimately have ∀i∈n. Init(x)(i) ∈ W(i) by simp
with <W:U:n→τ> show thesis using func1_1_L1 by simp
qed
ultimately have Init(x) ∈ FinProd(n→∪τ,W_U) using FinProd_def
by simp
with <U =FinProd(n→∪τ,W_U)> show thesis by simp
qed
moreover have x(n) ∈ V
proof -
from <W:succ(n)→τ> I have x(n) ∈ W(n) using func1_1_L1 by simp
moreover from A4 <W:U:n→τ> have W(n) = V using append Props
by simp
ultimately show thesis by simp
qed
ultimately have x∈B by simp
}

thus FinProd(succ(n)→∪τ,W) ⊆ B by auto
qed
moreover from A1 A2 have
ProductTopBase(succ(n),τ) = {FinProd(succ(n)→∪τ,W). W∈succ(n)→τ}
using fin_prod_def_nat by simp
ultimately show thesis by auto

720
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 $A1$: $\tau$ {is a topology} and $A2$: $n \in \text{nat}$ and

$A3$: $U \in \text{SeqProductTopology}(n,\tau)$ and $A4$: $V \in \tau$

shows $\{x : \text{succ}(n) \rightarrow \bigcup \tau. \text{Init}(x) \in U \land x(n) \in V\} \in \text{SeqProductTopology}($\text{succ}(n),$\tau)$

**Proof** -

from $A1$ have ProductTopBase($n,\tau$) {is a base for} $\text{SeqProductTopology}(n,\tau)$ and

$I$: ProductTopBase($\text{succ}(n),\tau$) {is a base for} $\text{SeqProductTopology}($\text{succ}(n),$\tau$) and

$II$: $\text{SeqProductTopology}($\text{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 $B \subseteq \text{ProductTopBase}(n,\tau)$ and $U = \bigcup B$ by auto

then have

$\{x : \text{succ}(n) \rightarrow \bigcup \tau. \text{Init}(x) \in U \land x(n) \in V\} = (\bigcup B. \{x : \text{succ}(n) \rightarrow \bigcup \tau. \text{Init}(x) \in B \land x(n) \in V\})$

by auto

moreover from $A1$ $A2$ $A4$ <\$B \subseteq \text{ProductTopBase}(n,\tau)$> have

$\forall B \in B. (\{x : \text{succ}(n) \rightarrow \bigcup \tau. \text{Init}(x) \in B \land x(n) \in V\} \in \text{ProductTopBase}(\text{succ}(n),\tau))$

using finite_prod_succ_base by auto

with $I$ $II$ have

$(\bigcup B. \{x : \text{succ}(n) \rightarrow \bigcup \tau. \text{Init}(x) \in B \land x(n) \in V\}) \in \text{SeqProductTopology}(\text{succ}(n),\tau)$

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 $A1$: $\tau$ {is a topology} and $A2$: $n \in \text{nat}$ and

$A3$: $f = \{\langle x, ($\text{Init}(x),x(n)\rangle. x \in \text{succ}(n) \rightarrow \bigcup \tau\}$ and

$A4$: $T = \text{SeqProductTopology}(n,\tau)$ and

$A5$: $S = \text{SeqProductTopology}(\text{succ}(n),\tau)$

shows IsAhomeomorphism($S,$ProductTopology($T,\tau$),$f$)

**Proof** -

let $C = \text{ProductCollection}(T,\tau)$
let $B = \text{ProductTopBase}(\text{succ}(n), \tau)$

from A1 A4 have $T$ {is a topology} using \text{seq_prod_top_is_top} by simp

with A1 A5 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 assms have $f \in \text{bij}(\bigcup S, \bigcup \text{ProductTopology}(T, \tau))$

using \text{lists_cart_prod 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 two_top_spaces0$(S, \text{ProductTopology}(T, \tau), f)$ using two_top_spaces0_def

moreover have $\forall W \in \bigcup C. f^{-1}(W) \in \bigcup S$

proof

fix $W$ assume $W \in \bigcup C$

then obtain $U V$ where $U \in T$ and $V \in \tau$ using \text{ProductCollection_def} by auto

from A1 A5 have $f: \text{succ}(n) \rightarrow \bigcup \tau \rightarrow \bigcup \text{ProductTopology}(T, \tau)$

using \text{seq_prod_top_is_top} by simp

with assm $W = U \times V$ show $f^{-1}(W) \in S$

using \text{ZF_fun_from_tot_val func1_1_L15 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 A1 A2 obtain $W_V$ where $W_V \in \text{succ}(n) \rightarrow \tau$ and $V = \text{FinProd}(\text{succ}(n) \rightarrow \bigcup \tau, W_V)$

using \text{fin_prod_def_nat} by auto

let $U = \text{FinProd}(n \rightarrow \bigcup \tau, \text{Init}(W_V))$

let $W = W_V(n)$

have $U \in T$

proof -

from A1 A2 have $W_V \in \text{succ}(n) \rightarrow \tau$ have $U \in \text{ProductTopBase}(n, \tau)$

using \text{fin_prod_def_nat init_props} by auto

with A1 A4 show thesis using \text{seq_prod_top_is_top base_sets_open} by blast

qed

moreover from A1 $W_V \in \text{succ}(n) \rightarrow \tau$ have $U \in T$ have $U \times W \in \text{ProductTopology}(T, \tau)$

using \text{apply_functype prod_open_open_prod} by simp

moreover have $f(V) = U \times W$

proof -

from A2 have $W_V: \text{succ}(n) \rightarrow \tau$ have $\text{Init}(W_V): n \rightarrow \tau$ and III: $\forall k \in n. \text{Init}(W_V)(k)$

722
= \mathcal{W}_V(k)

  \text{using init_props by auto}
  \text{then have domain(Init(\mathcal{W}_V))} = n \text{ using func1_1_L1 by simp}
  \text{have } f(V) = \{ \langle \text{Init}(x), x(n) \rangle . x \in V \}
  \text{proof -}
  \text{have } f(V) = \{ f(x) . x \in V \}
  \text{proof -}
  \text{from A1 A5 have } B \{ \text{is a base for} \} S \text{ using seq_prod_top_is_top}
  \text{by simp}
  \text{with } V \in B \text{ using IsAbaseFor_def by auto}
  \text{with } f : \bigcup S \rightarrow \bigcup \text{ProductTopology}(T, \tau) \text{ show thesis using func_imagedef}
  \text{by simp}
  \text{qed}
  \text{moreover have } \forall x \in V. f(x) = \langle \text{Init}(x), x(n) \rangle
  \text{proof -}
  \text{from A1 A3 A5}
  \text{< } V = \text{FinProd}(\text{succ}(n) \rightarrow \bigcup \tau, \mathcal{W}_V) \text{ > have } V \subseteq \bigcup S \text{ and}
  \text{fdef: } f = \{ \langle x, \langle \text{Init}(x), x(n) \rangle \rangle . x \in \bigcup S \} \text{ using seq_prod_top_is_top}
  \text{FinProd_def}
  \text{by auto}
  \text{from } f : \bigcup S \rightarrow \bigcup \text{ProductTopology}(T, \tau) \text{ fdef have } \forall x \in \bigcup S. f(x)
  = \langle \text{Init}(x), x(n) \rangle
  \text{by (rule ZF_fun_from_tot_val0)}
  \text{with } V \subseteq \bigcup S \text{ show thesis by auto}
  \text{qed}
  \text{ultimately show thesis by simp}
  \text{qed}
  \text{also have } \{ \langle \text{Init}(x), x(n) \rangle . x \in V \} = U \times \mathcal{W}
  \text{proof}

  \{ \text{ fix } y \text{ assume } y \in \{ \langle \text{Init}(x), x(n) \rangle . x \in V \}
  \text{ then obtain } x \text{ where I: } y = \langle \text{Init}(x), x(n) \rangle \text{ and } x \in V \text{ by auto}

  \text{with } V = \text{FinProd}(\text{succ}(n) \rightarrow \bigcup \tau, \mathcal{W}_V) \text{ have}
  \text{x : succ(n) \rightarrow \bigcup \tau and II: } \forall k \in \text{domain(\mathcal{W}_V)}. x(k) \in \mathcal{W}_V(k)
  \text{unfolding FinProd_def by auto}
  \text{with A2 } < \mathcal{W}_V : \text{succ}(n) \rightarrow \tau > \text{ have IV: } \forall k \in n. \text{ Init}(x)(k) = x(k)
  \text{using init_props by simp}
  \text{have Init(x) } \in U
  \text{proof -}
  \text{from A2 } < x : \text{succ}(n) \rightarrow \tau > \text{ have Init(x): } n \rightarrow \bigcup \tau \text{ using init_props}
  \text{by simp}
  \text{moreover have } \forall k \in \text{domain(Init(\mathcal{W}_V))). Init}(x)(k) \in \text{Init(\mathcal{W}_V))(k)
  \text{proof -}
  \text{from A2 } < \mathcal{W}_V : \text{succ}(n) \rightarrow \tau > \text{ have Init(\mathcal{W}_V): } n \rightarrow \tau \text{ using init_props}
  \text{by simp}
  \text{then have domain(Init(\mathcal{W}_V))} = n \text{ using func1_1_L1 by simp}
  \text{note III IV } < \text{domain(Init(\mathcal{W}_V))} = n >
  \text{moreover from II } < \mathcal{W}_V \in \text{succ}(n) \rightarrow \tau > \text{ have } \forall k \in n. \ x(k) \in
  \mathcal{W}_V(k)
ultimately show thesis by simp
qed
ultimately show \( \text{Init}(x) \in U \) using \( \text{FinProd}_{\text{def}} \) by simp
qed
moreover from \( \langle W, V \rangle : \text{succ}(n) \rightarrow \tau \) II have \( x(n) \in W \) using func1_1_L1
by simp
ultimately have \( \langle \text{Init}(x), x(n) \rangle \in U \times W \) by simp
with \( I \) have \( y \in U \times W \) by auto
\{ thus \( \{ \langle \text{Init}(x), x(n) \rangle. x \in V \} \subseteq U \times W \) by auto
\{ fix \( y \) assume \( y \in U \times W \)
then have \( \text{fst}(y) \in U \) and \( \text{snd}(y) \in W \) by auto
with \( \langle \text{domain}(\text{Init}(W_V)) = n \rangle \) have \( \text{fst}(y) : n \rightarrow \bigcup \tau \) and
\( V : \forall k \in n. \text{fst}(y)(k) \in \text{Init}(W_V)(k) \)
using \( \text{FinProd}_{\text{def}} \) by auto
from \( \langle W_V : \text{succ}(n) \rightarrow \tau \rangle \) have \( W \in \tau \) using apply_funtype by simp
with \( \langle \text{snd}(y) \in W \rangle \) have \( \text{snd}(y) \in \bigcup \tau \) by auto
let \( x = \text{Append}(\text{fst}(y), \text{snd}(y)) \)
have \( x \in V \) 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 \( x(n) = \text{snd}(y) \)
using append_props by auto
moreover from III \( V \) have \( \forall k \in n. \text{fst}(y)(k) \in W_V(k) \) by simp
moreover note \( \langle \text{snd}(y) \in W \rangle \)
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 \( \text{FinProd}_{\text{def}} \)
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 \( A2 \langle y \in U \times W \rangle \langle \text{fst}(y) : n \rightarrow \bigcup \tau \rangle \langle \text{snd}(y) \in \bigcup \tau \rangle \) have \( y = \langle \text{Init}(x), x(n) \rangle \)
using init_append append_props by auto
ultimately have \( y \in \{ \langle \text{Init}(x), x(n) \rangle. x \in V \} \) by auto
\} thus \( U \times W \subseteq \{ \langle \text{Init}(x), x(n) \rangle. x \in V \} \) by auto
qed
finally show \( f(V) = U \times W \) by simp
qed
ultimately show \( f(V) \in \text{ProductTopology}(T, \tau) \) by simp
qed
ultimately show thesis using two_top_spaces0.bij_base_open_homeo by
60  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.

60.1  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.

First we need to define what a directed set is:

definition
  IsDirectedSet (_ directs _ 90)
  where r directs D ≡ refl(D,r) ∧ trans(r) ∧ (∀x∈D. ∀y∈D. ∃z∈D. ⟨x,z⟩∈r ∧ ⟨y,z⟩∈r)

Any linear order is a directed set; in particular (ℕ, ≤).

lemma linorder_imp_directed:
  assumes IsLinOrder(X,r)
  shows r directs X
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,x)∈r) using refl_def by auto
    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

725
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 now what a directed set is.

definition 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 (_ → _ 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 ∈ Neigh
    then have ⟨U,U⟩ ∈ r using r_def by auto
  }
  then have refl(Neigh,r) using refl_def by auto
  moreover
  { fix U V W
    assume ⟨U,V⟩ ∈ r ⟨V,W⟩ ∈ r
    then have U ∈ Neigh W ∈ Neigh W⊆U using r_def by auto
    then have ⟨U,W⟩∈r using r_def by auto
  }
  then have trans(r) using trans_def by blast
  moreover

726
\{
  \text{fix } A, B \\
  \text{assume } p: A \in \text{Neigh} B \in \text{Neigh} \\
  \text{have } A \cap B \in \text{Neigh} \\
  \text{proof}\-
  \text{from } p \text{ have } A \cap B \in \text{Pow}(\bigcup T) \text{ using Neigh_def by auto} \\
  \text{moreover} \\
  \{ \text{from } p \text{ have } x \in \text{int}(A) x \in \text{int}(B) \text{ using Neigh_def by auto} \\
  \text{then have } x \in \text{int}(A) \cap \text{int}(B) \text{ by auto} \\
  \text{moreover} \}
  \text{have } \text{int}(A) \cap \text{int}(B) \subseteq \text{int}(A \cap B) \text{ using Top_2_L1 by auto} \\
  \text{moreover have } \text{int}(A) \cap \text{int}(B) \in T \\
  \text{using Top_2_L2 Top_2_L2 topSpaceAssum IsATopology_def by blast} \\
  \text{ultimately have } \text{int}(A) \cap \text{int}(B) \subseteq \text{int}(A) \cap \text{int}(B) \text{ using Top_2_L5 by auto} \}
  \text{ultimately have } x \in \text{int}(A \cap B) \text{ by auto} \\
  \text{ultimately show } \text{thesis using Neigh_def by auto} \}
  \text{qed}
\}

\text{ultimately from } \langle A \cap B \in \text{Neigh} \rangle \text{ have } \langle A, A \cap B \rangle \in r \land \langle B, A \cap B \rangle \in r \\
\text{using r_def p by auto} \\
\text{ultimately have } \exists z \in \text{Neigh}. \langle A, z \rangle \in r \land \langle B, z \rangle \in r \text{ by auto} \\
\}
\text{ultimately show } \text{thesis using IsDirectedSet_def by auto} \)
\text{qed}

There can be nets directed by the neighborhoods that converge to the point; 
if there is a choice function.

\text{theorem (in topology0) net_direct_neigh_converg:} \\
\text{assumes } x \in \bigcup T \\
\text{defines Neigh} \equiv \{ U \in \text{Pow}(\bigcup T). x \in \text{int}(U) \} \\
\text{defines } r \equiv \{ (U, V) \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 \\
\text{shows } (f, r) \rightarrow_N x \\
\text{proof } - \\
\text{from } \text{assms(4) have } \text{dom_def: } \text{Neigh} = \text{domain}(f) \text{ using Pi_def by auto} \\
\text{moreover} \\
\text{have } \bigcup T \in T \text{ using topSpaceAssum IsATopology_def by auto} \\
\text{then have } \text{int}(\bigcup T) = \bigcup T \text{ using Top_2_L3 by auto} \\
\text{with } \text{assms(1) have } \bigcup T \in \text{Neigh using Neigh_def by auto} \\
\text{then have } \bigcup T \in \text{domain}(\text{fst}((f, r))) \text{ using } \text{dom_def by auto} \\
\text{moreover from } \text{assms(4) dom_def have } \text{fst}((f, r)) : \text{domain}(\text{fst}((f, r))) \rightarrow \bigcup T \\
\text{by auto} \\
\text{moreover from } \text{assms(1, 2, 3) dom_def have } \text{snd}((f, r)) \text{ directs domain}(\text{fst}((f, r))) \\
\text{using directedset_neighborhoods by simp}
ultimately have Net: \( (f,r) \) \{is a net on\} \( \bigcup T \) unfolding IsNet_def by auto
{
  fix \( U \)
  assume \( U \in \text{Pow}(\bigcup T) \times \text{int}(U) \)
  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 \( A: W \in \text{domain}(f) \) \( \langle U, W \rangle \in r \)
    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 \( \exists V \in \text{domain}(f). \ \forall W \in \text{domain}(f). (\langle V, W \rangle \in r \rightarrow fW \in U) \) by auto
}
then have \( \forall U \in \text{Pow}(\bigcup T). \ (x \in \text{int}(U) \rightarrow \exists V \in \text{domain}(f). \ \forall W \in \text{domain}(f). ((V, W) \in r \rightarrow f(W) \in U)) \) by auto
with assms(1) Net show thesis using NetConverges_def by auto
qed

60.2 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\} \( X \equiv (0 \notin \mathcal{F}) \land (\forall x \in \mathcal{F} \land (\exists y \in \text{Pow}(X)) \land (\forall A \in \mathcal{F}. \ \forall B \in \mathcal{F}. \ A \cap B \in \mathcal{F}) \land (\forall B \in \mathcal{F}. \ \forall C \in \text{Pow}(X). \ B \subseteq C \rightarrow C \in \mathcal{F}) \)

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 \( \mathcal{C} \) \{is a base filter\} \( \mathcal{F} \equiv C \subseteq \mathcal{F} \land \mathcal{F} = \{A \in \text{Pow}(\bigcup \mathcal{F}). \ (\exists D \in \mathcal{C}. \ D \subseteq A)\} \)

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 \( \mathcal{C} \) \{satisfies the filter base condition\} \equiv (\forall A \in \mathcal{C}. \ \forall B \in \mathcal{C}. \exists D \in \mathcal{C}. \ D \subseteq A \cap B) \land \mathcal{C} \neq 0 \land 0 \notin \mathcal{C} \)
Every set of a filter contains a set from the filter’s base.

**Lemma basic_element_filter:**

**Assumes** $A \in \mathcal{F}$ and $C$ is a base filter.

**Shows** $\exists D \in C. D \subseteq A$.

**Proof**

- From assumptions (2) have $t : \mathcal{F} = \{ A \in \text{Pow}(\bigcup \mathcal{F}) \. \exists D \in C. D \subseteq A \}$ using IsBaseFilter_def.
  - With assumptions (1) have $A \in \text{Pow}(\bigcup \mathcal{F})$.
  - Then have $A \in \text{Pow}(\bigcup \mathcal{F})$ 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 $\bigcup C$.

**Proof**

- Fix $A, B$.
  - Assume $A \in F$ and $B \in F$.
  - With assumptions (1) have $\exists D \in C. D \subseteq A$ using basic_element_filter by simp.
  - Then obtain $D$ where $D \subseteq A$.
- From assumptions (2) perA perB have $\exists D \in C. D \subseteq A$.
  - Unfolding SatisfiesFilterBase_def by auto.
- Then obtain $D$ where $D \subseteq A$.
  - With subA subB AF BF have $A \cap B \in \text{Pow}(\bigcup \mathcal{F})$. $\exists D \in C. D \subseteq A$ by auto.
  - With assumptions (1) have $A \cap B \in F$ by auto.

**Moreover**

- Fix $A, B$.
  - Assume $A \in F$ and $B \in \text{Pow}(\bigcup \mathcal{F})$ and sub: $A \subseteq B$.
  - From assumptions (1) $A \subseteq B$ using basic_element_filter by auto.
  - Then obtain $D$ where $D \subseteq C$.
  - With sub BS have $B \subseteq \{ A \in \text{Pow}(\bigcup \mathcal{F}) \. \exists D \in C. D \subseteq A \}$ by auto.
  - With assumptions (1) have $B \subseteq C$ unfolding IsBaseFilter_def by auto.

**Moreover**

- From assumptions (2) have $C \neq \emptyset$ using SatisfiesFilterBase_def by auto.
  - Then obtain $D$ where $D \subseteq C$ by auto.
- With assumptions (1) have $D \subseteq \bigcup C$ using IsBaseFilter_def by auto.
  - With $< D \subseteq C \cdot$ have $\bigcup C \in \{ A \in \text{Pow}(\bigcup \mathcal{F}) \. \exists D \in C. D \subseteq A \}$ by auto.
  - With assumptions (1) have $\bigcup C \subseteq F$ unfolding IsBaseFilter_def by auto.

**Moreover**

729
assume $0 \in F$
with assms(1) have $\exists D \in C. D \subseteq 0$ using basic_element_filter by simp
then obtain $D$ where $D \in C \subseteq D \subseteq 0$ by auto
then have $D \in C$ $D = 0$ by auto
with assms(2) have False using SatisfiesFilterBase_def by auto
} then have $0 \notin F$ 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} $\exists$ and $F$ {is a filter on} $\bigcup F$
shows $C$ {satisfies the filter base condition}
proof-

\{
fix $A$ $B$
assume $AF$: $A \in C$ and $BF$: $B \in C$
then have $A \in F$ and $B \in F$ using assms(1) IsBaseFilter_def by auto
then have $A \cap B \in F$ using assms(2) IsFilter_def by auto
then have $\exists D \in C. D \subseteq A \cap B$ using assms(1) basic_element_filter by blast
\}
then have $\forall A \in C. \forall B \in C. \exists D \in C. D \subseteq A \cap B$ by auto
moreover
\{
assume $0 \in C$
then have $0 \in F$ using assms(1) IsBaseFilter_def by auto
then have False using assms(2) IsFilter_def by auto
\}
then have $0 \notin C$ by auto
moreover
\{
assume $C=0$
then have $\exists = 0$ 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} $\exists$
shows $(C$ {satisfies the filter base condition}) $\iff (\exists$ {is a filter on} $\bigcup \exists)$
using assms basic_filter_1 basic_filter_2 by auto

730
A base for a filter determines a filter up to the underlying set.

theorem base_unique_filter:
  assumes C {is a base filter} \& C {is a base filter} \f
  shows \f_1 = \f_2 \iff \bigcup \f_1 = \bigcup \f_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 \( C \subseteq \mathcal{P}(X) \) and \( C \neq 0 \)
  shows \( (C \text{ is a base filter}) \land \bigcup \{A \in \mathcal{P}(X). \exists D \in C. D \subseteq A\} = X \)
proof -
  from assms(1) have \( C \subseteq \mathcal{P}(X) \). \exists D \in C. D \subseteq A \) by auto
  moreover
  from assms(2) obtain D where D \in C by auto
  with \( \langle D \in C \rangle \) have \( X \in \{A \in \mathcal{P}(X). \exists D \in C. D \subseteq A\} \) by auto
  then show \( \bigcup \{A \in \mathcal{P}(X). D \in C. D \subseteq A\} = X \) by auto
  ultimately
  show \( C \text{ is a base filter} \) \& \( \exists D \in C. D \subseteq A \) 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 \( C \subseteq \mathcal{P}(X) \) and \( C \text{ satisfies the filter base condition} \)
  shows \( (C \text{ is a base filter}) \land \bigcup \mathcal{F} = X \) \iff \( \mathcal{F} \subseteq \mathcal{P}(X). \exists D \in C. D \subseteq A \)
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 \( C \subseteq \mathcal{P}(X) \) and \( C \text{ satisfies the filter base condition} \)
  shows \( C \text{ is a base filter} \) \& \( \exists D \in C. D \subseteq A \) and \( \bigcup \{A \in \mathcal{P}(X). \exists D \in C. D \subseteq A\} = X \)
proof -
  let \( \mathcal{F} = \{A \in \mathcal{P}(X). \exists D \in C. D \subseteq A\} \)
  from assms have \( C \text{ is a base filter} \) \& \( \bigcup \mathcal{F} = X \)
  using base_unique_filter_set2 by simp
  thus \( C \text{ is a base filter} \) \& \( \bigcup \mathcal{F} = X \)
  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.
The neighborhoods of a point form a filter that converges to that point.

**Lemma (in topology0) neigh_filter:**

assumes $x \in \bigcup T$

defines $\text{Neigh} \equiv \{ U \in \text{Pow}(\bigcup T). \ x \in \text{int}(U) \}$
suggests $\text{Neigh}$ is a filter on $\bigcup T$ and $\text{Neigh} \rightarrow_f x$

**Proof**

1. **Fix** $A, B$
   - Assume $p : A \in \text{Neigh}$ $B \in \text{Neigh}$
   - Have $A \cap B \in \text{Neigh}$
     - From $p$ have $A \cap B \in \text{Pow}(\bigcup T)$ using $\text{Neigh}\_\text{def}$ by auto
     - Moreover
       - From $p$ 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 $\text{Top}\_2\_L1$ by auto
           - Moreover have $\text{int}(A) \cap \text{int}(B) \subseteq T$
             - Using $\text{Top}\_2\_L2$ $\text{topSpaceAssum}$ $\text{IsATopology}\_\text{def}$ by blast
             - Ultimately have $\text{int}(A) \cap \text{int}(B) \subseteq \text{int}(A \cap B)$ using $\text{Top}\_2\_L5$ by auto
             - Ultimately have $x \in \text{int}(A \cap B)$ by auto
       - Ultimately show thesis using $\text{Neigh}\_\text{def}$ by auto
     -Qed
   - Moreover
     - Fix $A, B$
       - Assume $A \subseteq \text{Neigh}$ and $B \subseteq \text{Pow}(\bigcup T)$ and $\text{sub} : A \subseteq B$
         - From $\text{sub}$ have $\text{int}(A) \in T$ $\text{int}(A) \subseteq B$ using $\text{Top}\_2\_L2$ $\text{Top}\_2\_L1$ by auto
         - Then have $\text{int}(A) \subseteq \text{int}(B)$ using $\text{Top}\_2\_L5$ by auto
         - With $A$ have $x \in \text{int}(B)$ using $\text{Neigh}\_\text{def}$ by auto
         - With $B$ have $B \subseteq \text{Neigh}$ using $\text{Neigh}\_\text{def}$ by auto
     - Moreover
       - Assume $0 \in \text{Neigh}$
         - Then have $x \in \text{interior}(0, T)$ using $\text{Neigh}\_\text{def}$ by auto
         - Then have $x \in 0$ using $\text{Top}\_2\_L1$ by auto
         - Then have False by auto
     - Then have $0 \notin \text{Neigh}$ by auto
   - Moreover
have $\bigcup T \in T$ using topSpaceAssum IsATopology_def by auto
then have $\text{Interior(} \bigcup T, T) = \bigcup T$ using Top_2_L3 by auto
with assms(1) have $ab: \bigcup T \subseteq \text{Neigh}$ unfolding Neigh_def by auto
moreover have $\text{Neigh} \subseteq \text{Pow}(\bigcup T)$ using Neigh_def by auto
ultimately show $\text{Neigh} \{ \text{is a filter on} \} \bigcup T$ using IsFilter_def by auto
moreover from $ab$ have $\bigcup \text{Neigh} = \bigcup T$ unfolding Neigh_def by auto
ultimately show $\text{Neigh} \rightarrow_F x$ 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.

\begin{theorem}[in topology0] convergence_filter_base1:
  assumes $F \{ \text{is a filter on} \} \bigcup T$ and $C \{ \text{is a base filter} \}$
  and $F \rightarrow_F x$
  shows $\forall U \in \text{Pow}(\bigcup T). x \in \text{int}(U) \rightarrow (\exists D \in C. D \subseteq U)$ and $x \in \bigcup T$
\end{theorem}

\begin{proof}-
  \{ fix $U$
  assume $U \subseteq (\bigcup T)$ and $x \in \text{int}(U)$
  with assms(1,3) have $U \in F$ using FilterConverges_def by auto
  with assms(2) have $\exists D \in C. D \subseteq U$ using basic_element_filter by blast
  } thus $\forall U \in \text{Pow}(\bigcup T). x \in \text{int}(U) \rightarrow (\exists D \in C. D \subseteq U)$ 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.

\begin{theorem}[in topology0] convergence_filter_base2:
  assumes $F \{ \text{is a filter on} \} \bigcup T$ and $C \{ \text{is a base filter} \}$
  and $\forall U \in \text{Pow}(\bigcup T). x \in \text{int}(U) \rightarrow (\exists D \in C. D \subseteq U)$ and $x \in \bigcup T$
  shows $F \rightarrow_F x$
\end{theorem}

\begin{proof}-
  \{ fix $U$
  assume $AS: U \in \text{Pow}(\bigcup T)$ $x \in \text{int}(U)$
  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 $U \in \text{Pow}(\bigcup T)$
  using IsBaseFilter_def by auto
  with assms(1) have $U \in F$ 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:**

**Assumptions**
- F is a filter on \( \bigcup T \) and C is a base filter \( \mathcal{F} \).

**Shows**
- \( \mathcal{F} \rightarrow_{F} x \) if and only if \( (\forall U \in \text{Pow}(\bigcup T). \ x \in \text{int}(U) \rightarrow (\exists D \in C. \ D \subseteq U)) \land x \in \bigcup T \).

**Proof**

1. Assume \( \mathcal{F} \rightarrow_{F} x \) with assms.
2. Show \( (\forall U \in \text{Pow}(\bigcup T). \ x \in \text{int}(U) \rightarrow (\exists D \in C. \ D \subseteq U)) \land x \in \bigcup T \).
   - Using convergence_filter_base1 by simp.
3. Next, assume \( (\forall U \in \text{Pow}(\bigcup T). \ x \in \text{int}(U) \rightarrow (\exists D \in C. \ D \subseteq U)) \land x \in \bigcup T \) with assms.
4. Show \( \mathcal{F} \rightarrow_{F} x \) using convergence_filter_base2 by auto.

**Qed**

### 60.3 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**

NetOfFilter \((Net(_)) \) where

\[ \mathcal{F} \rightarrow_{Net(\_)} X \]

is a filter on \( \bigcup \mathcal{F} \) \implies Net(\mathcal{F}) \equiv \langle \{ (A,fst(A)) \mid A \in (\bigcup \mathcal{F}) \times \mathcal{F}. \ x \in F \} \}, \{ (A,B) \mid (A,B) \in (\bigcup \mathcal{F}) \times \mathcal{F}. \ x \in F \} \times (\bigcup \mathcal{F}) \times \mathcal{F}. \ x \in F \} \rangle \}

Net of a filter is indeed a net.

**Theorem** net_of_filter_is_net:

**Assumptions**
- F is a filter on X.

**Shows**
- \( \text{Net}(\mathcal{F}) \) is a net on X.

**Proof**

1. From assms have \( X \in \mathcal{F} \subseteq \text{Pow}(X) \) using IsFilter_def by auto.
2. Then have \( \bigcup \mathcal{F} = X \) by blast.
3. Let \( f = \{ (A,fst(A)) \mid A \in (\bigcup \mathcal{F}) \times \mathcal{F}. \ x \in F \} \}
4. Let \( r = \{ (A,B) \mid (A,B) \in (\bigcup \mathcal{F}) \times \mathcal{F}. \ x \in F \} \times (\bigcup \mathcal{F}) \times \mathcal{F}. \ x \in F \} \} \}
5. Have function(f) using function_def by auto.
6. Moreover have relation(f) using relation_def by auto.
7. Ultimately have \( f: \text{domain}(f) \rightarrow \text{range}(f) \) using function_imp_Pi by auto.
8. Have dom: \( \text{domain}(f) = \{ (x,F) \in (\bigcup \mathcal{F}) \times \mathcal{F}. \ x \in F \} \) by auto.
9. Have range(f) \subseteq \bigcup \mathcal{F} by auto.
10. With \( f: \text{domain}(f) \rightarrow \text{range}(f) \) have \( f: \text{domain}(f) \rightarrow \bigcup \mathcal{F} \) using fun_weaken_type by auto.

Moreover

- \( \text{fix t} \)
- Assume pp:t \in \text{domain}(f).
- Then have \( \text{snd}(t) \subseteq \text{snd}(t) \) by auto.
with dom pp have \( (t,t) \in r \) by auto
}
then have refl(domain(f),r) using refl_def by auto
moreover
{  
fix t1 t2 t3
assume \( (t1,t2) \in r \) \( (t2,t3) \in r \)
then have snd(t3) \subseteq snd(t1) t1 \in domain(f) t3 \in domain(f) using dom
by auto
then have \( (t1,t3) \in r \) by auto
}
then have trans(r) using trans_def by auto
moreover
{  
fix x y
assume as:x \in domain(f)y \in domain(f)
then have snd(x) \cap snd(y) \subseteq \emptyset
by auto
then have \( x,\langle x,snd(x) \cap snd(y) \rangle \) \in r
\( y,\langle y,snd(x) \cap snd(y) \rangle \) \in r
by auto
with as have \( \exists z \in domain(f). (x,z) \in r \) \( (y,z) \in r \) by blast
}
w
\ultimately have r directs domain(f) using IsDirectedSet_def by blast
}
moreover
{  
have p:X \in F and 0 /\in 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 \( q,X \) \in r by auto
then have domain(f) \neq 0 by blast
}
ultimately have \( (f,r) \) \{is a net on\} \( \bigcup F \) using IsNet_def by auto
then show \( (\text{Net}(F)) \) \{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 $\mathcal{F}$ {is a filter on} $\bigcup T$ and $\mathcal{F} \to_F x$
  shows $(\text{Net}(\mathcal{F})) \to_N x$
proof-
  from assms have $\bigcup T \in \mathcal{F}, \mathcal{F} \subseteq \text{Pow}(\bigcup T)$ using IsFilter_def by auto
  then have uu: $\bigcup F = \bigcup T$ by blast from assms(1)
  have fun: $\text{fst}(\text{Net}(F)) = \{\langle A, \text{fst}(A) \rangle. A \in \{\langle x, F \rangle \in (\bigcup F) \times F. x \in F\}\}$
    using NetOfFilter_def uu by auto
  then have dom_def: $\text{domain}(\text{fst}(\text{Net}(F))) = \{\langle x, F \rangle \in (\bigcup F) \times F. x \in F\}$ by auto
  from fun have fun: $\text{fst}(\text{Net}(F)) : \{\langle x, F \rangle \in (\bigcup F) \times F. x \in F\} \to (\bigcup F)$
    using ZF_fun_from_total by simp
  from assms(1) have NN: $(\text{Net}(F)) \text{ is a net on } \bigcup T$ 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: $U \in \text{Pow}(\bigcup T), x \in \text{int}(U)$
    with assms have $x \in F, U \subseteq F$ using Top_2_L1 FilterConverges_def by auto
    then have pp: $\langle x, U \rangle \in \text{domain}(\text{fst}(\text{Net}(F)))$ using dom_def by auto
    { fix $m$
      assume ASS: $m \in \text{domain}(\text{fst}(\text{Net}(F))), \langle x, U \rangle, m \in \text{snd}(\text{Net}(F))$
        from ASS(1) fun have $\text{fst}(\text{Net}(F))(m) = \text{fst}(m)$
          using func1_1_L1 ZF_fun_from_tot_val by simp
        with dir ASS have $\text{fst}(\text{Net}(F))(m) \in U$ using dom_def by auto
      }
      then have $\forall m \in \text{domain}(\text{fst}(\text{Net}(F))). (\langle x, U \rangle, m) \in \text{snd}(\text{Net}(F)) \longrightarrow \text{fst}(\text{Net}(F))m \in U)$
        by auto
      with pp have $\exists t \in \text{domain}(\text{fst}(\text{Net}(F))). \forall m \in \text{domain}(\text{fst}(\text{Net}(F))). (\langle t, m \rangle) \in \text{snd}(\text{Net}(F)) \longrightarrow \text{fst}(\text{Net}(F))m \in U)$
        by auto
    }
    then have $\forall U \in \text{Pow}(\bigcup T). (x \in \text{int}(U) \longrightarrow (\exists t \in \text{domain}(\text{fst}(\text{Net}(F))). \forall m \in \text{domain}(\text{fst}(\text{Net}(F))). (\langle t, m \rangle) \in \text{snd}(\text{Net}(F)) \longrightarrow \text{fst}(\text{Net}(F))m \in U))$
      by auto
    ultimately show thesis using NetConverges_def by auto
  }

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 $\mathcal{F}$ {is a filter on} $\bigcup T$ and $(\text{Net}(\mathcal{F})) \to_N x
shows $\mathcal{F} \rightarrow_F x$

proof:
from assms have $\bigcup T \in \mathcal{F} \subseteq \mathcal{P}(\bigcup T)$ using IsFilter_def by auto
then have uu: $\bigcup \mathcal{F} = \bigcup T$ by blast
have \(x \in \bigcup T\) using assms NetConverges_def net_of_filter_is_net by auto
moreover
{ fix \(U\)
assume \(U \in \mathcal{P}(\bigcup T)\) \(x \in \text{int}(U)\)
then obtain \(t\) where \(t\) : \(t \in \text{domain}(\text{fst}(\text{Net}(\mathcal{F})))\) and \(\text{reg}: \forall m \in \text{domain}(\text{fst}(\text{Net}(\mathcal{F}))). \langle t, m \rangle \in \text{snd}(\text{Net}(\mathcal{F})) \rightarrow \text{fst}(\text{Net}(\mathcal{F}))m \in U\)
using assms net_of_filter_is_net NetConverges_def by blast
with assms(1) uu obtain \(t_1\) \(t_2\) where \(t_{\text{def}}: t = \langle t_1, t_2 \rangle\) and \(t_2 \in \mathcal{F}\)
using NetOfFilter_def by auto
}

{ fix \(s\)
assume \(s \in t_2\)
then have \(\langle s, t_2 \rangle \in \bigcup \mathcal{F} \times \mathcal{F}. q_1 \in q_2\) using tFF by auto
moreover
from assms(1) uu have \(\text{domain}(\text{fst}(\text{Net}(\mathcal{F}))) = \bigcup \mathcal{F} \times \mathcal{F}. q_1 \in q_2\)
using NetOfFilter_def
by auto
ultimately
have \(tt: \langle s, t_2 \rangle \in \text{domain}(\text{fst}(\text{Net}(\mathcal{F})))\) by auto
moreover
from assms(1) uu \(t_{\text{def}}\) \(tt\) have \(\langle \langle t_1, t_2 \rangle, \langle s, t_2 \rangle \rangle \in \text{snd}(\text{Net}(\mathcal{F}))\)
using NetOfFilter_def
by auto
ultimately
have \(\text{function}(\text{fst}(\text{Net}(\mathcal{F})))\) \(\bigcup \mathcal{F} \subseteq U\) using \(\text{reg}\) \(t_{\text{def}}\) by auto
moreover
from assms(1) uu have \(\text{function}(\text{fst}(\text{Net}(\mathcal{F})))\) using NetOfFilter_def
function_def
by auto
moreover
from \(tt\) assms(1) uu have \(\langle \langle s, t_2 \rangle, s \rangle \in \text{fst}(\text{Net}(\mathcal{F}))\)
using NetOfFilter_def
by auto
ultimately
have \(s \in U\)
using NetOfFilter_def function_apply_equality by auto
}
then have \(t_2 \subseteq U\) by auto
with tFF assms(1) \(U \in \mathcal{P}(\bigcup T)\) have \(U \in \mathcal{F}\)
using IsFilter_def by auto
}
then have \(\{ U \in \mathcal{P}(\bigcup T). x \in \text{int}(U) \} \subseteq \mathcal{F}\)
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.
Theorem (in topology0) filter_conver_iff_net_of_filter_conver:
assumes \( F \) {is a filter on} \( T \)
shows \( (F \rightarrow F \ x) \iff ((\text{Net}(F)) \rightarrow_N x) \)
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 \( \text{Net}(F) \) is a net that doesn’t come into our attention as a first choice; maybe because we restrict ourselves to the antisymmetry 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.

definition FilterOfNet (Filter (_ .. _) 40) where
\( (N \ {is a net on} X) \implies \text{Filter N..X} \equiv \{A \in \text{Pow}(X). \exists D \in \{\{\text{fst}(N)\text{snd}(s). s \in \{s \in \text{domain(fst(N))} \times \text{domain(fst(N)). s \in \text{snd}(N) \land \text{fst}(s)=t0}\}. t0 \in \text{domain(fst(N))}. D \subseteq A\} \)

Filter of a net is indeed a filter

Theorem filter_of_net_is_filter:
assumes \( N \ {is a net on} X \)
shows \( \{\text{Filter N..X} \ {is a filter on} X \ and \{\{\text{fst}(N)\text{snd}(s). s \in \{s \in \text{domain(fst(N))} \times \text{domain(fst(N)). s \in \text{snd}(N) \land \text{fst}(s)=t0}\}. t0 \in \text{domain(fst(N))}\}\ \ {is a base filter}\ \} \ {is a filter on} X \ and \)
\( \{\{\text{fst}(N)\text{snd}(s). s \in \{s \in \text{domain(fst(N))} \times \text{domain(fst(N)). s \in \text{snd}(N) \land \text{fst}(s)=t0}\}. t0 \in \text{domain(fst(N))}\}\ \ {is a base filter}\ \} \ {is a filter on} X \ and \)
proof -
\( \text{let} C = \{\{\text{fst}(N)\text{snd}(s). s \in \{s \in \text{domain(fst(N))} \times \text{domain(fst(N)). s \in \text{snd}(N) \land \text{fst}(s)=t0}\}. t0 \in \text{domain(fst(N))}\}\ \ {is a base filter}\ \} \ {is a filter on} X \ and \)
\( \text{have} C \subseteq \text{Pow}(X) \)
proof -
\{ 
\( \text{fix} t \)
\( \text{assume} t \in C \)
\( \text{then obtain} t1 \text{ where} t1 \in \text{domain(fst(N))} \text{ and} \)
\( t\text{Def}: t = \{\text{fst}(N)\text{snd}(s). s \in \{s \in \text{domain(fst(N))} \times \text{domain(fst(N)). s \in \text{snd}(N) \land \text{fst}(s)=t1}\}\} \)
by auto
\{ 
\( \text{fix} x \)
\( \text{assume} x \in t \)
\}

738
with t_Def obtain ss where ss∈\{s∈domain(fst(N))×domain(fst(N)). s∈snd(N) ∧ fst(s)=t1\} and
  x_def: x = fst(N)(snd(ss)) by blast
then have snd(ss) ∈ domain(fst(N)) by auto
from assms have fst(N):domain(fst(N))→X unfolding IsNet_def
by simp
  with <snd(ss) ∈ domain(fst(N))> have x∈X using apply_funtype
x_def
  by auto
  hence t⊆X by auto
} thus thesis by blast qed
have sat: C {satisfies the filter base condition}
proof -
  from assms obtain t1 where t1∈domain(fst(N)) using IsNet_def by blast
  hence \{fst(N)snd(s). s∈\{s∈domain(fst(N))×domain(fst(N)). s∈snd(N) ∧ fst(s)=t1\}\}∈C
  by auto
  hence C≠0 by auto
moreover
  \{ 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 \langle q,q \rangle∈snd(N) ∧ fst(\langle q,q \rangle)=q unfolding IsNet_def
    IsDirectedSet_def refl_def
    by auto
    with q_dom have \langle q,q \rangle∈\{s∈domain(fst(N))×domain(fst(N)). s∈snd(N) ∧ fst(s)=q\}
    by auto
    with U_def have fst(N)(snd(\langle q,q \rangle)) ∈ 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 $B \in C$

with $pA$ obtain $qA$ $qB$ where per: $qA \in \text{domain}(f(N))$, $qB \in \text{domain}(f(N))$

and 

$A\_def$: $A=\{f(N)\text{snd}(s). s \in \{s \in \text{domain}(f(N)) \times \text{domain}(f(N)). s\}_{s \in \text{snd}(N)} \}$

$B\_def$: $B=\{f(N)\text{snd}(s). s \in \{s \in \text{domain}(f(N)) \times \text{domain}(f(N)). s\}_{s \in \text{snd}(N)} \}$

by blast

have dir: $\text{snd}(N)$ directs $\text{domain}(f(N))$ using assms IsNet\_def

by auto

with per obtain $qD$ where ine: $\langle qA,qD \rangle \in \text{snd}(N)$, $\langle qB,qD \rangle \in \text{snd}(N)$

and

perD: $qD \in \text{domain}(f(N))$ unfolding IsDirectedSet\_def

by blast

let $D = \{f(N)\text{snd}(s). s \in \{s \in \text{domain}(f(N)) \times \text{domain}(f(N)). s\}_{s \in \text{snd}(N)} \}$

from perD have $D \in C$ by auto

moreover

{ 
  fix $d$
  assume $d \in D$
  then obtain $sd$ where $sd \in \{s \in \text{domain}(f(N)) \times \text{domain}(f(N)). s\}_{s \in \text{snd}(N)}$

  $d\_def$: $d=f(N)\text{snd}(sd)$ by blast

  then have $sdN$: $sd \in \text{snd}(N)$ and $qdd$: $\text{fst}(sd)=qD$

  by auto

  then obtain $qI$ $aa$ where $sd=\langle aa,qI \rangle$ $qI \in \text{domain}(f(N))$

  by auto

  with $qdd$ have $sd\_def$: $sd=\langle qD,qI \rangle$ and $qIdom$: $qI \in \text{domain}(f(N))$

  by auto

  with $sdN$ have $\langle qD,qI \rangle \in \text{snd}(N)$ by auto

  from dir have trans($\text{snd}(N)$) unfolding IsDirectedSet\_def by auto

  then have $\langle qA,qD \rangle \in \text{snd}(N) \land \langle qD,qI \rangle \in \text{snd}(N) \longrightarrow \langle qA,qI \rangle \in \text{snd}(N)$

  and 

  $\langle qB,qD \rangle \in \text{snd}(N) \land \langle qD,qI \rangle \in \text{snd}(N) \longrightarrow \langle qB,qI \rangle \in \text{snd}(N)$

  using trans\_def by auto

  with ine $\langle qD,qI \rangle \in \text{snd}(N)$ have $\langle qA,qI \rangle \in \text{snd}(N)$, $\langle qB,qI \rangle \in \text{snd}(N)$

  by auto

  with $qIdom$ per have $\langle qA,qI \rangle \in \{s \in \text{domain}(f(N)) \times \text{domain}(f(N)). s\}_{s \in \text{snd}(N)}$

  $s\in\text{snd}(N) \land \text{fst}(s)=qA$

  $\langle qB,qI \rangle \in \{s \in \text{domain}(f(N)) \times \text{domain}(f(N)). s\}_{s \in \text{snd}(N)}$

  by auto

  then have $\text{fst}(N)(qI) \in A \cap B$ using $A\_def$ $B\_def$ by auto

  then have $\text{fst}(N)(\text{snd}(sd)) \in A \cap B$ using $sd\_def$ by auto

  then have $d \in A \cap B$ using $d\_def$ by auto

740
then have $D \subseteq A \cap B$ by blast
ultimately show $\exists D \in C$. $D \subseteq A \cap B$ by blast
qed
qed ultimately
show thesis unfolding SatisfiesFilterBase_def by blast
have
Base: $C$ {is a base filter} {A ∈ Pow(X). $\exists D \in C$. $D \subseteq A$}
proof -
  from $<C \subseteq \text{Pow}(X)>$ sat show $C$ {is a base filter} {A ∈ Pow(X). $\exists D \in C$. $D \subseteq A$}=$X$
  by (rule base_unique_filter_set3)
from $<C \subseteq \text{Pow}(X)>$ sat show $\bigcup \{A \in \text{Pow}(X). \exists D \in C. D \subseteq A\} = X$
by (rule 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 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 $N \rightarrow_N x$
sows ($\text{Filter N..(} \bigcup T) \rightarrow_F x$
proof -
  let $C = \{(\text{fst}(N)snd(s). s \in \{s \in \text{domain(fst(N))} \times \text{domain(fst(N))}. s \in \text{snd(N)}$
\wedge \text{fst(s)=t}\}. t \in \text{domain(fst(N))})$
from assms(1) have ($\text{Filter N..(} \bigcup T) \rightarrow_F x$
using filter_of_net_is_filter by auto
moreover have $\forall U \in \text{Pow}(\bigcup T). x \in \text{int}(U) \rightarrow (\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 $\exists t \in \text{domain(fst(N))}. (\forall m \in \text{domain(fst(N))}. (\langle t, m \rangle \in \text{snd(N)}$
\wedge $\text{fst}(N)m \in U))$
    using NetConverges_def by auto
    then obtain $t$ where $t \in \text{domain(fst(N))}$ and
    reg: $\forall m \in \text{domain(fst(N))}. (\langle t, m \rangle \in \text{snd(N)} \rightarrow \text{fst}(N)m \in U)$ by auto
    { fix $f$
assume \( f \in \{ \text{fst}(N) \, \text{snd}(s) \mid s \in \text{domain}(\text{fst}(N)) \times \text{domain}(\text{fst}(N)) \land \text{fst}(s) = t \} \)

then obtain \( s \) where \( s \in \{ s \in \text{domain}(\text{fst}(N)) \times \text{domain}(\text{fst}(N)) \mid s \in \text{snd}(N) \land \text{fst}(s) = t \} \)

\( f \) def: \( f = \text{fst}(N) \, \text{snd}(s) \)

by \( \text{blast} \)

hence \( s \in \text{domain}(\text{fst}(N)) \times \text{domain}(\text{fst}(N)) \land s \in \text{snd}(N) \land \text{fst}(s) = t \)

by \( \text{auto} \)

hence \( s = \langle t, \text{snd}(s) \rangle \land \text{snd}(s) \in \text{domain}(\text{fst}(N)) \)

by \( \text{auto} \)

 moreover from asms have \( x \in \bigcup T \) using \( \text{NetConverges_def} \) by auto

ultimately show \( (\text{Filter } N.\, (\bigcup T)) \to F \, x \) by (rule convergence_filter_base2)

qed

Convergence of a filter corresponding to a net implies convergence of the net.

\[ \text{theorem (in topology0) filter_of_net_conver_net_conver:} \]

\[ \text{assumes } N \text{ \{is a net on\} } \bigcup T \text{ and } (\text{Filter } N.\, (\bigcup T)) \to F \, x \]

\[ \text{shows } N \to_N x \]

\[ \text{proof -} \]

\[ \text{let } C = \{ \{ \text{fst}(N) \, \text{snd}(s) \mid s \in \{ s \in \text{domain}(\text{fst}(N)) \times \text{domain}(\text{fst}(N)) \land s \in \text{snd}(N) \land \text{fst}(s) = t \} \} \} \]

\[ t \in \text{domain}(\text{fst}(N)) \]

from asms have 1: \( (\text{Filter } N.\, (\bigcup T)) \text{ \{is a filter on\} } (\bigcup T) \)

\[ C \text{ \{is a base filter\} } (\text{Filter } N.\, (\bigcup T)) \quad (\text{Filter } N.\, (\bigcup T)) \to F \, x \]

using \( \text{filter_of_net_is_filter} \) by auto

then have reg: \( \forall U \in \text{Pow}(\bigcup T). \quad x \in \text{int}(U) \to (\exists D \in C. \quad D \subseteq U) \) by auto

(qed)

742
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} $\bigcup T$ shows $((\text{Filter } N..(\bigcup T)) \rightarrow F \ x) \longleftrightarrow (N \rightarrow_N x)$ using assms filter_of_net_conver_net_conver net_conver_filter_of_net_conver by auto

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 $\mathcal{F}$ {is a filter on} $\bigcup \mathcal{F}$ shows $(\mathcal{F} \cup \{0\})$ {is a topology} proof -

```
{ fix A B
 assume A\in(\mathcal{F} \cup \{0\})B\in(\mathcal{F} \cup \{0\})
 then have (A\in\mathcal{F} \land B\in\mathcal{F}) \lor (A\cap B=0) by auto
 with assms have A\cap B\in(\mathcal{F} \cup \{0\}) unfolding IsFilter_def by blast
}
```
then have \( \forall A \in (\mathcal{F} \cup \{0\}). \forall B \in (\mathcal{F} \cup \{0\}). A \cap B \in (\mathcal{F} \cup \{0\}) \) by auto
moreover
\[
\begin{align*}
&\text{fix } M \\
&\text{assume } A \in \mathcal{F} \cup \{0\} \\
&\text{then have } M \in \mathcal{F} \cup \{0\}. T \in \mathcal{F} \text{ by blast} \\
&\text{then obtain } T \text{ where } M = 0 \vee (T \in \mathcal{F} \land T \in M) \text{ by auto} \\
&\text{moreover from this } A \text{ have } \bigcup M \in \mathcal{F} \cup \{0\} \text{ by auto} \\
&\text{ultimately have } \bigcup M \in \mathcal{F} \cup \{0\} \text{ using IsFilter_def assms by auto} \\
&\text{then have } \bigcup M \in \mathcal{F} \cup \{0\} \text{ by auto} \\
&\text{ultimately show thesis using IsATopology_def by auto} \\
\end{align*}
\]
qed

We can use topology0 locale with filters.

**lemma** topology0_filter:

\[
\begin{align*}
&\text{assumes } \mathcal{F} \text{ is a filter on } \bigcup \mathcal{F} \\
&\text{shows } \text{topology0}(\mathcal{F} \cup \{0\}) \\
&\text{using top_of_filter topology0_def assms by auto} \\
\end{align*}
\]

The next abbreviation introduces notation where we want to specify the space where the filter convergence takes place.

**abbreviation** FilConvTop(_ \rightarrow_F _ {in} _)

\[
\begin{align*}
&\text{where } \mathcal{F} \rightarrow_F x \{\text{in}\} T \equiv \text{topology0.FilterConverges}(T, \mathcal{F}, x) \\
\end{align*}
\]

The next abbreviation introduces notation where we want to specify the space where the net convergence takes place.

**abbreviation** NetConvTop(_ \rightarrow_N _ {in} _)

\[
\begin{align*}
&\text{where } N \rightarrow_N x \{\text{in}\} T \equiv \text{topology0.NetConverges}(T, N, x) \\
\end{align*}
\]

Each point of a the union of a filter is a limit of that filter.

**lemma** lim_filter_top_of_filter:

\[
\begin{align*}
&\text{assumes } \mathcal{F} \text{ is a filter on } \bigcup \mathcal{F} \text{ and } x \in \bigcup \mathcal{F} \\
&\text{shows } \mathcal{F} \rightarrow_F x \{\text{in}\} (\mathcal{F} \cup \{0\}) \text{ using top_of_filter topology0_def assms by auto} \\
\end{align*}
\]

proof-

\[
\begin{align*}
&\text{have } \bigcup \mathcal{F} = \bigcup (\mathcal{F} \cup \{0\}) \text{ by auto} \\
&\text{with assms(1) have assms1: } \mathcal{F} \text{ is a filter on } \bigcup (\mathcal{F} \cup \{0\}) \text{ by auto} \\
&\text{fix } U \\
&\text{assume } U \in \text{Pow}(\bigcup (\mathcal{F} \cup \{0\})) \text{ x \in Interior}(U, (\mathcal{F} \cup \{0\})) \\
&\text{with assms(1) have Interior}(U, (\mathcal{F} \cup \{0\})) \in \mathcal{F} \text{ using topology0_def top_of_filter topology0.Top_2_L2 by blast} \\
&\text{moreover} \\
&\text{from assms(1) have Interior}(U, (\mathcal{F} \cup \{0\})) \subseteq U \text{ using topology0_def top_of_filter topology0.Top_2_L1 by auto} \\
&\text{moreover} \\
\end{align*}
\]
61 Topology and neighborhoods

This theory considers the relations between topology and systems of neighborhood filters.

61.1 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**

\[ \text{IsNeighSystem} \quad (\text{is a neighborhood system on} \quad X) \quad \equiv \quad (\forall x \in X. (M(x) \quad \text{is a filter on} \quad X) \quad \land \quad (\forall N \in M(x). \quad x \in N \quad \land \quad (\exists U \in M(x). \quad \forall y \in U. (N \in M(y)))) \]

A neighborhood system on \( X \) consists of collections of subsets of \( X \).

**lemma neighborhood_subset:**

**assumes** \( M \quad \text{is a neighborhood system on} \quad X \quad \text{and} \quad x \in X \quad \text{and} \quad N \in M(x) \)

**shows** \( N \subseteq X \quad \text{and} \quad x \in N \)

**proof** -

**from** \( M \quad \text{is a neighborhood system on} \quad X \quad \text{have} \quad M : X \rightarrow \text{Pow}(\text{Pow}(X)) \)

**unfolding** \( \text{IsNeighSystem_def} \quad \text{by simp} \)

**with** \( x \in X \quad \text{have} \quad M(x) \subseteq \text{Pow}(\text{Pow}(X)) \quad \text{using} \quad \text{apply_funtype} \quad \text{by blast} \)

**with** \( N \subseteq M(x) \quad \text{show} \quad N \subseteq X \quad \text{by blast} \)

**from** \( \text{assms} \quad \text{show} \quad x \in N \quad \text{using} \quad \text{IsNeighSystem_def} \quad \text{by simp} \)

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 $\exists U \in \mathcal{M}(x). U \subseteq N \land (\forall y \in U. (N \in \mathcal{M}(y)))$

**Proof**

- from assms obtain $\mathcal{W}$ where $\mathcal{W} \subseteq \mathcal{M}(x)$ and areNeigh: $\forall y \in \mathcal{W}. (N \in \mathcal{M}(y))$
  - unfolding IsNeighSystem_def by blast
  - let $U = N \cap \mathcal{W}$ from assms have $U \subseteq N$ by blast
  - moreover from areNeigh have $\forall y \in U. (N \in \mathcal{M}(y))$ by auto
  - ultimately show thesis by auto
  
  qed

### 61.2 Topology from neighborhood systems

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 $\mathcal{M}$ is a neighborhood system on $X$

defines $T_{\text{def}}$: $T \equiv \{U \in \mathcal{P}(X). \forall x \in U. U \in \mathcal{M}(x)\}$

shows $T$ is a topology and $\bigcup T = X$

**Proof**

- fix $U$ assume $U \in \mathcal{P}(T)$
  - have $\bigcup U \in T$
    - proof
      - from $<U \in \mathcal{P}(T)>$ $T_{\text{def}}$ have $\bigcup U \in \mathcal{P}(X)$ by blast
      - moreover
        - fix $x$ assume $x \in \bigcup U$
          - then obtain $U$ where $U \subseteq U$ and $x \in U$ by blast
          - with assms $<U \in \mathcal{P}(T)>$
            - have $U \in \mathcal{M}(x)$ and $U \subseteq \bigcup U$ and $\mathcal{M}(x)$ is a filter on $X$
              - unfolding IsNeighSystem_def by auto
              - with $<\bigcup U \in \mathcal{P}(X)>$ have $\bigcup U \in \mathcal{M}(x)$ unfolding IsFilter_def by simp
        
        ultimately show $\bigcup U \in T$ using $T_{\text{def}}$ by blast
      
    qed

  moreover

  - fix $U V$ assume $U \in T$ and $V \in T$
    - have $U \cap V \in T$
proof -
  from Tdef <U∈T> <U∈T> have U∩W ∈ Pow(X) by auto
  moreover
  { fix x assume x ∈ U∩W
    with assms <U∈T> <V∈T> Tdef have U ∈ M(x) V ∈ M(x) and M(x)
    {is a filter on} X
      unfolding IsNeighSystem_def by auto
    then have U∩W ∈ M(x) unfolding IsFilter_def by simp
  }
  ultimately show U∩W ∈ T using Tdef by simp
  qed
}
ultimately show T {is a topology} unfolding IsATopology_def by blast

from assms show ∪T = X unfolding IsNeighSystem_def IsFilter_def by blast
qed

Some sources (like Wikipedia) define the open sets generated by a neigh-
borhood 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 {U∈Pow(X). ∀x∈U. U ∈ M(x)} = {U∈Pow(X). ∀x∈U. ∃V ∈ M(x). V ⊆ U}
proof
  let T = {U∈Pow(X). ∀x∈U. U ∈ M(x)}
  let S = {U∈Pow(X). ∀x∈U. ∃V ∈ M(x). V ⊆ U}
  show S ⊆ T
  proof -
    { fix U assume U∈S
      then have U∈Pow(X) by simp
      moreover
        from assms <U∈S> <U∈Pow(X)> have ∀x∈U. U ∈ M(x)
          unfolding IsNeighSystem_def IsFilter_def by blast
        ultimately have U∈T by auto
      }
      thus thesis by auto
    qed
  show T ⊆ S by auto
  qed

61.3 Neighborhood system from topology

Once we have a topology T we can define a natural neighborhood system
on X = ∪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 a collection supersets of open sets containing $x$. We call that the "neighborhood system of $T$"

**definition**

$\text{NeighSystem} \ (\{\text{neighborhood system of}\} \ T) \equiv \{ \langle x, \{ V \in \bigcup T. \exists U \in T. (x \in U \land U \subseteq V) \} \rangle \mid x \in \bigcup T \}$

The next lemma shows that open sets are members of (what we will prove later to be) the natural neighborhood system on $X = \bigcup T$.

**lemma open_are_neighs:**

assumes $U \in T$ $x \in U$

shows $x \in \bigcup T$ and $U \in \{ V \in \bigcup T. \exists U \in T. (x \in U \land U \subseteq V) \}$

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 $T \ {\text{is a topology}}$ and $x \in \bigcup T$

defines $M \equiv \{ \text{neighborhood system of} \ T \}$

shows $M(x) \ {\text{is a filter on}} \ (\bigcup T)$

proof
- let $X = \bigcup T$
- let $\mathcal{F} = \{ V \in \text{Pow}(X). \exists U \in T. (x \in U \land U \subseteq V) \}$
- have $0 \notin \mathcal{F}$ by blast
- moreover have $X \in \mathcal{F}$

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 $\forall A \in \mathcal{F}$. $\forall B \in \mathcal{F}$. $A \cap B \in \mathcal{F}$

proof
- $\langle \text{fix} \ A \ B \ \text{assume} \ A \in \mathcal{F} \ B \in \mathcal{F} \ \text{then obtain} \ U_A \ U_B \ \text{where} \ U_A \in T \ x \in U_A \ U_A \subseteq A \ U_B \in T \ x \in U_B \ U_B \subseteq B \ \text{by auto} \rangle$

with $\langle T \ {\text{is a topology}} \rangle$ $\langle A \in \mathcal{F} \rangle$ $\langle B \in \mathcal{F} \rangle$ have $A \cap B \in \text{Pow}(X)$ and $U_A \cap U_B \in T$ $x \in U_A \cap U_B$ $U_A \cap U_B \subseteq A \cap B$ using IsATopology_def

by auto
- hence $A \cap B \in \mathcal{F}$ by blast

thus thesis by blast

qed

moreover have $\forall B \in \mathcal{F}$. $\forall C \in \text{Pow}(X)$. $B \subseteq C$ $\rightarrow$ $C \in \mathcal{F}$

proof
- $\langle \text{fix} \ B \ C \ \text{assume} \ B \in \mathcal{F} \ C \in \text{Pow}(X) \ B \subseteq C \ \text{then obtain} \ U \ \text{where} \ U \in T \ \text{and} \ x \in U \ U \subseteq B \ \text{by blast} \rangle$

with $\langle C \in \text{Pow}(X) \rangle$ $\langle B \subseteq C \rangle$ have $C \in \mathcal{F}$ by blast

748
\{\text{thus thesis by auto}\}
\begin{array}{l}
\text{qed}
\end{array}
\text{ultimately have } \exists \exists \text {is a filter on} \ X \text{ unfolding } \text{IsFilter_def by blast}
\text{with } \text{Hdef} \ x \in X \text{ show } M(x) \text{ is a filter on} \ X \text{ using } ZF\text{\_fun_from_tot_val1}
\text{NeighSystem_def}
\text{by } \text{simp}
\text{qed}

The next theorem states that the the natural neighborhood system on \( X = \bigcup T \) indeed is a neighborhood system.

\text{theorem neigh_from_topology:}
\text{assumes } T \text{ is a topology}
\text{shows } (\text{neighborhood system of} \ T) \text{ is a neighborhood system on} \ (\bigcup T)
\text{proof -}
\text{let } X = \bigcup T
\text{let } M = \text{neighborhood system of} \ T
\text{have } M : X \to \text{Pow(Pow(X))}
\text{proof -}
\{ \text{fix } x \text{ assume } x \in X
\text{ hence } \{V \in \text{Pow(} \bigcup T \}. \exists U \in T. (x \in U \land U \subseteq V) \} \in \text{Pow(Pow(X))} \text{ by auto}
\} \text{ hence } \forall x \in X. \{V \in \text{Pow(} \bigcup T \}. \exists U \in T. (x \in U \land U \subseteq V) \} \in \text{Pow(Pow(X))} \text{ by auto}
\text{then show thesis using } ZF\text{\_fun_from_total NeighSystem_def by simp}
\text{qed}
\text{moreover from } \text{assms have } \forall x \in X. (M(x) \text{ is a filter on} \ X)
\text{using neighs_is_filter NeighSystem_def by auto}
\text{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)))
\text{proof -}
\{ \text{fix } x \text{ N assume } x \in X \text{ N } \in M(x)
\text{ let } \exists = \{V \in \text{Pow(X).} \exists U \in T. (x \in U \land U \subseteq V)\}
\text{from } <x \in X> \text{ have } M(x) = \exists using ZF\text{\_fun_from_tot_val1 NeighSystem_def}
\text{by simp}
\text{with } <N \in M(x)> \text{ have } N \in \exists \text{ by simp}
\text{hence } x \in N \text{ by blast}
\text{from } <N \in \exists> \text{ obtain } U \text{ where } U \in T \ x \in U \text{ and } U \subseteq N \text{ by blast}
\text{with } <N \in \exists> \text{ <M(x) = \exists> have } U \in M(x) \text{ by auto}
\text{moreover from } \text{assms <U \in T> <U \subseteq N> <N \in \exists> have } \forall y \in U. (N \in M(y))
\text{using ZF\text{\_fun_from_tot_val1 open_are_neighs neighs_is_filter NeighSystem_def IsFilter_def by auto}
\text{ultimately have } \exists U \in M(x). \forall y \in U. (N \in M(y)) \text{ by blast}
\text{with } <x \in N> \text{ have } x \in N \land (\exists U \in M(x). \forall y \in U. (N \in M(y))) \text{ by simp}
\} \text{ thus thesis by auto}
\text{qed}
\text{ultimately show thesis unfolding } \text{IsNeighSystem_def by blast}
\text{qed}
\text{end}

749
62 Topology - examples

theory Topology_ZF_examples imports Topology_ZF Cardinal_ZF

begin

This theory deals with some concrete examples of topologies.

62.1 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
CoCardinal$(X,T)$ ≡ \{F∈\Pow(X). X-F ≺ T\}∪\{0\}

For any set and any infinite cardinal we prove that CoCardinal$(X,Q)$ forms a topology. The proof is done with an infinite cardinal, but it is obvious that the set $Q$ 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 $Q=1$ 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:$M\in\Pow(T)$
  hence $M\subseteq T$ by auto
  then have $M\subseteq\Pow(X)$ using CoCardinal_def by auto
  then have $\bigcup M\subseteq\Pow(X)$ by auto
  moreover
  { assume B:$M=0$
    then have $\bigcup M\subseteq T$ using CoCardinal_def by auto
  }
  moreover
  { assume B:$M=\{0\}$
    then have $\bigcup M\subseteq T$ using CoCardinal_def by auto
  }
}
assume B: M ≠ 0 M̸=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)∪(X-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
assms 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 topology.

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 T1. 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))
62.2 **Total set, Closed sets, Interior, Closure and Boundary**

There are several assertions that can be done to the \( \text{CoCardinal}(X, T) \) topology. In each case, we will not assume sufficient conditions for \( \text{CoCardinal}(X, T) \) 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** \( T \neq 0 \)
- **Shows** \( \bigcup \text{CoCardinal}(X, T) = X \)

**Proof**
1. Have \( X : X - X = 0 \) by auto
2. Have \( 0 \preceq 0 \) by auto
   - With assumptions have \( 0 \preceq 11 \preceq T \) using \( \text{not}_0 \_\text{is}_\text{lepoll}_1 \_\text{lepoll}_\text{imp}_\text{lesspoll}_\text{succ} \) by auto
   - Then have \( 0 \preceq T \) using \( \text{lesspoll}_\text{trans} \) by auto
   - With \( X \) have \( (X - X) - T \) by auto
   - Then have \( X \in \text{CoCardinal}(X, T) \) using \( \text{CoCardinal}_\text{def} \) by auto
   - Hence \( X \in \bigcup \text{CoCardinal}(X, T) \) using \( \text{CoCardinal}_\text{def} \) by auto

**QED**

The closed sets are the small subsets of \( X \) and \( X \) itself.

**Lemma closed_sets_cocardinal:**
- **Assumes** \( T \neq 0 \)
- **Shows** \( D \{ \text{is closed in} \ \text{CoCardinal}(X, T) \} \iff (D \in \text{Pow}(X) \land D \prec T) \lor D = X \)

**Proof**
1. Assume \( A : D \subseteq X \) \( - D \in \text{CoCardinal}(X, T) \) \( D \neq X \)
   - From \( A(1,3) \) have \( X - (X - D) = D \) \( X \neq 0 \) by auto
   - With \( A(2) \) have \( D \prec T \) using \( \text{CoCardinal}_\text{def} \) by simp
2. With assumptions have \( D \{ \text{is closed in} \ \text{CoCardinal}(X, T) \} \rightarrow (D \in \text{Pow}(X) \land D \prec T) \lor D = X \) using \( \text{IsClosed}_\text{def} \)
   - \( \text{union}_\text{cocardinal} \) by auto
   - 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 \( X - D \in \text{CoCardinal}(X, T) \) using \( \text{CoCardinal}_\text{def} \) by auto
      - With assumptions have \( (D \in \text{Pow}(X) \land D \prec T) \rightarrow D \{ \text{is closed in} \ \text{CoCardinal}(X, T) \} \) using \( \text{union}_\text{cocardinal} \)
        - \( \text{IsClosed}_\text{def} \) by auto
        - Moreover
          - Have \( X - X = 0 \) by auto
            - Then have \( X - X \in \text{CoCardinal}(X, T) \) using \( \text{CoCardinal}_\text{def} \) by auto

752
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 ≠ 0 and A ⊆ X

shows Interior(A,CoCardinal(X,T)) = (if ((X-A) ⊆ T) then A else 0)

**proof**-

from assms(2) have dif_dif:X-(X-A)=A by blast

{ assume (X-A) ⊆ T then have (X-A) ∈ Pow(X) ∧ (X-A) ⊆ 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) ∈ CoCardinal(X,T) using IsClosed_def union_cocardinal by auto
  with dif_dif have A ∈ CoCardinal(X,T) by auto
  hence a1:A ⊆ U {U ∈ CoCardinal(X,T). U ⊆ A} by auto
  have a2: U ∈ {U ∈ CoCardinal(X,T). U ⊆ A} by blast
  from a1 a2 have Interior(A,CoCardinal(X,T))=A using Interior_def by auto
}

moreover

{ assume as:¬((X-A) ⊆ T)
  { fix U assume U ⊆ A hence X-A ⊆ X-U by blast
    then have Q:X-A ⊆ X-U using subset_imp_lepoll by auto
    { assume X-U ⊆ T
      with Q have X-A ⊆ T using lesspoll_trans1 by auto
      with as have False by auto
    } hence ¬((X-U) ⊆ T) by auto
    then have U ∉ CoCardinal(X,T) \ U=0 using CoCardinal_def by auto
  } hence {U ∈ CoCardinal(X,T). U ⊆ A} ⊆ {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 ≠ 0 is too weak for CoCardinal(X,T) to be a topology.
lemma X_closedcov_cocardinal:
assumes T≠0 A⊆X
shows X∈ClosedCovers(A,CoCardinal(X,T)) using ClosedCovers_def
using union_cocardinal closed_sets_cocardinal assms by auto

The closure of a set is itself if it is closed or X if it isn’t closed.

lemma closure_set_cocardinal:
assumes T≠0 A⊆X
shows Closure(A,CoCardinal(X,T))=(if (A≺T) then A else X)
proof-
{ assume A≺T
  with assms have A {is closed in} CoCardinal(X,T) using closed_sets_cocardinal
  by auto
  with assms(2) have A∈ {D∈Pow(X). D {is closed in} CoCardinal(X,T)}
  by auto
  with assms(1) have S:A∈ClosedCovers(A,CoCardinal(X,T)) using ClosedCovers_def
  using union_cocardinal by auto
  hence 11:⋂ClosedCovers(A,CoCardinal(X,T))⊆A by blast
  from S have 12:A⊆⋂ClosedCovers(A,CoCardinal(X,T))
  unfolding ClosedCovers_def by auto
  from 11 12 have Closure(A,CoCardinal(X,T))=A using Closure_def
  by auto
} moreover
{ assume as:¬ A≺T
  { fix U
    assume A⊆U
    then have Q:A⊆U using subset_imp_lepoll by auto
    { assume U≺T
      with Q have A≺T using lesspoll_trans1 by auto
      with as have False by auto
    }
    hence ¬ U≺T by auto
    with assms(1) have ¬(U {is closed in} CoCardinal(X,T)) ∨ U=X using closed_sets_cocardinal
    by auto
  }
  with assms(1) have ∀ U∈Pow(X). U {is closed in} CoCardinal(X,T) ∧ A⊆U→U=X
  by auto
  with assms(1) have ClosedCovers(A,CoCardinal(X,T))⊆X
  using union_cocardinal using ClosedCovers_def by auto
  with assms have ClosedCovers(A,CoCardinal(X,T))=X using X_closedcov_cocardinal
  by auto
  then have Closure(A,CoCardinal(X,T)) = X using Closure_def by auto
}
ultimately show thesis by auto
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.

\begin{lemma}\textbf{boundary_cocardinal:}
\begin{quote}
assumes $T \neq 0 \subseteq X$
shows $\text{Boundary}(A, \text{CoCardinal}(X,T)) = \begin{cases} 
\text{if } A \prec T \text{ then } (\text{if } (X-A) \prec T \text{ then } 0 \text{ else } A) \text{ else } (\text{if } (X-A) \prec T \text{ then } X-A \text{ else } X) \end{cases}
\end{quote}
\end{lemma}

\begin{proof}
from \text{assms}(2) have $X-A \subseteq X$ by auto
\begin{quote}
\begin{assumption}
A \prec T \quad X-A \prec T
\end{assumption}
with \text{assms} $<X-A \subseteq X>$ have
\begin{quote}
Closure(X-A, \text{CoCardinal}(X,T)) = X-A \quad \text{and} \quad Closure(A, \text{CoCardinal}(X,T)) = A
\end{quote}
using \text{closure_set_cocardinal} by auto
with \text{assms}(1) have $\text{Boundary}(A, \text{CoCardinal}(X,T)) = 0$
using \text{Boundary_def} \quad \text{union_cocardinal} by auto
\end{quote}
\end{quote}
moreover
\begin{quote}
\begin{assumption}
\neg(A \prec T) \quad \neg(X-A \prec T)
\end{assumption}
with \text{assms} $<X-A \subseteq X>$ have
\begin{quote}
Closure(X-A, \text{CoCardinal}(X,T)) = X \quad \text{and} \quad Closure(A, \text{CoCardinal}(X,T)) = X
\end{quote}
using \text{closure_set_cocardinal} by auto
with \text{assms}(1) have $\text{Boundary}(A, \text{CoCardinal}(X,T)) = X$
using \text{Boundary_def} \quad \text{union_cocardinal} by auto
\end{quote}
\end{quote}
moreover
\begin{quote}
\begin{assumption}
A \prec T \quad \neg(X-A \prec T)
\end{assumption}
with \text{assms} $<X-A \subseteq X>$ have
\begin{quote}
Closure(X-A, \text{CoCardinal}(X,T)) = X \quad \text{and} \quad Closure(A, \text{CoCardinal}(X,T)) = A
\end{quote}
using \text{closure_set_cocardinal} by auto
with \text{assms} have $\text{Boundary}(A, \text{CoCardinal}(X,T)) = A$
using \text{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 assms by auto
    with A have U∈CoCardinal(X,T) using CoCardinal_def 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 show CoCardinal(X,1) ⊆ {0, X} by auto
  have 0 ∈ CoCardinal(X,1) using CoCardinal_def by auto
  moreover
  have 0≺1 and 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 $\text{CoCardinal}(X,T)$ topology are also $\text{CoCardinal}$ topologies.

**Lemma subspace_cocardinal:**

shows $\text{CoCardinal}(X,T) \{\text{restricted to}\} Y = \text{CoCardinal}(Y \cap X,T)$

**Proof**

{ fix $M$
  assume $M \in (\text{CoCardinal}(X,T) \{\text{restricted to}\} Y)$
  then obtain $A$ where $A1: A \in \text{CoCardinal}(X,T)$ $M=Y \cap A$ using RestrictedTo_def
  by auto
  then have $M \in \text{Pow}(X \cap Y)$ using CoCardinal_def by auto
  moreover
  from $A1$ have $(Y \cap X)-M = (Y \cap X)-A$ using CoCardinal_def by auto
  with $<(Y \cap X)-M = (Y \cap X)-A> \implies (Y \cap X)-M \subseteq X-A$ by auto
  then have $(Y \cap X)-M \subseteq X-A$ using subset_imp_lepoll by auto
  with $A1$ have $(Y \cap X)-M \prec T \lor M=0$ using lesspoll_trans1 CoCardinal_def by auto
  ultimately have $M \in \text{CoCardinal}(Y \cap X, T)$ using CoCardinal_def by auto
  }
  then show $\text{CoCardinal}(X,T) \{\text{restricted to}\} Y \subseteq \text{CoCardinal}(Y \cap X,T)$ by auto

{ fix $M$
  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 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 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 CoCardinal_def by auto
    then have $AT:Y \cap A \in \text{CoCardinal}(X,T) \{\text{restricted to}\} Y$ using RestrictedTo_def
    by auto
    have $Y \cap A=Y \cap M$ by blast
    also from $A1$ have $...=M$ 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
  }

757
ultimately have \( M \in \text{CoCardinal}(X,T) \) \{restricted to\} \( Y \) by \text{auto} 

then show \( \text{CoCardinal}(Y \cap X, T) \subseteq \text{CoCardinal}(X,T) \) \{restricted to\} \( Y \) by \text{auto} 

qed

62.3 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** \( \text{ExcludedSet}(X,U) \equiv \{ F \in \text{Pow}(X). U \cap F=0 \} \cup \{ X \} \)

For any set; we prove that \( \text{ExcludedSet}(X,Q) \) forms a topology.

**theorem** \( \text{excludedset_is_topology} \): 

shows \( \text{ExcludedSet}(X,Q) \) \{is a topology\}

**proof**-

\{ 

fix \( M \)

assume \( M \in \text{Pow}(\text{ExcludedSet}(X,Q)) \)

then have \( A: \{ F \in \text{Pow}(X). Q \cap F=0 \} \cup \{ X \} \) using \( \text{ExcludedSet_def} \) by \text{auto} 

hence \( \bigcup M \in \text{Pow}(X) \) by \text{auto} 

moreover 

\{ 

have \( B: Q \cap \bigcup M = \bigcup \{ Q \cap T \mid T \in M \} \) by \text{auto} 

\{ 

assume \( X \notin M \) 

with \( A \) have \( M \subseteq \{ F \in \text{Pow}(X). Q \cap F=0 \} \) by \text{auto} 

with \( B \) have \( Q \cap \bigcup M=0 \) by \text{auto} 

\} 

moreover 

\{ 

assume \( X \in M \) 

with \( A \) have \( \bigcup M=X \) by \text{auto} 

\} 

ultimately have \( Q \cap \bigcup M=0 \lor \bigcup M=X \) by \text{auto} 

\} 

ultimately have \( \bigcup M \in \text{ExcludedSet}(X,Q) \) using \( \text{ExcludedSet_def} \) by \text{auto} 

\} 

moreover 

\{ 

fix \( U \) \( V \) 

assume \( U \in \text{ExcludedSet}(X,Q) \) \( V \in \text{ExcludedSet}(X,Q) \) 

then have \( U \in \text{Pow}(X) \) \( V \in \text{Pow}(X) \) \( U=V \lor U \cap Q=0 \lor V \cap Q=0 \) using \( \text{ExcludedSet_def} \) by \text{auto} 

\}

758
hence $U \in \text{Pow}(X) \land V \in \text{Pow}(X) \implies (U \cap V) = X \lor Q \cap (U \cap V) = 0$ 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 $\text{topology0}$ when discussing excluded set topology.

**Theorem** topology0_excludedset:
shows $\text{topology0}(\text{ExcludedSet}(X, T))$
using topology0_def excludedset_is_topology by auto

Choosing a singleton set, it is considered a point in excluded topology.

**Definition**
$\text{ExcludedPoint}(X, p) \equiv \text{ExcludedSet}(X, \{p\})$

### 62.4 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 $X$

**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 $D \ (\text{is closed in}) \text{ExcludedSet}(X, T) \iff (D \in \text{Pow}(X) \land (X \cap T) \subseteq D) \lor D = 0$

**Proof**
\{
  \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) have B: X-(X-D)=D by auto}
  \text{from A(2) 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 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) have } T \cap (X-D)=0 \text{ by auto}
  \text{with A(4) have } x \notin X-D \text{ by auto}
  \text{with A(5) have } x \notin D \text{ by auto}
\}
moreover
\{
  \text{assume A: } X \cap T \subseteq D \subseteq X
\}

759
from A(1) have X-D X-(X∩T) by auto
also have .... = X-T by auto
finally have T∩(X-D) = 0 by auto
moreover
have X-D ∈ Pow(X) by auto
ultimately have X-D ∈ExcludedSet(X,T) 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 ⊆ X
shows Interior(A,ExcludedSet(X,T)) = (if A=X then X else A-T)
proof-
{ assume A:A≠X
from asms have A-T ∈ExcludedSet(X,T) using ExcludedSet_def by auto
then have A-T ⊆ Interior(A,ExcludedSet(X,T))
using Interior_def by auto
moreover
{ fix U
assume U ∈ExcludedSet(X,T) U⊆A
then have T∪U=0 ∨ U=XU⊆A using ExcludedSet_def by auto
with A asms have T∪U=0U⊆A by auto
then have U-T=UU-T⊆A-T by auto
then have U⊆A-T by auto
}
then have Interior(A,ExcludedSet(X,T))⊆A-T using Interior_def by auto
ultimately have Interior(A,ExcludedSet(X,T))=A-T by auto
}
moreover
have X∈ExcludedSet(X,T) using ExcludedSet_def
union_excludedset by auto
then have Interior(X,ExcludedSet(X,T)) = X using topology0.Top_2_L3
topology0_excludedset by auto
ultimately show thesis by auto
qed

The closure of a set is itself if it is 0 or the union with T.

lemma closure_set_excludedset:
assumes A ⊆ X
shows Closure(A,ExcludedSet(X,T))=(if A=0 then 0 else A ∪(X∩ T))
proof-
have 0∈ClosedCovers(0,ExcludedSet(X,T)) using ClosedCovers_def
closed_sets_excludedset by auto
then have Closure(0,ExcludedSet(X,T))\subseteq 0 using Closure_def by auto
hence Closure(0,ExcludedSet(X,T))=0 by blast
moreover
{ assume A:A\neq 0
  with assms have (A|(X\cap T)) {is closed in}ExcludedSet(X,T) using closed_sets_excludedset
  by blast
  then have (A \cup (X\cap T)) \in \{ D \in Pow(X). D {is closed in}ExcludedSet(X,T) \} \wedge A \subseteq D
    using assms by auto
  then have (A \cup (X\cap T)) \in ClosedCovers(A,ExcludedSet(X,T)) unfolding
    ClosedCovers_def
    using union_excludedset by auto
  then have 11:\bigcap \text{ClosedCovers}(A,\text{ExcludedSet}(X,T)) \subseteq (A \cup (X\cap T)) by blast
  { fix U
    assume U \in \text{ClosedCovers}(A,\text{ExcludedSet}(X,T))
    then have U {is closed in}ExcludedSet(X,T) and A \subseteq U using ClosedCovers_def
      union_excludedset by auto
    then have U=0\lor (X\cap T) \subseteq U and A \subseteq U using closed_sets_excludedset
      by auto
    with A have (X\cap T) \subseteq UA \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 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:
  assumes A\subseteq X
  shows Boundary(A,\text{ExcludedSet}(X,T)) = (if A=0\lor A=X then 0 else X\cap T)
proof-
{ have Closure(0,\text{ExcludedSet}(X,T))=0\lor Closure(X - 0,\text{ExcludedSet}(X,T))=X
  using closure_set_excludedset by auto
  then have 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 Closure(X,ExcludedSet(X,T)) = X and Closure(X-X,ExcludedSet(X,T)) = 0
    using closure_set_excludedset by auto
  then have Boundary(X,ExcludedSet(X,T)) = 0
    unfolding Boundary_def
    using union_excludedset by auto
}
moreover
{
  assume A≠0 and A≠X
  then have X-A≠0 using assms by auto
  with assms <A≠0> <A⊆X> have Closure(A,ExcludedSet(X,T)) = A ∪ (X∩T)
    using closure_set_excludedset by simp
  moreover from <A⊆X> have X-A ⊆ X by blast
  with <X-A≠0> have Closure(X-A,ExcludedSet(X,T)) = (X-A) ∪ (X∩T)
    using closure_set_excludedset by simp
  ultimately have Boundary(A,ExcludedSet(X,T)) = X∩T
    using Boundary_def union_excludedset by auto
}
ultimately have thesis by auto
qed

62.5 Special cases and subspaces

This section provides some miscellaneous facts about excluded set topolo-
gies.

The excluded set topology is equal in the sets T and X∩T.

lemma smaller_excludedset:
  shows ExcludedSet(X,T) = ExcludedSet(X,(X∩T))
proof
  show ExcludedSet(X,T) ⊆ ExcludedSet(X, X∩T) and ExcludedSet(X, X∩T) ⊆ 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∩X=0
  shows ExcludedSet(X,T) = Pow(X)
proof
  from assms show ExcludedSet(X,T) ⊆ Pow(X) using smaller_excludedset
  ExcludedSet_def
The topological subspaces of the $\text{ExcludedSet}(X,T)$ topology are also $\text{ExcludedSet}$ topologies.

**Lemma**  subspaces_excludedset:

- **shows** $\text{ExcludedSet}(X,T) \{\text{restricted to}\} Y = \text{ExcludedSet}(Y \cap X, T)$
- **proof**
  
  \[
  \begin{align*}
  \text{fix } M & \\
  \text{assume } M & \in (\text{ExcludedSet}(X,T) \{\text{restricted to}\} Y) \\
  \text{then obtain } A & \text{ where } A1:A: \text{ExcludedSet}(X,T) \cap Y \cap X & \text{unfolding RestrictedTo_def} & \text{by auto} \\
  \text{then have } M & \in \text{Pow}(Y \cap X) \text{ unfolding ExcludedSet_def by auto} \\
  \text{moreover} & \\
  \text{from } A1 & \text{ have } T \cap M = T \cap Y \cap X \text{ unfolding ExcludedSet_def by blast} \\
  \text{ultimately have } M & \in \text{ExcludedSet}(Y \cap X, T) \text{ unfolding ExcludedSet_def} & \text{by auto} \\
  \end{align*}
  \]
  
  then show $\text{ExcludedSet}(X,T) \{\text{restricted to}\} Y \subseteq \text{ExcludedSet}(Y \cap X, T)$

**Proof**

\[
\begin{align*}
\text{fix } M & \\
\text{let } A & = M \cup ((X \cap Y - T) - Y) \\
\text{assume } A: M & \in \text{ExcludedSet}(Y \cap X, T) \\
\text{assume } M & = Y \cap X \\
\text{then have } M & \in \text{ExcludedSet}(X,T) \{\text{restricted to}\} Y \text{ unfolding RestrictedTo_def} & \text{ExcludedSet_def by auto} \\
\text{moreover} & \\
\text{assume } AS: M & \neq Y \cap X \\
\text{from } A & \text{ assume } A1:(M \in \text{Pow}(Y \cap X) \land T \cap M = 0) \text{ unfolding ExcludedSet_def} & \text{by auto} \\
\text{then have } A & \in \text{Pow}(X) \text{ by blast} \\
\text{moreover} & \\
\text{have } T \cap A = T \cap M & \text{ by blast} \\
\text{with } A1 & \text{ have } T \cap A = 0 \text{ by auto} \\
\text{ultimately have } A & \in \text{ExcludedSet}(X,T) \text{ unfolding ExcludedSet_def by auto} \\
\text{then have } AT: Y \cap A & \in \text{ExcludedSet}(X,T) \{\text{restricted to}\} Y \text{ unfolding RestrictedTo_def} & \text{by auto} \\
\text{have } Y \cap A & = Y \cap M \text{ by blast} \\
\text{also have } \ldots = M \text{ using } A1 \text{ by auto} \\
\text{finally have } Y \cap A & = M \text{ by simp}
\end{align*}
\]
with AT have $M \in \text{ExcludedSet}(X, T)$ {restricted to} $Y$ by auto 
ultimately have $M \in \text{ExcludedSet}(X, T)$ {restricted to} $Y$ by auto 
then show $\text{ExcludedSet}(Y \cap X, T) \subseteq \text{ExcludedSet}(X, T)$ {restricted to} $Y$ by auto 
qed

62.6 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($X, U$) ≡ \{F ∈ Pow($X$). $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}
proof-
  { 
    fix $M$  
    assume $M \in \text{Pow}(\text{IncludedSet}(X, Q))$ 
    then have $A: M \subseteq \{F \in \text{Pow}(X). \ Q \subseteq F\} \cup \{0\}$ using IncludedSet_def by auto 
    then have $\bigcup M \in \text{Pow}(X)$ by auto 
    moreover 
    have $Q \subseteq \bigcup \bigcup M=0$ using $A$ by blast 
    ultimately have $\bigcup M \in \text{IncludedSet}(X, Q)$ using IncludedSet_def by auto 
  } 
moreover 
  { 
    fix $U$ $V$  
    assume $U \in \text{IncludedSet}(X, Q)$ $V \in \text{IncludedSet}(X, Q)$ 
    then have $U \in \text{Pow}(X) V \in \text{Pow}(X) U=0 \lor Q \subseteq U \lor V$ using IncludedSet_def by auto 
    then have $(U \cap V) \in \text{IncludedSet}(X, Q)$ 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 topology0_includedset:
shows topology0(\text{ IncludedSet}(X, T))
Choosing a singleton set, it is considered a point excluded topology. In the following lemmas and theorems, when necessary it will be considered that \( T \neq 0 \) and \( T \subseteq X \). These cases will appear in the special cases section.

**62.7 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 \( T \subseteq X \)
- shows \( \bigcup \text{IncludedSet}(X,T) = X \)
  ```
  proof
  from assms have \( X \in \text{IncludedSet}(X,T) \) using IncludedSet_def by auto
  then show \( \bigcup \text{IncludedSet}(X,T) = X \) 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 \( D \ {\text{is closed in}} \ \text{IncludedSet}(X,T) \iff (D \in \mathcal{P}(X) \land (D \cap T)=0) \lor D=X \)
  ```
  proof
  have \( X-X=0 \) by blast
  then have \( X-X \in \text{IncludedSet}(X,T) \) using IncludedSet_def by auto
  moreover
  { assume \( A:D \subseteq X \land D \in \text{IncludedSet}(X,T) \land D \neq X \)
    from \( A(2) \) have \( T \subseteq (X-D) \lor X-D=0 \) using IncludedSet_def by auto
    moreover
    { assume \( A(1) \) 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 \( A(1) \) 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**

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 $T \cap A = 0$
  - Then have $A \{\text{is closed in}\} \text{IncludedSet}(X, T)$ using closed_sets_includedset assms by auto
  - 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 $T \cap A \neq 0$
    - Have $X \in \text{ClosedCovers}(A, \text{IncludedSet}(X, T))$ using ClosedCovers_def closed_sets_includedset union_includedset assms by auto
    - Then have $11: \bigcap \text{ClosedCovers}(A, \text{IncludedSet}(X, T)) \subseteq X$ using Closure_def by auto
  - Moreover
    - Fix $U$
      - Assume $U \in \text{ClosedCovers}(A, \text{IncludedSet}(X, T))$
        - Then have $U \{\text{is closed in}\} \text{IncludedSet}(X, T) A \subseteq U$ using ClosedCovers_def by auto
then have $U = X \lor (T \cap A) = 0$ using `closed_sets_includedset` `assms(2)` by auto
then have $U = X \lor (T \cap A) = 0$ by auto
then have $U = X$ using `AS` by auto

} then have $X \subseteq \bigcap \text{ClosedCovers}(A, \text{IncludedSet}(X, T))$ using `topology0.Top_3_L3` `topology0_includedset` union_includedset `assms by auto`
ultimately have $\bigcap \text{ClosedCovers}(A, \text{IncludedSet}(X, T)) = X$ by auto
then 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
Closure$(A, \text{IncludedSet}(X, T)) = X$ and $\text{Closure}(X - A, \text{IncludedSet}(X, T))$
  $= (X - A)$
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)` $\langle T \cap A = 0 \rangle$ $\langle X - A \subseteq X \rangle$ have
Closure$(A, \text{IncludedSet}(X, T)) = A$ and $\text{Closure}(X - A, \text{IncludedSet}(X, T))$
$= X$
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

767
with assms(1,2) \langle \langle T \setminus A \neq 0 \rangle \cap X \subseteq X \rangle 
\text{ have } 
\text{ Closure}(A, \text{IncludedSet}(X,T)) = X \text{ and } \text{ Closure}(X-A, \text{IncludedSet}(X,T)) = X 
\text{ using } \text{closure_set_includedset} \text{ by auto} 
\text{ with } \text{assms}(2) \text{ have } \text{Boundary}(A, \text{IncludedSet}(X,T)) = X 
\text{ using } \text{Boundary_def union_includedset} \text{ by auto} 
\} 
\text{ ultimately show thesis by auto} 
\text{ qed} 

62.8 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 $T=0$

\textbf{lemma smaller_includedset:} 
\text{ shows } \text{IncludedSet}(X,0) = \text{Pow}(X) 
\text{proof} 
\text{ show } \text{IncludedSet}(X,0) \subseteq \text{Pow}(X) \text{ and } \text{Pow}(X) \subseteq \text{IncludedSet}(X,0) 
\text{ unfolding } \text{IncludedSet_def} \text{ by auto} 
\text{ qed} 

If the set which is included is not a subset of $X$, then the topology is trivial.

\textbf{lemma empty_includedset:} 
\text{ assumes } \lnot\,(T \subseteq X) 
\text{ shows } \text{IncludedSet}(X,T) = \{0\} 
\text{proof} 
\text{ from } \text{assms show } \text{IncludedSet}(X,T) \subseteq \{0\} \text{ and } \{0\} \subseteq \text{IncludedSet}(X,T) 
\text{ unfolding } \text{IncludedSet_def} \text{ by auto} 
\text{ qed} 

The topological subspaces of the $\text{IncludedSet}(X,T)$ topology are also 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.

\textbf{lemma subspace_includedset:} 
\text{ assumes } T \subseteq X 
\text{ shows } \text{IncludedSet}(X,T) \{\text{restricted to}\} Y = \text{IncludedSet}(Y \cap X, Y \cap T) 
\text{proof} 
\{ 
\text{ fix } M 
\text{ assume } M \in (\text{IncludedSet}(X,T) \{\text{restricted to}\} Y) 
\text{ then obtain } A \text{ where } A \vdash A : \text{IncludedSet}(X,T) M = Y \cap A \text{ unfolding } \text{RestrictedTo_def} 
\text{ by auto} 
\}
then have $M \in \text{Pow}(X \cap Y)$ unfolding IncludedSet_def by auto
moreover
from A1 have $Y \cap T \subseteq M \vee M = 0$ 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 $A : M \neq 0$
from A1 have $A1 : M \in \text{Pow}(Y \cap X) \land Y \cap T \subseteq M$ unfolding IncludedSet_def
by auto
then have $A \in \text{Pow}(X)$ using assms by blast
moreover
have $T \subseteq A$ by blast
ultimately have $A \in \text{IncludedSet}(X, T)$ unfolding IncludedSet_def by auto
then have $AT : Y \cap A \in \text{IncludedSet}(X, T) \{\text{restricted to}\} Y$ 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 $M \in \text{IncludedSet}(X, T) \{\text{restricted to}\} Y$
by auto
}
ultimately have $M \in \text{IncludedSet}(X, T) \{\text{restricted to}\} Y$ by auto

thus $\text{IncludedSet}(Y \cap X, Y \cap T) \subseteq \text{IncludedSet}(X, T) \{\text{restricted to}\} Y$ by auto
qed

63 More examples in topology

theory Topology_ZF_examples_1
imports Topology_ZF_1 Order_ZF

769
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.

63.1 New ideas using a base for a topology

63.2 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.

\[ \text{definition} \]
\[ \text{TopologyWithBase \ (TopologyBase \ _ \ 50) where} \]
\[ \text{\quad \quad U \ {\text{satisfies the base condition}} \implies \text{TopologyBase U = THE \ T. \ U \ {\text{is a base for}} \ T} \]

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.

\[ \text{theorem Base_topology_is_a_topology:} \]
\[ \text{\quad \quad assumes \ U \ {\text{satisfies the base condition}} \]}
\[ \text{\quad \quad shows \ (TopologyBase U) \ {\text{is a topology}} \ and \ U \ {\text{is a base for}} \ (TopologyBase U) \]
\[ \text{\quad \quad proof} \]
\[ \text{\quad \quad \quad from \ assms \ obtain \ T \ where \ U \ {\text{is a base for}} \ T \ using} \]
\[ \text{\quad \quad \quad \quad Top_1_2_T1(2) \ by \ blast} \]
\[ \text{\quad \quad \quad then \ have \ \exists ! T. \ U \ {\text{is a base for}} \ T \ using \ same_base_same_top \ ex1I\[\text{where} \]
\[ \text{\quad \quad \quad \quad P= \lambda T. \ U \ {\text{is a base for}} \ T} \ by \ blast \]
\[ \text{\quad \quad \quad with \ assms \ show \ U \ {\text{is a base for}} \ (TopologyBase U) \ using \ theI\[\text{where} \]
\[ \text{\quad \quad \quad \quad P= \lambda T. \ U \ {\text{is a base for}} \ T \] \text{TopologyWithBase_def by auto} \]
\[ \text{\quad \quad \quad with \ assms \ show \ (TopologyBase U) \ {\text{is a topology}} \ using \ Top_1_2_T1(1) \]}
\[ \text{\quad \quad \quad \quad IsAbaseFor_def by auto} \]
\[ \text{\quad \quad qed} \]

A base doesn’t need the empty set.

\[ \text{lemma base_no_0:} \]
\[ \text{\quad \quad shows \ B\{is a base for\}T \longleftrightarrow \ (B\{-0\})\{is a base for\}T} \]
\[ \text{\quad \quad proof} \]
\[ \text{\quad \quad \quad \quad \{ \text{fix M} \]}
\[ \text{\quad \quad \quad \quad \quad assume \ M\in\{\bigcup A \ , \ A \in \text{Pow}(B)\} \]}
\[ \text{\quad \quad \quad \quad \quad then \ obtain \ Q \ where \ M=\bigcup Q\in\text{Pow}(B) \ by \ auto} \]
\[ \text{\quad \quad \quad \quad \quad hence \ M=\bigcup (Q\{-0\})Q\{-0\}\in\text{Pow}(B\{-0\}) \ by \ auto} \]
\[ \text{\quad \quad \quad \quad \quad hence \ M\in\{\bigcup A \ , \ A \in \text{Pow}(B \ - \ {0})\} \ by \ auto} \]
\[ \text{\quad \quad \quad \quad \}} \]

770
hence \{\bigcup A . A \in \operatorname{Pow}(B)\} \subseteq \{\bigcup A . A \in \operatorname{Pow}(B - \{0\})\} by \text{blast}

moreover

\{ \\
  \text{fix } M \\
  \text{assume } M \in \{\bigcup A . A \in \operatorname{Pow}(B - \{0\})\} \\
  \text{then obtain } \emptyset \text{ where } M = \bigcup \{Q \in \operatorname{Pow}(B - \{0\}) \text{ by auto} \\
  \text{hence } M = \bigcup \{Q \in \operatorname{Pow}(B) \text{ by auto} \\
  \text{hence } M \in \{\bigcup A . A \in \operatorname{Pow}(B)\} \text{ by auto} \\
}\}

hence \{\bigcup A . A \in \operatorname{Pow}(B - \{0\})\} \subseteq \{\bigcup A . A \in \operatorname{Pow}(B)\} by \text{auto}

ultimately have \{\bigcup A . A \in \operatorname{Pow}(B - \{0\})\} = \{\bigcup A . A \in \operatorname{Pow}(B)\} by \text{auto}

then show \(B\{\text{is a base for}\}T \iff (B-\{0\}\{\text{is a base for}\}T\) using \text{IsBaseFor_def}

\text{by auto}

qed

The interior of a set is the union of all the sets of the base which are fully contained by it.

\text{lemma interior_set_base_topology:}

\text{assumes } U \{\text{is a base for}\} T, T \{\text{is a topology}\}
\text{shows } \operatorname{Interior}(A, T) = \bigcup \{T \in U. T \subseteq A\}

\text{proof}

\text{have } \{U \subseteq U. U \subseteq A\} \subseteq U \text{ by auto}
\text{with asm(1) have } \bigcup \{U \subseteq U. U \subseteq A\} \subseteq T \\
\text{using \text{IsBaseFor_def by auto} }
\text{moreover have } \bigcup \{U \subseteq U. U \subseteq A\} \subseteq A \text{ by blast}
\text{ultimately show } \bigcup \{U \subseteq U. U \subseteq A\} \subseteq \operatorname{Interior}(A, T) \\
\text{using asm(2) topology0.Top_2_L5 topology0_def by auto }

\{ \\
  \text{fix } x \\
  \text{assume } x \in \operatorname{Interior}(A, T) \\
  \text{with asm obtains } V \text{ where } V \in U V \subseteq \operatorname{Interior}(A, T) x \in V \\
  \text{using point_open_base_neigh topology0.Top_2_L2 topology0_def by blast} \\
  \text{with asm have } V \in U x \in V V \subseteq A \text{ using topology0.Top_2_L1 topology0_def by auto} \\
  \text{hence } x \in \bigcup \{U \subseteq U. U \subseteq A\} \text{ by auto} \\
\}

\text{thus } \operatorname{Interior}(A, T) \subseteq \bigcup \{T \in U. T \subseteq A\} \text{ by auto}

\text{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 \text{cl_inter_neigh} and \text{inter_neigh_cl}.

\text{It states that it is only necessary to check the sets of the base, not all the open sets.}

\text{lemma closure_set_base_topology:}

\text{assumes } U \{\text{is a base for}\} Q, Q \{\text{is a topology}\} A \subseteq \bigcup Q
\text{shows } \operatorname{Closure}(A, Q) = \{x \in \bigcup Q. \forall T \in U. x \in T \rightarrow A \cap T \neq 0\}

\text{proof}
{ fix x 
  assume A:x∈Closure(A,Q) 
  with assms(2,3) have B:x∈∪Q using topology0_def topology0.Top_3_L11(1) 
    by blast 
  moreover 
  { fix T 
    assume T∈U x∈T 
    with assms(1) have T∈Qx∈T using base_sets_open by auto 
    with assms(2,3) A have A∩T ≠ 0 using topology0_def topology0.cl_inter_neigh 
      by auto 
  } 
    hence ∀T∈U. x∈T→A∩T≠0 by auto 
    ultimately have x∈{x∈∪Q. ∀T∈U. x∈T→A∩T≠0} by auto 
  } 
  thus Closure(A, Q) ⊆ {x∈∪Q. ∀T∈U. x∈T→A∩T≠0} by auto 
  
  { fix x 
    assume AS:x∈{x∈∪Q. ∀T∈U. x∈T→A∩T≠0} 
    hence x∈∪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∈Wx∈WW by auto 
          { assume R∩A=0 
            with RR(2) TT(1) have WW∩A=0 by auto 
              with TT(1) RR(1) have WW∈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 \( (B \text{ restricted to } Y) \) {is a base for} \( (T \text{ restricted to } Y) \)

**Proof:**

from assms have \( (B \text{ restricted to } Y) \subseteq (T \text{ restricted to } Y) \)
unfolding IsAbaseFor_def RestrictedTo_def by auto
moreover have \( (T \text{ restricted to } Y) = \bigsqcup \{A. A \in \text{Pow}(B \text{ restricted to } Y)\} \)
proof
	{ fix \( U \) assume \( U \in (T \text{ restricted to } Y) \)
then obtain \( W \) where \( W \in T \) and \( U = W \cap Y \) unfolding RestrictedTo_def
by blast
with assms obtain \( C \) where \( C \subseteq B \) and \( W=\bigcup C \) unfolding IsAbaseFor_def
by blast
let \( A=(\forall V \in C. V \cap Y) \in \text{Pow}(B) \)
from \( A \in \text{Pow}(B \text{ restricted to } Y) \) and \( U=(\bigcup A) \)
unfolding RestrictedTo_def by auto
hence \( U \in (\bigcup \{A. A \in \text{Pow}(B \text{ restricted to } Y)\}) \) by blast
} thus \( (T \text{ restricted to } Y) \subseteq (\bigcup \{A. A \in \text{Pow}(B \text{ restricted to } Y)\}) \)
by auto
{ fix \( U \) assume \( U \in (\bigcup \{A. A \in \text{Pow}(B \text{ restricted to } Y)\}) \)
then obtain \( A \) where \( A \subseteq (B \text{ restricted to } Y) \) and \( U=\bigcup A \)
by auto
let \( A_0=(\forall C \in B. Y \cap C) \)
from \( A \) have \( A_0 \subseteq B \) and \( A = A_0 \text{ restricted to } Y \) unfolding RestrictedTo_def
by auto
with \( U = (\bigcup A) \) have \( A_0 \subseteq B \) and \( U = \bigcup (A_0 \text{ restricted to } Y) \)
by auto
with assms have \( U \in (T \text{ restricted to } Y) \) unfolding RestrictedTo_def
IsAbaseFor_def
by auto
} thus \( (\bigcup \{A. A \in \text{Pow}(B \text{ restricted to } Y)\}) \subseteq (T \text{ restricted to } Y) \)
by blast
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 \( B=\text{is a base for}T \)
\( B_2=\text{is a base for}T_2 \)
\( B \subseteq B_2 \)
shows \( T \subseteq T_2 \)

**Proof:**

{ fix \( x \)
assume \( x \in T \)
with assms(1) obtain \( M \) where \( M \subseteq Bx=\bigcup M \) using IsAbaseFor_def by auto

773
with assms(3) have $M \subseteq B \cup M$ by auto
with assms(2) show $x \in T$ using IsBaseFor_def by auto

qed

63.3 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
$B \subseteq \text{is a base for} \implies \text{DualBase} B = \{\bigcup \{T - U \mid U \in B\}, \bigcup \{T\}\}$

lemma closed_inter_dual_base:
assumes $D \subseteq \text{is closed in} \implies B \subseteq \text{is a base for} \implies D \subseteq \bigcap \{M \mid M \subseteq \text{DualBase} B\}$
obtains $M$ where $M \subseteq \text{DualBase} B \implies D = \bigcap M$ proof-
assume $K: \forall M. M \subseteq \text{DualBase} B \implies D = \bigcap M \implies \text{thesis}$
{
assume $AS: D \neq \bigcup T$
from assms(1) have $D \subseteq \text{Pow}(\bigcup T) \cup T \subseteq T$ using IsClosed_def by auto
hence $A: \bigcup T - (\bigcup T - D) = D \cup T - D$ by auto
with assms(2) obtain $Q$ where $Q \subseteq \text{Pow}(B) \cup T - D = \bigcup Q$ using IsBaseFor_def by auto
{
assume $Q = 0$
then have $\bigcup Q = 0$ by auto
with QQ(2) have $\bigcup T - D = 0$ by auto
with D(1) have $D = \bigcup T$ by auto
with AS have False by auto
}
hence $Q \neq 0$ by auto
from QQ(2) A(1) have $D = \bigcup T - \bigcup Q$ by auto
with $Q \neq 0$ have $D = \bigcap \{T - R \mid R \in Q\}$ by auto
moreover
with assms(2) QQ(1) have $\{T - R \mid R \in Q\} \subseteq \text{DualBase} B \implies \text{DualBase}_T$ by auto
with calculation $K$ have thesis by auto
}
moreover
{
assume $AS: D = \bigcup T$
with assms(2) have $\{T\} \subseteq \text{DualBase} B \implies \text{DualBase}_T$ by auto
moreover
have $\bigcup T = \bigcap \{T\}$ by auto
with calculation $K$ AS have thesis by auto
}
ultimately show thesis by auto

774
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∈Pow(∪Q) using topology0.Top_3_L1 topology0_def

topology0.Top_3_L9 base_sets_open by auto
}

then have SUB:{T∈DualBase U Q. A⊆T}⊆{T∈Pow(∪Q). (T{is closed in}Q)A⊆T}

by blast

with calculation assms(3) have ∩{T∈Pow(∪Q). (T{is closed in}Q)A⊆T}⊆∩{T∈DualBase U Q. A⊆T}

by auto

then show Closure(A,Q)⊆∩{T∈DualBase U Q. A⊆T} using Closure_def ClosedCovers_def by auto

{fix x
assume A:x∈∩{T∈DualBase U Q. A⊆T}

{fix T
assume B:x∈T∈U

{assume C:A∩T=0

from B(2) assms(1) have ∪Q-T∈DualBase U Q using DualBase_def by auto

moreover

from C assms(3) have A⊆∪Q-T by auto

moreover

from B(1) have x∉∪Q-T by auto

ultimately have x∉∩{T∈DualBase U Q. A⊆T} by auto

with A have False by auto

} hence A∩T#0 by auto

} hence ∀T∈U. x∈T→A∩T#0 by auto

moreover

qed
from T A assms(3) have x∈∪Q by auto
  with calculation assms have x∈Closure(A,Q) using closure_set_base_topology
    by auto

  } thus ∩{T ∈ DualBase U Q . A ⊆ T} ⊆ Closure(A, Q) by auto
qed

63.4 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
  IsAPartition (_{is a partition of}_ 90) where
  (U {is a partition of} X) ≡ (∪U=X ∧ (∀A∈U. ∀B∈U. A=B ∨ A∩B=0) ∧ 0∉U)

A subcollection of a partition is a partition of its union.

lemma subpartition:
  assumes U {is a partition of} X V ⊆ U
  shows V{is a partition of}∪V
  using assms unfolding IsAPartition_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 {restricted to} Y)-{0}) {is a partition of} (X ∩ 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 ⊆ P P {is a partition of} X
  shows X - ∪R=∪(P-R)
proof
  { fix x
    assume x∈X - ∪R
    hence P:x∈Xx∉∪R by auto
    { fix T
      assume T∈R
      with P(2) have x∉T by auto
    }
    with P(1) assms(2) obtain Q where Q∈(P-R)x∈Q using IsAPartition_def
    by auto
    hence x∈∪(P-R) by auto
  }

776
thus $X - \bigcup R \subseteq \bigcup (P-R)$ by auto

\{
  fix x
  assume $x \in \bigcup (P-R)$
  then obtain $Q$ where $Q \in P-R x \in Q$ by auto
  hence $C: Q \in P-R x \in Q$ by auto
  then have $x \in \bigcup P$ by auto
  with assms(2) have $x \in X$ using IsAPartition_def by auto
  moreover
  \{
    assume $x \in R$
    then obtain $t$ where $t \in R x \in t$ by auto
    with $C(3)$ assms(1) have $t \cap Q \neq 0 t \in P$ 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 $x \notin R$ by auto
  ultimately have $x \in X - \bigcup R$ by auto
  \}
thus $\bigcup (P-R) \subseteq X - \bigcup R$ by auto
qed

63.5 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 assms have $A: U = V \lor U \cap V = 0$ using IsAPartition_def by auto
  \{
    fix $x$
    assume $ASS: x \in U \lor V$
    with $A$ have $U = V$ by auto
    with $AS$ $ASS$ have $U \in P x \in U \land U \subseteq U \cap V$ by auto
    hence $\exists W \in P. x \in W \land W \subseteq U \cap V$ by auto
  } hence $(\forall x \in U \lor V. \exists W \in P. x \in W \land W \subseteq U \cap V)$ 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) ==> PTopology X U = 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 assms 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 assms by auto

63.6 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
  using assms Ptopology_is_a_topology(2) Top_1_2_L5
  IsAPartition_def by auto

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) <-> 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
    { fix D
      assume D {is closed in} (PTopology X T)
      with assms have A:D\in Pow(X)X-D\in(PTopology X T)
        using IsClosed_def union_ptopology by auto
      from A(2) B obtain R where Q:R \subseteq T X-D=\bigcup R using Top_1_2_L1[where B=T and U=X-D]
        by auto
      from A(1) have X-(X-D)=D by blast
      with Q(2) have D=X-\bigcup R by auto
      with Q(1) assms have D=\bigcup(T-R) using diff_union_is_union_diff
        by auto
      with B show D \in (PTopology X T) using IsAbaseFor_def by auto
    }
    { fix D

778
assume \( D \in (\text{PTopology } X \ T) \)
with \( B \) obtain \( R \) where \( Q : R \subseteq T = \bigcup R \) using \text{IsAbaseFor_def} by auto
hence \( X-D = X- \bigcup R \) by auto
with \( Q(1) \) assms have \( X-D = \bigcup (T-R) \) using \text{diff_union_is_union_diff} by auto
with \( B \) have \( X-D \in (\text{PTopology } X \ T) \) using \text{IsAbaseFor_def} by auto
moreover
from \( Q \) have \( D \subseteq \bigcup T \) by auto
with assms have \( D \subseteq X \) using \text{IsAPartition_def} by auto
with calculation assms show \( D \) is closed in \( (\text{PTopology } X \ T) \) using \text{IsClosed_def union_ptopology} by auto
\}
qed

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.

\textbf{lemma} \text{interior_set_ptopology}: \text{assumes} \( U \) is a partition of \( X \)
\text{shows} \( \text{Interior}(A, (\text{PTopology } X \ U)) = \bigcap \{T \in \text{DualBase U (PTopology } X \ U). T=X \lor T \cup A \neq X\} \)
\text{proof}
\{
fix \( x \)
assume \( x \in \text{Interior}(A, (\text{PTopology } X \ U)) \)
with assms obtain \( R \) where \( A: x \in R \in (\text{PTopology } X \ U) R \subseteq A \)
using \text{topology0.open_open_neigh topology0_ptopology topology0.Top_2_L2 topology0.Top_2_L1} by auto
with assms obtain \( B \) where \( B: B \subseteq U = \bigcup B \) using \text{Ptopology_is_a_topology(2) IsAbaseFor_def} by auto
from \( A(1, 3) \) assms have \( XX: x \in XX \in \{T \in \text{DualBase U (PTopology } X \ U). T=X \lor T \cup A \neq X\} \)
using \text{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 \( C:S \subseteq Bx \subseteq S \) by auto
\{
fix \( T \)
assume \( AS: T \in \text{DualBase U (PTopology } X \ U) T \cup A \neq X \)
from \( AS(1) \) assms obtain \( w \) where \( (T=X-w \lor w \subseteq U) \lor (T=X) \)
using \text{DualBase_def union_ptopology Ptopology_is_a_topology(2)} by auto
with assms(2) \( AS(2) \) have \( D: T=X-w \subseteq U \) by auto
from \( D(2) \) have \( w \subseteq U \) by auto
with assms(1) have \( w \subseteq \bigcup (\text{PTopology } X \ U) \) using \text{Ptopology_is_a_topology(2) Top_1_2_L5[of UPTopology X U]} by auto
with assms(1) have \( w \subseteq X \) using \text{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)≠∅ by auto
  with \langle X-T∈U \rangle 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 \langle X-S∪A=X \rangle 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
  assume p:x∈\{T∈DualBase U (PTopology X U). T=X∪T∪A≠X\}
  hence noE:\{T∈DualBase U (PTopology X U). T=X∪T∪A≠X\}≠∅ by auto
  \{
    fix T
    assume T∈DualBase U (PTopology X U)
    with assms(1) obtain w where T=X∪(w∈U∧T=X-w) using DualBase_def
    Ptopology_is_a_topology(2) union_ptopology by auto
    with assms(1) have T=X∪(w∈(PTopology X U)∧T=X-w) using base_sets_open
    Ptopology_is_a_topology(2) by blast
    with assms(1) have T\{is closed in\}(PTopology X U) using topology0.Top_3_L1[where T=PTopology X U]
    topology0_ptopology topology0.Top_3_L9[where T=PTopology X U]
    union_ptopology by auto
  \}
  moreover
  from assms(1) p have X∈\{T∈DualBase U (PTopology X U). T=X∪T∪A≠X\} and
  X:x∈X using Ptopology_is_a_topology(2)
  DualBase_def union_ptopology by auto
  with calculation assms(1) have (∩\{T∈DualBase U (PTopology X U).
  T=X∪T∪A≠X\}) (is closed in)(PTopology X U)
  using topology0.Top_3_L4[where K=\{T∈DualBase U (PTopology X U).
  T=X∪T∪A≠X\}] topology0_ptopology[where U=U and X=X]
  by auto
  with assms(1) have ab:(\{T∈DualBase U (PTopology X U). T=X∪T∪A≠X\}∈(PTopology X U)
  using closed_sets_ptopology by auto

780
with assms(1) obtain B where $B \in \text{Pow}(U)(\bigcap \{T \in \text{DualBase } U \ (\text{PTopology } X \ U) \ . \ T = X \lor T \cup A \neq X\}) = \bigcup B$
  using $\text{PTopology\_is\_a\_topology}(2)$ $\text{IsAbaseFor\_def}$ by auto
  with p obtain R where $x \in R \subseteq \bigcap \{T \in \text{DualBase } U \ (\text{PTopology } X \ U) \ . \ T = X \lor T \cup A \neq X\}$
  by auto
  with assms(1) have $R : x \in R \subseteq (\bigcap \{T \in \text{DualBase } U \ (\text{PTopology } X \ U) \ . \ T = X \lor T \cup A \neq X\})$
  by (safe, blast, simp, blast)
  
  assume $(X-R) \cup A \neq X$
  with R(4) have $X-R \subseteq \{T \in \text{DualBase } U \ (\text{PTopology } X \ U) \ . \ T = X \lor T \cup A \neq X\}$
  by auto
  hence $\bigcap \{T \in \text{DualBase } U \ (\text{PTopology } X \ U) \ . \ T = X \lor T \cup A \neq X\} \subseteq X-R$
  by auto
  hence $R=0$ by blast
  with R(1) have False by auto

hence $I : (X-R) \cup A = X$ by auto

fix y
assume ASR : $y \in R$
with R(2) have $y \in \bigcup (\text{PTopology } X \ U)$ by auto
with assms(1) have $XX : y \in X$ using union_ptopology by auto
with I have $y \in (X-R) \cup A$ by auto
with XX have $y \notin R \lor y \in A$ by auto
with ASR have $y \in A$ by auto

hence $R \subseteq A$ by auto
with R(1,2) have $\exists R \in (\text{PTopology } X \ U) . (x \in R \land R \subseteq A)$ by auto
with assms(1) have $x \in \text{Interior}(A, (\text{PTopology } X \ U))$ using topology0.Top_2_L6
  topology0_ptopology by auto

thus $\bigcap \{T \in \text{DualBase } U \ (\text{PTopology } X \ U) \ . \ T = X \lor T \cup A \neq X\} \subseteq \text{Interior}(A, (\text{PTopology } X \ U))$
  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 $U \ (\text{is a partition of}) \ X \subseteq X$
  shows $\text{Closure}(A, (\text{PTopology } X \ U)) = \bigcup \{T \in U . T \cap A \neq \emptyset\}$
proof
  fix x
  assume $A : x \in \text{Closure}(A, (\text{PTopology } X \ U))$

  781
with assms have $x\in\bigcup(\text{PTopology } X U)$ using topology0.Top_3_L11(1)[where
T=\text{PTopology } X U
and A=A] topology0.ptopology union_ptopology by auto
with assms(1) have $x\in\bigcup\{T\in U . T \cap A \neq 0\}$ by auto
then obtain $W$ where $B:x\in W\subseteq U$ by auto
moreover
from assms $B(2)$ have $W\in(\text{PTopology } X U)$A\subseteq X using base_sets_open Ptopology_is_a_topology(2)
by (safe,blast)
with calculation assms(1) have $A\cap W\neq 0$ using topology0.ptopology[where
U=U and X=X]
topology0.cl_inter_neigh union_ptopology by auto
with B have $x\in\bigcup\{T\subseteq X . T \cap A \neq 0\}$ by auto
thus $\text{Closure}(A, \text{PTopology } X U) \subseteq \bigcup\{T\subseteq X . T \cap A \neq 0\}$ by auto
{fix $x$
assume $x\in\bigcup\{T \subseteq X . T \cap A \neq 0\}$
then obtain $T$ where $A:x\in TT\cap A \neq 0$ by auto
from assms have $A\subseteq\bigcup(\text{PTopology } X U)$ using union_ptopology by auto
moreover
from A(1,2) assms(1) have $x\in\bigcup(\text{PTopology } X U)$ using Top_1_2_L5[where
B=U and T=\text{PTopology } X U]
Ptopology_is_a_topology(2) by auto
moreover
{fix $Q$
assume $B:Q\in(\text{PTopology } X U)x\in Q$
with assms(1) obtain $M$ where $C:Q=\bigcup MM\subseteq U$ using
Ptopology_is_a_topology(2)
IsAbaseFor_def by auto
from B(2) C(1) obtain $R$ where $D:R\subseteq Mx\in R$ by auto
with C(2) A(1,2) have $R\cap T\neq \emptyset\subseteq UT\subseteq U$ by auto
with assms(1) have $R\cap T\subseteq UT\subseteq U$ by auto
with C(1) D(1) have $T\subseteq Q$ by auto
with A(3) have $Q\cap A \neq 0$ by auto
}
then have $\forall Q\in(\text{PTopology } X U). x\in Q \rightarrow Q\cap A \neq 0$ by auto
with calculation assms(1) have $x\in\text{Closure}(A,(\text{PTopology } X U))$ using
topology0.inter_neigh_cl
topology0_ptopology by auto
}
then show $\bigcup\{T \subseteq U . T \cap A \neq 0\} \subseteq \text{Closure}(A, \text{PTopology } X U)$ 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} \( X \)

shows \( \text{Boundary}(A, (\text{PTopology } X U)) = \bigcup \{ T \in U . T \cap A \neq 0 \} \)

**Proof:**

from assms have \( \text{Closure}(A, (\text{PTopology } X U)) = \bigcup \{ T \in U . T \cap A \neq 0 \} \) using

\( \text{closure_set_ptopology} \) by auto

moreover from assms(1) have \( \text{Interior}(A, (\text{PTopology } X U)) = \bigcup \{ T \in U . T \subseteq A \} \)

using \( \text{interior_set_base_topology Ptopology_is_a_topology[where U=U and X=X]} \) by auto

with calculation assms have \( \text{A:Boundary}(A, (\text{PTopology } X U)) = \bigcup \{ T \in U . T \cap A \neq 0 \} \)

using \( \text{topology0.Top_3_L12 topology0_ptopology union_ptopology} \) by auto

from assms(1) have \( \{ \{ T \in U . T \cap A \neq 0 \} \} \) {is a partition of} \( \bigcup \{ \{ T \in U . T \cap A \neq 0 \} \} \)

using subpartition by blast

moreover

\[
\begin{align*}
\{ & \text{fix } T \\
& \text{assume } T \in U \subseteq A \\
& \text{with assms(1) have } T \cap A = T \neq 0 \text{ using } \text{IsAPartition_def by auto} \\
& \text{with } <T \in U> \text{ have } T \cap A \neq 0 \text{ by auto}
\end{align*}
\]

then have \( \{ T \in U . T \subseteq A \} \subseteq \{ T \in U . T \cap A \neq 0 \} \) by auto

ultimately have \( \bigcup \{ T \in U . T \cap A \neq 0 \} \subseteq \{ T \in U . T \subseteq A \} = \bigcup \{ \{ T \in U . T \cap A \neq 0 \} \} \)

using \( \text{diff_union_is_union_diff} \) by auto

also have \( \ldots = \bigcup \{ \{ T \in U . T \cap A \neq 0 \} \land (T \subseteq A) \} \) by blast

with calculation A show thesis by auto

**Qed**

### 63.7 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 \( \text{IsAPartition_def by auto} \)

**Lemma indiscrete_partition:**

assumes \( X \neq 0 \)

shows \( \{ X \} \) {is a partition of} \( X \)

using assms \( \text{IsAPartition_def by auto} \)
theorem discrete_ptopology:
  shows (PTopology X \{x : x ∈ X\}) = Pow(X)
proof
{  fix t
  assume t ∈ (PTopology X \{x : x ∈ X\})
  hence t ⊆ ∪ (PTopology X \{x : x ∈ X\}) by auto
  then have t ∈ Pow(X) using union_ptopology
    discrete_partition by auto
}  thus (PTopology X \{x : x ∈ X\}) ⊆ Pow(X) by auto
{  fix t
    assume A : t ∈ Pow(X)
  have ∪ (\{x : x ∈ t\}) = t by auto
    moreover from A have \{x : x ∈ t\} ∈ Pow(\{x : x ∈ X\}) by auto
    hence ∪ (\{x : x ∈ t\}) ∈ (\{A : A ∈ Pow(\{x : x ∈ X\})\}) by auto
    ultimately have t ∈ (PTopology X \{x : x ∈ X\}) using Ttopology_is_a_topology(2)
      discrete_partition IsAbaseFor_def by auto
}  thus Pow(X) ⊆ (PTopology X \{x : x ∈ X\}) by auto
qed

theorem indiscrete_ptopology:
  assumes X ≠ 0
  shows (PTopology X \{x\}) = \{0,X\}
proof
{  fix T
    assume T ∈ (PTopology X \{x\})
      with asms obtain M where M ⊆ \{X\} ∪ M = T using Ttopology_is_a_topology(2)
        indiscrete_partition IsAbaseFor_def by auto
    then have T = 0 ∨ T = X by auto
}  then show (PTopology X \{x\}) ⊆ \{0,X\} by auto
  from asms have 0 ∈ (PTopology X \{x\}) using Ttopology_is_a_topology(1)
    empty_open
      indiscrete_partition by auto
    moreover
      from asms have ∪ (PTopology X \{x\}) ∈ (PTopology X \{x\}) using union_open
        Ttopology_is_a_topology(1)
          indiscrete_partition by auto
        with asms have X ∈ (PTopology X \{x\}) using union_ptopology indiscrete_partition
          by auto
        ultimately show \{0,X\} ⊆ (PTopology X \{x\}) by auto
    qed

The topological subspaces of the (PTopology X U) are partition topologies.
lemma subspace_ptopology:
  assumes U{is a partition of}X
  shows (PTopology X U) {restricted to} Y=(PTopology (X∩Y) ((U {restricted to} Y)-{0}))
proof-
  from assms have U{is a base for}(PTopology X U) using Ptopology_is_a_topology(2)
    by auto
  then have (U{restricted to} Y){is a base for}(PTopology X U){restricted to} Y
    using subspace_base_topology by auto
  then have (U{restricted to} Y)-{0}){is a base for}(PTopology X U){restricted to} Y
    by auto
  moreover
  from assms have ((U{restricted to} Y)-{0}) {is a partition of} (X∩Y)
    using restriction_partition by auto
  then have ((U{restricted to} Y)-{0}){is a base for}(PTopology (X∩Y)
    ((U {restricted to} Y)-{0}))
    using Ptopology_is_a_topology(2) by auto
  ultimately show thesis using same_base_same_top by auto
qed

63.8 Order topologies

63.9 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∈Xbv∈Xcu∈Xcv∈XIsLinOrder(X,r)
  shows IntervalX(X,r,ru,cv)∩IntervalX(X,r,bv,cv)=IntervalX(X,r,GreaterOf(r,ru,bv),SmallerOf(r,cu,cv))
proof
  have T:GreaterOf(r,ru,bv)∈XSmallerOf(r,cu,cv)∈X using assms
    GreaterOf_def SmallerOf_def by (cases (ru,bv)∈r,simp,simp,cases (cu,cv)∈r,simp,simp)
  { fix x
    assume ASS:x∈IntervalX(X,r,ru,cv)∩IntervalX(X,r,bv,cv)
    
    785
then have \( x \in \text{IntervalX}(X, r, bu, cu) \) by auto
then have \( BB: x \in X \in \text{Interval}(r, bu, cu) \neq \text{bux} \neq \text{cux} \in \text{Interval}(r, bv, cv) \neq \text{bvx} \neq \text{cv} \)
using \( \text{IntervalX_def assms by auto} \)
then have \( x \in X \) by auto
moreover
have \( x \neq \text{GreaterOf}(r, bu, bv) x \neq \text{SmallerOf}(r, cu, cv) \)
proof-
show \( x \neq \text{GreaterOf}(r, bu, bv) \) using \( \text{GreaterOf_def BB(6,3) by (cases } (bu, bv) \in r, \text{simp+)} \)
show \( x \neq \text{SmallerOf}(r, cu, cv) \) using \( \text{SmallerOf_def BB(7,4) by (cases } (cu, cv) \in r, \text{simp+)} \)
qed
moreover
have \( (bu, x) \in r (x, cu) \in r (bv, x) \in r (x, cv) \in r \) using \( \text{BB(2,5) Order_ZF_2_L1A by auto} \)
then have \( (\text{GreaterOf}(r, bu, bv), x) \in r (x, \text{SmallerOf}(r, cu, cv)) \in r \) using \( \text{GreaterOf_def SmallerOf_def} \)
by (cases \( (bu, bv) \in r, \text{simp, simp, cases } (cu, cv) \in r, \text{simp, simp} \)
then have \( x \in \text{Interval}(r, \text{GreaterOf}(r, bu, bv), \text{SmallerOf}(r, cu, cv)) \)
using \( \text{Order_ZF_2_L1 by auto} \)
ultimately
have \( x \in \text{IntervalX}(X, r, \text{GreaterOf}(r, bu, bv), \text{SmallerOf}(r, cu, cv)) \) using \( \text{IntervalX_def T by auto} \)
\}
then show \( \text{IntervalX}(X, r, bu, cu) \cap \text{IntervalX}(X, r, bv, cv) \subseteq \text{IntervalX}(X, r, \text{GreaterOf}(r, bu, bv), \text{SmallerOf}(r, cu, cv)) \)
by auto
\}
fix \( x \)
assume \( x \in \text{IntervalX}(X, r, \text{GreaterOf}(r, bu, bv), \text{SmallerOf}(r, cu, cv)) \)
then have \( BB: x \in X \in \text{Interval}(r, \text{GreaterOf}(r, bu, bv), \text{SmallerOf}(r, cu, cv)) \neq \text{GreaterOf}(r, bu, bv) \)
using \( \text{IntervalX_def T by auto} \)
then have \( x \in X \) by auto
moreover
from \( \text{BB(2) have CC: } (\text{GreaterOf}(r, bu, bv), x) \in r (x, \text{SmallerOf}(r, cu, cv)) \in r \)
using \( \text{Order_ZF_2_L1A by auto} \)
\{
\{
assume \( AS: (bu, bv) \in r \)
then have \( \text{GreaterOf}(r, bu, bv) = bv \) using \( \text{GreaterOf_def by auto} \)
then have \( (bv, x) \in r \) using \( \text{CC(1) by auto} \)
with \( AS \) have \( (bu, x) \in r (bv, x) \in r \) using \( \text{assms IsLinOrder_def trans_def} \)
by (safe, blast) \}
moreover
\{
assume \( AS: (bu, bv) \notin r \)
then have \( \text{GreaterOf}(r, bu, bv) = bu \) using \( \text{GreaterOf_def by auto} \)
then have \( (bu, x) \in r \) using \( \text{CC(1) by auto} \)
\}
786
from AS have ⟨bv, bu⟩∈ r using assms IsLinOrder_def IsTotal_def
assms by auto
with ⟨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, bv, bu) = 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 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 CC(2) by auto
with AS have ⟨x, cu⟩∈ r ⟨x, cv⟩∈ r using assms 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 CC(2) by auto
from AS have ⟨cv, cu⟩∈ r using assms IsLinOrder_def IsTotal_def
by auto
with ⟨x, cv⟩∈ r have ⟨x, cv⟩∈ r ⟨x, cu⟩∈ r using assms IsLinOrder_def
trans_def by (safe, blast)
}
ultimately have R: ⟨x, cv⟩∈ r ⟨x, cu⟩∈ r by auto
moreover
{
assume AS: x = cv
then have ⟨cv, cu⟩∈ r using R(2) by auto
then have SmallerOf(r, cv, cu) = 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 (cu,cv)∈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 ⟨x,cu⟩∈r ⟨x,cv⟩∈rx ≠ cu ≠ cv by auto
}

ultimately have x∈IntervalX(X,r,bu,cu) x∈IntervalX(X,r,bv,cv) using Order_ZF_2_L1 IntervalX_def

assms by auto
then have x∈IntervalX(X,r,bu,cu) ∩ IntervalX(X,r,bv,cv) by auto

then show IntervalX(X,r,GreaterOf(r,bu,bv),SmallerOf(r,cu,cv)) ⊆ IntervalX(X,r,bu,cu) ∩ IntervalX(X,r,bv,cv)
by auto
qed

lemma inter_rray_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∈IntervalX(X,r,bu,cu) x∈Interval(r,bv,cv) by auto
  then have BB:x∈x≠bu≠bv≠cv(bu,x)∈rx using RightRayX_def IntervalX_def
by auto
then have ⟨bv,x⟩∈r(x,cv)∈r using Order_ZF_2_L1A by auto
with ⟨bu,x⟩∈r have ⟨GreaterOf(r,bu,bv),x⟩∈r using GreaterOf_def
by (cases ⟨bu,bv⟩∈r,simp+)
with ⟨x,cv⟩∈r have x∈Interval(r,GreaterOf(r,bu,bv),cv) using Order_ZF_2_L1 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∈x∈x∈Interval(r,GreaterOf(r,bu,bv),cv)x≠cvx≠GreaterOf(r,
bu, bv) using IntervalX_def by auto
then have R:⟨GreaterOf(r, bu, bv),x⟩∈r⟨x,cv⟩∈r using Order_ZF_2_L1A
by auto
with ⟨x≠cv⟩ have ⟨x,cv⟩∈rx≠cv by auto
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 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 ⟨bu,x⟩∈r (bv,x)∈r using assms unfolding IsLinOrder_def
trans_def by (safe,blast)
}
ultimately have T:⟨bu,x⟩∈r (bv,x)∈r by auto
moreover
{ 
  assume AS:x=bu
  then have ⟨bv,bu⟩∈r using T(2) by auto
  then have GreaterOf(r,bu,bv)=bu unfolding GreaterOf_def using
assms unfolding IsLinOrder_def
antisym_def by auto
  with ⟨x≠GreaterOf(r,bu,bv)⟩ have False using AS by auto
}
moreover
{ 
  assume AS:x=bv
  then have ⟨bu,bv⟩∈r using T(1) by auto
  then have GreaterOf(r,bu,bv)=bv unfolding GreaterOf_def by auto
  with ⟨x≠GreaterOf(r,bu,bv)⟩ have False using AS by auto
}
ultimately have ⟨bu,x⟩∈r (bv,x)∈rx≠bux≠bv by auto
}
with calculation ⟨x∈X⟩ have x∈RightRayX(X, r, bu)x∈IntervalX(X, r, bv, cv) unfolding RightRayX_def IntervalX_def
using Order_ZF_2_L1 by auto
then have x∈RightRayX(X, r, bu) ∩ IntervalX(X, r, bv, cv) by auto
}
then show IntervalX(X, r, GreaterOf(r, bu, bv), cv) ⊆ RightRayX(X, r, bu) ∩ IntervalX(X, r, bv, cv) by auto
}
lemma inter_lray_interval:
  assumes bv∈Xcu∈Xcv∈IsLinOrder(X,r)
  shows LeftRayX(X,r,cu)∩IntervalX(X,r,bv,cv)=IntervalX(X,r,bv,SmallerOf(r,cu,cv))
proof
{  
  fix x assume x∈LeftRayX(X,r,cu)∩IntervalX(X,r,bv,cv)
  then have B:x≠cu∈X,cu∈r(bv,x)∈r(x,cv)∈r\ bx≠cv unfolding LeftRayX_def IntervalX_def
  Interval_def Interval_def
  by auto
  from B(7,1) have C:x∈IntervalX(X,r,bv,SmallerOf(r,cu,cv)) using B C IntervalX_def
  Order_ZF_2_L1 by auto
}  
then show LeftRayX(X,r,cu)∩IntervalX(X,r,bv,cv)⊆IntervalX(X,r,bv,SmallerOf(r,cu,cv))
proof
{  
  fix x assume x∈IntervalX(X,r,bv,SmallerOf(r,cu,cv))
  then have R:x∈X,bv∈r,x∈r, SmallerOf(r,cu,cv)=cu using IsLinOrder_def IsTotal_def
  then have \langle x,cu\rangle∈r\ bv≠SmallerOf(r,cu,cv) using SmallerOf_def by (cases \langle cu,cv\rangle∈r,simp+)
  then have \langle x,bv\rangle∈r\ bv≠SmallerOf(r,cu,cv) using B C IntervalX_def
  Order_ZF_2_L1 by auto
  from B(7,1) have C:x∈IntervalX(X,r,bv,SmallerOf(r,cu,cv)) using B C IntervalX_def
  Interval_def Interval_def
  by auto
  then have \langle bv,x\rangle∈r\ bx≠bv by auto
  moreover
  {  
    assume AS:\langle cu,cv\rangle∈r
    then have SmallerOf(r,cu,cv)=cu using SmallerOf_def by auto
    then have \langle x,cu\rangle∈r\ x∈r, using R(3) by auto
    with AS have \langle x,cu\rangle∈r\ x∈r, using assms unfolding IsLinOrder_def
    trans_def by (safe, blast)
  }
  moreover
  {  
    assume AS:\langle cu,cv\rangle∉r
    then have SmallerOf(r,cu,cv)=cv using SmallerOf_def by auto
    then have \langle x,cv\rangle∈r\ x∈r, using R(3) by auto
    from AS have \langle cv,cu\rangle∉r\ cv∈r using assms IsLinOrder_def IsTotal_def
    trans_def by (safe, blast)
  }
  ultimately have T:\langle x,cv\rangle∈r\ x∈r by auto
  moreover
  {  
    assume AS:\langle cu,cv\rangle∈r
    then have SmallerOf(r,cu,cv)=cu using SmallerOf_def by auto
    then have \langle x,cu\rangle∈r\ x∈r, using R(3) by auto
    from AS have \langle cv,cu\rangle∉r\ cv∈r using assms IsLinOrder_def IsTotal_def
    trans_def by (safe, blast)
  }
ultimately have T:\langle x,cv\rangle∈r\ x∈r by auto
moreover
}
assume $AS: x = cu$
then have $\langle cu, cv \rangle \in r$ using T(1) by auto
then have $\operatorname{SmallerOf}(r, cu, cv) = cu$ using $\operatorname{SmallerOf\_def}$ asms $\operatorname{IsLinOrder\_def}$
$\operatorname{antisym\_def}$ by auto
with $x \neq \operatorname{SmallerOf}(r, cu, cv)$ have False using $AS$ by auto

moreover
{
assume $AS: x = cv$
then have $\langle cv, cu \rangle \in r$ using T(2) by auto
then have $\operatorname{SmallerOf}(r, cu, cv) = cv$ using $\operatorname{SmallerOf\_def}$ asms $\operatorname{IsLinOrder\_def}$
$\operatorname{antisym\_def}$ by auto
with $x \neq \operatorname{SmallerOf}(r, cu, cv)$ have False using $AS$ by auto
}
ultimately have $\langle x, cu \rangle \in r \langle x, cv \rangle \in r$ $x \neq cu \neq cv$ by auto

with calculation $x \in X$ have $x \in \operatorname{LeftRayX}(X, r, cu) \cap \operatorname{IntervalX}(X, r, bv, cv)$
using $\operatorname{LeftRayX\_def} \ \operatorname{IntervalX\_def} \ \operatorname{Interval\_def}$ by auto
then have $x \in \operatorname{LeftRayX}(X, r, cu) \cap \operatorname{IntervalX}(X, r, bv, cv)$ by auto
then show $\operatorname{IntervalX}(X, r, bv, \operatorname{SmallerOf}(r, cu, cv)) \subseteq \operatorname{LeftRayX}(X, r, cu)$

qed

lemma inter_lray_rray:
assumes $bu \in X \ \operatorname{cv} \in X \ \operatorname{IsLinOrder}(X, r)$
shows $\operatorname{LeftRayX}(X, r, bu) \cap \operatorname{RightRayX}(X, r, cv) = \operatorname{IntervalX}(X, r, cv, bu)$
unfolding $\operatorname{LeftRayX\_def} \ \operatorname{RightRayX\_def} \ \operatorname{IntervalX\_def} \ \operatorname{Interval\_def}$ by auto

lemma inter_lray_lray:
assumes $bu \in X \ \operatorname{cv} \in X \ \operatorname{IsLinOrder}(X, r)$
shows $\operatorname{LeftRayX}(X, r, bu) \cap \operatorname{LeftRayX}(X, r, cv) = \operatorname{LeftRayX}(X, r, \operatorname{SmallerOf}(r, bu, cv))$
proof
{
fix $x$
assume $x \in \operatorname{LeftRayX}(X, r, bu) \cap \operatorname{LeftRayX}(X, r, cv)$
then have $B: x \in X \ \operatorname{bu} \in X \ \operatorname{cv} \in X \ \operatorname{r} \ \operatorname{rx} \neq bu \neq cv$ using $\operatorname{LeftRayX\_def}$ by auto
then have $C: (x, \operatorname{SmallerOf}(r, bu, cv)) \in r$ using $\operatorname{SmallerOf\_def}$ by (cases $\langle bu, cv \rangle \in X \ \operatorname{r}$, auto)
from $B$ have $D: x \neq \operatorname{SmallerOf}(r, bu, cv)$ using $\operatorname{SmallerOf\_def}$ by (cases $\langle bu, cv \rangle \in X \ \operatorname{r}$, auto)
from $B \ C \ D$ have $x \in \operatorname{LeftRayX}(X, r, \operatorname{SmallerOf}(r, bu, cv))$ using $\operatorname{LeftRayX\_def}$ by auto
}
then show $\operatorname{LeftRayX}(X, r, bu) \cap \operatorname{LeftRayX}(X, r, cv) \subseteq \operatorname{LeftRayX}(X, r, \operatorname{SmallerOf}(r, bu, cv))$ by auto
{
fix $x$

791
assume \( x \in \text{LeftRayX}(X, r, \text{SmallerOf}(r, bu, cv)) \)
then have \( R : x \in X, \text{SmallerOf}(r, bu, cv) \in r \neq \text{SmallerOf}(r, bu, cv) \) using \( \text{LeftRayX_def} \) by auto
{
    
    assume AS:\( \langle bu, cv \rangle \in r \)
    then have \( \text{SmallerOf}(r, bu, cv) = bu \) using \( \text{SmallerOf_def} \) by auto
    then have \( \langle x, bu \rangle \in r \) using \( R(2) \) by auto
    with AS have \( \langle x, bu \rangle \in r \langle x, cv \rangle \in r \) using \( \text{assms IsLinOrder_def trans_def} \)
by(safe, blast)
}
moreover
{
    assume AS:\( \langle bu, cv \rangle \not\in r \)
    then have \( \text{SmallerOf}(r, bu, cv) = cv \) using \( \text{SmallerOf_def} \) 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 \( \text{assms IsLinOrder_def IsTotal_def asssms by auto} \)
    with \( \langle x, cv \rangle \in r \) have \( \langle x, bu \rangle \in r \langle x, cv \rangle \in r \) using \( \text{assms IsLinOrder_def trans_def by(safe, blast)} \)
}
ultimately have T:\( \langle x, cv \rangle \in r \langle x, bu \rangle \in r \) by auto
moreover
{
    assume AS:\( x = bu \)
    then have \( \langle bu, cv \rangle \in r \) using \( T(1) \) by auto
    then have \( \text{SmallerOf}(r, bu, cv) = bu \) using \( \text{SmallerOf_def assms IsLinOrder_def antisym_def by auto} \)
    with \( \langle x \neq \text{SmallerOf}(r, bu, cv) \rangle \) have False using AS by auto
}
moreover
{
    assume AS:\( x = cv \)
    then have \( \langle cv, bu \rangle \in r \) using \( T(2) \) by auto
    then have \( \text{SmallerOf}(r, bu, cv) = cv \) using \( \text{SmallerOf_def assms IsLinOrder_def antisym_def by auto} \)
    with \( \langle x \neq \text{SmallerOf}(r, bu, cv) \rangle \) have False using AS by auto
}
ultimately have \( \langle x, bu \rangle \in r \langle x, cv \rangle \in r \langle x \neq bu \neq cv \rangle \) by auto
}
with \( \langle x \in X \rangle \) have \( x \in \text{LeftRayX}(X, r, bu) \cap \text{LeftRayX}(X, r, cv) \) using \( \text{LeftRayX_def by auto} \)
}
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 inter_rray_rray:
    assumes \( bu \in X, cv \in X \) \( \text{IsLinOrder}(X, r) \)
shows \( \text{RightRayX}(X, r, bu) \cap \text{RightRayX}(X, r, cv) = \text{RightRayX}(X, r, \text{GreaterOf}(r, bu, cv)) \)

proof

\[
\begin{align*}
\text{fix } x \\
\text{assume } x \in \text{RightRayX}(X, r, bu) \cap \text{RightRayX}(X, r, cv) \\
\text{then have } B: x \in X, \langle bu, x \rangle \in r, \langle cv, x \rangle \in r, bu \neq x, cv \neq x \\
\text{using } \text{RightRayX_def} \text{ by auto} \\
\text{from } B \text{ have } D: x \neq \text{GreaterOf}(r, bu, cv) \\
\text{using } \text{GreaterOf_def} \text{ by } (\text{cases } (bu, cv) \in r, \text{auto}) \\
\text{from } B, C, D \text{ have } x \in \text{RightRayX}(X, r, \text{GreaterOf}(r, bu, cv)) \\
\text{using } \text{RightRayX_def} \text{ by auto} \\
\end{align*}
\]

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

\[
\begin{align*}
\text{fix } x \\
\text{assume } x \in \text{RightRayX}(X, r, \text{GreaterOf}(r, bu, cv)) \\
\text{then have } R: x \in X, \langle \text{GreaterOf}(r, bu, cv), x \rangle \in r, bu \neq x, cv \neq x \\
\text{using } \text{RightRayX_def} \text{ by auto} \\
\text{\{ \\
\text{assume } AS: \langle bu, cv \rangle \in r \\
\text{then have } \text{GreaterOf}(r, bu, cv) = cv \text{ using } \text{GreaterOf_def} \text{ by auto} \\
\text{then have } \langle cv, x \rangle \in r \text{ using } \text{R(2)} \text{ by auto} \\
\text{with } AS \text{ have } \langle bu, x \rangle \in r \langle cv, x \rangle \in r \text{ using } \text{assms } \text{IsLinOrder_def} \text{ trans_def} \\
\text{by } (\text{safe, blast}) \}
\end{align*}
\]

moreover

\[
\begin{align*}
\text{assume } AS: \langle bu, cv \rangle \notin r \\
\text{then have } \text{GreaterOf}(r, bu, cv) = bu \text{ using } \text{GreaterOf_def} \text{ by auto} \\
\text{then have } \langle bu, x \rangle \in r \text{ using } \text{R(2)} \text{ by auto} \\
\text{from } AS \text{ have } \langle cv, bu \rangle \in r \text{ using } \text{assms } \text{IsLinOrder_def} \text{ IsTotal_def} \\
\text{assms by auto} \\
\text{with } \langle bu, x \rangle \in r \text{ have } \langle cv, x \rangle \in r \langle bu, x \rangle \in r \text{ using } \text{assms } \text{IsLinOrder_def} \text{ trans_def} \text{ by } (\text{safe, blast}) \}
\end{align*}
\]

ultimately have \( T: \langle cv, x \rangle \in r, \langle bu, x \rangle \in r \) by auto

moreover

\[
\begin{align*}
\text{assume } AS: x = bu \\
\text{then have } \langle cv, bu \rangle \in r \text{ using } \text{T(1)} \text{ by auto} \\
\text{then have } \text{GreaterOf}(r, bu, cv) = bu \text{ using } \text{GreaterOf_def} \text{ assms } \text{IsLinOrder_def} \text{ antisym_def} \text{ by auto} \\
\text{with } x \neq \text{GreaterOf}(r, bu, cv) \text{ have False using } AS \text{ by auto} \\
\end{align*}
\]

moreover

\[
\begin{align*}
\end{align*}
\]
assume AS: \(x = cv\)
then have \((bu, cv) \in r\) using T(2) by auto
then have \(\text{GreaterOf}(r, bu, cv) = cv\) using \(\text{GreaterOf_def}\) assm IsLinOrder_def 
antisym_def by auto
with \(x \neq \text{GreaterOf}(r, bu, cv)\) have False using AS by auto
ultimately have \((bu, x) \in r\) \((cv, x) \in r\) \(\neq bu \neq cv\) by auto
with \(x \in X\) have \(x \in \text{RightRayX}(X, r, bu) \cap \text{RightRayX}(X, r, cv)\) using RightRayX_def by auto
then show \(\text{RightRayX}(X, r, \text{GreaterOf}(r, bu, cv)) \subseteq \text{RightRayX}(X, r, bu) \cap \text{RightRayX}(X, r, cv)\) by auto
qed

The open intervals and rays satisfy the base condition.

lemma intervals_rays_base_condition:
  assumes IsLinOrder(X, r)
  shows \(\{\text{IntervalX}(X, r, b, c). (b, c) \in X \times X\} \cup \{\text{LeftRayX}(X, r, b). b \in X\} \cup \{\text{RightRayX}(X, r, b). b \in X\}\) \{satisfies the base condition\}
proof-
  let I=\(\{\text{IntervalX}(X, r, b, c). (b, c) \in X \times X\}\)
  let R=\(\{\text{RightRayX}(X, r, b). b \in X\}\)
  let L=\(\{\text{LeftRayX}(X, r, b). b \in X\}\)
  let B=\(\{\text{IntervalX}(X, r, b, c). (b, c) \in X \times X\} \cup \{\text{LeftRayX}(X, r, b). b \in X\} \cup \{\text{RightRayX}(X, r, b). b \in X\}\) 
  { fix U V 
    assume A:U\(\in B\)V\(\in B\) 
    then have dU:U\(\in I\)V\(\in I\)U\(\in L\)V\(\in L\)and dV:V\(\in I\)V\(\in I\)V\(\in L\)V\(\in L\)R by auto 
    { assume S:V\(\in I\) 
      { assume U\(\in I\) 
        with S obtain bu cu bv cv where A:U=\(\text{IntervalX}(X, r, bu, cu)\)V=\(\text{IntervalX}(X, r, bv, cv)\)bu\(\in X\ubv\(\in X\) 
        by auto 
        then have SmallerOf(r, cu, cv)\(\in X\)GreaterOf(r, bu, bv)\(\in X\) by (cases \(cu, cv\)\(\in X\)Simp add:SmallerOf_def A, simp add:SmallerOf_def A, simp add:GreaterOf_def A, simp add:GreaterOf_def A) 
      } 
      moreover 
      { assume U\(\in L\) 
        with S obtain bu bv cv where A:U=\(\text{LeftRayX}(X, r, bu, cv)\)V=\(\text{IntervalX}(X, r, bv, cv)\)bu\(\in Xbv\(\in Xcv\(\in X\) 
        by auto 
        then have SmallerOf(r, bu, cv)\(\in X\) using SmallerOf_def by (cases \(bu, cv\)\(\in X\)auto) 
      } 
    } 
  } 

with A have \( U \cap V \in B \) using `inter_lray_interval` assms by auto
}
moreover
{
  assume \( U \in R \)
  with \( S \) obtain \( cu \, bv \, cv \) where A: \( U = \text{RightRay}(X, r, cu) \) \( V = \text{Interval}(X, r, bv, cv) \)
           cu \in Xbv \in Xcv \in X
  by auto
  then have `GreaterOf(r, cu, bv) \in X` using `GreaterOf_def` by (cases (cu, bv) \in r, auto)
  with A have \( U \cap V \in B \) using `inter_lray_interval` assms by auto
}
ultimately have \( U \cap V \in B \) using dU by auto
}
moreover
{
  assume \( S : V \in L \)
  {
    assume \( U \in I \)
    with \( S \) obtain \( bu \, bv \, cv \) where A: \( V = \text{LeftRay}(X, r, bu) \) \( U = \text{Interval}(X, r, bv, cv) \)
                   bu \in Xbv \in Xcv \in X
    by auto
    then have `SmallerOf(r, bu, cv) \in X` using `SmallerOf_def` by (cases (bu, cv) \in 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 `inter_lray_interval` assms by auto
  }
moreover
{
  assume \( U \in R \)
  with \( S \) obtain \( bu \, cv \) where A: \( V = \text{LeftRay}(X, r, bu) \) \( U = \text{RightRay}(X, r, cv) \)
                   bu \in Xcv \in X
  by auto
  have \( U \cap V = V \cap U \) by auto
  with A have \( U \cap V \in B \) using `inter_lray_rray` assms by auto
}
moreover
{
  assume \( U \in L \)
  with \( S \) obtain \( bu \, bv \) where A: \( U = \text{LeftRay}(X, r, bu) \) \( V = \text{LeftRay}(X, r, bv) \)
                   bu \in Xbv \in X
  by auto
  then have `SmallerOf(r, bu, bv) \in X` using `SmallerOf_def` by (cases (bu, bv) \in r, auto)
  with A have \( U \cap V \in B \) using `inter_lray_lray` assms by auto
}
ultimately have \( U \cap V \in B \) using dU by auto
}
moreover
{
  assume \( S : V \in R \)
  {
    assume \( U \in I \)
    with \( S \) obtain \( cu \, bv \) where A: \( U = \text{LeftRay}(X, r, cu) \) \( V = \text{LeftRay}(X, r, bv) \)
                   cu \in Xbv \in X
    by auto
    then have `GreaterOf(r, cu, bv) \in X` using `GreaterOf_def` by (cases (cu, bv) \in r, auto)
    with A have \( U \cap V \in B \) using `inter_lray_interval` assms by auto
  }
ultimately have \( U \cap V \in B \) using dU by auto
}
assume \( U \in I \) with \( S \) obtain \( cu, bv, cv \) where \( A:V=\text{RightRay}(X,r,cu) \lor \text{Interval}(X,r,bv,cv) \)
by auto
then have \( \text{GreaterOf}(r,cu,bv) \in X \) using \( \text{GreaterOf_def} \) by (cases
\( (cu,bv) \in r, \text{auto} \)
have \( U \cup V = \text{V} \cup U \) by auto
with \( A < \text{GreaterOf}(r,cu,bv) \in X \) have \( U \cup V \in B \) using \( \text{inter_rRay_interval} \)
sassms by auto
}
moreover
{
assume \( U \in L \) with \( S \) obtain \( bu, cv \) where \( A:U=\text{LeftRay}(X,r,bu) \lor \text{RightRay}(X,r,cv) \)
by auto
then have \( U \cup V \in B \) using \( \text{inter_lRay_rRay} \) assms by auto
}
moreover
{
assume \( U \in R \) with \( S \) obtain \( cu, cv \) where \( A:U=\text{RightRay}(X,r,cu) \lor \text{RightRay}(X,r,cv) \)
by auto
then have \( \text{GreaterOf}(r,cu,cv) \in X \) using \( \text{GreaterOf_def} \) by (cases
\( (cu,cv) \in r, \text{auto} \)
with \( A \) have \( U \cup V \in B \) using \( \text{inter_rRay_rRay} \) assms by auto
}
ultimately have \( U \cup V \in B \) using \( dU \) by auto
ultimately have \( S:U \cup V \in B \) using \( dV \) by auto
{
fix \( x \)
assume \( x \in U \cap V \)
then have \( x \in U \cup V \) by auto
then have \( \exists W. W \in \text{B} \land x \in W \land W \subseteq U \cup V \) using \( S \) by blast
then have \( \exists W \in \text{B} . x \in W \land W \subseteq U \cup V \) by blast
}
hence \( (\forall x \in U \cup V . \exists W \in \text{B} . x \in W \land W \subseteq U \cup V) \) by auto
}
then show thesis using \( \text{SatisfiesBaseCondition_def} \) 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.

definition
\( \text{OrderTopology}(\text{OrdTopology} \_ \_ 50) \) where
\( \text{IsLinOrder}(X,r) \implies \text{OrdTopology}(X,r) \equiv \text{TopologyBase}\{\text{Interval}(X,r,b,c) . \}
\( (b,c) \in X \times X \} \cup \{ \text{LeftRay}(X,r,b) . b \in X \} \cup \{ \text{RightRay}(X,r,b) . b \in X \} \)

theorem \( \text{Ordtopology_is_a_topology} \):
assumes IsLinOrder(X,r)
shows (OrdTopology X r) {is a topology} and (IntervalX(X,r,b,c). (b,c)∈X×X)∪{LeftRayX(X,r,b). b∈X}∪{RightRayX(X,r,b). b∈X} {is a base for} (OrdTopology X r)
using assms Base_topology_is_a_topology intervals_rays_base_condition

OrderTopology_def by auto

lemma topology0_ordtopology:
assumes IsLinOrder(X,r)
shows topology0(OrdTopology X r)
using Ordtopology_is_a_topology topology0_def assms by auto

63.10 Total set

The topology is defined in the set X, when X has more than one point

lemma union_ordtopology:
assumes IsLinOrder(X,r) ∃x y. x≠y ∧ x∈X ∧ y∈X
shows ∪ (OrdTopology X r)=X
proof
let B={IntervalX(X,r,b,c). (b,c)∈X×X}∪{LeftRayX(X,r,b). b∈X}∪{RightRayX(X,r,b). b∈X}
   have base:B {is a base for} (OrdTopology X r) using Ordtopology_is_a_topology(2) assms(1)
   by auto
from assms(2) obtain x y where T:x≠y ∧ x∈X ∧ y∈X by auto
then have B:x∈LeftRayX(X,r,y)∧y∈RightRayX(X,r,y) using LeftRayX_def
RightRayX_def
assms(1) IsLinOrder_def IsTotal_def by auto
then have x∈∪B using T by auto
then have x:x∈∪(OrdTopology X r) using Top_1_2_L5 base by auto
{
  fix z
  assume z:z∈X
  {
    assume x=z
    then have z∈∪(OrdTopology X r) using x by auto
  }
  moreover
  {
    assume x≠z
    with z T have z∈LeftRayX(X,r,x)∧z∈RightRayX(X,r,x) using LeftRayX_def
RightRayX_def
assms(1) IsLinOrder_def IsTotal_def by auto
then have z∈∪B by auto
then have z∈∪(OrdTopology X r) using Top_1_2_L5 base by auto
  }
  ultimately have z∈∪(OrdTopology X r) by auto
}
then show X⊆∪(OrdTopology X r) by auto

797
have \( \bigcup B \subseteq X \) using \( \text{IntervalX_def LeftRayX_def RightRayX_def} \) by auto
then show \( \bigcup (\text{OrdTopology X r}) \subseteq X \) using \( \text{Top_1_2_L5 base by auto} \)
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.

63.11 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, it is necessary to consider \( X \) as an element of the base or that limit point wouldn’t be in any basic open set.

63.11.1 Right and Left Order topologies are topologies

**lemma lefrays_base_condition:**
assumes \( \text{IsLinOrder(X,r)} \)
shows \( \{\text{LeftRayX(X,r,b).} \ b \in X\} \cup \{X\} \) {satisfies the base condition}
proof-
{  
  fix \( U \ V \)
  assume \( \text{U} \in \{\text{LeftRayX(X,r,b).} \ b \in X\} \cup \{X\} \)
  then obtain \( b \ c \) where \( A:(b \in X \land U=\text{LeftRayX(X,r,b)}) \lor U=X (c \in X \land V=\text{LeftRayX(X,r,c)}) \lor V=X \)
  unfolding \( \text{LeftRayX_def} \) by auto
  then have \( U \cap V=\text{LeftRayX(X,r,SmallerOf(r,b,c))} \land b \in X \land c \in X \land V \subseteq X \land U \subseteq X \)
  using \( \text{inter_lray_lray assms by auto} \)
  moreover
  have \( b \in X \land c \in X \rightarrow \text{SmallerOf(r,b,c) \in X} \) unfolding \( \text{SmallerOf_def by (cases} \)
  \( b,c) \in r,auto \)
  ultimately have \( U \cap V \in \{\text{LeftRayX(X,r,b).} \ b \in X\} \cup \{X\} \) by auto
  hence \( \forall x \in U \cap V \exists W \in \{\text{LeftRayX(X,r,b).} \ b \in X\} \cup \{X\}. \ x \in W \land W \subseteq U \cap V \) by blast
}
moreover
then show thesis using \( \text{SatisfiesBaseCondition_def by auto} \)
qed

**lemma rightrays_base_condition:**
assumes \( \text{IsLinOrder(X,r)} \)
shows \( \{\text{RightRayX(X,r,b).} \ b \in X\} \cup \{X\} \) {satisfies the base condition}
proof-
{  
  fix \( U \ V \)
}

798
\begin{verbatim}
assume U \in \{ \text{RightRayX}(X,r,b) \mid b \in X \} \cup \{ X \}
then obtain b, c where A: (b \in X \land U = \text{RightRayX}(X,r,b)) \lor (V = X) \lor (U = \text{RightRayX}(X,r,c)) \lor (V = X)
unfolding RightRayX_def by auto
then have \((U \cap V = \text{RightRayX}(X,r,\text{GreaterOf}(r,b,c)) \land b \in X \land c \in X) \lor (U \cap V = X) \lor (U \cap V = \text{RightRayX}(X,r,b) \land b \in X)\)
using inter_ray_ray assms by auto
moreover have b \in X \land c \in X \rightarrow \text{GreaterOf}(r,b,c) \in X
using GreaterOf_def by (cases \langle b,c \rangle \in r,auto)
ultimately have U \cap V \in \{ \text{RightRayX}(X,r,b) \mid b \in X \} \cup \{ X \} by auto
hence \forall x \in U \cap V. \exists W \in \{ \text{RightRayX}(X,r,b) \mid 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 \ X \ r \ 50) \ where
IsLinOrder(X,r) \implies LOrdTopology X r \equiv \text{TopologyBase} \ \{ \text{LeftRayX}(X,r,b) \mid b \in X \} \cup \{ X \}

definition RightOrderTopology \ (ROrdTopology \ X \ r \ 50) \ where
IsLinOrder(X,r) \implies ROrdTopology X r \equiv \text{TopologyBase} \ \{ \text{RightRayX}(X,r,b) \mid b \in X \} \cup \{ X \}

theorem LOrdtopology_ROrdtopology_are_topologies:
assumes IsLinOrder(X,r)
shows (LOrdTopology X r) \ (is a topology) \ and \ \{ \text{LeftRayX}(X,r,b) \mid b \in X \} \ (is a base for) \ (LOrdTopology X r)
\and (ROrdTopology X r) \ (is a topology) \ and \ \{ \text{RightRayX}(X,r,b) \mid b \in X \} \ (is a base for) \ (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

63.11.2 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]]
\and Top_1_2_L5[OF LOrdtopology_ROrdtopology_are_topologies(4)[OF assms]]
unfolding LeftRayX_def RightRayX_def by auto
\end{verbatim}
63.12 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**

$$\text{joinT} (\text{joinT M})$$

where

$$\forall T \in M. \ T \text{ is a topology} \land (\forall Q \in M. \ \bigcup Q = \bigcup T) \implies (\text{joinT M} \equiv \text{T.} \ (\bigcup M) \text{ is a subbase for } T)$$

First let's prove 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 $$\{\bigcap A. \ A \in \text{FinPow}(B)\} \text{ satisfies the base condition}$$

proof-

{ fix U V 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 MR:Finite(M)Finite(R)M\subseteq BR \subseteq B U=\bigcap MV=\bigcap R using FinPow_def by auto

{ fix x assume AS:x\in U \cup V then have N:M \neq 0R \neq 0 using MR(5,6) by auto have Finite(M \cup R) using MR(1,2) by auto moreover have M \cup R \subseteq \text{Pow}(B) using MR(3,4) by auto ultimately have M \cup R \subseteq \text{FinPow}(B) using FinPow_def by auto then have \bigcap (M \cup R)\in\{\bigcap A. \ A \in \text{FinPow}(B)\} by auto moreover from N have \bigcap (M \cup R)\subseteq M \cap (M \cup R) \subseteq R by auto then have \bigcap (M \cup R)\subseteq U \cap V using MR(5,6) by auto moreover

{ fix S assume S\subseteq M \cup R then have S\subseteq M \cup S is R by auto then have x\in S using AS MR(5,6) by auto

} then have x\in \bigcap (M \cup R) using N by auto ultimately have \exists W\in\{\bigcap A. \ A \in \text{FinPow}(B)\}. x\in W \land W \subseteq U \cup V by blast

} then have (\forall x \in U \cup V. \ \exists W\in\{\bigcap A. \ A \in \text{FinPow}(B)\}. x\in W \land W \subseteq U \cup V) by auto

800
then have $\forall U \subseteq \bigcap A. A \in \text{FinPow}(B) \land \forall V \subseteq \bigcap A. A \in \text{FinPow}(B)$

$\rightarrow$

$(\forall x \in U \cap V. \exists W \subseteq \bigcap A. A \in \text{FinPow}(B))$ by auto

then show $(\bigcap A. A \in \text{FinPow}(B)) \text{ satisfies the base condition}$

using SatisfiesBaseCondition_def by auto

done

definition Top_subbase:
assumes $T = \{\bigcup A. A \in \text{FinPow} \{\bigcap A. A \in \text{FinPow}(B)\}\}$

shows $T \text{ is a topology}$ and $B \text{ is a subbase for } T$

proof

fix $S$
assume $S \subseteq B$

then have $\{S\} \subseteq \text{FinPow}(B) \cap \{S\} = S$ using FinPow_def auto

then have $\{S\} \subseteq \text{Pow} \{\bigcap A. A \in \text{FinPow}(B)\}$ by (blast+)

then have $\bigcup \{S\} \subseteq \bigcup \{\bigcap A. A \in \text{FinPow}(B)\}$ by blast

then have $S \subseteq (\bigcap A. A \in \text{FinPow}(B))$ by auto

then have $S \subseteq T$ using assms auto

done

A subbase defines a unique topology.

definition same_subbase_same_top:
assumes $B \text{ is a subbase for } T$ and $B \text{ is a subbase for } S$

shows $T = S$

using IsAsubBaseFor_def assms same_base_same_top auto

done

64 Properties in Topology

definition Topology_ZF_properties imports Topology_ZF_examples Topology_ZF_examples_1

begin

This theory deals with topological properties which make use of cardinals.
64.1 Properties of compactness

It is already defined what is a compact topological space, but the is a generalization which may be useful sometimes.

**definition**

\[\text{IsCompactOfCard (~is compact of cardinal~, in_90)}\]

where

\[K \text{~is compact of cardinal~} \subseteq \bigcup T \wedge \]

\[(\forall M \in \text{Pow}(T). K \subseteq \bigcup M \rightarrow \exists N \in \text{Pow}(M). K \subseteq \bigcup N \wedge N \mathbin{\prec} \text{nat})]\]

The usual compact property is the one defined over the cardinal of the natural numbers.

**lemma** Compact_is_card_nat:

shows \(K \text{~is compact in~} T \iff (K \text{~is compact of cardinal~} \text{nat~} \text{in~} T)\)

**proof**

{assume \(K \text{~is compact in~} T\)
then have \(\text{sub:} K \subseteq \bigcup T\) and \(\text{reg:} (\forall M \in \text{Pow}(T). K \subseteq \bigcup M \rightarrow \exists N \in \text{FinPow}(M). K \subseteq \bigcup N)\)
using IsCompact_def by auto
{fix \(M\)
assume \(M \in \text{Pow}(T). K \subseteq \bigcup M\)
with \(\text{reg}\) obtain \(N\) where \(N \in \text{FinPow}(M)\) \(K \subseteq \bigcup N\) by blast
then have \(\text{Finite}(N)\) using FinPow_def by auto
then obtain \(n\) where \(A(1)\) have \(n \approx \text{nat}\) using n_lesspoll_nat by auto
with \(\text{A(2)}\) have \(n \approx \text{nat}\) using lesspoll_def eq_lepoll_trans by auto
moreover
{assume \(N \approx \text{nat}\)
then have \(\approx N\) by auto
with \(\text{A(2)}\) have \(\approx n\) using eqpoll_sym by auto
then have \(n \approx \text{nat}\) using eqpoll_trans by auto
with \(\langle n, \text{nat}\rangle\) have \(\text{False}\) using lesspoll_def by auto
}
then have \(~(N \approx \text{nat})\) by auto
with \(\text{calculation:}\ K \subseteq \bigcup N \wedge N \in \text{FinPow}(M)\) have \(N \approx \text{nat}\) \(K \subseteq \bigcup N \wedge N \approx \text{nat}\) by auto
using lesspoll_def
FinPow_def by auto
hence \((\exists N \in \text{Pow}(M). K \subseteq \bigcup N \wedge N \approx \text{nat})\) by auto
}
with \(\text{sub}\) show \(K \text{~is compact of cardinal~} \text{nat~} \text{in~} T\) using IsCompactOfCard_def
Card_nat by auto
}
{assume \(K \text{~is compact of cardinal~} \text{nat~} \text{in~} T\)
then have \(\text{sub:} K \subseteq \bigcup T\) and \(\text{reg:} (\forall M \in \text{Pow}(T). K \subseteq \bigcup M \rightarrow \exists N \in \text{Pow}(M). K \subseteq \bigcup N \wedge N \approx \text{nat})\)
using IsCompactOfCard_def by auto
}
Another property of this kind widely used is the Lindeloef property; it is the one on the successor of the natural numbers.

definition
  IsLindeloef (\_\{is lindeloef in\}_ 90) where
  K {is lindeloef in} T \equiv K{is compact of cardinal}csucc(nat){in}T

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.

definition
  UnionCompact:
  assumes K{is compact of cardinal}Q{in}T K1{is compact of cardinal}Q{in}T
  InfCard(Q)
  shows (K \cup K1){is compact of cardinal}Q{in}T unfolding IsCompactOfCard_def by auto
proof(safe)
  from assms(1) show Card(Q) unfolding IsCompactOfCard_def by auto
  fix x assume x \in K then show x \in T unfolding IsCompactOfCard_def by auto
qed
next
  fix x assume x ∈ K 1 then show x ∈ ∪ T using assms(2) unfolding IsCompactOfCard_def
  by blast
next
  fix M assume M ⊆ T K u K 1 ⊆ M
  then have K ⊆ ∪ N ∧ N ⊆ Q ∃ N ∈ Pow(M). K 1 ⊆ ∪ N ∧ N ⊆ Q using assms unfolding IsCompactOfCard_def
  by auto
  then obtain NK NK 1 where NK ∈ Pow(M) NK 1 ∈ Pow(M) K ⊆ ∪ NK ⊆ ∪ NK 1 ⊆ Q by auto
  then have NK ∪ NK 1 ⊆ Q ∪ K 1 ⊆ ∪ (NK ∪ NK 1) using assms(3) less_less_imp_un_less by auto
  then show ∃ N ∈ Pow(M). K ∪ K 1 ⊆ ∪ N ∧ 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.

theorem compact_coarser:
  assumes T 1 ⊆ T and ∪ T 1 = ∪ T and (K) {is compact of cardinal} Q {in} T
  shows (K) {is compact of cardinal} Q 1 {in} T
proof-
  { fix M
    assume AS: M ∈ Pow(T 1) K ⊆ M
    then have M ∈ Pow(T) K ⊆ M using assms(1) by auto
    then have ∃ N ∈ Pow(M). K ⊆ ∪ N ∧ N ⊆ Q using assms(3) unfolding IsCompactOfCard_def
    by auto
  } then show thesis using assms(2,3) 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 ⊲ Q 1 and (K) {is compact of cardinal} Q {in} T and Card(Q 1)
  shows (K) {is compact of cardinal} Q 1 {in} T
proof-
  { fix M
    assume AS: M ∈ Pow(T) K ⊆ M
    then have ∃ N ∈ Pow(M). K ⊆ ∪ N ∧ N ⊆ Q using assms(2) unfolding IsCompactOfCard_def
    by auto
    then have ∃ N ∈ Pow(M). K ⊆ ∪ N ∧ N ⊆ Q 1 using assms(1) lesspoll_trans2 unfolding IsCompactOfCard_def by auto
  } then show thesis using assms(2,3) unfolding IsCompactOfCard_def by auto

804
A closed subspace of a compact space of any cardinality, is also compact of the same cardinality.

**theorem compact_closed:**
- assumes $K \{\text{is compact of cardinal}\} Q \{\text{in} T\}$
- and $R \{\text{is closed in} T\}$
- shows $(K \cap R) \{\text{is compact of cardinal}\} Q \{\text{in} T\}$

**proof**
- \{ fix $M$
  - assume $AS : M \in \text{Pow}(T) (K \cap R) \subseteq \bigcup M$
  - 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 $\bigcup T-R \in T$ have $K \subseteq \bigcup (M \cup (\bigcup T-R))$ and $M \cup (\bigcup T-R) \in \text{Pow}(T)$
  - proof (safe)
    - { fix $x$
      - assume $x \in M$
      - with $AS(1)$ show $x \in T$ by auto }
    - { fix $x$
      - assume $x \in K$
      - have $x \in R \vee x \not\in R$ by auto
        - with $\langle x \in K \rangle$ have $x \in K \cap R \vee x \in K-R$ by auto
        - with $AS(2) \langle K-R \subseteq (\bigcup T-R) \rangle$ have $x \in \bigcup (M \cup (\bigcup T-R))$ by auto
        - then show $x \in \bigcup (M \cup (\bigcup T-R))$ by auto }
  - qed
  - with assms(1) have $\exists N \in \text{Pow}(\bigcup (\bigcup T-R)). K \subseteq \bigcup N \land N \prec Q$ unfolding IsCompactOfCard_def by auto
  - then obtain $N$ where $\langle \text{cub} : N \in \text{Pow}(\bigcup (\bigcup T-R)) \rangle K \subseteq \bigcup N \prec Q$ by auto
  - have $N-(\bigcup T-R) \in \text{Pow}(N) K \cap R \subseteq \bigcup (N-(\bigcup T-R))$ and $N-(\bigcup T-R) \prec Q$
  - proof (safe)
    - { fix $x$
      - assume $x \in N \not\in M$
      - then show $x = \bigcup T-R$ using cub(1) by auto }
    - { fix $x$
      - assume $x \in K \not\in R$
      - then have $x \not\in \bigcup T-R \vee x \in K-R$ by auto
        - then show $x \in (N-(\bigcup T-R))$ 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 $\exists N \in \text{Pow}(M). K \cap R \subseteq \bigcup N \land N \prec Q$ by auto

then have $\forall M \in \text{Pow}(T). (K \cap R \subseteq \bigcup M \rightarrow (\exists N \in \text{Pow}(M). K \cap R \subseteq \bigcup N \land N \prec Q))$ by auto

then show thesis using IsCompactOfCard_def assms(1) by auto

qed

64.2 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

806
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* \( AS: \text{Closure}(A,T) = \bigcup T \)

  - *Fix* \( U \)
    - *Assume* \( U \text{open}: U \in T \text{ and } U \neq 0 \)
      - *Then* \( U \cap \bigcup T \neq 0 \) *by auto*
      - *With* \( AS \) *have* \( U \cap \text{Closure}(A,T) \neq 0 \) *by auto*
      - *With asms* \( U \text{open} \) *have* \( U \cap A \neq 0 \) *using topology0.cl_inter_neigh topology0_def* *by blast*

  - *Then show* \( \forall U \in T. U \neq 0 \rightarrow A \cap U \neq 0 \) *by auto*

- *Next* *Assume* \( AS: \forall U \in T. U \neq 0 \rightarrow A \cap U \neq 0 \)

  - *Fix* \( x \)
    - *Assume* \( A: x \in \bigcup T \)
      - *Then* \( \forall U \in T. x \in U \rightarrow U \cap A \neq 0 \) *using AS by auto*
      - *With asms* \( A \) *have* \( x \in \text{Closure}(A,T) \) *using topology0.inter_neigh_cl topology0_def* *by auto*

  - *Then* \( \bigcup T \subseteq \text{Closure}(A,T) \) *by auto*
    - *With asms* *show* \( \text{Closure}(A,T) = \bigcup T \) *using topology0.Top_3_L11(1) topology0_def* *by blast*

**64.3 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 \( csucc(Q) \) is separable of
The axiom of \( Q \) choice.

In the article [4] there is a proof of this statement for \( Q = N \), with more equivalences.

If a topology is of second type of cardinal \( csucc(Q) \), 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_choice_imp_second_imp_separable:**

- Assumes \( T \) is of second type of cardinal \( csucc(Q) \)
- and the axiom of \( Q \) choice holds for subsets \( \bigcup T \)
- and \( T \) is a topology
- Shows \( T \) is separable of cardinal \( csucc(Q) \)

**Proof**

- From assms(1) have \( \exists B. (B \text{ is a base for } T) \land (B \prec csucc(Q)) \)
- Using IsSecondOfCard_def by auto
- Then obtain \( B \) where base:(\( B \text{ is a base for } T \)) \land (B \prec csucc(Q))
- Let \( N = \lambda b \in B. b \)
- Let \( B = B - \{0\} \)
- Have \( B - \{0\} \subseteq B \) by auto
- With base have prec: \( B - \{0\} \prec csucc(Q) \) using subset_imp_lepoll lesspoll_trans1 by blast
- From base have \( \forall t \in B. Nt \subseteq \bigcup T \) using baseOpen by auto
- With baseOpen have \( \exists f. f:Pi(B, \lambda t. Nt) \land (\forall t \in B. ft \in Nt) \) using Card_less_csucc_eq_le car by auto
- Moreover with \( \forall U \in T. U \neq 0 \) have \( \{fb. b \in B \} \subseteq U \) using Top_1_2_L1 base by auto
- With \( f2 \) have \( fb \in U \) by auto
- With \( A1 \) have \( \{fb. b \in B \} \cap U \neq 0 \) by auto
- Then have \( \forall U \in T. U \neq 0 \) \rightarrow \( \{fb. b \in B \} \cap U \neq 0 \) by auto
- Have \( \{fb. b \in B \} \subseteq \bigcup T \) 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 : B → range(f) using f range_of_fun by auto
then have f ∈ surj(B, range(f)) using fun_is_surj by auto
then have des1: range(f) ≤ B using surj_fun_inv_2[of fRange(f)Q] prec
Card_less_csuucc_eq_le car
then have f ∈ surj(Q, range(f)) using fun_is_surj by auto
ultimately show thesis using IsSeparableOfCard_def by auto
qed

The next theorem resolves that the axiom of Q choice for subsets of ∪ T
is necessary for second type spaces to be separable of the same cardinal
csuucc(Q).

theorem second_imp_separable_imp_Q_choice:
  assumes ∀ T. (T is a topology) ∧ (T is of second type of cardinal)csuucc(Q))
  → (T is separable of cardinal)csuucc(Q))
  and Card(Q)
  shows {the axiom of} Q {choice holds}
proof-
  { fix N M
    assume AS:M ≤ Q ∧ (∀ t ∈ M. Nt ≠ 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))×{i}. i ∈ range(h)}
    { fix j
      assume AS2:j ∈ range(h)
      from bij have converse(h):range(h) → M using bij_def inj_def by auto
      with AS2 have converse(h)j ∈ M by simp
      with AS have N(converse(h)j) ≠ 0 by auto
      then have (N(converse(h)j))×{j} ≠ 0 by auto
    }
    then have noEmpty:0 ∉ T by auto
    moreover
    { fix A B
      assume AS2:A∈TB∈TA∩B ≠ 0
      then obtain j t where A_def:A=N(converse(h)j)×{j} and B_def:B=N(converse(h)t)×{t}
      and Range:j ∈ range(h) t ∈ range(h) by auto
      from AS2(3) obtain x where x∈A∩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 \(f = \{(i, (N(\text{converse}(h)i)) \times \{i\}). i \in \text{range}(h)\}\)
have f: \(\text{range}(h) \rightarrow T\) using functionI[of f] Pi_def by auto
then have \(f \in \text{surj}(\text{range}(h), T)\) 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 frange(h)TQ] assms(2)
Card_is_Ord lepoll_trans
subset_imp_lepoll
by auto
then obtain D where sub:D \(\in\) Pow(\(\bigcup\) \(\tau\)) and clos: Closure(D, \(\tau\)) = \(\bigcup\) \(\tau\)
and cardd:D \(\prec\) csucc(Q)
using IsSeparableOfCard_def by auto
then have D \(\approx\) Q using Card_less_csucc_eq_le assms(2)
by auto
then obtain r where r:r \(\in\) inj(D, Q)
using lepoll_def by auto
then have bij2: converse(r):bij(range(r), D)
using inj_bij_range bij_converse_bij
by auto
then have surj2: converse(r):surj(range(r), D)
using bij_def by auto
let \(R = \lambda i \in \text{range}(h). \{j \in \text{range}(r). \text{converse}(r)j \in ((N(\text{converse}(h)i)) \times \{i\})\}\}
{ fix i
assume AS:i \(\in\) range(h)
then have T: (N(\text{converse}(h)i)) \times \{i\} \(\in\) T by auto
then have P: (N(\text{converse}(h)i)) \times \{i\} \(\in\) T using base unfolding IsAbaseFor_def
by blast
with top sub clos have \(\forall U \in \tau. U \neq 0 \rightarrow D \cap U \neq 0\) using dense_int_open
by auto
with P have (N(\text{converse}(h)i)) \times \{i\} \(\neq\) 0 \(\rightarrow\) D \(\cap\) (N(\text{converse}(h)i)) \times \{i\} \(\neq\) 0
by auto
with T noEmpty have D \(\cap\) (N(\text{converse}(h)i)) \times \{i\} \(\neq\) 0 by auto
then obtain x where x:D and px:x \(\in\) (N(\text{converse}(h)i)) \times \{i\} by auto
with surj2 obtain j where j \(\in\) range(r) and converse(r)j = x unfolding surj_def by blast
with px have j \(\in\) \(\{j \in \text{range}(r). \text{converse}(r)j \in ((N(\text{converse}(h)i)) \times \{i\})\}\}
by auto
then have Ri \(\neq\) 0 using beta_if[of range(h) _ i] AS by auto

810
then have \( \forall i \in \text{range}(h). R_i \neq 0 \) by auto

\{ fix i j \
assume \( i \in \text{range}(h) \) and \( j \in R_i \) from \( j \) \( i \) have \( \text{converse}(r) j \in ((N(\text{converse}(h)i)) \times \{i\}) \) using beta_if by auto \}

then have \( \forall i \in \text{range}(h). \forall j \in R_i. \text{converse}(r) j \in ((N(\text{converse}(h)i)) \times \{i\}) \) by auto

let \( E = \{ \langle m, \text{fst}(\text{converse}(r)(\mu j. j \in R(hm))) \rangle. m \in M \} \) have \( ff : \text{function}(E) \) unfolding function_def by auto

moreover \{ fix \( m \) assume \( M : m \in M \) with inj have \( \text{hm} : \text{hm} \in \text{range}(h) \) using apply_rangeI inj_def by auto \{ fix \( j \) assume \( j \in R(hm) \) using beta_if by auto from \( r \) have \( r : \text{surj}(D, \text{range}(r)) \) using fun_is_surj inj_def by auto with \( < j \in \text{range}(r) \) obtain \( d \) where \( d \in D \) and \( rd = j \) using surj_def by auto \}

then have \( j \in Q \) using \( r \) inj_def by auto \}

then have \( \text{subcar} : R(hm) \subseteq Q \) by blast from \( \text{nonE} \) \( \text{hm} \) obtain \( \text{ee} \) where \( P : \text{ee} \in R(hm) \) by blast with \( \text{subcar} \) have \( \text{ee} \in Q \) by auto then have \( \text{Ord}(\text{ee}) \) using assms(2) Card_is_Ord Ord_in_Ord by auto with \( P \) have \( (\mu j. j \in R(hm)) \in R(hm) \) using LeastI[where \( i = \text{ee} \) and \( P = \lambda j. j \in R(hm) \)] by auto with \( \text{pp} \) \( \text{hm} \) have \( \text{converse}(r)(\mu j. j \in R(hm)) \in ((N(\text{converse}(h)(hm))) \times \{\text{hm}\}) \) by auto then have \( \text{converse}(r)(\mu j. j \in R(hm)) \in ((N(m)) \times \{\text{hm}\}) \) using left_inverse[OF inj M] by simp then have \( \text{fst}(\text{converse}(r)(\mu j. j \in R(hm))) \in (N(m)) \) by auto ultimately have \( \text{thesis1} : \forall m \in M. \text{Em} \in (N(m)) \) using function_apply_equality by auto \{ fix \( e \) assume \( e \in E \) then obtain \( m \) where \( m \in M \) and \( e = (m, \text{Em}) \) using function_apply_equality ff by auto with \( \text{thesis1} \) have \( e \in \Sigma(M, \lambda t. Nt) \) by auto \}
then have $E \in \text{Pow}(\Sigma(M, \lambda t. Nt))$ by auto
with $ff$ have $E \in \Pi(M, \lambda m. Nm)$ using $\Pi$ _iff_ by auto
then have $(\exists f. f : \Pi(M, \lambda t. Nt) \land (\forall t \in M. ft \in Nt))$ 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 $\text{Card}(Q)$
shows $(\forall T. (T\text{ is a topology} \land (T\text{ is of second type of cardinal} csucc(Q)))
\rightarrow (T\text{ is separable of cardinal} csucc(Q)))$
$\leftrightarrow \{\text{the axiom of } Q \text{ choice holds}\}$
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 $B \subseteq Q \text{ B is a base for} T$
shows $\exists N. \{N_i. i \in Q\} \text{ is a base for} T$
proof-
from assms obtain $f$ where $f\_def : f \in \text{inj}(B, Q)$ unfolding lepoll_def by auto
let $ff = \{ (b, fb) . b \in B \}$ have 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: $ff : B \rightarrow \text{range}(ff)$ using function_imp_Pi[of $ff$] by auto
then have injj: $ff \in \text{inj}(B, \text{range}(ff))$ unfolding inj_def
proof
{fix $w x$
  assume AS: $w \in B . x \in B \{ (b, f b) . b \in B \} \ w = \{ (b, f b) . b \in B \} \ x$
  then have $fw = fx$ using apply_equality[OF _ fun] by auto
  then have $w = x$ using $f\_def$ inj_def AS(1,2) by auto
}
then show $\forall w \in B . \forall x \in B . \{ (b, f b) . b \in B \} \ w = \{ (b, f b) . b \in B \}$ $w = x$ by auto
qed
then have bij: $ff \in \text{bij}(B, \text{range}(ff))$ using inj_bij_range by auto
from fun have range$(ff) = \{ fb . b \in B \}$ by auto
with $f\_def$ have ran: $\text{range}(ff) \subseteq Q$ using inj_def by auto
let $N = \{ i, (\text{if } i \in \text{range}(ff) \text{ then converse}(ff_i) \text{ else } 0) . i \in Q \}$
have FN: function(N) unfolding function_def by auto
have B ⊆ \{Ni. i ∈ Q\}
proof
  fix t
  assume a: t ∈ B
  from bij have rr: ff: B \to \text{range}(ff) unfolding bij_def inj_def by auto
  have ig: ff \in \text{range}(ff) using apply_type[OF _ rr] by auto
  have r: ff \in \text{range}(ff) using apply_type[OF _ a] f_def unfolding inj_def
  by auto
  with r have N(ff) = \text{converse}(ff)(ff) using function_apply_equality[OF _ FN] by auto
  then have N(ff) = t using left_inverse[OF injj a] by auto
  then have \exists i ∈ Q. t = Ni using t(1) by auto
  then show t ∈ \{Ni. i ∈ Q\} by simp
qed
moreover
have \forall r ∈ \{Ni. i ∈ Q\} - B. r = 0
proof
  fix r
  assume r ∈ \{Ni. i ∈ Q\} - B
  then obtain j where R: j ∈ Q \and r = N(j) by auto
  { assume AS: j \in \text{range}(ff)
    with R(1) have N(j) = \text{converse}(ff)(j) using function_apply_equality[OF _ FN] by auto
    then have N(j) \in \text{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 \notin \text{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 \lor \{Ni. i ∈ Q\} = B \cup \{0\} by blast
moreover
have (B \cup \{0\}) - \{0\} = B - \{0\} by blast
then have (B \cup \{0\}) - \{0\} \{is a base for\} T using base_no_0[of BT assms(2)] by auto
ultimately
have \{Ni. i ∈ Q\} \{is a base for\} T using assms(2) by auto
then show thesis by auto
qed
64.4 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 ≲ Q ∧ (∀ t ∈ M. N t ≠ 0 ∧ N t ⊆ Pow(Q)))
  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 = λ U ∈ M. {i ∈ Q. N(i) ⊆ U}
    have inj: α ∈ inj(M, Pow(Q)) unfolding inj_def
    proof
      show (λ U ∈ M. {i ∈ Q. N(i) ⊆ U}) ∈ M → Pow(Q) using lam_type[of M](λ U. {i ∈ Q. N(i) ⊆ U}) by auto
      { fix w x
        assume AS: w ∈ M x ∈ M {i ∈ Q. N(i) ⊆ U} = {i ∈ Q. N(i) ⊆ x}
        from AS (1,2) A (2) have w ∈ T x ∈ T by auto
        then have w = Interior(w, T) x = Interior(x, T) using assms (3) topology0.Top_2_L3[of T]
        topology0_def[of T] by auto
        then have UN: w = (∪ {B ∈ {N(i). i ∈ Q}. B ⊆ w}) x = (∪ {B ∈ {N(i). i ∈ Q}. B ⊆ x})
        using interior_set_base_topology assms (3) base by auto
        { fix b
          assume b ∈ w
          then have b ∈ (∪ {B ∈ {N(i). i ∈ Q}. B ⊆ w}) using UN (1) by auto
          then obtain S where S: S ⊆ {N(i). i ∈ Q} b ∈ S S ⊆ w by blast
          then obtain j where j: j ∈ QS = N(j) by auto
          then have j ∈ i ∈ Q. N(i) ⊆ w} using S (3) by auto
          then have N(j) ⊆ x ∈ N(j) j ∈ Q using S (2) A (3) j by auto
          then have b ∈ (∪ {B ∈ {N(i). i ∈ Q}. B ⊆ x}) by auto
          then have b ∈ x using UN (2) by auto
        }
      }
      moreover
      
      814
fix b
assume b∈x
then have b∈⋃{B∈{N(i). i∈Q}. B⊆x} by UN(2) by auto
then obtain S where S:S∈{N(i). i∈Q} b∈S S⊆x by blast
then obtain j where j:j∈Q=N(j) by auto
then have j∈{i ∈ Q. N(i) ⊆ x} using S(3) by auto
then have j∈{i ∈ Q. N(i) ⊆ w} using S(2) j(2) by auto
then have b∈⋃{B∈{N(i). i∈Q}. B⊆w} by auto
then have b∈w using UN(2) by auto
ultimately have w=x by auto
}
then show ∀w∈M. ∀x∈M. (λU∈M. {i ∈ Q. N i ⊆ U}) w = (λU∈M. {i ∈ Q. N(i) ⊆ U}) x −→ w = x by auto
let X=λi∈Q. {α∈V∈M. N(i)⊆V}
let M={i ∈ Q. Xi=0}
have subMQ:M ⊆ Q by auto
then have ddd:M ≲ Q using subset_imp_lepoll by auto
then have M ≲ Q ∀i∈M. Xi∈Pow(Q) by auto
then have M ≲ Q ∀i∈M. Xi≲Pow(Q) using subset_imp_lepoll by auto
then have (∃f. f:Pi(M,λt. Xt)) ∧ (∀t∈M. ft∈Xt)) using reg[of MX] by auto
then obtain f where f:Pi(M,λt. Xt)!!,t∈M ⇒ ft∈Xt) by auto
{
fix m
assume S:m∈M
from f(2) S obtain YY where YY:(YY∈M) (fm=αYY) by auto
then have Y:(YY∈M)∧(fm=αYY) by auto
moreover
{
fix U
assume U∈M\(fm=αU)\)
then have U=YY using inj inj_def YY by auto
}
then have r:∀x. x∈M\(fm=αx) ⇒ x=YY by blast
have ∃!YY. YY∈M ∧ fm=αYY using ex1I[of %Y. Y∈M ∧ fm=αY,OF Y r] by auto
}
then have ex1YY:∀m∈M. ∃!YY. YY∈M ∧ fm=αYY by auto
let YYm={(m,(THE YY. YY∈M ∧ fm=αYY))). m∈M} have aux:∀m. m∈M ⇒ YYm=THE YY. YY∈M ∧ fm=αYY) unfolding apply_def by auto
have ree:∀m∈M. (YYm)∈M ∧ fm=α(YYm) proof
fix m
assume $C : m \in M$
then have $\exists ! YY. YY \in M \land fm = \alpha YY$ using ex1YY by auto
then have $(THE YY. YY \in M \land fm = \alpha YY) \in M \land fm = \alpha YY$ using theI[of $YY. YY \in M \land fm = \alpha YY$] by blast
then show $(YYmm) \in M \land fm = \alpha (YYmm)$ apply (simp only: aux[OF C]) done

have $tt : \forall m. m \in M \implies N(m) \subseteq YYmm$
proof-
  fix $m$
  assume $D : m \in M$
  then have $QQ : m \in Q$ by auto
from $D$ have $t : (YYmm) \in M \land fm = \alpha (YYmm)$ using ree by blast
  then have $fm = \alpha (YYmm)$ by blast
then have $(\alpha (YYmm))(\lambda i \in Q. \{ \alpha U. U \in \{ V \in M. N(i) \subseteq V \}))m$ using f(2)[OF D] by auto
then have $(\alpha (YYmm)) \subseteq (\lambda i \in Q. \{ \alpha U. U \in \{ V \in M. N(m) \subseteq V \}))$ using $Q$
  by auto
then have $r : U \in N(m)$ using $t$ by auto
then have $YYmm = U$ using inj_apply_equality[OF inj] by blast
then show $N(m) \subseteq YYmm$ using $r$ by auto
qed

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 $r : t \in M \subseteq N(t)$ by auto
then have $s \in YYmt$ using $tt$[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 $AT : x \in \bigcup T$
with $A$ obtain $U$ where $BB : U \in MU \in T x \in U$ by auto
then obtain $j$ where $BC : j \in Q \subseteq U x \in N(j)$ using point_open_base_neigh[OF base,of $Ux$] by auto
then have $Xj \neq 0$ using $BB(1)$ by auto
then have $j \in M$ using $BC(1)$ by auto
then have $x \in (\bigcup m \in M. N(m))$ using $BC(3)$ 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 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

816
moreover have f:function(YYm) unfolding function_def by auto
moreover have d:domain(YYm)=M by auto
moreover have r:range(YYm)=YYmM by auto
ultimately have fun:YYm:M \rightarrow YYmM using function_imp_Pi[of YYm] by auto
have YYm \subseteq M by auto
with ddd have Rw:YYmM \subseteq Q using lepoll_trans by blast

In the following proof, we have chosen an infinite cardinal to be able to apply the equation Q × Q ≈ Q. For finite cardinals; both, the assumption and the axiom of choice, are always true.

theorem second_imp_compact_imp_Q_choice_PowQ:
  assumes \( \forall T. (T\text{ is a topology} \land (T\text{ is of second type of cardinal})\text{csucc}(Q)) \rightarrow ((\bigcup T)\text{is compact of cardinal})\text{csucc}(Q)\{in\} T\)
  and InfCard(Q)
  shows \{the axiom of\} Q \{choice holds for subsets\} (Pow(Q))
proof-
  \{ fix N M
    assume AS:M \subseteq Q \land (\forall t\in M. Nt\neq 0 \land Nt\subseteq Pow(Q)) \}

817
then obtain $h$ where $h \in \text{inj}(M, Q)$ using lepoll_def by auto

have discTop: Pow($Q \times M$) {is a topology} using Pow_is_top by auto

{ fix $A$
  assume $AS : A \in (\bigcup\{\{i\}. \ i \in A\}$ by auto
  with $AS$ have $\exists T \in \text{Pow}(\bigcup\{\{i\}. \ i \in Q \times M\}). A = \bigcup T$ by auto
  then have $A \in (\bigcup\{U. \ U \in \text{Pow}(\bigcup\{\{i\}. \ i \in Q \times M\})\}$ by auto
}

moreover

{ fix $A$
  assume $AS : A \in (\bigcup\{U. \ U \in \text{Pow}(\bigcup\{\{i\}. \ i \in Q \times M\})\}$
  then have $A \in \text{Pow}(Q \times M)$ by auto
}

ultimately

have $\text{base}:\{\{x\}. \ x \in Q \times M\}$ {is a base for} Pow($Q \times M$) unfolding IsAbaseFor_def by blast

let $f = \{\langle i, \{i\}\rangle. \ i \in Q \times M\}$

have $fff : f \in \text{Pow}(Q \times M)$ using Pi_def function_def by auto

then have $f \in \text{inj}(Q \times M, \bigcup\{\{i\}. \ i \in Q \times M\})$ unfolding inj_def using apply_equality by auto

then have $f \in \text{bij}(Q \times M, \bigcup\{\{i\}. \ i \in Q \times M\})$ unfolding bij_def surj_def using $fff$ apply_equality $fff$ by auto

then have $Q \times M \approx \bigcup\{\{i\}. \ i \in Q \times M\}$ using eqpoll_def by auto

then have $\bigcup\{\{i\}. \ i \in Q \times M\} \approx Q \times M$ using eqpoll_sym by auto

then have $\bigcup\{\{i\}. \ i \in Q \times M\} \lesseq Q \times Q$ using $\text{AS prod_lepoll_mono[of QMQ]}$ lepoll_refl[of $Q$]

lepoll_trans by blast

then have $\bigcup\{\{i\}. \ i \in Q \times M\} \lesseq Q$ using InfCard_square_eqpoll assms(2) lepoll_eq_trans by auto

then have $\bigcup\{\{i\}. \ i \in Q \times M\} \approx \text{csucc}(Q)$ using Card_less_csucc_eq_le assms(2)

InfCard_is_Card by auto

then have Pow($Q \times M$) {is of second type of cardinal} csucc($Q$) using IsSecondOfCard_def base by auto

then have comp:($Q \times M$) {is compact of cardinal} csucc($Q$) {in} Pow($Q \times M$) using discTop assms(1) by auto

{ fix $W$
  assume $W \in \text{Pow}(Q \times M)$
  then have $T:W$ {is closed in} Pow($Q \times M$) and $(Q \times M) \cap W = W$ using IsClosed_def by auto
  with compact_closed[OF comp T] have $(W$ {is compact of cardinal} csucc($Q$) {in} Pow($Q \times M$) by auto
}

then have subCompact: $\forall W \in \text{Pow}(Q \times M). \ (W$ {is compact of cardinal} csucc($Q$) {in} Pow($Q \times M$))

818
by auto
let cub=\{((U)\times\{t\}. U\in\mathbb{N}t). t\in\mathbb{M}\}
from AS have (\bigcup \text{cub})\in\text{Pow}(\mathbb{Q}\times\mathbb{M}) by auto
with subCompact have Ncomp:((\bigcup \text{cub}) \text{ is compact of cardinal})_{\text{csucc}(\mathbb{Q})\in\text{Pow}(\mathbb{Q}\times\mathbb{M})} by auto
have cond:((\bigcup \text{cub})\in\text{Pow}(\mathbb{Q}\times\mathbb{M})\land \bigcup \text{cub}\subseteq \bigcup \text{cub}\text{ using AS by auto}
have \exists S\in\text{Pow}(\text{cub}). (\bigcup \text{cub}) \subseteq \bigcup S \land S \prec \text{csucc}(\mathbb{Q})
proof-
\{ have ((\bigcup \text{cub}) \text{ is compact of cardinal})_{\text{csucc}(\mathbb{Q})\in\text{Pow}(\mathbb{Q}\times\mathbb{M})} using Ncomp by auto
then have \forall M\in\text{Pow}(\mathbb{Q}\times\mathbb{M})\ldots by auto
unfolding IsCompactOfCard_def by auto
with cond have \exists S\in\text{Pow}(\text{cub}). (\bigcup \text{cub}) \subseteq \bigcup S \land S \prec \text{csucc}(\mathbb{Q}) by auto \} then show thesis by auto
qed
then have ttt:\exists S\in\text{Pow}(\text{cub}). (\bigcup \text{cub}) \subseteq \bigcup S \land S \prec \mathbb{Q} using Card_less_csucc_eq_le
assms(2) InfCard_is_Card by auto
then obtain S where S_def:S\in\text{Pow}(\text{cub})(\bigcup \text{cub}) \subseteq \bigcup S \subseteq \mathbb{Q} by auto
\{ fix t
assume AA:t\in\mathbb{M}nt\neq\{0\}
from AA(1) AS have Nt\neq0 by auto
with AA(2) obtain U where G:U\in\mathbb{N}t and notEm:U\neq0 by blast
then have U\times\{t\}\subseteq \text{cub}\text{ using AA by auto}
then have U\times\{t\}\subseteq \text{cub}\text{ by auto}
with G notEm AA have \exists s. (s,t)\in\text{cub}\text{ by auto}
\}
then have \forall t\in\mathbb{M}. (Nt\neq\{0\}) \rightarrow (\exists s. (s,t)\in\text{cub}) by auto
then have A:\forall t\in\mathbb{M}. (Nt\neq\{0\}) \rightarrow (\exists s. (s,t)\in\text{S}) using S_def(2) by blast
from S_def(1) have B:\forall e\in\mathbb{S}. \exists t\in\mathbb{M}. \exists U\in\mathbb{N}t. f=U\times\{t\} by blast
from A B have \forall t\in\mathbb{M}. (Nt\neq\{0\}) \rightarrow (\exists U\in\mathbb{N}t. U\times\{t\}\in\mathbb{S}) by blast
then have noEmp:\forall t\in\mathbb{M}. (Nt\neq\{0\}) \rightarrow (\exists U\in\mathbb{N}t. U\times\{t\}\neq\{0\}) by auto
from S_def(3) obtain r where r:r:\text{inj}(S,\mathbb{Q}) using lepoll_def by auto
then have bij2:converse(r):bij(range(r),\mathbb{S}) using inj_bij_range bij_converse_bij
by auto
then have surj2:converse(r):surj(range(r),\mathbb{S}) using bij_def by auto
let R=\lambda t:\mathbb{M}. \{j\in\text{range}(r). converse(r)j\in(\{U\times\{t\}. U\in\mathbb{N}t\})\}
\{ fix t
assume AA:t\in\mathbb{M}nt\neq\{0\}
then have (S\cap(\{U\times\{t\}. U\in\mathbb{N}t\})\neq\{0\}) using noEmp by auto
then obtain s where ss:s\in\mathbb{S}ss\in(\{U\times\{t\}. U\in\mathbb{N}t\}) by blast
then obtain j where converse(r)j=s j\in\text{range}(r) using surj2 unfolding surj_def by auto
then have j\in(\{j\in\text{range}(r). converse(r)j\in(\{U\times\{t\}. U\in\mathbb{N}t\})\}) using ss
819
by auto
then have Rt\neq 0 using beta_if AA by auto

\}
then have nonE:\forall t \in M. Nt \neq \{0\} \rightarrow Rt \neq 0 by auto
\{
fix t j
assume t \in M j \in Rt
then have converse(r)(j) \in \{U \times \{t\}. U \in Nt\} using beta_if by auto
\}
then have pp:\forall t \in M. \forall j \in Rt. converse(r)(j) \in \{U \times \{t\}. U \in Nt\} by auto
have reg:\forall t U V. U \times \{t\} = V \times \{t\} \rightarrow U = V
proof-
\{
fix t U V
assume AA:U \times \{t\} = V \times \{t\}
\{
fix v
assume v \in V
then have \langle v, t \rangle \in V \times \{t\} by auto
then have \langle v, t \rangle \in U \times \{t\} using AA by auto
then have v \in U by auto
\}
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 \langle u, t \rangle \in V \times \{t\} using AA 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 Nt=\{0\} then 0 else (THE U. converse(r)(\mu j. j \in Rt)=U \times \{t\})). t \in M\}
have ff:function(E) unfolding function_def by auto
moreover
\{
fix t
assume pm:t \in M
\{
assume nonEE:Nt \neq \{0\}
\{
fix j
assume j \in Rt
with pm(1) have j \in range(r) using beta_if by auto
\}
\}
\}
820
from r have r:surj(S,range(r)) using fun_is_surj inj_def by auto
with \langle j\in range(r)\rangle obtain d where d\in S and rd=j using surj_def
by auto
then have j\in Q using r inj_def by auto
with \langle j\in range(r)\rangle obtain d where d\in S and rd=j using surj_def
by auto
then have j\in Q using r inj_def by auto
with sub have ee\in Q by blast
then have \sub: R_t \subseteq Q by blast
from nonE pm nonEE obtain ee where P: ee\in R_t by blast
with sub have ee\in Q by auto
then have Ord(ee) using assms(2) Card_is_Ord Ord_in_Ord InfCard_is_Card by blast
with P have (\\mu. j. j\in R_t)\in R_t by auto
with pp pm have converse(r)(\mu. j. j\in R_t)\in U\times\{t\}. U\in N_t by auto
then obtain w where converse(r)(\mu. j. j\in R_t)=w\times\{t\} and s: w\in N_t by auto
then have (THE U. converse(r)(\mu. j. j\in R_t)=U\times\{t\})=w using reg by auto
with s have (THE U. converse(r)(\mu. j. j\in R_t)=U\times\{t\})\in N_t by auto
ultimately have thesis1: \forall t\in M. Et\in N_t using function_apply_equality by auto
fix e assume e\in E
then obtain m where m\in M and e=\langle m, E_m \rangle using function_apply_equality
ff by auto
with thesis1 have e\in Sigma(M, \lambda t. N_t) by auto
then have E\in Pow(Sigma(M, \lambda t. N_t)) by auto
with ff have E\in Pi(M, \lambda m. N_m) using Pi_iff by auto
then have (\\exists f. f: Pi(M, \lambda t. N_t) \land (\forall t\in M. ft\in N_t)) 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 (\forall T. (T\{is a topology\} \land (T\{is of second type of cardinal\} csucc(Q))))
\rightarrow ((\bigcup T)\{is compact of cardinal\} csucc(Q)\{in\} T))
\leftarrow (\{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 (\kappa, Pow(\kappa)), for \kappa an infinite
cardinal, is compact of its successor cardinal; then all topological spaces which are of second type of the successor cardinal of \( \kappa \) are also compact of that cardinal.

**Theorem Q_csuccQ_comp_eq_Q_choice.Pow:**

assumes \( \text{InfCard(Q)} \) \( (Q) \{\text{is compact of cardinal} \} \) \( \text{csucc(Q)} \{\text{in} \} \) \( \text{Pow(Q)} \)

shows \( \forall T. (T\{\text{is a topology}\} \land (T\{\text{is of second type of cardinal} \} \text{csucc(Q)})) \rightarrow ((\bigcup T)\{\text{is compact of cardinal} \} \text{csucc(Q)}\{\text{in} \} T) \)

**Proof:**

fix \( T \)

\{ assume \( \text{top:T} \{\text{is a topology}\} \) and \( \text{sec:T}\{\text{is of second type of cardinal} \} \text{csucc(Q)} \)

from \( \text{assms} \) have \( \text{Card(csucc(Q)) Card(Q)} \) using \( \text{InfCard_is_Card Card_is_Ord} \)

by \( \text{auto} \)

moreover have \( \bigcup T \subseteq \bigcup T \) by \( \text{auto} \)

moreover \{ fix \( M \)

assume \( \text{MT:M} \{\text{in} \} \text{Pow(T)} \) and \( \text{cover:}\bigcup T \subseteq M \)

from \( \text{sec obtain} \) \( B \) where \( B \{\text{is a base for} \} T \) \( B<\text{csucc(Q)} \) using \( \text{IsSecondOfCard_def} \)

by \( \text{auto} \)

with \( \langle \text{Card(Q)} \rangle \) obtain \( N \) where \( \text{base:}\{\text{Ni. i}\in Q\}\{\text{is a base for}\}T \) using \( \text{Card_less_csucc_eq_le} \)

by \( \text{auto} \)

base_to_indexed_base by \( \text{blast} \)

let \( S=\{\langle u,\{i\in Q. N i \subseteq u\}\rangle. u\in M\} \)

have \( \text{function(S)} \) unfolding \( \text{function_def} \) by \( \text{auto} \)

then have \( S:\text{inj}(M,\text{Pow(Q)}) \) unfolding \( \text{inj_def} \)

proof \{ fix \( w \) \( x \)

assume \( \text{AS:w}\in Mx\in M=\{\langle u,\{i\in Q. N i \subseteq u\}\rangle. u\in M\} \)

with \( \langle S:M\rightarrow \text{Pow(Q)} \rangle \) have \( \text{ASS:}\{i\in Q. N i \subseteq w\}=\{i\in Q. N i \subseteq x\} \) using \( \text{apply_equality} \) by \( \text{auto} \)

from \( \text{AS(1,2) MT have w}\in Tx\in T \)

then have \( w=\text{Interior}(w,T)x=\text{Interior}(x,T) \) using \( \text{top topology0.Top_2_L3[of T]} \)

by \( \text{auto} \)

then have \( \text{UN:w}=\{\bigcup \{B\in (N(i). i\in Q). B\subseteq w\}\}=\{\bigcup \{B\in N(i). i\in Q\}. B\subseteq x\} \)

by \( \text{auto} \)

using \( \text{interior_set_base_topology top base by auto} \)

\{ fix \( b \)

assume \( b\in w \)

then have \( b\in \{B\in (N(i). i\in Q). B\subseteq w\} \) using \( \text{UN(1)} \) by \( \text{auto} \)

then obtain \( S \) where \( S:S\in (N(i). i\in Q) \) \( b\in S \) \( S\subseteq w \) by \( \text{blast} \)

then obtain \( j \) where \( j:j\in QS=N(j) \) by \( \text{auto} \)

then have \( j\in \{i\in Q. N(i) \subseteq w\} \) using \( \text{S(3)} \) by \( \text{auto} \)
then have $N(j) \subseteq x \in N(j)$ using $\text{S}(2)$ by auto

then have $b \in (\bigcup \{B \subseteq N(i). i \in Q \})$ using $\text{UN}(2)$ by auto

moreover
{
    fix $b$
    assume $b \in x$
    then have $b \in (\bigcup \{B \subseteq N(i). i \in Q \})$ using $\text{UN}(2)$ by auto
}

ultimately have $w = x$ by auto

then have $\forall w \in M. \forall x \in M. \{\langle u, \{i \in Q. N(i) \subseteq u\} \rangle. u \in M\} w \rightarrow \text{w} = x$ by auto

qed

then have $\exists S \in \text{bij}(M, \text{range}(S))$ using $\text{fun_is_surj}$ unfolding $\text{bij_def}$ $\text{inj_def}$$\text{surj_def}$

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 $(\bigcup (\text{range}(S))) \{\text{is closed in}\} \text{Pow}(Q)$ using $\text{IsClosed_def}$

from this(2) have $(\bigcup (\text{range}(S))) \{\text{is compact of cardinal}\} \text{csucc}(Q)$ by auto

moreover
have $\bigcup (\text{range}(S)) \subseteq \bigcup (\text{range}(S))$ by auto
ultimately have $\exists S \in \text{Pow}(\text{range}(S)). (\bigcup (\text{range}(S)) \subseteq \bigcup S \wedge S \prec \text{csucc}(Q)$ by auto

then obtain $S$ where $S \subseteq \text{range}(S)$ using $\text{bijeq_def}$

with $S : \text{bij}(M, \text{range}(S))$ have $\text{con:converse}(S) \subseteq \text{bij}(\text{range}(S), M)$ using $\text{bij_converse_bij}$ by auto

then have $r_1 : \text{restrict}(\text{converse}(S), SS) \subseteq \text{bij}(SS, \text{converse}(S)SS)$ using $\text{restrict_bij_def SS_def(1)}$ by auto
then have $r_2 : \text{converse}(\text{restrict}(\text{converse}(S), SS)) \subseteq \text{bij}(\text{converse}(S)SS, SS)$ using $\text{bij_converse_bij}$ by auto

{ 
    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

823
with MT have $R \in T \land x \in R$ by auto
then have $\exists V \in \{ N_i. \ i \in Q \}. \ V \subseteq R \land x \in V$ using point_open_base_neigh base by force
then obtain $j$ where $j \in Q \land N_j \subseteq R$ and $x_p : x \in N_j$ by auto with $\langle R : M \rightarrow \text{Pow}(Q) \rangle < S \in \text{bij}(M, \text{range}(S))>$ have $SR \in \text{range}(S)$
$\land j \in SR$ using apply_equality
bij_def inj_def by auto
from exI[where $P = \lambda t. \ t \in \text{range}(S) \land j \in t$, OF this] have $\exists A \in \text{range}(S)$.
j$\in A$ unfolding Bex_def by auto
then have $j \in (\bigcup \text{range}(S)))$ by auto
then have $j \in \bigcup SS$ using SS_def(2) by blast
then obtain $SR$ where $SR \subseteq R$ and $x_p : x \in SR$ by auto moreover have converse(restrict(converse(S), SS))$\in \text{surj}(\text{converse}(S)SS, SS)$ using rr bij_def by auto
ultimately obtain $RR$ where converse(restrict(converse(S), SS))$RR$=$SR$ and $p : RR \in \text{surj}(\text{converse}(S)SS, SS)$ unfolding surj_def by blast
then have converse(restrict(converse(S), SS)))$\in \text{inj}(\text{converse}(S)SS, SS)$ 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 $r_1$ have restrict(converse(S), SS)$\in SS \rightarrow \text{converse}(S)SS$ unfolding bij_def inj_def by auto
then have relation(restrict(converse(S), SS))$\in \text{relation}(\text{converse}(S), SS))$ using Pi_def relation_def by auto
then have converse(converse(restrict(converse(S), SS)))$= \text{restrict}(\text{converse}(S), SS))$ using relation_converse_converse by auto
ultimately have $RR =$ restrict(converse(S), SS)$RR$=$SR$ by auto
with $< SR \in SS >$ have eq: $RR =$ converse(S)SR unfolding restrict by auto
then have converse(converse(S))$RR =$ converse(converse(S))(converse(S)SR) by auto
moreover with $< SR \in SS >$ have $SR \in \text{range}(S)$ 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
then have $SRR =$ SR using relation_converse_converse[of S] unfolding relation_def by auto
moreover have converse(S)$: \text{range}(S) \rightarrow M$ using con bij_def inj_def by auto
with $< SR \in \text{range}(S) >$ have converse(S)SR$\in M$ using apply_funtype

824
by auto
with eq have RR∈M by auto
ultimately have SR={i∈Q. Ni⊆RR} using <S:M→Pow(Q)> apply_equality

by auto
then have Nj⊆RR using <j∈SR> 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
ultimately
have (converse(S)SS)⊆M by auto
}
then have converse(S)SS∈Pow(M) by auto
moreover
with rr have converse(S)SS≈SS using eqpoll_def by auto
then have converse(S)SS≺csucc(Q) using SS_def(3) eq_lesspoll_trans
by auto
ultimately
have ∃N∈Pow(M). T⊆N ∧ N≺csucc(Q) by auto
}
then have ∀M∈Pow(T). T⊆M → (∃N∈Pow(M). T⊆N ∧ N≺csucc(Q))
by auto
ultimately have (∪T){is compact of cardinal}csucc(Q){in}T unfolding IsCompactOfCard_def
by auto
then show (T {is a topology}) ∧ (T {is of second type of cardinal}csucc(Q)) → ((∪T){is compact of cardinal}csucc(Q) {in}T)
by auto
qed

theorem Q_disc_is_second_card_csuccQ:
  assumes InfCard(Q)
  shows Pow(Q){is of second type of cardinal}csucc(Q)
proof-
  { fix A
assume AS: A ∈ Pow(Q)
have A = ⋃ {{i}. i ∈ A} by auto
with AS have ∃ T ∈ Pow({{i}. i ∈ Q}). A = ⋃ T by auto
then have A ∈ (⋃ U. U ∈ Pow({{i}. i ∈ Q})) by auto
}
moreover
{
  fix A
  assume AS: A ∈ (⋃ U. U ∈ Pow({{i}. i ∈ Q}))
  then have A ∈ Pow(Q) by auto
}
ultimately
have base: {{x}. x ∈ Q} {is a base for} Pow(Q) unfolding IsAbaseFor_def
by blast
let f = {{i, {i}}. i ∈ Q}
have f ∈ Q → {{x}. x ∈ Q} unfolding Pi_def function_def by auto
then have f ∈ inj(Q, {{x}. x ∈ Q}) unfolding inj_def using apply_equality
by auto
moreover
from < f ∈ Q → {{x}. x ∈ Q} have f ∈ surj(Q, {{x}. x ∈ Q}) unfolding surj_def
using apply_equality
by auto
ultimately have f ∈ bij(Q, {{x}. x ∈ Q}) unfolding bij_def by auto
then have Q ≈ {{x}. x ∈ Q} using eqpoll_def by auto
then have {{x}. x ∈ Q} ≲ Q using eqpoll_sym by auto
then have {{x}. x ∈ Q} ≼ 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.

theorem Q_disc_comp_csuccQ_eq_Q_choice_csuccQ:
  assumes InfCard(Q)
  shows (Q {is compact of cardinal} csucc(Q) {in} Pow(Q)) ←→ ({the axiom of} Q{choice holds for subsets}(Pow(Q)))
proof
  assume Q {is compact of cardinal} csucc(Q) {in} Pow(Q)
  with assms show (the axiom of) Q{choice holds for subsets}(Pow(Q)) using
  Q_choice_Pow_eq_secon_imp_comp Q_csuccQ_comp_eq_Q_choice_Pow
  by auto
next
  assume {the axiom of} Q{choice holds for subsets}(Pow(Q))
  with assms show Q {is compact of cardinal} csucc(Q) {in} Pow(Q)
  using Q_disc_is_second_card_csuccQ Q_choice_Pow_eq_secon_imp_comp Pow_is_top[of Q]
  by force
qed

826
65 Topology 5

theory Topology_ZF_5 imports Topology_ZF_properties Topology_ZF_examples_1
  Topology_ZF_4
begin

65.1 Some results for separation axioms

First we will give a global characterization of $T_1$-spaces; which is interesting because it involves the cardinal $\aleph_0$.

lemma (in topology0) T1_cocardinal_coarser:
  shows $(T \{\text{is } T_1\}) \iff (\text{CoFinite } (\bigcup T) \subseteq T)$
proof
  { assume AS: T \{\text{is } T_1\}
    
    \fix \text{x assume } p : x \in \bigcup T
    
    \fix \text{y assume } y \in (\bigcup T) - \{x\}
    
    with AS \text{ p obtain } U \text{ where } U \in T \ y \in U \ x \notin U \text{ using } \text{isT1_def by blast}
    then have \text{U \in T} \ y \in U \ U \subseteq (\bigcup T) - \{x\} \text{ by auto}
    then have \\exists U \in T. \ y \in U \ \land \ U \subseteq (\bigcup T) - \{x\} \text{ by auto}
  }
  then have \\forall y \in (\bigcup T) - \{x\}. \ \exists U \in T. \ y \in U \ \land \ U \subseteq (\bigcup T) - \{x\} \text{ by auto}
  then have \text{U - \{x\} }\in T \text{ using } \text{open_neigh_open by auto}
  with p have \{x\} \{\text{is closed in} T \text{ using } \text{IsClosed_def by auto}
  then have \text{pointCl : } \forall x \in \bigcup T. \ \{x\} \{\text{is closed in} T \text{ by auto}
  {
    \fix A
    assume AS2: A \in \text{FinPow}(\bigcup T)
    let p = (A, {{x} | x \in A})
    have p \in A \rightarrow {{x} | x \in A} \text{ using } \text{Pi_def unfolding } \text{function_def by auto}
    then have \text{p : bij(A, {{x} | x \in A}) unfolding } \text{bij_def inj_def surj_def}
    using \text{apply_equality}
    by auto
    then have A \approx {{x} | x \in A} \text{ unfolding } \text{eqpoll_def by auto}
    with AS2 have \text{Finite}({{x} | x \in A}) \text{ unfolding } \text{FinPow_def using } \text{eqpoll_imp_Finite_iff}
    by auto
    then have \text{{x} | x \in A} \in \text{FinPow}({D \in \text{Pow}(\bigcup T). D \{\text{is closed in} T})
  }
  using AS2 \text{ pointCl unfolding } \text{FinPow_def}
  by (safe, blast+)
  then have (\bigcup {{x} | x \in A}) \{\text{is closed in} T \text{ using } \text{fin_union_cl_is_cl}
  by auto

827
moreover
have \( \bigcup \{ x \mid x \in A \} = A \) by auto
ultimately have \( A \) {is closed in} \( T \) by simp

then have reg: \( \forall A \in \text{FinPow}(\bigcup T). A \) {is closed in} \( T \) by auto

fix \( U \)
assume AS2: \( U \in \text{CoCardinal}(\bigcup T, \text{nat}) \)
then have \( U \in \text{Pow}(\bigcup T) \) \( U = 0 \lor (\bigcup T - U) \prec \text{nat} \) using CoCardinal_def by auto
then have \( U \in \text{Pow}(\bigcup T) \) \( U \in T \lor (\bigcup T - U) \in T \) using lesspoll_nat_is_Finite
by auto
moreover
then have \( (\bigcup T - (\bigcup T - U)) = U \) by blast
ultimately have \( U \in T \) by auto

then show (\( \text{CoFinite}(\bigcup T) \)) \( \subseteq T \) using Cofinite_def by auto

assume (\( \text{CoFinite}(\bigcup T) \)) \( \subseteq T \)
then have AS: \( \text{CoCardinal}(\bigcup T, \text{nat}) \subseteq T \) using Cofinite_def by auto

fix \( x \) \( y \)
assume AS2: \( x \in \bigcup T, y \in \bigcup T \) \( x \neq y \)
have Finite(\( \{ y \} \)) by auto
then obtain n where \( \{ y \} \approx n \) \( n \in \text{nat} \) using Finite_def by auto
then have \( \{ y \} \prec \text{nat} \) using n_lesspoll_nat_eq_lesspoll_trans by auto
then have \( \{ y \} \) {is closed in} \( \text{CoCardinal}(\bigcup T, \text{nat}) \) using closed_sets_cocardinal
AS2(2) by auto
then have \( (\bigcup T - \{ y \}) \in \text{CoCardinal}(\bigcup T, \text{nat}) \) using union_cocardinal
IsClosed_def by auto
with AS have \( (\bigcup T - \{ y \}) \in T \) by auto
moreover
with AS2(1,3) have \( x \in ((\bigcup T) - \{ y \}) \land y \notin ((\bigcup T) - \{ y \}) \) by auto
ultimately have \( \exists V \in T. x \in V \land y \notin V \) by (safe, auto)
then show \( T \) {is \( T_1 \)} using 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 topology0) T1_ifs_singleton_closed:
shows \( (T \) {is \( T_1 \}) \iff (\forall x \in \bigcup T. \{ x \} \) {is closed in} \( T \))
proof
assume AS:T \{is T\}
\{ 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
with p have \{x\} {is closed in}T using IsClosed_def by auto
\}
then show pointCl:\∀x∈∪T. \{x\} {is closed in} T by auto
next
assume pointCl:\∀x∈∪T. \{x\} {is closed in} T
\{ 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 then have \((\bigcup T-(\bigcup T-U))=U\) by blast ultimately have \(U \in T\) by auto

} then have \((\text{CoFinite } (\bigcup T)) \subseteq T\) using Cofinite_def by auto then show \(T\) \{is \(T_1\}\) using T1_cocardinal_coarser by 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 \(Q_1 \preceq Q_2\)
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 \(x \in \text{Pow}(X)\) using CoCardinal_def by auto
with assms have \(x \in \text{Pow}(X)\) using lesspoll_trans2 by auto
then show \(x \in \text{CoCardinal}(X, Q_2)\) using CoCardinal_def by auto

qed

**corollary cocardinal_is_T1:**
fixes \(X K\)
assumes \(\text{InfCard}(K)\)
shows \(\text{CoCardinal}(X,K) \{\text{is } T_1\}\)

**proof**
- have \(\text{nat} \leq K\) using InfCard_def assms by auto
then have \(\text{nat} \subseteq K\) using le_imp_subset by auto
then have \(\text{nat} \subseteq K, K \neq 0\) using subset_imp_lepoll by auto
then have \(\text{CoCardinal}(X, \text{nat}) \subseteq \text{CoCardinal}(X,K) \bigcup \text{CoCardinal}(X,K)=X\)
using order_cocardinal_top
union_cocardinal by auto
then show thesis using topology0.T1_cocardinal_coarser topology0_CoCardinal assms Cofinite_def by auto

qed

In \(T_2\)-spaces, filters and nets have at most one limit point.

**lemma (in topology0) T2_imp_unique_limit_filter:**
assumes \(T\) \{is \(T_2\}\) \(\mathcal{F}\) \{is a filter on\}\(\bigcup T\) \(\mathcal{F} \rightarrow x\) \(\mathcal{F} \rightarrow y\)
shows \(x=y\)

**proof**
- assume \(x \neq y\)
from assms(3,4) have \(x \in \bigcup T\) \(y \in \bigcup T\) using FilterConverges_def assms(2) by auto

830
with \( x \neq y \) have \( \exists U \in T. \exists V \in T. x \in U \land y \in V \land U \cap V = 0 \) using assms(1) isT2_def by auto
then obtain \( U, V \) where \( x \in U \) \( y \in V \) \( U \cap V = 0 \) \( U \in T \) \( V \in T \) by auto
then have \( U \in F \) \( V \in F \) using FilterConverges_def assms(2) assms(3,4) by auto
then have \( U \cap V \in F \) using IsFilter_def assms(2) by auto
with \( U \cap V = 0 \) have \( 0 \in F \) by auto
then have \( False \) using IsFilter_def assms(2) by auto
\}
then show thesis by auto qed

\[
\text{lemma (in topology0) T2_imp_unique_limit_net: }
\begin{align*}
\text{assumes } & T \{ \text{is } T2 \} \quad N \{ \text{is a net on} \} \bigcup T N \to N x N \to y \\
\text{shows } & x = y
\end{align*}
\]
\proof-
have \( \text{(Filter } N..(\bigcup T)) \{ \text{is a filter on} \} (\bigcup T) \ (\text{Filter } N..(\bigcup T)) \to_F x \ \text{(Filter } N..(\bigcup T)) \to_F y \)
using filter_of_net_is_filter(1) net_conver_filter_of_net_conver assms(2) assms(3,4) by auto
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.

\[
\text{lemma (in topology0) unique_limit_filter_imp_T2: }
\begin{align*}
\text{assumes } & \forall x \in \bigcup T. \forall y \in \bigcup T. \forall \emptyset. ((\emptyset \{ \text{is a filter on} \} \bigcup T) \land (\emptyset \to_F x) \\
\text{shows } & T \{ \text{is } T2 \}
\end{align*}
\]
\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) \longrightarrow U \cap V \neq 0 
\let Ux = \{ A \in \text{Pow}(\bigcup T). x \in \text{int}(A) \} 
\let Uy = \{ A \in \text{Pow}(\bigcup T). y \in \text{int}(A) \} 
\let FF = Ux \cup Uy \cup \{ A \cap B. (A, B) \in Ux \times Uy \} 
\have sat:FF \{ \text{satisfies the filter base condition} \} 
\proof- 
\{ \fix A B 
\assume A \in FF B \in FF 
\{ \assume A \in Ux 
\{ \assume B \in Ux 

831
with \(<x \in \bigcup T > \langle A \in Ux \rangle\) have \(A \cap B \in Ux\) using neigh_filter(1)

IsFilter_def by auto
then have \(A \cap B \in FF\) by auto
}
mOREOVER
{ assume \(B \in Uy\)
with \(<A \in Ux>\) have \(A \cap B \in FF\) by auto
}
mOREOVER
{ assume \(B \in \{A \cap B. \langle A, B \rangle \in Ux \times Uy\}\)
then obtain \(A, B\) where \(B = A \cap B\) \(A \cap B \in Ux\) \(B \in Uy\) by auto
with \(<x \in \bigcup T > \langle A \in Ux \rangle\) have \(A \cap B \in AA \cap BB\) \(A \cap A \in Ux\) using neigh_filter(1)

IsFilter_def by auto
with \(<BB \in Uy>\) have \(A \cap B \in \{A \cap B. \langle A, B \rangle \in Ux \times Uy\}\) 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 \(A \in Uy\)
{ assume \(B \in Uy\)
with \(<y \in \bigcup T > \langle A \in Uy \rangle\) have \(A \cap B \in Uy\) using neigh_filter(1)

IsFilter_def by auto
then have \(A \cap B \in FF\) by auto
}
mOREOVER
{ assume \(B \in Ux\)
with \(<A \in Uy>\) 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 \(B \in \{A \cap B. \langle A, B \rangle \in Ux \times Uy\}\)
then obtain AA BB where \(B = A \cap BB\) \(AA \cap BB \in Ux\) \(BB \in Uy\) by auto
with \(<y \in \bigcup T > \langle A \in Uy \rangle\) have \(A \cap B = AA \cap (A \cap BB)\) \(AA \in Ux\) \(BB \in Uy\) using neigh_filter(1)

IsFilter_def by auto
with \(<AA \in Ux>\) have \(A \cap B \in \{A \cap B. \langle A, B \rangle \in Ux \times Uy\}\) by auto
then have \(A \cap B \in FF\) by auto
}
ultimately have \(A \cap B \in FF\) using \(<B \in FF>\) by auto
}
mOREVER
assume $A \in \{A \cap B. (A,B) \in Ux \times Uy\}$
then obtain $AA BB$ where $A=AA \cap BB AA \in Ux BB \in Uy$ by auto
{
assume $B \in Uy$
with $<BB \in Uy> \ y \in \bigcup T$ have $B \cap BB \in Uy$ using neigh_filter(1)
IsFilter_def auto
moreover from $<A=AA \cap BB$ have $A \cap B = AA \cap (B \cap BB)$ by auto
ultimately have $A \cap BB \in FF$ using $<AA \in Ux> BB \in Uy$ by auto
} 
moreover 
{
assume $B \in Ux$
with $<AA \in Ux> x \in \bigcup T$ have $B \cap AA \in Ux$ using neigh_filter(1)
IsFilter_def auto
moreover from $<A=AA \cap BB$ have $A \cap B = (A \cap AA) \cap BB$ by auto
ultimately have $A \cap BB \in FF$ using $<B \cap AA \in Ux BB \in Uy$ by auto
} 
moreover 
{
assume $B \in \{A \cap B. (A,B) \in Ux \times Uy\}$
then obtain $AA2 BB2$ where $B=AA2 \cap BB2 AA2 \in Ux BB2 \in Uy$ by auto
from $<B=AA2 \cap BB2> A=AA \cap BB$ have $A \cap B = (AA \cap AA2) \cap (BB \cap BB2)$
by auto
moreover from $<AA \in Ux> AA2 \in Ux > x \in \bigcup T$ have $AA \cap AA2 \in Ux$ using neigh_filter(1)
IsFilter_def auto
moreover from $<BB \in Uy> BB2 \in Uy > y \in \bigcup T$ have $BB \cap BB2 \in Uy$ using neigh_filter(1)
IsFilter_def auto
ultimately have $A \cap BB \in FF$ by auto
} 
ultimately have $A \cap BB \in FF$ by auto
} 
ultimately have $A \cap BB \in FF$ 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 $\bigcup T \in Ux$ using $<x \in \bigcup T$ neigh_filter(1) IsFilter_def by auto
then have $FF \neq 0$ by auto
moreover 
{
assume $0 \in FF$
moreover have $0 \in Ux$ using $<x \in \bigcup T$ neigh_filter(1) IsFilter_def by auto
moreover have $0 \in Uy$ using $<y \in \bigcup T$ neigh_filter(1) IsFilter_def by auto
ultimately have $0 \in \{A \cap B. (A,B) \in Ux \times Uy\}$ by auto
then obtain $A B$ where $0=A \cap B A \in Ux B \in Uy$ by auto

833
then have $x \in \text{int}(A) \land y \in \text{int}(B)$ by auto
moreover with $0 = A \cap B$ have $\text{int}(A) \cap \text{int}(B) = 0$ using Top_2_L1 by auto
moreover have $\text{int}(A) \subseteq \text{int}(B) \subseteq T$ using Top_2_L2 by auto
ultimately have False using $\forall U \in T. \forall V \in T. x \in U \land y \in V \implies U \cap V \neq 0$ by auto
}
then have $0 \notin \text{FF}$ by auto
ultimately show thesis using SatisfiesFilterBase_def by auto
qed
moreover have $\text{FF} \subseteq \text{Pow}(\bigcup T)$ by auto
ultimately have bas: $\text{FF}$ is a base filter using $\forall U \in T. \exists D \in \text{FF}. D \subseteq A$
} then have $\bigcup \{\text{FF} \subseteq \text{Pow}(\bigcup T)\}$ by auto
ultimately have $\text{fil} \subseteq \text{Pow}(\bigcup T)$ using basic_filter sat by auto
have $\forall U \in T. x \in \text{int}(U) \implies (\exists D \in \text{FF}. D \subseteq U)$ by auto
then have $\bigcup \{\text{fil} \subseteq \text{Pow}(\bigcup T)\}$ by auto
ultimately have $x = y$ using assms $\text{fil} \subseteq \text{Pow}(\bigcup T)$ by blast
with $x \neq y$ have False by auto
}
then have $\exists U \in T. \exists V \in T. x \in U \land y \in V \land U \cap V = 0$ by blast
}
then show thesis using isT2_def by auto
qed

lemma (in topology0) unique_limit_net_imp_T2:
assumes $\forall x \in U. \forall y \in U. \forall N. (\{N \text{ is a net on } U\} \land (N \rightarrow N x) \land (N \rightarrow N y)) \implies x = y$
shows $T$ is T2
proof-
{
fix $x$ $y$ $\mathcal{F}$
assume $x \in U \land y \in U \land \mathcal{F}$ is a filter on $U$ \implies (Net($\mathcal{F}$) is a net on $U$) \land (Net($\mathcal{F}$) \rightarrow N x \land (Net($\mathcal{F}$) \rightarrow N y)
using filter_conver_net_of_filter_conver net_of_filter_is_net by auto
with $x \in U$ \land $y \in U$ have $x = y$ using assms by blast
}
then have $\forall x \in U. \forall y \in U. (\forall \mathcal{F} \text{ is a filter on } U) \land (\mathcal{F} \rightarrow F x) \land (\mathcal{F} \rightarrow F y) \implies x = y$ by auto
then show thesis using unique_limit_filter_imp_T2 by auto

834
This results make easy to check if a space is $T_2$.

The topology which comes from a filter as in $\mathcal{F}$ (is a filter on $\bigcup \mathcal{F} \Rightarrow (\mathcal{F} \cup \{0\})$ (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_card1:
  assumes $\bigcup \mathcal{F} \neq \{x\}$
  shows $\bigcup \mathcal{F} = \{x\}$
proof-
  { fix $y$ assume $y \in \bigcup \mathcal{F}$
    then have $\mathcal{F} \rightarrow F \{\text{in } (\mathcal{F} \cup \{0\}) \text{ using } \lim_{\mathcal{F} \to \mathcal{F}} \text{ top_of_filter } \text{assms}(2)$
    by auto
    moreover
    have $\mathcal{F} \rightarrow F \{\text{in } (\mathcal{F} \cup \{0\}) \text{ using } \lim_{\mathcal{F} \to \mathcal{F}} \text{ top_of_filter } \text{assms}(2,3)$
    by auto
    moreover
    have $\bigcup \mathcal{F} = \bigcup (\mathcal{F} \cup \{0\})$ by auto
    ultimately
    have $y = x$ using topology0.T2_imp_unique_limit_filter[OF topology0_filter[OF assms(2)] assms(1)] assms(2)
    by auto
  }
  then have $\bigcup \mathcal{F} \subseteq \{x\}$ by auto
  with assms(3) show thesis by auto
qed

There are more separation axioms that just $T_0$, $T_1$ or $T_2$

definition isRegular (_{is regular} 90)
  where $T$ (is regular) $\equiv$ $\forall A. A$ (is closed in) $\Rightarrow (\forall x \in \bigcup T-A. \exists \mathcal{U} \in T. \exists \mathcal{V} \in T. A \subseteq \mathcal{U} \land x \in \mathcal{V} \land \mathcal{U} \cap \mathcal{V} = \emptyset)$

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) $\Rightarrow (\forall B. B$ (is closed in) $\Rightarrow (\exists \mathcal{U} \in T. \exists \mathcal{V} \in T. A \subseteq \mathcal{U} \land \mathcal{V} \subseteq \bigcup \mathcal{U} \cap \mathcal{V} = \emptyset))$

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 T4} shows T{is T3}
proof-
  from assms have nor:T{is normal} using isT4_def by auto
  from assms have T{is T1} using isT4_def by auto
  then have Cofinite (⋃T) ⊆ T using T1_cocardinal_coarser by auto
  { fix A
    assume AS:A{is closed in}T
    { fix x
      assume x ∈ ⋃T-A
      have Finite({x}) by auto
      then obtain n where {x} ≈ n n ∈ nat unfolding Finite_def by auto
      then have {x} ≲ n n ∈ nat using eqpoll_imp_lepoll by auto
      then have {x} ≺ nat using n_lesspoll_nat lesspoll_trans1 by auto
      with <x ∈ ⋃T-A> have {x} {is closed in} (Cofinite (⋃T)) using Cofinite_def
        closed_sets_cocardinal by auto
      then have ⋃T-{x} ∈ Cofinite(⋃T) unfolding IsClosed_def using union_cocardinal
        Cofinite_def by auto
      with <Cofinite (⋃T) ⊆ T> have ⋃T-{x} ∈ T using Cofinite_def by auto
      with <Cofinite (⋃T) ⊆ T> have ⋃T-{x} ∈ T by auto
      with <x ∈ ⋃T-A> have {x} {is closed in} (Cofinite (⋃T)) using Cofinite_def
        closed_sets_cocardinal by auto
      then have ∀x ∈ ⋃T-A. ∃U ∈ T. ∃V ∈ T. A ⊆ U ∧ {x} ⊆ V ∧ U ∩ V = 0 using IsNormal_def
        by blast
      then have T{is regular} using IsRegular_def by blast
      with T{is T1} show thesis using isT3_def by auto
    qed
  }
lemma (in topology0) T3_is_T2:
  assumes T{is T3} shows T{is T2}
proof-
  from assms have T{is regular} using isT3_def by auto
  from assms have T{is T1} using isT3_def by auto
  then have Cofinite (⋃T) ⊆ T using T1_cocardinal_coarser by auto
  { fix x y
    assume x ∈ ⋃Ty ∈ ⋃Tx ≠ y
    have Finite({x}) by auto
    then obtain n where {x} ≈ n n ∈ nat unfolding Finite_def by auto
    then have {x} ≲ n n ∈ nat using eqpoll_imp_lepoll by auto
    then have {x} ≺ nat using n_lesspoll_nat lesspoll_trans1 by auto
    with <x ∈ ⋃T> have {x} {is closed in} (Cofinite (⋃T)) using Cofinite_def
then have $\bigcup T - \{x\} \in \text{Cofinite}(\bigcup T)$ unfolding \texttt{IsClosed_def} using \texttt{union_cocardinal} \texttt{Cofinite_def} by auto

with $\langle \text{Cofinite } (\bigcup T) \subseteq T \rangle$ have $\bigcup T - \{x\} \in \text{Cofinite}(\bigcup T)$ unfolding \texttt{IsClosed_def} by auto

with $\langle \text{is regular} \rangle$ have $\exists U \in T. \{x\} \subseteq U \land y \in U \land x \neq y$ unfolding \texttt{IsRegular_def} by force

then have $\exists U \in T. \{x\} \subseteq U$ unfolding \texttt{IsClosed_def} by auto

with $\langle \text{T is regular} \rangle$ have $\exists U \in T. \{x\} \subseteq U \land y \in U \land x \neq y$ unfolding \texttt{IsRegular_def} by force

then have $\exists U \in T. \{x\} \subseteq U$ unfolding \texttt{isT2_def} by auto

qed
then have $c(V) = \bigcup T - \text{int}(\bigcup T-V)$ using Top_3_L11(2) [of $\bigcup T-V$] by auto
ultimately have $F \subseteq \text{int}(\bigcup T-V)$ by auto moreover
have $\forall V \ni (\text{int}(\bigcup T-V)) \cap V = 0$ by auto moreover
note $<x \in V, <V \in T>$. ultimately
have $V \ni \text{int}(\bigcup T-V) \subseteq \bigcup T-V$ by Top_2_L1 by auto
then have $V \cap (\text{int}(\bigcup T-V)) = 0$ by auto
moreover have $\exists x \in V, <V \in T>$ ultimately
have $\exists U \ni x \in V \cap \text{int}(\bigcup T-V) = 0$ by auto
using Top_2_L2 by auto
then have $\bigcup T - F \subseteq \text{int}(\bigcup T-V)$ and $x \in V$ and $(\text{int}(\bigcup T-V)) \cap V = 0$ using Top_2_L2 by auto
then have $\forall x \in \bigcup T - F, \exists U \ni x \in V$ and $U \cap V = 0$ by auto
then show thesis using IsRegular_def by blast qed

lemma (in topology0) regular_eq:
shows $T \{\text{is regular}\} \iff (\forall x \in \bigcup T. \forall U \ni x \in U \iff (\exists V \ni x \in V \wedge c(V) \subseteq U))$
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 $T \{\text{is T}_2\}, A \{\text{is compact in} T\}$
shows $\exists U \ni x \in \bigcup T \neg x \in A$

proof-
assume $A = 0$
then have $A \subseteq \bigcup T \cap (\bigcup T - 0) = 0$ 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 $<x \notin A>$ assms(4) have $x \neq y$ by auto
moreover from $<y \in A>$ have $x \in \bigcup T \forall y \in \bigcup T$ using assms(2,3) unfolding IsCompact_def by auto
ultimately obtain $U \ni V$ where $U \in T \cap U \cap V = 0 \ni x \in \bigcup U \ni y \in V$ using assms(1) unfolding isT2_def by blast
then have $\exists (U, V) \ni x \in V$ by auto
}
then have $\forall y \in A. \exists (U, V) \ni y \in V$ by auto
then have $A \subseteq \{ \text{snd}(B). B \in U \}$ by auto
moreover have $\{ \text{snd}(B). B \in U \} \in \text{Pow}(T)$ by auto
ultimately have $\exists N \in \text{FinPow}( \{ \text{snd}(B). B \in U \} ) \ni A \cup N$ using assms(2) unfolding IsCompact_def by auto

838
then obtain \( N \) where \( ss : N \in \text{FinPow}(\{ \text{snd}(B), B \in U \}) \) \( A \subseteq N \) by auto
with \( \{ \text{snd}(B), B \in U \} \in \text{Pow}(T) \) have \( A \subseteq N \in \text{Pow}(T) \) unfolding \( \text{FinPow} \_ \text{def} \)
by auto
then have \( N : A \subseteq N \cup N \in T \) using \( \text{topSpaceAssum} \) unfolding \( \text{IsATopology} \_ \text{def} \)
by auto
from \( ss \) have \( \text{Finite}(N) N \subseteq \{ \text{snd}(B), B \in U \} \) unfolding \( \text{FinPow} \_ \text{def} \) by auto
then obtain \( n \) where \( n \in \text{nat} N = n \) unfolding \( \text{Finite} \_ \text{def} \) by auto
then have \( N \subseteq n \) using \( \text{eqpoll} \_ \text{imp} \_ \text{lepoll} \) by auto
from \( \text{noEmpty} \) \( A \subseteq N \Rightarrow \) \( \text{NnoEmpty} : N \neq 0 \) by auto
let \( QQ = \{ (n, (\text{fst}(B), B \in (A \in \text{U}, \text{snd}(A) = n)) \}, n \in N \} \)
have \( QQ_{\text{IP}} : QQ : N \rightarrow \{ \text{fst}(B). B \in (A \in \text{U}, \text{snd}(A) = n) \}, n \in N \} \) unfolding \( \text{Pi} \_ \text{def} \)
function \_ definition \( \text{domain} \_ \text{def} \)
by auto
\[
\begin{align*}
  \text{fix } n & \text{ assume } n \in N \\
  \text{with } & \{ N \subseteq \{ \text{snd}(B), B \in U \} \} \text{ obtain } B \text{ where } n = \text{snd}(B) B \in U \text{ by auto} \\
  \text{then have } & \text{fst}(B) \in \{ \text{fst}(B). B \in (A \in \text{U}, \text{snd}(A) = n) \} \text{ by auto} \\
  \text{then have } & \{ \text{fst}(B). B \in (A \in \text{U}, \text{snd}(A) = n) \} \neq 0 \text{ by auto moreover} \\
  \text{from } & \{ n \subseteq N \} \text{ have } \{ (n, (\text{fst}(B). B \in (A \in \text{U}, \text{snd}(A) = n)) \in QQ \} \text{ by auto} \\
  \text{with } & \text{QQ}_{\text{IP}} \text{ have } QQ_{n} = \{ \text{fst}(B). B \in (A \in \text{U}, \text{snd}(A) = n) \} \text{ using } \text{apply} \_ \text{equality} \\
\end{align*}
\]
by auto
ultimately have \( QQ_{n} \neq 0 \) by auto
\[
\begin{align*}
  \text{then have } & \forall n \in N. \ QQ_{n} \neq 0 \text{ by auto} \\
  \text{with } & \{ n \in \text{nat} N \subseteq n \} \text{ have } \exists f. f \in \Pi (N, \lambda t. QQ_{t}) \wedge (\forall t \in N. ft \in QQ_{t}) \text{ using } \text{finite} \_ \text{choice} \text{ unfolding } (\text{AxiomCardinalChoice} \_ \text{Gen} \_ \text{def}) \\
  \text{by auto} \\
  \text{then obtain } & f \text{ where } fPI : f \in \Pi (N, \lambda t. QQ_{t}) (\forall t \in N. ft \in QQ_{t}) \text{ by auto} \\
  \text{from } & fPI(1) \text{ NnoEmpty have range}(f) \neq 0 \text{ unfolding } \text{Pi} \_ \text{def} \text{ range} \_ \text{def} \text{ domain} \_ \text{def} \text{ converse} \_ \text{def} \text{ by } (\text{safe}, \text{blast}) \\
\end{align*}
\]
\[
\begin{align*}
  \text{fix } t & \text{ assume } t \in N \\
  \text{then have } & ft \in QQ_{t} \text{ using } fPI(2) \text{ by auto} \\
  \text{with } & \{ t \subseteq N \} \text{ have } ft \in \bigcup (QQ_{N}) \text{ QQ}_{t} \subseteq \bigcup (QQ_{N}) \text{ using } \text{func} \_ \text{image} \_ \text{def} \text{ QQ}_{\text{IP}} \\
\end{align*}
\]
by auto
\[
\begin{align*}
  \text{then have } & \text{reg} : \forall t \in N. ft \in \bigcup (QQ_{N}) \forall t \in N. QQ_{t} \subseteq \bigcup (QQ_{N}) \text{ by auto} \\
  \text{fix } tt & \text{ assume } tt \in f \\
  \text{with } & fPI(1) \text{ have } tt \subseteq \Sigma \in (N, ()(QQ)) \text{ unfolding } \text{Pi} \_ \text{def} \text{ by auto} \\
  \text{then have } & tt \subseteq (\bigcup xa \in N. \bigcup y \in QQ_{xa}. \{ xa, y \}) \text{ unfolding } \Sigma \_ \text{def} \text{ by auto} \\
  \text{then obtain } & xa \ y \text{ where } xa \in N y \in QQ_{xa} tt = (xa, y) \text{ by auto} \\
  \text{with } & \text{reg}(2) \text{ have } y \in \bigcup (QQ_{N}) \text{ by blast} \\
  \text{with } & \{ tt = (xa, y) \} \text{ have } tt \in (\bigcup xa \in N. \bigcup y \in \bigcup (QQ_{N}). \{ (xa, y) \}) \\
\end{align*}
\]
by auto
\[
\begin{align*}
  \text{then have } & tt \in N \times (\bigcup (QQ_{N})) \text{ unfolding } \Sigma \_ \text{def} \text{ by auto} \\
  \text{then have } & \text{ffun} : f : N \rightarrow \bigcup (QQ_{N}) \text{ using } fPI(1) \text{ unfolding } \text{Pi} \_ \text{def} \text{ by auto} \\
  \text{then have } & f \in \text{surj}(N, \text{range}(f)) \text{ using } \text{fun} \_ \text{is} \_ \text{surj} \text{ by auto} \\
\end{align*}
\]

839
with \(<N \subseteq \text{n} \subseteq \text{n} \in \text{nat}> have \text{range}(f) \subseteq N\) using surj_fun_inv_2 nat_into_Ord by auto

with \(<N \subseteq \text{n} \subseteq \text{n} \in \text{nat}> have \text{range}(f) \subseteq n\) using lepoll_trans by blast

with \(<\text{n} \in \text{nat}> have \text{Finite}(\text{range}(f))\) using n_lesspoll_nat lesspoll_nat_is_Finite

lesspoll_trans1 by auto

moreover from ffun have \(\text{rr}: \text{range}(f) \subseteq \bigcup (\text{QQ} \text{N})\) unfolding Pi_def by auto

then have \(\text{range}(f) \subseteq T\) by auto

ultimately have \(\text{range}(f) \in \text{FinPow}(T)\) unfolding FinPow_def by auto

then have \(\bigcap \text{range}(f) \in T\) using fin_inter_open_open <\text{range}(f) \neq 0> by auto

moreover { fix \(S\) assume \(S \in \text{range}(f)\)
  with \(\text{rr}\) have \(\exists B \in (\text{QQ} \text{N}). S \in B\) using Union_iff by auto
  then obtain \(B\) where \(B \in (\text{QQ} \text{N})\). \(S \in B\) by auto
  then have \(\exists \text{rr} \in \text{N}. \langle \text{rr}, B \rangle \in \text{QQ}\) unfolding image_def by auto
  then have \(\exists \text{rr} \in \text{N}. B = \{\text{fst}(B). B \in \{A \in U. \text{snd}(A) = \text{rr}\}\}\) by auto
  with \(\text{rr} \in \text{N}\) obtain \(\text{rr}\) where \(\langle \text{rr}, B \rangle \in \text{QQ}\) by auto
  then have \(\exists \text{rr} \in \text{N}. B = \{\text{fst}(B). B \in \{A \in U. \text{snd}(A) = \text{rr}\}\}\) by auto
  with \(\text{rr} \in \text{N}\) fPI(2) have \(\text{rr} \in \text{QQ} \text{t}\) by auto
  with \(\text{rr} \in \text{N}\) have \(\text{rr} \in \{\text{fst}(B). B \in \{A \in U. \text{snd}(A) = \text{rr}\}\}\) using apply_equality

QQP1 by auto

then have \(\langle \text{ft}, \text{t} \rangle \in U\) by auto

then have \(\text{ft} \cap \text{t} = 0\) by auto

with \(\text{y} \in \text{t}\) yft have False by auto

} then have \((\bigcup \text{N}) \cap (\bigcap \text{range}(f)) = 0\) by blast moreover note NN

ultimately have thesis by auto

} ultimately show thesis by auto

qed

A Hausdorff space separates compact spaces from other compact spaces.

theorem (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 \(\exists U \in T. \exists V \in T. A \subseteq U \land B \subseteq V \land U \cap V = 0\)

proof -


assume \( B = 0 \)
then have \( A \subseteq \bigcup T \land B \subseteq 0 \land ((\bigcup T) \cap 0 = 0) \) using assms(2) unfolding IsCompact_def by auto
moreover
have \( 0 \in T \) using empty_open topSpaceAssum by auto
moreover
have \( \bigcup T \in T \) using topSpaceAssum unfolding IsATopology_def by auto
ultimately
have thesis by auto
}
moreover

\begin{itemize}
\item \( \) assume \( \text{noEmpty} : B \neq 0 \)
let \( U = \{(U, V) \in T \times T. A \subseteq U \land U \cap V = 0\} \)
\item \( \) fix \( y \) assume \( y \in B \)
then have \( y \in \bigcup T \) using assms(3) unfolding IsCompact_def by auto
\end{itemize}

ultimately

\begin{itemize}
\item \( \) have \( \exists (U, V) \in U. y \in V \) by auto
\item \( \) then have \( \forall y \in B. \exists (U, V) \in U. y \in V \) by auto
\item \( \) then have \( B \subseteq \bigcup \{\text{snd}(B). B \in U\} \) by auto
\item \( \) then obtain \( N \) where \( ss : N \in \text{FinPow}(\{\text{snd}(B). B \in U\}). B \subseteq \bigcup N \) using assms(3) unfolding IsCompact_def by auto
\item \( \) then obtain \( nn : B \subseteq \bigcup N \bigcup n \in T \) using topSpaceAssum unfolding IsATopology_def by auto
\item \( \) from \( ss \) have \( \text{Finite}(N) N \subseteq \{\text{snd}(B). B \in U\} \) unfolding FinPow_def by auto
\item \( \) then obtain \( n \) where \( n \in \text{nat} N \neq n \) unfolding Finite_def by auto
\item \( \) then have \( N \subseteq n \) using eqpoll_imp_lepoll by auto
\item \( \) let \( QQ = \{ (n, \{f\text{st}(B). B \in \{A \in U. \text{snd}(A) = n\}\}). n \in N\} \)
\item \( \) have \( QQ_1 : QQ : N \rightarrow \{f\text{st}(B). B \in \{A \in U. \text{snd}(A) = n\}. n \in N\} \) unfolding Pi_def function_def domain_def by auto
\item \( \) 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
\item \( \) then have \( \text{fst}(B) \in \{f\text{st}(B). B \in \{A \in U. \text{snd}(A) = n\}\} \) by auto
\item \( \) then have \( \{f\text{st}(B). B \in \{A \in U. \text{snd}(A) = n\}\} \neq 0 \) by auto
moreover
\item \( \) from \( \langle n \in N\rangle \) have \( \langle n, \{f\text{st}(B). B \in \{A \in U. \text{snd}(A) = n\}\}\rangle \in QQ \) by auto
with \( QQ_1 \) have \( QQ_n = \{f\text{st}(B). B \in \{A \in U. \text{snd}(A) = n\}\} \) using apply_equality by auto
\item \( \) ultimately
\item \( \) have \( QQ_n \neq 0 \) by auto
\end{itemize}

ultimately

\begin{itemize}
\item \( \) have \( \forall n \in N. QQ_n \neq 0 \) by auto
\item \( \) with \( \langle n \in \text{nat} \rangle \) have \( \exists f : f \in \text{Pi}(N, \lambda t. QQ_t) \land (\forall t \in N. ft \in QQ_t) \) using finite_choice unfolding AxiomCardinalChoiceGen_def

841
by auto
then obtain f where fPI:f∈Π(N,∀t.QQt) (∀t∈N. ft∈QQt) by auto
from fPI(1) NnoEmpty have range(f)≠∅ unfolding Pi_def range_def domain_def
converse_def by (safe,blast)
{ 
  fix t assume t∈N
  then have ft∈QQt using fPI(2) by auto
with <t∈N> have ft∈∪(QQN) QQt⊆∪(QQN) using func_imagedef QQPi
by auto
}
then have reg:∀t∈N. ft∈∪(QQN) ∀t∈N. QQt⊆∪(QQN) by auto
{ 
  fix tt assume tt∈f
  with fPI(1) have tt∈Sigma(N, ()(QQ)) unfolding Pi_def
  by auto
  then have tt∈∪(xa∈N. ∪y∈QQxa. {⟨xa,y⟩}) unfolding Sigma_def
  by auto
  then obtain xa y where xa∈N y∈QQxa tt=⟨xa,y⟩ by auto
  with reg(2) have y∈∪(QQN) by blast
  with <tt=(xa,y)> <xa∈N> have tt∈∪(xa∈N. ∪y∈∪(QQN). {⟨xa,y⟩})
  by auto
  then have tt∈N×(∪(QQN)) unfolding Sigma_def by auto
}
then have ffun:N→∪(QQN) using fPI(1) unfolding Pi_def by auto
then have f∈surj(N,range(f)) using fun_is_surj by auto
with <N≤n> <n∈nat> have range(f)≤N using surj_fun_inv_2 nat_into_Ord
by auto
with <N≤n> have range(f)≤n using lepoll_trans by blast
with <n∈nat> have Finite(range(f)) using n_lesspoll_nat lesspoll_nat_is_Finite
lesspoll_trans1 by auto
moreover from ffun have rr:range(f)⊆∪(QQN) unfolding Pi_def by auto
then have range(f)⊆T by auto
ultimately have range(f)∈FinPow(T) unfolding FinPow_def by auto
then have ∩range(f)∈T by auto
moreover
{ 
  fix S assume S∈range(f)
  with rr have S∈∪(QQN) by blast
  then have ∃B∈(QQN). S∈B using Union_iff by auto
  then obtain B where B∈(QQN) S⊆B by auto
  then have ∃rr∈N. (rr,B)∈QQ unfolding image_def by auto
  then have ∃rr∈N. B={fst(B). B∈{A∈U. snd(A)=rr}} by auto
  with <S∈B> obtain rr where ⟨S,rr⟩∈U by auto
  then have A⊆S by auto
}
then have A⊆∩range(f) using ∩range(f)≠∅ by auto
moreover
{ 
  fix y assume y∈(∪N)∩(∩range(f))
  then have reg:(∀S∈range(f). y∈S)∧∃t∈N. y∈t) by auto
}
then obtain $t$ where $t \in \mathbb{N}$ by auto  
then have $(t, \{\text{fst}(B) \mid B \in \{A \in U. \text{snd}(A) = t\}) \in \mathbb{Q}$ by auto  
with reg have yft: $y \inft$ by auto  
with $<t \in \mathbb{N}>$ have ft$\in \mathbb{Q}$ by auto  
with $<t \in \mathbb{N}>$ have ft$\in \{\text{fst}(B) \mid B \in \{A \in U. \text{snd}(A) = t\}\}$ using apply_equality

QQP1 by auto  
then have $(f(t), t) \in U$ by auto  
then have $f(t) = 0$ by auto  
with $<y \in ft>$ yft have False by auto  
}

then have $(\bigcap_{\text{range}(f)}) \cap (\bigcup_{\mathbb{N}}) = 0$ by blast  
moreover  
note $NN$  
ultimately have thesis by auto  
}  
ultimately show thesis by auto  
qed

A compact Hausdorff space is normal.

corollary (in topology0) $T_2$-compact_is_normal:  
assumes $T$ is $T_2$  
shows $T$ is normal using IsNormal_def

proof-  
from assms(2) have car_nat:$(\bigcup T)\{\text{is compact of cardinal}\} \text{nat(in)T}$ using Compact_is_card_nat by auto  
}  
fix A B assume A(is closed in)T B(is closed in)T\{\text{is compact of cardinal}\}nat(in)T  
then have con:$(\bigcup T)\{\text{is compact of cardinal}\} \text{nat(in)T}$ $(\bigcup T)\cap B\{\text{is compact of cardinal}\} \text{nat(in)T}$ using compact_closed[OF car_nat]  
by auto  
from $<A(is closed in)T>\cdot<B(is closed in)T>$ have $(\bigcup T)\cap A = A(\bigcup T)\cap B = B$  
unfolding IsClosed_def by auto  
with con have A(is compact of cardinal) \text{nat(in)T} B(is compact of cardinal) \text{nat(in)T}$  
by auto  
then have A(is compact in)T B(is compact in)T using Compact_is_card_nat  
by auto  
with $<A\cap B = 0>$ have $F \subseteq T. \exists V \subseteq T. A \subseteq U \land B \subseteq V \land U \cap V = 0$ using $T_2$-compact_compact

assms(1) by auto  
}  
then show \( \forall A. A \{\text{is closed in} \} T \longrightarrow (\forall B. B \{\text{is closed in} \} T \land A \land B = 0 \longrightarrow (\exists U \subseteq T. \exists V \subseteq T. A \subseteq U \land B \subseteq V \land U \cap V = 0))  
by auto  
qed

65.2 Hereditability

A topological property is hereditary if whenever a space has it, every subspace also has it.

definition IsHer (_{is hereditary} 90)

843
where \( P \) is hereditary \( \equiv \forall T. (T \text{ is a topology}) \land P(T) \rightarrow (\forall A \in \text{Pow}(\bigcup T). P(T \text{ restricted to } A)) \)

**Lemma subspaces_of_subspaces:**

assumes \( A \subseteq B \subseteq \bigcup T \)

shows \( T \text{ restricted to } A = (T \text{ restricted to } B) \text{ restricted to } A \)

**Proof**

from \( \text{assms} \) have \( S: \forall S \in T. A \cap (B \cap S) = A \cap S \) by \( \text{auto} \)

then show \( T \text{ restricted to } A \subseteq T \text{ restricted to } B \text{ restricted to } A \)

unfolding \( \text{RestrictedTo_def} \) by \( \text{auto} \)

from \( S \) show \( T \text{ restricted to } B \text{ restricted to } A \subseteq T \text{ restricted to } A \)

unfolding \( \text{RestrictedTo_def} \) by \( \text{auto} \)

qed

The separation properties \( T_0, T_1, T_2 \) and \( T_3 \) are hereditary.

**Theorem regular_here:**

assumes \( T \text{ is regular} \) \( A \in \text{Pow}(\bigcup T) \)

shows \( (T \text{ restricted to } A) \text{ is regular} \)

**Proof**

\{- \}

fix \( C \)

assume \( A: C \text{ (is closed in) } (T \text{ restricted to } A) \)

\{- fix y assume y \in \bigcup (T \text{ restricted to } A) y \notin C \}

with \( A \) have \( (\bigcup (T \text{ restricted to } A)) - C \subseteq (T \text{ restricted to } A) C \subseteq \bigcup (T \text{ restricted to } A) \) \( y \in \bigcup (T \text{ restricted to } A) y \notin C \)

unfolding \( \text{IsClosed_def} \)

by \( \text{auto} \)

moreover with \( \text{assms(2)} \) have \( \bigcup (T \text{ restricted to } A) = A \)

unfolding \( \text{RestrictedTo_def} \)

by \( \text{auto} \)

ultimately have \( A - C \subseteq T \text{ restricted to } A \)

\( y \in A \cap y \notin C \subseteq \text{Pow}(A) \)

by \( \text{auto} \)

then obtain \( S \) where \( S \subseteq T \) \( A \cap S = A - C \) \( y \in A \cap y \notin C \)

unfolding \( \text{RestrictedTo_def} \)

by \( \text{auto} \)

ultimately have \( A - C \subseteq T \text{ restricted to } A \)

\( y \in A \cap y \notin C \subseteq \text{Pow}(A) \)

by \( \text{auto} \)

then obtain \( S \) where \( S \subseteq T \) \( A \cap S = A - C \) \( y \in A \cap y \notin C \)

unfolding \( \text{RestrictedTo_def} \)

by \( \text{auto} \)

ultimately have \( A - C \subseteq T \text{ restricted to } A \)

\( y \in A \cap y \notin C \subseteq \text{Pow}(A) \)

by \( \text{auto} \)

then obtain \( S \) where \( S \subseteq T \) \( A \cap S = A - C \) \( y \in A \cap y \notin C \)

unfolding \( \text{RestrictedTo_def} \)

by \( \text{auto} \)

ultimately have \( A - C \subseteq T \text{ restricted to } A \)

\( y \in A \cap y \notin C \subseteq \text{Pow}(A) \)

by \( \text{auto} \)

then obtain \( S \) where \( S \subseteq T \) \( A \cap S = A - C \) \( y \in A \cap y \notin C \)

unfolding \( \text{RestrictedTo_def} \)

by \( \text{auto} \)

ultimately have \( A - C \subseteq T \text{ restricted to } A \)

\( y \in A \cap y \notin C \subseteq \text{Pow}(A) \)

by \( \text{auto} \)

then obtain \( S \) where \( S \subseteq T \) \( A \cap S = A - C \) \( y \in A \cap y \notin C \)

unfolding \( \text{RestrictedTo_def} \)

by \( \text{auto} \)

ultimately have \( A - C \subseteq T \text{ restricted to } A \)

\( y \in A \cap y \notin C \subseteq \text{Pow}(A) \)

by \( \text{auto} \)

then obtain \( S \) where \( S \subseteq T \) \( A \cap S = A - C \) \( y \in A \cap y \notin C \)

unfolding \( \text{RestrictedTo_def} \)

by \( \text{auto} \)
with $C \subseteq \mathcal{P}(A) \Rightarrow \forall y \in A$ have $C \subseteq A \cap Uy \subseteq A \cap V \cap y \cap V = 0$ by auto
ultimately have $\exists U \in (\mathcal{T}_{\text{restricted to}A}). \exists V \in (\mathcal{T}_{\text{restricted to}A}). C \subseteq U \land y \in V \land U \cap V = 0$ by auto

} then have $\forall C. (\mathcal{T}_{\text{restricted to}A}) - C. \exists U \in (\mathcal{T}_{\text{restricted to}A}). \exists V \in (\mathcal{T}_{\text{restricted to}A}). C \subseteq U \land y \in V \land U \cap V = 0$ by blast
then have 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 $T\{\text{T\_1}\} \mathcal{A} \in \mathcal{P}(\bigcup T)$ shows $(\mathcal{T}_{\text{restricted to}A})\{\text{T\_1}\}$
proof-
from assms(2) have $\mathcal{U}(\mathcal{T}_{\text{restricted to}A}) = A$ unfolding RestrictedTo_def
by auto

{ fix x y
assume $x \in A \land y \notin y$
with $A \in \mathcal{P}(\bigcup T)$ have $x \in \bigcup Ty \in \bigcup T x \neq y$ by auto
then have $\exists U \in T. x \in U \land y \notin U$ using assms(1) isT1_def by auto
then obtain $U$ where $U \in T \land x \in U \land y \notin U$ unfolding RestrictedTo_def
by auto

then have $\exists U \in (\mathcal{T}_{\text{restricted to}A}). x \in U \land y \notin U$ by blast
}
with un have $\forall y \in A \land y \notin y \in A \land y \notin y \in A \land y \notin y$ by auto
then show thesis using isT1_def by auto
qed

corollary here_T1:
shows isT1 {is hereditary} using T1_here IsHer_def by auto

lemma here_and:
assumes P {is hereditary} Q {is hereditary}
shows $(\Lambda T. P(T) \land Q(T))$ {is hereditary} using assms unfolding IsHer_def
by auto

corollary here_T3:
shows isT3 {is hereditary} using here_and[OF here_T1 here_regular]
unfolding IsHer_def isT3_def.
lemma T2_here:
  assumes T\{is T_2\} \{A \in \text{Pow}(\bigcup T)\} shows \((T\{\text{restricted to}A\}\{is T_2\})\)
proof-
  from assms(2) have un:\bigcup(T\{\text{restricted to}A\})=A unfolding RestrictedTo_def by auto
  { fix x y
    assume x\in A y\in A \neq y
    with \(A \in \text{Pow}(\bigcup T)\) have x\in A y\notin A 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 assms(1) isT2_def by auto
    then obtain U V where U \in T V \in T x \in U \land y \notin U \lor (y \in U \land x \notin U) using auto
    with \(x \in A \land y \notin A\) \(\exists U \in T\{\text{restricted to}A\}\)(x \in U \land y \notin U \lor (y \in U \land x \notin U)) unfolding RestrictedTo_def by auto
    then have \(\exists U \in T\{\text{restricted to}A\}. \exists V \in T\{\text{restricted to}A\}. x \in U \land y \in V \land U \cap V = 0\) by auto
    then show thesis using isT2_def by auto
  }
  with un have \(\forall x y. x \in \bigcup(T\{\text{restricted to}A\}) \land y \in \bigcup(T\{\text{restricted to}A\}) \land x \neq y \rightarrow (\exists U \in T\{\text{restricted to}A\}. x \in U \land y \notin U \lor (y \in U \land x \notin U))\) unfolding RestrictedTo_def by auto
  then show thesis using isT2_def by auto
qed

corollary here_T2:
  shows isT2 \{is hereditary\} using T2_here IsHer_def by auto

lemma T0_here:
  assumes T\{is T_0\} \{A \in \text{Pow}(\bigcup T)\} shows \((T\{\text{restricted to}A\}\{is T_0\})\)
proof-
  from assms(2) have un:\bigcup(T\{\text{restricted to}A\})=A unfolding RestrictedTo_def by auto
  { fix x y
    assume x\in A y\in A \neq y
    with \(A \in \text{Pow}(\bigcup T)\) have x\in A y\notin A by auto
    then have \(\exists U \in T. (x \in U \land y \notin U) \lor (y \in U \land x \notin U)\) using assms(1) isT0_def by auto
    then obtain U where U \in T \(x \in U \land y \notin U\) \(y \in U \land x \notin U\) unfolding RestrictedTo_def by auto
    with \(x \in A \land y \notin A\) \((x \in A \land y \notin A) \land (y \in A \land x \notin A)\) unfolding RestrictedTo_def by auto
    then have \(\exists U \in T\{\text{restricted to}A\}. (x \in U \land y \notin U) \lor (y \in U \land x \notin U)\) unfolding RestrictedTo_def by auto
    then show thesis using isT0_def by auto
  }
  with un have \(\forall x y. x \in \bigcup(T\{\text{restricted to}A\}) \land y \in \bigcup(T\{\text{restricted to}A\}) \land x \neq y \rightarrow (\exists U \in T\{\text{restricted to}A\}. x \in U \land y \notin U \lor (y \in U \land x \notin U))\) unfolding RestrictedTo_def 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

65.3 Spectrum and anti-properties

The spectrum of a topological property is a class of sets such that all topolo-
gies 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 topo-
logical 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} _ 99)
    where Spec(K,P) ≡ ∀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 indiscrete 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 $(A \{\text{is in the spectrum of} \; isT4} \iff A \preceq 1$

**Proof**

assume $A \{\text{is in the spectrum of} \; isT4}$
then have $\forall T. ((T \{\text{is a topology} \} \land \bigcup T \approx A) \rightarrow (T \{\text{T}_4}\})$
using **Spec_def** by auto

\[
\begin{align*}
& \text{assume } A \neq 0 \\
& \text{then obtain } x \text{ where } x \in A \text{ by auto} \\
& \text{then have } x \in \bigcup \{A\} \text{ by auto} \\
& \text{moreover} \\
& \text{then have } \{A\} \{\text{is a filter on}\} \bigcup \{A\} \text{ using **IsFilter_def** by auto} \\
& \text{moreover} \\
& \text{then have } \left(\{A\} \cup \{0\}\right) \{\text{is a topology} \} \land \bigcup \left(\{A\} \cup \{0\}\right) = A \text{ using **top_of_filter**} \\
& \text{by auto} \\
& \text{then have } \top: \left(\{A\} \cup \{0\}\right) \{\text{is a topology} \} \cup \left(\{A\} \cup \{0\}\right) = A \text{ using **eqpoll_refl**} \\
& \text{by auto} \\
& \text{then have } \left(\{A\} \cup \{0\}\right) \{\text{is T}_4\} \text{ using **isT4_def** by auto} \\
& \text{ultimately have } \bigcup \{A\} = \{x\} \text{ using **filter_T2_imp_card1**[of \{A\}x]} \text{ by auto} \\
& \text{then have } A = \{x\} \text{ by auto} \\
& \text{then have } A \approx 1 \text{ using **singleton_eqpoll_1** by auto} \\
& \text{moreover} \\
& \text{have } A = 0 \rightarrow A = 0 \text{ by auto} \\
& \text{ultimately have } A \approx 1 \lor A = 0 \text{ by blast} \\
& \text{then show } A \preceq 1 \text{ using **empty_lepollI** eqpoll_imp_lepoll eq_lepoll_trans** by auto} \\
& \text{next} \\
& \text{assume } A \preceq 1 \\
& \text{have } A = 0 \lor A \neq 0 \text{ by auto} \\
& \text{then obtain } E \text{ where } A = 0 \lor E \in A \text{ by auto} \\
& \text{then have } A \approx 0 \lor E \in A \text{ by auto} \\
& \text{with } A \preceq 1 \text{ have } A \approx 0 \lor E = \{E\} \text{ using **lepoll_1_is_sing** by auto} \\
& \text{then have } A \approx 0 \lor E \approx 1 \text{ using **singleton_eqpoll_1** by auto} \\
& \text{fix } T \\
& \text{assume } A \{\text{T is a topology} \} \cup T = A \\
& \{ \\
& \text{assume } A \approx 0 \\
& \text{with } A \{\text{T is a topology} \} \text{ and empty: } \cup T = 0 \text{ using **eqpoll_trans**} \\
& \text{eqpoll_0_is_0** by auto} \\
& \text{then have } T \{\text{T}_2\} \text{ using **isT2_def** by auto} \\
& \text{then have } T \{\text{T}_1\} \text{ using **T2_is_T1** by auto} \\
& \text{moreover} \\
& \text{from empty have } T \subseteq \{0\} \text{ by auto} \\
& \end{align*}
\]
with AS(1) have $T=\{0\}$ using empty_open by auto
from empty have $\forall A. A$ is closed in $T$ $\rightarrow$ $A=0$ using IsClosed_def by auto
have $\exists U \in T. \exists V \in T. 0 \subseteq U \land 0 \subseteq V \land U \cap V=0$ using empty_open AS(1) by auto
then have $T$ is normal using IsNormal_def by auto
moreover
{
assume $A \approx 1$
with AS have $T$ is a topology} and NONempty $\cup T \approx 1$ using eqpoll_trans[of $\cup T A1$] by auto
then have $\cup T < 1$ using eqpoll_imp_lepoll by auto
moreover
{
assume $\cup T = 0$
then have $0 \approx \cup T$ by auto
with NONempty have $0 \approx 1$ 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 $\cup T \neq 0$ by auto
then obtain $R$ where $R \in \cup T$ by blast
ultimately have $\cup T = \{R\}$ using lepoll_1_is_sing by auto
{
fix $x$ $y$
assume $x$ is closed in $T$ $y$ is closed in $T$ $x \cap y = 0$
then have $x \subseteq \cup T \subseteq T$ using IsClosed_def by auto
then have $x=0 \lor y = 0$ using $x \subseteq \cup T \subseteq \{R\}$ by force
{
assume $x = 0$
then have $x \subseteq 0 \subseteq \cup T$ using $y \subseteq \cup T$ by auto
moreover
have $0 \in \cup T \subseteq T$ using AS(1) IsATopology_def empty_open by auto
ultimately have $\exists U \in T. \exists V \in T. x \subseteq U \land y \subseteq V \land U \cap V = 0$ by auto
}
moreover
{
assume $x \neq 0$
with $x = 0 \lor y = 0$ have $y = 0$ by auto
then have $x \subseteq \cup T \subseteq 0$ using $x \subseteq \cup T$ by auto
moreover
have $0 \in \cup T \subseteq T$ using AS(1) IsATopology_def empty_open by auto
ultimately have $\exists U \in T. \exists V \in T. x \subseteq U \land y \subseteq V \land U \cap V = 0$ by auto
}
ultimately

849
have \((\exists U \in T. \exists V \in T. x \subseteq U \land y \subseteq V \land U \cap V = 0)\) by blast
}
then have \(T\{\text{is normal}\}\) using IsNormal_def by auto
moreover
\{ 
  fix \ x \ y 
  assume \(x \in \bigcup T\ y \in \bigcup T x \neq y\)
  with \(\bigcup T = \{R\}\) have False by auto
  then have \(\exists U \in T. x \in U \land y \notin U\) by auto
\}
then have \(T\{\text{is } T_1\}\) using isT1_def by auto
ultimately have \(T\{\text{is } T_4\}\) using isT4_def by auto
\}
ultimately have \(T\{\text{is } T_4\}\) using \(<A \approx 0 \lor A \approx 1>\) by auto
\}
then have \(\forall T. (\text{T is a topology} \land \bigcup T \approx A) \longrightarrow (T\{\text{is } T_4\})\) by auto
then show \(A\) is in the spectrum of \(isT4\) using Spec_def by auto
qed

If the topological properties are related, then so are the spectra.

lemma \(\text{P_imp_Q_spec_inv}\):
  assumes \(\forall T. \text{T is a topology} \longrightarrow (Q(T) \longrightarrow P(T))\) \(A\) {is in the spectrum of} \(Q\)
  shows \(A\) {is in the spectrum of} \(P\)
proof-
  from assms(2) have \(\forall T. \text{T is a topology} \land \bigcup T \approx A \longrightarrow Q(T)\) using Spec_def by auto
  with assms(1) have \(\forall T. \text{T is a topology} \land \bigcup T \approx A \longrightarrow P(T)\) by auto
  then show thesis using Spec_def by auto
qed

Since we already now 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 \((A\) is in the spectrum of \(isT0) \iff A \approx 1\)
proof
  assume \(A\) {is in the spectrum of} \(isT0\)
  then have \(\text{reg:} \forall T. ((T\text{ is a topology} \land \bigcup T \approx A) \longrightarrow (T\{\text{is } T_0\}))\) 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 \(\{A\}\) 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
  \}
  \}

850
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 \(\langle \{A\} \cup \{0\} \rangle \) 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 \(\text{dis} : \langle y \in A \rangle \langle x \in \bigcup \{A\} \rangle\) have False by auto

then have \(\forall y \in A. y = x\) by auto

with \(\langle x \in \bigcup \{A\} \rangle\) have \(A = \{x\}\) by blast

then have \(A = 1\) using singleton_eqpoll_1 by auto

moreover
have \(A = 0\) \(\longrightarrow A = 0\) by auto
ultimately have \(A = 1\lor A = 0\) by blast
then show \(A \lesssim 1\) using empty_lepollI eqpoll_imp_lepoll eq_lepoll_trans by auto

next
assume \(A \lesssim 1\)

fix \(T\)
assume \(T\text{ is a topology}\)
then have \((T \text{ is } T_0) \longrightarrow (T \text{ is } T_0)\) using topology0.T4_is_T3 topology0.T3_is_T2 T2_is_T1 T1_is_T0 topology0_def by auto

then have \(\forall T. (T \text{ is a topology}) \longrightarrow ((T \text{ is } T_0) \longrightarrow (T \text{ is } T_0))\) by auto
then have \(A \{ \text{is in the spectrum of} \} \text{ is } T_4\) \(\longrightarrow A \{ \text{is in the spectrum of} \} \text{ is } T_0\)
using P_imp_Q_spec_inv[of \(\lambda T. (T \text{ is } T_0)\)] topology0_def by auto
then show \(A \{ \text{is in the spectrum of} \} \text{ is } T_0\) using T4_spectrum <\(A \lesssim 1\)> by auto

qed

theorem T1_spectrum:
shows \((A \{ \text{is in the spectrum of} \} \text{ is } T_1) \longleftrightarrow A \lesssim 1\)
proof-

note T2_is_T1 topology0.T3_is_T2 topology0.T4_is_T3
then have \(A \{ \text{is in the spectrum of} \} \text{ is } T_4\) \(\longrightarrow A \{ \text{is in the spectrum of} \} \text{ is } T_1\)
using P_imp_Q_spec_inv[of isT4isT1] topology0_def by auto
moreover
note T1_is_T0
then have \(A \{ \text{is in the spectrum of} \} \text{ is } T_1\) \(\longrightarrow A \{ \text{is in the spectrum of} \} \text{ is } T_0\)
using P_imp_Q_spec_inv[of isT1isT0] by auto

851
moreover note T0_spectrum T4_spectrum
ultimately show thesis by blast
qed

theorem T2_spectrum:
shows (A {is in the spectrum of} isT2) \iff A \subseteq 1
proof-
- note topology0.T3_is_T2 topology0.T4_is_T3
then have (A {is in the spectrum of} isT4) \longrightarrow (A {is in the spectrum of} isT2)
  using P_imp_Q_spec_inv[of isT4isT2] topology0_def by auto
moreover
- note T2_is_T1
then have (A {is in the spectrum of} isT2) \longrightarrow (A {is in the spectrum of} isT1)
  using P_imp_Q_spec_inv[of isT2isT1] by auto
moreover
- note T1_spectrum T4_spectrum
ultimately show thesis by blast
qed

theorem T3_spectrum:
shows (A {is in the spectrum of} isT3) \iff A \subseteq 1
proof-
- note topology0.T4_is_T3
then have (A {is in the spectrum of} isT4) \longrightarrow (A {is in the spectrum of} isT3)
  using P_imp_Q_spec_inv[of isT4isT3] topology0_def by auto
moreover
- note topology0.T3_is_T2
then have (A {is in the spectrum of} isT3) \longrightarrow (A {is in the spectrum of} 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 {is in the spectrum of} (\lambda T. (\bigcup T) {is compact in}T)) \iff Finite(A)
proof
- assume A {is in the spectrum of} (\lambda T. (\bigcup T) {is compact in}T)
then have reg:\forall T. T{is a topology} \land \bigcup T\approx A \longrightarrow ((\bigcup T) {is compact in}T) using Spec_def by auto
  have Pow(A){is a topology} \land \bigcup Pow(A)=A using Pow_is_top by auto
then have Pow(A){is a topology} \land \bigcup Pow(A)=A using eqpoll_refl by auto
with reg have A is compact in Pow(A) by auto
moreover
have \{\{x\}. x\in A\}\in Pow(Pow(A)) by auto
moreover
have \bigcup \{\{x\}. x\in A\}=A by auto
ultimately have \exists N\in FinPow(\{\{x\}. x\in A\}). A\subseteq N using IsCompact_def by auto
then obtain N where N\in FinPow(\{\{x\}. x\in A\}) A\subseteq N by auto
then have N\subseteq \{\{x\}. x\in A\} Finite(N) A\subseteq \bigcup N using FinPow_def by auto
\{ 
fix t
assume t\in \{\{x\}. x\in A\} 
then obtain x where x\in A=t by auto
with \langle A\cup N\rangle have x\in N by auto 
then obtain B where B\subseteq N by auto
with \langle N\subseteq \{\{x\}. x\in A\}\rangle have B={x} by auto
with \langle t=x\rangle \langle B\subseteq N\rangle have t\in N by auto 
\}
with \langle N\subseteq \{\{x\}. x\in A\}\rangle have N={\{x\}. x\in A} by auto
with \langle Finite(N)\rangle have 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: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 \langle Finite(\{\{x\}. x\in A\})\rangle show Finite(A) using eqpoll_imp_Finite_iff by auto
next
assume Finite(A)
\{ 
fix T assume T is a topology} \bigcup T=A
with \langle Finite(A)\rangle 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\subseteq Pow(T) \bigcup T\subseteq M
with \langle Finite(T)\rangle have Finite(M) using subset_Finite by auto
with \langle \bigcup T\subseteq M\rangle have \exists N\in FinPow(M). \bigcup T\subseteq N using FinPow_def by auto
\}
then have \bigcup T is compact in T unfolding IsCompact_def by auto
\}
then show A \{is in the spectrum of\} (\lambda T. \bigcup T 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) $K$ (choice holds for subsets) $(\text{Pow}(K))$ $\text{Card}(K)$
- **Shows**: ($A \in \text{spectrum of} (\lambda T. ((\bigcup T)\text{is compact of cardinal})$ $\text{csucc}(K)\{\text{in}\}T))$ $\iff (A \preceq K)$

**Proof**

- **Assume**: $A \in \text{spectrum of} (\lambda T. ((\bigcup T)\text{is compact of cardinal})$ $\text{csucc}(K)\{\text{in}\}T))$
- **Then have**: $\forall T. T\text{is a topology} \land \bigcup T \approx A \rightarrow ((\bigcup T)\text{is compact of cardinal})$ $\text{csucc}(K)\{\text{in}\}T)$ using $\text{Spec_def}$ by auto
- **Then have**: $A\text{is compact of cardinal} \ \text{csucc}(K)\{\text{in}\}A$ using $\text{Pow_is_top[of A]}$ by auto
- **Then have**: $\forall M \in \text{Pow}(\text{Pow}(A))$. $A \subseteq \bigcup M \rightarrow (\exists N \in \text{Pow}(M). A \subseteq \bigcup N \land N\prec \text{csucc}(K))$
- **Unfolding**: $\text{isCompactDefCard_def}$ by auto
  - **Moreover**: $A=\bigcup \{x\}. x \in A$ by auto
  - **Ultimately have**: $\exists N \in \text{Pow}(\{x\}. x \in A))$. $A \subseteq \bigcup N \land N\prec \text{csucc}(K)$ by auto
  - **Then obtain**: $N$ where $N \in \text{Pow}(\{x\}. x \in A)$ $A \subseteq \bigcup N \land N\prec \text{csucc}(K)$ by auto
  - **Then have**: $N\subseteq \{x\}. x \in A$ $N\prec \text{csucc}(K) A \subseteq \bigcup N$ using $\text{FinPow_def}$ by auto
  - **With**: $N\subseteq \{x\}. x \in A$ and $N\prec \text{csucc}(K)$ by auto
  - **Let**: $B=\{x\}. x \in A$
  - **From**: $N=\{x\}. x \in A)$ and $B=\{x\}$ by auto
  - **With**: $t=\{x\}$ and $B=\{x\}$ by auto
  - **Then have**: $A \preceq N$ using $\text{lepoll_def}$ by auto
  - **With**: $N \prec \text{csucc}(K)$ have $A \prec \text{csucc}(K)$ using $\text{lesspoll_trans1}$ by auto
  - **Then show**: $A \preceq K$ using $\text{Card_less_csucc_eq_le assms(2)}$ by auto

**Next**

- **Assume**: $A \preceq K$
  - **Fix**: $T$\{is a topology\}$\bigcup T \approx A$
  - **Have**: $\text{Pow}(\bigcup T)\{\text{is a topology}\}$ using $\text{Pow_is_top}$ by auto
  - **Fix**: $B$
    - **Assume**: $A:B \in \text{Pow}(\bigcup T)$
    - **Then have**: $\{\{i\}. i \in B\} \subseteq \{\{i\}. i \in \bigcup T\}$ by auto

854
moreover
have \( B = \bigcup \{ \{ i \}. \ i \in B \} \) by auto
ultimately have \( \exists S \in \text{Pow}(\bigcup \{ i \}. \ i \in T \}). \ B = \bigcup S \) by auto
then have \( B \in (\bigcup U. \ U \in \text{Pow}(\bigcup \{ i \}. \ i \in T \})) \) by auto

moreover
\[
\begin{align*}
\text{fix } \ & B \\
\text{assume } \ & \text{AS: } B \in (\bigcup U. \ U \in \text{Pow}(\bigcup \{ i \}. \ i \in T \)) \\
\text{then have } \ & B \in \text{Pow}(\bigcup T) \text{ by auto}
\end{align*}
\]
ultimately
have base: \( \{ x \}. \ x \in \bigcup T \} \) {is a base for}\( \text{Pow}(\bigcup T) \) unfolding \text{IsAbaseFor_def}
by auto
let \( f = \{(i, \{i\}). \ i \in \bigcup T\} \)
have \( f : f : (\bigcup T) \rightarrow \{ x \}. \ x \in \bigcup T \) using \text{Pi_def function_def} by auto
moreover
\[
\begin{align*}
\text{fix } \ & w \ x \\
\text{assume as: } \ & w \in \bigcup Tx \in \bigcup Tfw = fx \\
\text{with } f \text{ have } \ & fw = \{x\} \text{ using } \text{apply_equality} \text{ by auto} \\
\text{with } as(3) \text{ have } \ & w = x \text{ by auto}
\end{align*}
\]
with \( f \) have \( f : \text{inj}(\bigcup T, \{ x \}. \ x \in \bigcup T) \) unfolding \text{inj_def} by auto
moreover
\[
\begin{align*}
\text{fix } \ & xa \\
\text{assume xa: } \ & \{ x \}. \ x \in \bigcup T \\
\text{then obtain } \ & x \text{ where } x \in \bigcup Txa = x \text{ by auto} \\
\text{with } f \text{ have } \ & fx = xa \text{ using } \text{apply_equality} \text{ by auto} \\
\text{with } <x \in \bigcup T> \text{ have } \ & \exists x \in \bigcup T. \ fx = xa \text{ by auto}
\end{align*}
\]
then have \( \forall xa \in \{ x \}. \ x \in \bigcup T). \ \exists x \in \bigcup T. \ fx = xa \text{ by blast}
ultimately have \( f : \text{bij}(\bigcup T, \{ x \}. \ x \in \bigcup T) \) unfolding \text{bij_def surj_def} by auto
moreover
\[
\begin{align*}
\text{then have } \ & (\bigcup T) \approx \{ x \}. \ x \in \bigcup T) \text{ using } \text{eqpoll_def} \text{ by auto} \\
\text{then have } \ & \{ x \}. \ x \in \bigcup T) \approx (\bigcup T) \text{ using } \text{eqpoll_sym} \text{ by auto} \\
\text{with } <\bigcup T \approx A> \text{ have } \{ x \}. \ x \in \bigcup T) \approx A \text{ using } \text{eqpoll_trans} \text{ by blast} \\
\text{then have } \ & \{ x \}. \ x \in \bigcup T) \approx A \text{ using } \text{eqpoll_imp_lepoll} \text{ by auto} \\
\text{with } <A \leq K> \text{ have } \{ x \}. \ x \in \bigcup T) \leq K \text{ using } \text{lepoll_trans} \text{ by blast} \\
\text{then have } \ & \{ x \}. \ x \in \bigcup T) \text{ using } \text{assms(2) Card_less_csucc_eq_le} \\
\text{by auto}
\end{align*}
\]
with base have \( \text{Pow}(\bigcup T) \) {is of second type of cardinal}\( \text{csucc(K)} \) unfolding \text{IsSecondOfCard_def} by auto
moreover
\[
\begin{align*}
\text{have } \ & (\bigcup \text{Pow}(\bigcup T)) = \bigcup T \text{ by auto} \\
\text{with calculation } \text{assms(1) } <\text{Pow}(\bigcup T)\{\text{is a topology}> \text{ have } (\bigcup T) \text{ {is} compact of cardinal}\text{csucc(K)}\{\text{in}\text{Pow}(\bigcup T)} \\
\text{using } \text{compact_of_cardinal_Q[of KPow(\bigcup T)]} \text{ by auto}
\end{align*}
\]
moreover
  have \( \mathcal{T} \subseteq \text{Pow}(\bigcup \mathcal{T}) \) by auto
ultimately have \( (\bigcup \mathcal{T}) \) {is compact of cardinal}csucc(K){in}T using
  compact_coarser by auto
}

then show A {is in the spectrum of} \( (\lambda \mathcal{T}. ((\bigcup \mathcal{T}) \text{is compact of cardinal} \text{csucc}(K) \text{in} T)) \) using Spec_def by auto
  qed

theorem compactK_spectrum_reverse:
  assumes \( \forall A. (A \text{ in the spectrum of} (\lambda \mathcal{T}. ((\bigcup \mathcal{T}) \text{is compact of cardinal} \text{csucc}(K) \text{in} T))) \) ←→ \( A \lessapprox K \) InfCard(K)
  shows {the axiom of}K{choice holds for subsets}(Pow(K))
proof-
  have K\lessapprox K using lepoll_refl by auto
then have K {is in the spectrum of} \( (\lambda \mathcal{T}. ((\bigcup \mathcal{T}) \text{is compact of cardinal} \text{csucc}(K) \text{in} T)) \) using assms(1) by auto
moreover
  have \( \text{Pow}(K) \) {is a topology} using 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} csucc(K){in}Pow(K) using Spec_def by auto
then show thesis using Q_disc_comp_csuccQ_eq_Q_choice_csuccQ assms(2) by auto
  qed

This last theorem states that if one of the forms of the axiom of choice re-
lated 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
  antiProperty (_{is anti-})_ 50
  where T{is anti-}P \equiv \forall A\in\text{Pow}(\bigcup T). P(T\{\text{restricted to}\}A) \rightarrow (A \text{ in the spectrum of} P)

abbreviation
  ANTI(P) \equiv \lambda T. (T{is anti-})P

A first, very simple, but very useful result is the following: when the prop-
erties are related and the spectra are equal, then the anti-properties are
related in the opposite direction.

theorem (in topology0) eq_spect_rev_imp_anti:
assumes $\forall T. \text{T is a topology} \rightarrow P(T) \rightarrow Q(T)$ $\forall A. (A \text{ is in the spectrum of } Q) \rightarrow (A \text{ is in the spectrum of } P)$

and $T \text{ is anti-Q}$

shows $T \text{ is anti-P}$

proof-

fix $A$
assume $A \in \text{Pow}(\bigcup T) P(T\text{ restricted to } A)$
with assm(1) have $Q(T\text{ restricted to } A)$ using Top_1_L4 by auto
with assm(3) $\langle A \in \text{Pow}(\bigcup T) \rangle$ have $A \text{ is in the spectrum of } Q$ using antiProperty_def by auto

with assm(2) have $A \text{ is in the spectrum of } P$ by auto

} then show thesis using antiProperty_def by auto
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:**

assumes $\forall T. ((T \text{ is a topology}) \land P(T) \land Q(T)) \rightarrow ((\bigcup T \text{ is in the spectrum of } P))$

and $Q \text{ is hereditary}$

shows $\forall T. T \text{ is a topology} \rightarrow (Q(T) \rightarrow (T \text{ is anti-} P))$

proof

fix $T$

assume $T \text{ is a topology}$

\{ Assume $Q(T)$

\{ Assume $\neg (T \text{ is anti-} P)$

then obtain $A$ where $A \in \text{Pow}(\bigcup T) P(T\text{ restricted to } A) \neg (A \text{ is in the spectrum of } P)$

unfolding antiProperty_def by auto

from $\langle Q(T) \rangle \langle T \text{ is a topology} \rangle \langle A \in \text{Pow}(\bigcup T) \rangle$ assm(2) have $Q(T\text{ restricted to } A)$

unfolding IsHer_def by auto

moreover note $\langle P(T\text{ restricted to } A) \rangle$ assm(1)

moreover from $\langle T \text{ is a topology} \rangle$ have $(T\text{ restricted to } A) \text{ is a topology}$

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 $A \text{ is in the spectrum of } P$ by auto

with $\langle \neg (A \text{ is in the spectrum of } P) \rangle$ have False by auto

\}

857
then have \( T\{\text{is anti-}\}P \) by auto

} then have \( Q(T) \rightarrow (T\{\text{is anti-}\}P) \) by auto

} then show \( (T \{\text{is a topology}\}) \rightarrow (Q(T) \rightarrow (T\{\text{is anti-}\}P)) \) 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-}ANTI(P)

proof-

{ assume \( \neg(T\{\text{is anti-}\}ANTI(P)) \)
   then have \( \exists A \in \text{Pow}(\bigcup T). (\{\text{restricted to}\}A\{\text{is anti-}\}P) \land \neg(A\{\text{is in the spectrum of}\}ANTI(P)) \)

   unfolding antiProperty_def[of _ ANTI(P)] by auto

   then obtain A where A_def:A \in Pow(\bigcup T) \land \neg(A\{\text{is in the spectrum of}\}ANTI(P))

   by auto

   from \( \langle A\in\text{Pow}(\bigcup T) \rangle \) have tot:(\bigcup (\{\text{restricted to}\}A)=A unfolding RestrictedTo_def

   by auto

   from A_def have reg:P(\bigcup (\{\text{restricted to}\}A)) unfolding antiProperty_def by auto

   have \( \forall B\in\text{Pow}(A). (\{\text{restricted to}\}A\{\text{restricted to}\}B=\text{restricted to}\)B using subspace_of_subspace <\(A\in\text{Pow}(\bigcup T)\rangle \) by auto

   then have \( \forall B\in\text{Pow}(A). P(\text{restricted to}\)B \rightarrow (B\{\text{is in the spectrum of}\}P)

   using reg tot

   by force

   moreover

   have \( \forall B\in\text{Pow}(A). P(\text{restricted to}\)B) using assms <\(A\in\text{Pow}(\bigcup T)\rangle \) unfolding isHer_def using topologyAssum by blast

   ultimately have reg2:\(\forall B\in\text{Pow}(A). (B\{\text{is in the spectrum of}\}P) \)

   from \( \langle \neg(A\{\text{is in the spectrum of}\}ANTI(P)) \rangle \) have \( \exists T. T\{\text{is a topology}\} \)

   \( \land \bigcup T\approx A \land \neg(T\{\text{is anti-}\}P) \)

   unfolding Spec_def by auto

   then obtain S where S{is a topology} \( \bigcup S\approx A \land \neg(S\{\text{is anti-}\}P) \) by auto

   from \( \langle \neg(S\{\text{is anti-}\}P) \rangle \) have \( \exists B\in\text{Pow}(\bigcup S). P(S\{\text{restricted to}\)B \land

   \neg(B\{is in the spectrum of\}P) \)

   unfolding antiProperty_def by auto

   then obtain B where B_def:\(\neg(B\{is in the spectrum of\})P \) B\in\text{Pow}(\bigcup S)

   by auto

   then have B\subseteq\bigcup S using subset_imp_lepoll by auto

   with \( \langle \bigcup S\approx A \rangle \) have B\subseteq A using lepoll_eq_trans by auto

   then obtain f where f\in\text{inj}(B,A) unfolding lepoll_def by auto

   then have f\text{bij}(B,range(f)) using inj_bij_range by auto

   then have B\subseteq range(f) unfolding eqpoll_def by auto

   with B_def(1) have \( \neg(\text{range}(f)\{\text{is in the spectrum of}\}P) \)

using eqpoll_iff_spec

858
by auto 
moreover 
with \( f \in \text{inj}(B,A) \) have range(f) \( \subseteq \) A unfolding inj_def Pi_def by auto 
with reg2 have range(f) {is in the spectrum of} P by auto 
ultimately have False by auto 
} 
then show thesis by auto 
qed 

The anti-properties are always hereditary 

theorem anti_here: 
  shows ANTI(P) {is hereditary} 
proof- 
{ 
  fix T 
  assume T {is a topology} ANTI(P,T) 
  { 
    fix A 
    assume A \in\Pow(\bigcup T) 
    then have \( \bigcup (T{\text{restricted to}}A)=A \) unfolding RestrictedTo_def by auto 
  moreover 
  { 
    fix B 
    assume B \in\Pow(A)P((T{\text{restricted to}}A){\text{restricted to}}B) 
    with \( A \in\Pow(\bigcup T) \) have B \in\Pow(\bigcup T)P(T{\text{restricted to}}B) using subspace_of_subspace by auto 
    with \( ANTI(P,T) \) have B {is in the spectrum of} P unfolding antiProperty_def by auto 
  } 
  ultimately have \( \forall B \in\Pow(\bigcup (T{\text{restricted to}}A)). (P((T{\text{restricted to}}A){\text{restricted to}}B)) \rightarrow (B {\text{is in the spectrum of}} P) \) by auto 
  then have ANTI(P,(T{\text{restricted to}}A)) unfolding antiProperty_def by auto 
  } 
  then have \( \forall A \in\Pow(\bigcup T). ANTI(P,(T{\text{restricted to}}A)) \) by auto 
  } 
then show thesis using IsHer_def by auto 
qed 

corollary {in topology0} anti_imp_anti3: 
  assumes T {is anti-} P 
  shows T {is anti-} ANTI(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 {is anti-} isT0) \iff T={0,\bigcup T} 

859
proof
assume $T=\{0, \bigcup T\}$
{
  fix $A$
  assume $A \in \text{Pow}(\bigcup T)(\text{restricted to} A) \{\text{is } T_0\}$
  {
    fix $B$
    assume $B \in T(\text{restricted to} A)$
    then obtain $S$ where $S \subseteq T$ and $B = A \cap S$ unfolding RestrictedTo_def by auto
    with $T=\{0, \bigcup T\}$ have $S \in \bigcup T$ by auto
  }
moreover
  have $0 \in \bigcup T$ by auto
  with $T=\{0, \bigcup T\}$ have $0 \subseteq T$ 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
  assume $A \neq 0$
  then obtain $x$ where $x \in A$ by blast
  {
    fix $y$
    assume $y \in A \setminus \{x\}$
    with $\{0, A\}$ have $U$ where $U \subseteq \{0, A\}$ 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 $\{0, A\}$ have $\text{False}$ by auto
  }
then have $\forall y \in A. \ y = x$ by auto
  with $x \in A$ 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 $A \in \text{is in the spectrum of isT0}$ using T0_spectrum by auto
}
moreover
  assume $A = 0$
  then have $A \approx 0$ by auto

860
then have \( A \preceq 1 \) using empty_lepollI eq_lepoll_trans by auto
then have \( A \) is in the spectrum of \( isT0 \) using T0_spectrum by auto
}

ultimately have \( A \) is in the spectrum of \( isT0 \) using auto

then show \( T \) is anti- \( isT0 \) using antiProperty_def by auto
next
assume \( T \) is anti- \( isT0 \)
then have \( \forall A \in \text{Pow}(\bigcup T). (T \text{ restricted to } A) \) is \( T_0 \) −→ (\( A \) is in the spectrum of \( isT0 \)) using antiProperty_def by auto
then have \( \exists U \in (T \text{ restricted to } \{x,y\}). (b1 \in U \land b2 \not\in U) \lor (b2 \in U \land b1 \not\in U) \) by auto
moreover
from \( b1 \in \bigcup \{T \text{ restricted to } \{x,y\}\} \land b2 \in \bigcup \{T \text{ restricted to } \{x,y\}\} \lor b1 \neq b2 \) by auto
ultimately have \( b1 = x \lor b2 = y \lor (b1 = y \land b2 = x) \) by auto
with \( x \neq y \) have \( (b1 \in \{x\} \land b2 \not\in \{x\}) \lor (b2 \in \{x\} \land b1 \not\in \{x\}) \) by auto
moreover
from \( x \in \bigcup T-A \land x \in A \land \{x,y\} = T \) by auto
with \( A \) in \( (T \text{ restricted to } \{x,y\}) \) unfolding RestrictedTo_def by auto
ultimately have \( \exists U \in (T \text{ restricted to } \{x,y\}). (b1 \in U \land b2 \not\in U) \lor (b2 \in U \land b1 \not\in U) \) by auto
}
then have \( (T \text{ restricted to } \{x,y\}) \) is \( T_0 \) using isT0_def by auto
ultimately have \( \{x,y\} \preceq 1 \) using reg by auto
moreover
have \( x \in \{x,y\} \) by auto
ultimately have \( \{x,y\} = \{x\} \) using lepoll_1_is_sing[of \( \{x,y\} \times \) by auto
moreover
have \( y \in \{x,y\} \) by auto
ultimately have \( y \in \{x\} \) by auto
then have \( y = x \) by auto
with \( x \neq y \) have False by auto
}
then have \( T \subseteq \{0, \bigcup T\} \) by auto
moreover
from topSpaceAssum have \( 0 \in T \lor T \in T \) using IsATopology_def empty_open by auto

861
ultimately show $T=\{0, \bigcup T\}$ by auto
qed

lemma indiscrete_spectrum:
  shows $(A \{is in the spectrum of\}(\lambda T. T=\{0, \bigcup T\})) \iff A \leq 1$
proof
  assume $(A \{is in the spectrum of\}(\lambda T. T=\{0, \bigcup T\}))$
  then have reg: $\forall T. ((T\{is a topology\} \land \bigcup T \approx A) \implies T = \{0, \bigcup T\})$ using Spec_def by auto
  moreover
  have $\bigcup \text{Pow}(A) = A$ by auto
  then have $\bigcup \text{Pow}(A) \approx A$ by auto
  moreover
  have $\text{Pow}(A) \{is a topology\}$ using Pow_is_top by auto
  ultimately have $\mathcal{P} : \text{Pow}(A) = \{0, A\}$ 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 $\mathcal{P}$ have $\{x\} = A$ by auto
  then have $A \approx 1$ using singleton_eqpoll_1 by auto
  then have $A \leq 1$ using eqpoll_imp_lepoll by auto
  \}
  moreover
  \{
  assume $A = 0$
  then have $A \approx 0$ by auto
  then have $A \leq 1$ using empty_lepollI eq_lepoll_trans by auto
  \}
  ultimately show $A \leq 1$ by auto
next
assume $A \leq 1$
\{
  fix $T$
  assume $T\{is a topology\} \bigcup T \approx A$
  \{
  assume $A = 0$
  with $\langle \bigcup T \approx A \rangle$ have $\bigcup T \approx 0$ by auto
  then have $\bigcup T = 0$ using eqpoll_0_is_0 by auto
  then have $T \leq \{0\}$ by auto
  with $\langle T\{is a topology\} \rangle$ have $T = \{0\}$ using empty_open by auto
  then have $T = \{0, \bigcup T\}$ by auto
  \}
  moreover
  \{
  assume $A \neq 0$
  then obtain $E$ where $E \in A$ by blast
  with $\langle A \leq 1 \rangle$ have $A = \{E\}$ using lepoll_1_is_sing by auto
  then have $A \approx 1$ using singleton_eqpoll_1 by auto
  \}
  \}
with \( \bigcup T \approx A \) have \( \text{NONempty:} \bigcup T \approx 1 \) using eqpoll_trans by blast
then have \( \bigcup T \lesseqeq 1 \) using eqpoll_imp_lepoll by auto
moreover
\{
  assume \( \bigcup T = 0 \)
  then have \( 0 \approx \bigcup T \) by auto
  with \( \text{NONempty} \) have \( 0 \approx 1 \) 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 \( R \in \bigcup T \) by blast
ultimately have \( \bigcup T = \{R\} \) using lepoll_1_is_sing by auto
moreover
have \( T \subseteq \text{Pow}(\bigcup T) \) 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 \( 0 \in T \bigcup T \in T \) using IsATopology_def by auto
moreover
note \( \langle \bigcup T = \{R\} \rangle \)
ultimately have \( T = \{0,\bigcup T\} \) by auto
\}
ultimately have \( T = \{0,\bigcup T\} \) 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 \( \langle T \text{ is anti-} (\lambda T. T = \{0,\bigcup T\}) \rangle \Longleftrightarrow T \{is T_0\} \)
proof
  assume \( T \{is T_0\} \)
  \{
    fix \( A \)
    assume \( A \subseteq \text{Pow}(\bigcup T) T\{\text{restricted to}\}A = \{0,\bigcup (T\{\text{restricted to}\}A)\} \)
    then have \( \text{un:} \bigcup (T\{\text{restricted to}\}A) = A \{\text{restricted to}\}A = \{0, A\} \) using RestrictedTo_def by auto
    from \( \langle T \{is T_0\} \rangle \langle A \subseteq \text{Pow}(\bigcup T) \rangle \) have \( T\{\text{restricted to}\}A = \{is T_0\} \) using T0_here by auto
    \{
      assume \( A = 0 \)
      then have \( A \approx 0 \) by auto
      then have \( A \leq 1 \) using empty_lepollI eq_lepoll_trans 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 (T\{\text{restricted to}A\})E \in (T\{\text{restricted to}A\}) 
  by auto 
  with \ (T\{\text{restricted to}A\})\{\text{is } T_0\} \neq E \ have \ \exists U \in (T\{\text{restricted to}A\}) 
  unfolding isT0_def by blast 
  then obtain \( U \) where \( U \in (T\{\text{restricted to}A\}) (E \in U \land y \notin U) \lor (E \notin U \land y \in U) 
  by auto 
  with \ (T\{\text{restricted to}A\})\{\text{is } T_0\}\{\text{is } T_0\} \ y \neq E 
  have \ \exists U \in (\bigcup (T\{\text{restricted to}A\})) \ (E \in U \land y \notin U) \lor (E \notin U \land y \in U) 
  by auto 
\}

ultimately have \( A \subseteq 1 \) by auto 
then have \( A \) \( \{\text{is in the spectrum of}\} \{\lambda T. T=\{0, U\}\} \) using indiscrete_spectrum by auto 

then show \( T \{\text{is anti-}\} \{\lambda T. T=\{0, U\}\} \) unfolding antiProperty_def by auto 
next 
assume \( T \{\text{is anti-}\} \{\lambda T. T=\{0, U\}\} 
\rightarrow \ (A \{\text{is in the spectrum of}\} \{\lambda T. T=\{0, U\}\}) \) using antiProperty_def by auto 
then have \( \forall A \in \text{Pow} (\bigcup T). (T\{\text{restricted to}A\})\{\text{is } T_0\}\{\text{is } T_0\} = A \) unfolding RestrictedTo_def by auto 
moreover 
have \( \forall A \in \text{Pow} (\bigcup T). \bigcup (T\{\text{restricted to}A\})=A \) unfolding RestrictedTo_def by auto 
ultimately have reg: \( \forall A \in \text{Pow} (\bigcup T). (T\{\text{restricted to}A\})=\{0, A\} \rightarrow A \subseteq 1 \) by auto 

\{ 
  fix \ x y 
  assume \ x \in U Ty \in U Tx \neq y 
  \{ 
    assume \ \forall U \in T. (x \in U \land y \notin U) \lor (x \notin U \land y \in U) 
    then have \( T\{\text{restricted to}\}\{x,y\} \subseteq \{0, \{x,y\}\} \) unfolding RestrictedTo_def by auto 
  moreove 
from \( \forall x \in U Ty \in U \) have emp:0\(\in T\{x,y\}\cap 0=0 \) and tot: \( \{x,y\} \cap 0 \cap U T \{\in T\} \) using topSpaceAssum empty_open IsATopology_def by auto 
from emp have 0\(\in T\{\text{restricted to}\}\{x,y\}\) unfolding RestrictedTo_def by auto 

864
moreover
from tot have \(\{x,y\} \in T\{\text{restricted to}\}\{x,y\}\) unfolding RestrictedTo_def
by auto
ultimately have \(T\{\text{restricted to}\}\{x,y\}\{x,y\} \leq 1\) 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 \(y = x\) by auto
then have \(\{x,y\} = \{x\}\) using lepoll_1_is_sing[of \(\{x,y\}\) x] by auto
moreover
have \(y \in \{x\}\) by auto
ultimately have \(y \in \{x\}\) by auto
then have \(y = x\) by auto
then have \(\{x,y\} = \{x\}\) using lepoll_1_is_sing[of \(\{x,y\}\) x] by auto
moreover
have \(x \in \{x\}\) by auto
ultimately have \(x \in \{x\}\) by auto
then have \(x = y\) by auto
then have \(\{x,y\} = \{x\}\) unfolding \(<x\neq y>\) by auto
}
then have \(\exists U \in T. (x \notin U \land y \notin U) \lor (x \in U \land y \in U)\)
by auto
then show \(T\{\text{is } T_0}\) using isT0_def by auto
qed

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 \((T\{\text{is anti-}\} \iff T_0) \iff (\text{IsLinOrder}(T,\langle U,V \rangle \in \text{Pow}(\bigcup T) \times \text{Pow}(\bigcup T). U \subseteq V))\)
proof
assume \(T\{\text{is anti-}\} = T_0\)
let \(r = \langle U,V \rangle \in \text{Pow}(\bigcup T) \times \text{Pow}(\bigcup T). U \subseteq V\)
have antisym \((r)\) unfolding antisym_def by auto
moreover
have trans \((r)\) unfolding trans_def by auto
moreover
{ fix A B
  assume A \(\in\) TB \(\in\) T
  { assume \(\neg(\{A \subseteq B \lor B \subseteq A\})\)
    then have \(A \cap B \neq \emptyset \neq A \cap B\) by auto
    then obtain \(x y\) where \(x \in A, y \in B, y \notin A, x \neq y\) by blast
    then have \(\{x,y\} \cap A = \{x\}, \{x,y\} \cap B = \{y\}\) by auto
    moreover
    from \(<A \in T, \neq B \in T>\) have \(\{x,y\} \cap A \in T\{\text{restricted to}\}\{x,y\}\{x,y\} \cap B \in T\{\text{restricted to}\}\{x,y\}\) unfolding
    RestrictedTo_def by auto
  }
}

865
ultimately have open_set:{x}∈T{restricted to}{x,y}{y}∈T{restricted to}{x,y} by auto
    have x∈∪Ty∈∪T using <A∈T→B∈T>→<x∈A→<y∈B> by auto
    then have sub:{x,y}∈Pow(∪T) by auto
    then have tot:∪(T{restricted to}{x,y})={x,y} unfolding RestrictedTo_def by auto
    { fix s t
      assume s∈∪(T{restricted to}{x,y})t∈∪(T{restricted to}{x,y})s≠t
      with tot have s∈{x,y}t∈{x,y}s≠t by auto
      then have (s=x∧t=y)∨(s=y∧t=x) by auto
      with open_set have ∃U∈(T{restricted to}{x,y}). s∈U∧t∉U using <x≠y> by auto
    }
    then have (T{restricted to}{x,y}){is T_1} unfolding isT1_def by auto
    with sub <T{is anti-}isT1> tot have {x,y} {is in the spectrum of}isT1 using antiProperty_def
    by auto
    then have (x,y)≤1 using T1_spectrum by auto
    moreover have x∈{x,y} by auto
    ultimately have x={x,y} using lepoll_1_is_sing[of {x,y}x] by auto
    moreover have y∈{x,y} by auto
    ultimately have y∈{x} by auto
    then have x=y by auto
    then have False using <x∈A¬y∉A> by auto
    }
    then have A⊆B∨B⊆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)∈Pow(∪T)×Pow(∪T). U⊆V})
    then have ordTot:∀S∈T. ∀B∈T. S⊆BVB⊆S unfolding IsLinOrder_def IsTotal_def by auto
    {
      fix A
      assume A∈Pow(∪T) and T1:(T{restricted to}A) {is T_1}
      then have tot:∪(T{restricted to}A)=A unfolding RestrictedTo_def by auto
      {
        fix U V
        assume U∈T{restricted to}AV∈T{restricted to}A
        then obtain AU AV where AU∈TAV∈TU=A∩AV=A∩AV unfolding RestrictedTo_def by auto
    }

866
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

\begin{align*}
& \text{assume } A = 0 \\
& \text{then have } A \approx 0 \text{ by auto} \\
& \text{moreover} \\
& \text{have } 0 \less 1 \text{ using empty_lepollI by auto} \\
& \text{ultimately have } A \less 1 \text{ using eq_lepoll_trans by auto} \\
& \text{then have } A \{\text{is in the spectrum of}\} isT1 \text{ using T1_spectrum by auto} \\
& \text{moreover} \\
& \text{assume } A \neq 0 \\
& \text{then obtain } t \text{ where } t \in A \text{ by blast} \\
& \{ \\
& \text{fix } y \\
& \text{assume } y \in A \neq t \\
& \text{with } <t \in A> \text{ tot } T1 \text{ obtain } U \text{ where } U \in (T_{\text{restricted to}A})y \in Ut \notin U \\
& \text{unfolding isT1_def} \\
& \text{by auto} \\
& \text{from } <y \neq t> \text{ have } t \neq y \text{ by auto} \\
& \text{with } <y \in A> <t \in A> \text{ tot } T1 \text{ obtain } V \text{ where } V \in (T_{\text{restricted to}A})t \in V \notin V \\
& \text{unfolding isT1_def} \\
& \text{by auto} \\
& \text{with } <y \in U> <t \notin U> \text{ have } \neg (U \subseteq V \lor V \subseteq U) \text{ by auto} \\
& \text{with ordTotSub } <U \subseteq (T_{\text{restricted to}A})> <V \subseteq (T_{\text{restricted to}A})> \text{ have False by auto} \\
& \text{then have } \forall y \in A. y = t \text{ by auto} \\
& \text{with } <t \in A> \text{ have } A = \{t\} \text{ by blast} \\
& \text{then have } A \approx 1 \text{ using singleton_eqpoll_1 by auto} \\
& \text{then have } A \less 1 \text{ using eqpoll_imp_lepoll by auto} \\
& \text{then have } A \{\text{is in the spectrum of}\} isT1 \text{ using T1_spectrum by auto} \\
& \text{ultimately} \\
& \text{have } A \{\text{is in the spectrum of}\} isT1 \text{ by auto} \\
& \text{then show } T \{\text{is anti-}\} isT1 \text{ using antiProperty_def by auto} \\
\end{align*}

qed

**corollary linordtop_here:**

shows $\forall T. \text{IsLinOrder}(T, \{\langle U, V \rangle \in \text{Pow}(\bigcup T) \times \text{Pow}(\bigcup T). U \subseteq V\}) \{\text{is hereditary}\}$

using anti_T1 anti_here[of isT1] by auto

**theorem (in topology0) anti_T4:**

shows $(T \{\text{is anti-}\} isT4) \iff (\text{IsLinOrder}(T, \{\langle U, V \rangle \in \text{Pow}(\bigcup T) \times \text{Pow}(\bigcup T). U \subseteq V\}))$
proof
assume T{is anti-}isT4
let r={(⟨U,V⟩∈Pow(⋃T)×Pow(⋃T). U⊆V)
have antisym(r) unfolding antisym_def by auto
moreover
have trans(r) unfolding trans_def by auto
moreover
{fix A B
assume A∈TB∈T
{assume ¬(A⊆B∨B⊆A)
then have A-B≠0B-A≠0 by auto
then obtain x y where x∈Ax≠By∈Bx≠y by blast
then have (x,y)∩A={x}(x,y)∩B={y} by auto
moreover
from A∈TB∈T have {x,y}∩A∈T{x,y}∩B∈T{x,y} unfolding RestrictedTo_def by auto
ultimately have open_set:{x}∈T{x,y}∈T{x,y} unfolding RestrictedTo_def by auto
usually have open_set:{x}∈T{x,y}∈T{x,y} by auto
have x∈⋃y∈⋃T using A∈TB∈T{x}∈T{x,y} by auto
then have sub:{x,y}∈Pow(⋃T) by auto
then have tot:{x{restricted to}{x,y}∈T{restricted to}{x,y}}={x,y} unfolding RestrictedTo_def by auto
by auto
{fix s t
assume s∈⋃{T{restricted to}{x,y}}t∈⋃{T{restricted to}{x,y}}s≠t
with tot have s∈{x,y}t∈{x,y}s≠t by auto
then have (s=x∧t=y)∨(s=y∧t=x) by auto
with open_set have ∃U∈{T{restricted to}{x,y}}. s∈U∧t∈U using x≠y by auto
}then have {T{restricted to}{x,y}}{is T_1} unfolding isT1_def by auto
moreover
{fix s
assume AS:s{is closed in}{T{restricted to}{x,y}}
{fix t
assume AS2:t{is closed in}{T{restricted to}{x,y}}s∩t=0
have {T{restricted to}{x,y}}{is a topology} using Top_1_L4 by auto
with tot have 0∈{T{restricted to}{x,y}}{x,y}∈{T{restricted to}{x,y}} using empty_open
union_open[where A=T{restricted to}{x,y}] by auto
moreover
note open_set

868
moreover have \( T_{\text{restricted to}}\{x,y\} \subseteq \text{Pow}(\bigcup( T_{\text{restricted to}}\{x,y\})) \) by blast

with \textit{tot} have \( T_{\text{restricted to}}\{x,y\} \subseteq \text{Pow}\{x,y\} \) by auto
ultimately have \( T_{\text{restricted to}}\{x,y\} = \{0,\{x\},\{y\},\{x,y\}\} \) by blast
moreover have \( \{0,\{x\},\{y\},\{x,y\}\} = \text{Pow}\{x,y\} \) by blast
ultimately have \( P: T_{\text{restricted to}}\{x,y\} = \text{Pow}\{x,y\} \) by simp

with \textit{P} have \( A \subseteq \text{Pow}\{x,y\} \). \( A \subseteq \{x, y\} \land \{x, y\} - A \in \text{Pow}\{x, y\} \) using \textit{IsClosed_def}
by simp

with \textit{P} have \( s, t \in \text{Pow}\{x,y\} \). \( s \subseteq T_{\text{restricted to}}\{x,y\} \) by auto
then have \( \exists U \in ( T_{\text{restricted to}}\{x,y\} ). \exists V \in ( T_{\text{restricted to}}\{x,y\} ). s \subseteq U \land t \subseteq V \land U \cap V = 0 \) by auto
then have \( \forall t. t_{\text{closed in}}( T_{\text{restricted to}}\{x,y\} ) \land s \cap t = 0 \rightarrow ( \exists U \in ( T_{\text{restricted to}}\{x,y\} ). \exists V \in ( T_{\text{restricted to}}\{x,y\} ). s \subseteq U \land t \subseteq V \land U \cap V = 0 ) \) by auto
ultimately have \( ( T_{\text{restricted to}}\{x,y\} )_{\text{is normal}} \) using \textit{IsNormal_def} by auto
with \textit{sub} \( T_{\text{is anti}} \equiv \text{isT4} \) tot have \( \{x,y\} \) is in the spectrum of \( \text{isT4} \) using \textit{antiProperty_def} by auto
then have \( \{x,y\} \preceq 1 \) using \textit{T4_spectrum} by auto
moreover have \( x \in \{x,y\} \) by auto
ultimately have \( \{x\} = \{x,y\} \) using \textit{lepoll_1_is_sing[of \{x,y\}] isT4_def} 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 \textit{<x\in A>-<y\notin A>} by auto
then have $A \subseteq B \cup B \subseteq A$ by auto

next
assume $\text{IsLinOrder}(T, \{(U,V) \in \text{Pow}(\bigcup T) \times \text{Pow}(\bigcup T) . U \subseteq V\})$
then have $T\{\text{is anti-}\} \text{isT1}$ using $\text{anti_T1}$ by auto
moreover
have $\forall T. T\{\text{is a topology}\} \rightarrow (T\{\text{T}_4\}) \rightarrow (T\{\text{T}_1\})$ using $\text{topology0.T4_is_T3 T3_is_T2 T2_is_T1 topology0_def}$ by auto
moreover
have $\forall A. (A \{\text{is in the spectrum of}\} \text{isT1}) \rightarrow (A \{\text{is in the spectrum of}\} \text{isT4})$ using $\text{T1_spectrum T4_spectrum}$ by auto
ultimately show $T\{\text{is anti-}\} \text{isT4}$ using $\text{eq_spect_rev_imp_anti[of isT4 isT1]}$ by auto
qed

theorem (in topology0) $\text{anti_T3}$:
shows $(T\{\text{is anti-}\} \text{isT3}) \iff (\text{IsLinOrder}(T, \{(U,V) \in \text{Pow}(\bigcup T) \times \text{Pow}(\bigcup T) . U \subseteq V\}))$
proof
assume $T\{\text{is anti-}\} \text{isT3}$
moreover
have $\forall T. T\{\text{is a topology}\} \rightarrow (T\{\text{T}_4\}) \rightarrow (T\{\text{T}_3\})$ using $\text{topology0.T4_is_T3 T3_is_T2 T2_is_T1 topology0_def}$ by auto
moreover
have $\forall A. (A \{\text{is in the spectrum of}\} \text{isT3}) \rightarrow (A \{\text{is in the spectrum of}\} \text{isT4})$ using $\text{T3_spectrum T4_spectrum}$ by auto
ultimately have $T\{\text{is anti-}\} \text{isT4}$ using $\text{eq_spect_rev_imp_anti[of isT4 isT3]}$ by auto
then show $\text{IsLinOrder}(T, \{(U,V) \in \text{Pow}(\bigcup T) \times \text{Pow}(\bigcup T) . U \subseteq V\})$ using $\text{anti_T4}$ by auto
next
assume $\text{IsLinOrder}(T, \{(U,V) \in \text{Pow}(\bigcup T) \times \text{Pow}(\bigcup T) . U \subseteq V\})$
then have $T\{\text{is anti-}\} \text{isT1}$ using $\text{anti_T1}$ by auto
moreover
have $\forall T. T\{\text{is a topology}\} \rightarrow (T\{\text{T}_4\}) \rightarrow (T\{\text{T}_3\})$ using $\text{topology0.T4_is_T3 T3_is_T2 T2_is_T1 topology0_def}$ by auto
moreover
have $\forall A. (A \{\text{is in the spectrum of}\} \text{isT3}) \rightarrow (A \{\text{is in the spectrum of}\} \text{isT4})$ using $\text{T3_spectrum T4_spectrum}$ by auto
ultimately show $T\{\text{is anti-}\} \text{isT3}$ using $\text{eq_spect_rev_imp_anti[of isT3 isT1]}$
by auto
qed

thm (in topology0) anti_T2:
shows (T is anti-1 T2) \iff (IsLinOrder(T,\{(U,V) \in Pow(\bigcup T) \times Pow(\bigcup T). U \subseteq V\}))
proof
  assume T is anti-1 T2
  moreover
  have \( \forall T. \ T \text{ a topology} \longrightarrow (T \text{ is } T_4) \longrightarrow (T \text{ is } T_2) \) using topology0.T4_is_T3
    topology0.T3_is_T2 topology0_def by auto
  moreover
  have \( \forall A. \ (A \text{ is in the spectrum of } isT2) \longrightarrow (A \text{ is in the spectrum of } isT4) \) using T2_spectrum T4_spectrum
    by auto
  ultimately have T is anti-1 isT4 using eq_spect_rev_imp_anti[of isT4 isT2]
    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 reg: \( \forall T. \ T \text{ a topology} \longrightarrow (T \text{ is } T_1) \longrightarrow (T \text{ is } T_2) \) 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 is anti-1 isT2 using eq_spect_rev_imp_anti[of isT2 isT1]
    by auto
qed

lem linord_spectrum:
shows \( A \text{ is in the spectrum of } (\lambda T. \ IsLinOrder(T,\{(U,V) \in Pow(\bigcup T) \times Pow(\bigcup T). U \subseteq V\})) \iff A \leq 1 \)
proof
  assume A is in the spectrum of \( (\lambda T. \ IsLinOrder(T,\{(U,V) \in Pow(\bigcup T) \times Pow(\bigcup T). U \subseteq V\})) \)
  then have reg: \( \forall T. \ T \text{ a topology} \land \bigcup T \approx A \longrightarrow IsLinOrder(T,\{(U,V) \in Pow(\bigcup T) \times Pow(\bigcup T). U \subseteq V\}) \)
    using Spec_def by auto
  \{ assume A=0
    moreover
    have 0 \leq 1 using empty_lepollI by auto
    ultimately have A \leq 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 \( \text{Pow}(A) \) \{is a topology\} using \( \text{Pow_is_top} \) by auto
    moreover
    have \( \bigcup \text{Pow}(A) = A \) by auto
    then have \( \bigcup \text{Pow}(A) \approx A \) by auto
    note reg
    ultimately have \( \text{IsLinOrder}(\text{Pow}(A), \{ \langle \bigcup \text{Pow}(A) \times \bigcup \text{Pow}(A) \rangle. U \subseteq V \}) \)
    by auto
    then have \( \text{IsLinOrder}(\text{Pow}(A), \{ \langle \bigcup \text{Pow}(A) \times \bigcup \text{Pow}(A) \rangle. U \subseteq V \}) \) by auto
    with \( \langle x \in A \rangle < y \in A \rangle \) have \( \{x\} \subseteq \{y\} \lor \{y\} \subseteq \{x\} \) unfolding \( \text{IsLinOrder_def} \)
    \( \text{IsTotal_def} \) by auto
    then have \( x = y \) by auto
  }
  ultimately have \( A = \{x\} \) by blast
  then have \( A \approx 1 \) using \( \text{singleton_eqpoll_1} \) by auto
  then have \( A \subseteq 1 \) using \( \text{eqpoll_imp_lepoll} \) by auto
}
ultimately show \( A \subseteq 1 \) by auto
next

assume \( A \subseteq 1 \)
then have \( \text{ind}:A\{\text{is in the spectrum of}\}(\lambda T. T = \{0, U T\}) \) using \( \text{indiscrete_spectrum} \)
by auto
{
  fix \( T \)
  assume \( \text{AS:T} \{\text{is a topology}\} T = \{0, U T\} \)
  have \( \text{trans} \{\langle \bigcup \text{Pow}(U T) \times \bigcup \text{Pow}(U T) \rangle. U \subseteq V \} \) unfolding \( \text{trans_def} \) by auto
  moreover
  have \( \text{antisym} \{\langle \bigcup \text{Pow}(U T) \times \bigcup \text{Pow}(U T) \rangle. U \subseteq V \} \) unfolding \( \text{antisym_def} \)
  by auto
  moreover
  have \( \{ \langle U, V \rangle \in \text{Pow}(U T) \times \text{Pow}(U T). U \subseteq V \} \) is total on \( T \)
  proof-
  {
    fix \( aa \) \( b \)
    assume \( aa \in T b \in T \)
    with \( \text{AS}(2) \) have \( aa \in \{0, U T\} b \in \{0, U T\} \) by auto
    then have \( aa = 0 \lor aa = 0 \lor b = U T \) by auto
    then have \( aa \subseteq b \lor b \subseteq aa \) by auto
    then have \( \langle aa, b \rangle \in \text{Collect}(\text{Pow}(U T) \times \text{Pow}(U T), \text{split}(\subseteq)) \)
    \lor \( \langle b, aa \rangle \in \text{Collect}(\text{Pow}(U T) \times \text{Pow}(U T), \text{split}(\subseteq)) \)
    using \( \langle aa \in T \rangle < b \in T \rangle \) by auto
  }
}
}

872
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}(λ T. 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}(λ T. IsLinOrder(T, {{U,V}∈Pow(∪T)×Pow(∪T). U⊆V})) unfolding IsLinOrder_def by auto
    qed

theorem (in topology0) anti_linord:
says T{is anti-}(λ T. IsLinOrder(T, {{U,V} ∈ Pow(∪T)×Pow(∪T) . U ⊆ V})) ↔ T{is T_1}

proof
assume AS:T{is anti-}(λ T. IsLinOrder(T, {{U,V} ∈ Pow(∪T)×Pow(∪T) . U ⊆ V}))
{ assume ~(T{is T_1})
    then obtain x y where x∈∪T y∈∪T x≠y∀ U∈T. x∉U y∈U unfolding isT1_def by auto
    { assume {x}∈T{restricted to}{x,y}
        then obtain U where U∈T {x}={x,y}∩∪U unfolding RestrictedTo_def by auto
        moreover
        have x∈{x} by auto
        ultimately have U∈T∈U by auto
        moreover
        { assume y∈U
            then have y∈{x,y}∩∪U by auto
            with ⟨x⟩={x,y}∩∪U have y∈{x} by auto
            with ⟨x≠y⟩ have False by auto
        }
    }
    then have y∉U by auto
    moreover
    note ⟨U∈T. x∉U y∈U⟩
    ultimately have False by auto
    }
    then have {x}∉T{restricted to}{x,y} by auto
    moreover
    have tot:{∪T{restricted to}{x,y}}={x,y} using ⟨x∈∪T y∈∪T⟩ unfolding RestrictedTo_def by auto
    moreover
    have T{restricted to}{x,y}⊆Pow(∪T{restricted to}{x,y})) by auto

873
ultimately have $T_{\text{restricted to}} \{x, y\} \subseteq \text{Pow}\{x, y\} - \{x\}$ by auto
moreover
have $\text{Pow}\{x, y\} = \{0, \{x\}, \{x, y\}\}$ by blast
ultimately have $T_{\text{restricted to}} \{x, y\} \subseteq \{0, \{x\}, \{y\}\}$ by auto
moreover
have $\text{IsLinOrder}\{0, \{x\}, \{y\}\}, \langle U, V \rangle \in \text{Pow}\{x, y\} \times \text{Pow}\{x, y\}$. $U \subseteq V$) proof
have antisym(Collect($\text{Pow}\{x, y\} \times \text{Pow}\{x, y\}$, split($\subseteq$))) using antisym_def by auto
moreover
have trans(Collect($\text{Pow}\{x, y\} \times \text{Pow}\{x, y\}$, split($\subseteq$))) using trans_def by auto
moreover
have $\text{Collect}(\text{Pow}\{x, y\} \times \text{Pow}\{x, y\})$ is total on $\{0, \{x\}, \{x, y\}\}$ using IsTotal_def by auto
ultimately show $\text{IsLinOrder}\{0, \{x\}, \{y\}\}, \langle U, V \rangle \in \text{Pow}\{x, y\} \times \text{Pow}\{x, y\}$. $U \subseteq V$) using IsLinOrder_def by auto
qed
ultimately have $\text{IsLinOrder}(T_{\text{restricted to}} \{x, y\}, \langle U, V \rangle \in \bigcup(T_{\text{restricted to}} \{x, y\}) \times \text{Pow}(\bigcup(T_{\text{restricted to}} \{x, y\}))$. $U \subseteq V$) using ord_linear_subset by auto
then have $\text{IsLinOrder}(T_{\text{restricted to}} \{x, y\}, \text{Collect}(\bigcup(T_{\text{restricted to}} \{x, y\}) \times \text{Pow}(\bigcup(T_{\text{restricted to}} \{x, y\})))$ by auto
moreover
from $\langle x \in \bigcup T, y \in \bigcup T \rangle$ have $\{x, y\} \in \text{Pow}(\bigcup T)$ by auto
moreover
note AS
ultimately have $\{x, y\}$ is in the spectrum of $\lambda T. \text{IsLinOrder}(T, \langle U, V \rangle \in \text{Pow}(\bigcup T) \times \text{Pow}(\bigcup T)$. $U \subseteq V$) unfolding antiProperty_def by simp
then have $\{x, y\} \leq 1$ using linord_spectrum by auto
moreover
have $x \in \{x, y\}$ by auto
ultimately have $\{x\} = \{x, y\}$ using lepoll_1_is_sing[of $\{x, y\} \times \{x\}$] 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 False using $\langle x \neq y \rangle$ by auto
}
then show $T$ is $T_1$ by auto
next
assume $T_1 : T$ is $T_1$
{
fix A
assume \( A\text{\_def}:A \in \text{Pow}\left(\bigcup T\right)\) IsLinOrder\((T\text{\_restricted to}A),\{(U,V) \in \text{Pow}\left(\bigcup (T\text{\_restricted to}A)\right)\times\text{Pow}\left(\bigcup (T\text{\_restricted to}A)\right)\mid U \subseteq V\}\)\)

\[
\begin{align*}
\text{fix } x \\
\text{assume AS1}:x \in A \\
\text{fix } y \\
\text{assume AS}:y \in A \\
\text{with AS1} \text{ have } \{x,y\} \in \text{Pow}\left(\bigcup T\right) \\
\text{using } A\text{\_def}(1) \text{ linordtop}\text{\_here} \\
\text{ultimately have } U \text{ where } x \in U \\
\text{using } A\text{\_def}(2) \text{ linordtop}\text{\_here} \\
\text{ultimately obtain } V \text{ where } y \in V \\
\text{unfolding } isT1\text{\_def} \text{ by auto} \\
\text{moreover} \\
\text{from AS(2) tot T11 obtain } W \\
\text{ultimately have } x \in W \\
\text{unfolding } isT1\text{\_def} \text{ by auto} \\
\text{moreover} \\
\text{from AS(2) tot T11 obtain } V \\
\text{ultimately have } x \in V \\
\text{unfolding } isT1\text{\_def} \text{ by auto} \\
\text{moreover} \\
\text{from AS(2) tot T11 obtain } U \\
\text{ultimately have } x \in U \\
\text{unfolding } isT1\text{\_def} \text{ by auto} \\
\text{moreover} \\
\text{have } (T\text{\_restricted to}A) \text{ is a topology} \\
\text{using } \text{Top}\_1\_L4 \text{ by auto} \\
\text{moreover} \\
\text{note } A\text{\_def}(2) \text{ linordtop}\text{\_here} \\
\text{ultimately have } V \in \text{Pow}\left(\bigcup (T\text{\_restricted to}A)\right) \text{ IsLinOrder}\((T\text{\_restricted to}A)\{\text{restricted to} A\} \text{\_restricted to} B,B\text{\_restricted to} B) ,\{(U,V) \in \text{Pow}\left(\bigcup ((T\text{\_restricted to}A)\{\text{restricted to} A\} \text{\_restricted to} B)\right)\times\text{Pow}\left(\bigcup ((T\text{\_restricted to}A)\{\text{restricted to} A\} \text{\_restricted to} B)\right)\mid U \subseteq V\}\) \\
\text{unfolding } isHer\text{\_def} \text{ by auto} \\
\text{moreover} \\
\text{have tot: } \bigcup (T\text{\_restricted to}A)=A \\
\text{unfolding }RestrictedTo\_def \\
\text{using } A\in \text{Pow}\left(\bigcup T\right) \text{ by auto} \\
\text{ultimately have } V \in \text{Pow}(A) \text{ IsLinOrder}\((T\text{\_restricted to}A)\{\text{restricted to} A\} \text{\_restricted to} B,B\text{\_restricted to} B) ,\{(U,V) \in \text{Pow}\left(\bigcup ((T\text{\_restricted to}A)\{\text{restricted to} A\} \text{\_restricted to} B)\right)\times\text{Pow}\left(\bigcup ((T\text{\_restricted to}A)\{\text{restricted to} A\} \text{\_restricted to} B)\right)\mid U \subseteq V\}\) \\
\text{by auto} \\
\text{moreover}
have ∀B∈Pow(A). (T\{restricted to}A){restricted to}B=T\{restricted
to}B using subspace_of_subspace \langle A∈Pow(∪T) \rangle by auto

ultimately

have ∀B∈Pow(A). IsLinOrder((T\{restricted to}B) ,\{(U,V)∈Pow(∪((T\{restricted
to}A){restricted to}B))×Pow((∪((T\{restricted to}A){restricted to}B))\} by auto
U⊆V\rangle)

moreover

have ∀B∈Pow(A). \bigcup((T\{restricted to}A){restricted to}B)=B using
\langle A∈Pow(∪T) \rangle unfolding RestrictedTo_def by auto

ultimately have ∀B∈Pow(A). IsLinOrder((T\{restricted to}B) ,\{⟨U,V⟩∈Pow(B)×Pow(B) U⊆V⟩\} by auto

with \{x,y\}∈Pow(A)> have IsLinOrder((T\{restricted to}\{x,y\})
,{⟨U,V⟩∈Pow({x,y})×Pow({x,y}) U⊆V}) by auto

} ultimately have False using tot by auto

then have A={x} using AS1 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\(\{\text{is in the spectrum of}\}(λT. IsLinOrder(T, {⟨U,V⟩∈Pow(∪T)×Pow(∪T).\}
U⊆V\rangle))\) using linord_spectrum
by auto

} moreover
{
assume A=0
then have A≈0 by auto
moreover
have 0≤1 using empty_lepollI by auto
ultimately have A≤1 using eqpoll_imp_lepoll by auto
then have A\{\text{is in the spectrum of}\}(λT. IsLinOrder(T, {⟨U,V⟩∈Pow(∪T)×Pow(∪T).\}
U⊆V\rangle)) using linord_spectrum
by auto

} ultimately have A\{\text{is in the spectrum of}\}(λT. IsLinOrder(T, {⟨U,V⟩∈Pow(∪T)×Pow(∪T).\}
U⊆V\rangle)) by blast

} then show T\{is anti-\}(λT. IsLinOrder(T, \{⟨U,V⟩∈ Pow(∪T)×Pow(∪T) . U ⊆ 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 (_\{is anti-compact\})
where T\{is anti-compact\} ≡ T\{is anti-\}(λT. (∪T)\{is compact in}T)

definition
IsAntiLin (_{is anti-lindelof})

where \( T \)\{is anti-lindelof\} \equiv T\{is anti-\}(\lambda T. (\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

66 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.

66.1 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 \( \mathcal{F} \) \{is a filter on\} \( X \) \( \implies \) \( f:X \to Y \implies f[\mathcal{F}]..Y \equiv \{A \in \text{Pow}(Y). \exists D \in \{f(B) \cdot B \in \mathcal{F}\}. D \subseteq A\}\)

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 \( \text{domain}(f) \) without any change in the definition.

lemma base_image_filter:

assumes \( \mathcal{F} \) \{is a filter on\} \( X \) \( f:X \to Y \)

shows \( \{fB \cdot B \in \mathcal{F}\} \) \{is a base filter\} (\( f[\mathcal{F}]..Y \)) and \( (f[\mathcal{F}]..Y) \) \{is a filter on\} \( Y \)

proof-

{ 
  assume \( 0 \in \{fB \cdot B \in \mathcal{F}\}\)
  then obtain \( B \) where \( B \in \mathcal{F} \) and \( f_B:fB=0 \) by auto
  then have \( B \in \text{Pow}(X) \) using assms(1) IsFilter_def by auto
  then have \( fB=\{fb. b \in B\} \) using image_fun assms(2) by auto
  with \( f_B \) have \( \{fb. b \in B\}=0 \) by auto
  then have \( B=0 \) by auto
  with \( B \in \mathcal{F} \) have False using IsFilter_def assms(1) by auto
}

then have \( 0 \notin \{fB \cdot B \in \mathcal{F}\} \) by auto
moreover
from assms(1) obtain S where $S \in \mathfrak{F}$ using IsFilter_def by auto
then have $fS \in \{fB . B \in \mathfrak{F}\}$ by auto
then have $nA : (fB . B \in \mathfrak{F}) \neq 0$ by auto
moreover
\[
\begin{align*}
\text{fix } A & \ B \\
\text{assume } A \in \{fB . B \in \mathfrak{F}\} \text{ and } B \in \{fB . B \in \mathfrak{F}\} \\
\text{then obtain } AB BB \text{ where } A = fAB \ B = fBB \ AB \in \mathfrak{F} \ BB \in \mathfrak{F} \text{ by auto} \\
\text{then have } I : f(AB \cap BB) \subseteq A \cap B \text{ by auto} \\
\end{align*}
\]
moreover
from assms(1) $I < AB \cap BB$ have $AB \cap BB \in \mathfrak{F}$ using IsFilter_def by auto
ultimately have $\exists D \in \{fB . B \in \mathfrak{F}\}. D \subseteq A \cap B$ by auto
ultimately have $\forall A \in \{fB . B \in \mathfrak{F}\}. \forall B \in \{fB . B \in \mathfrak{F}\}. \exists D \in \{fB . B \in \mathfrak{F}\}. D \subseteq A \cap B$ by auto
ultimately have $sbc : \{fB . B \in \mathfrak{F}\}$ satisfies the filter base condition
using SatisfiesFilterBase_def by auto
moreover
\[
\begin{align*}
\text{fix } t & \\
\text{assume } t \in \{fB . B \in \mathfrak{F}\} \\
\text{then obtain } B \text{ where } B \in \mathfrak{F} \text{ and } im_def : fB = t \text{ by auto} \\
\text{with assms(1) have } B \in \text{Pow}(X) \text{ unfolding IsFilter_def by auto} \\
\text{with im_def assms(2) have } t = \{fx. x \in B\} \text{ using image_fun by auto} \\
\text{with assms(2) have } t \subseteq Y \text{ using apply_funtype by auto} \\
\end{align*}
\]
then have $nB : \{fB . B \in \mathfrak{F}\} \subseteq \text{Pow}(Y)$ by auto
ultimately have $((\{fB . B \in \mathfrak{F}\} \text{ is a base filter} \{A \in \text{Pow}(Y). \exists D \in \{fB . B \in \mathfrak{F}\}. D \subseteq A\}) \land (\bigcup \{A \in \text{Pow}(Y). \exists D \in \{fB . B \in \mathfrak{F}\}. D \subseteq A\} = Y))$ using base_unique_filter_set2
by force
then have $\{fB . B \in \mathfrak{F}\} \text{ is a base filter} \{A \in \text{Pow}(Y). \exists D \in \{fB . B \in \mathfrak{F}\}. D \subseteq A\}$ by auto
with assms show $\{fB . B \in \mathfrak{F}\} \text{ is a base filter} \{f[\mathfrak{F}]..Y\}$ using ImageFilter_def by auto
moreover
note sbc
moreover
\[
\begin{align*}
\text{from } nA \text{ obtain } D \text{ where } I : D \in \{fB . B \in \mathfrak{F}\} \text{ by blast} \\
\text{moreover from } nB \text{ have } D \subseteq Y \text{ by auto} \\
\text{ultimately have } Y \in \{A \in \text{Pow}(Y). \exists D \in \{fB . B \in \mathfrak{F}\}. D \subseteq A\} \text{ by auto} \\
\end{align*}
\]
then have $\bigcup \{A \in \text{Pow}(Y). \exists D \in \{fB . B \in \mathfrak{F}\}. D \subseteq A\} = Y$ by auto
ultimately show $\{f[\mathfrak{F}]..Y\} \text{ is a filter on } Y$ using basic_filter

878
ImageFilter_def assms by auto

qed

66.2 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 ∈ ⋃ τ1 IsContinuous(τ1,τ2,f) f:(⋃ τ1)→(⋃ τ2) x ∈ ⋃ τ1
  shows ∀ U ∈ τ2. f(x) ∈ U −→ (∃ V ∈ τ1. x ∈ V ∧ f(V) ⊆ U)
  proof
  { fix U
    assume AS: U ∈ τ2 f(x) ∈ U
    then have f - (U) ∈ τ1 using assms(2) IsContinuous_def by auto
    moreover
    from assms(3) have f(f - (U)) ⊆ U using function_image_vimage fun_is_fun
    by auto
    moreover
    from assms(3) assms(4) AS(2) have x ∈ f - (U) using func1_1_L15 by auto
    ultimately have ∃ V ∈ τ1. x ∈ V ∧ f(V) ⊆ U by auto
  }
  then show ∀ U ∈ τ2. f(x) ∈ U −→ (∃ V ∈ τ1. x ∈ V ∧ f(V) ⊆ U) by auto
qed

A function that is continuous at every point of its domain is continuous.

lemma continuous_all_x_imp_cont_global:
  assumes ∀ x ∈ ⋃ τ1. ∀ U ∈ τ2. fx ∈ U −→ (∃ V ∈ τ1. x ∈ V ∧ f(V) ⊆ U) f ∈ (⋃ τ1)→(⋃ τ2)
  and τ1 {is a topology}
  shows IsContinuous(τ1,τ2,f)
  proof
  { fix U
    assume U ∈ τ2
    { fix x
      assume AS: x ∈ f - U
      note <U ∈ τ2>
      moreover
      from assms(2) have f - U ⊆ ⋃ τ1 using func1_1_L6A by auto
      with AS have x ∈ ⋃ τ1 by auto
      with assms(1) have ∀ U ∈ τ2. fx ∈ U −→ (∃ V ∈ τ1. x ∈ V ∧ f(V) ⊆ U) by auto
      moreover
      from AS assms(2) have fx ∈ U using func1_1_L15 by auto
      ultimately have ∃ V ∈ τ1. x ∈ V ∧ f(V) ⊆ U by auto
    }
  }
  ultimately have ∃ V ∈ τ1. x ∈ V ∧ f(V) ⊆ U by auto

879
then obtain \( V \) where \( V \in \tau_2 \) \( x \in f(V) \subseteq U \) by auto

moreover
from \( I \) have \( V \subseteq \bigcup \tau_1 \) by auto
moreover
from \text{assms}(2) \( \forall V \in \tau_1 \) \( x \in V \land V \subseteq f(U) \) by blast
with \( \forall x \in f(U) \) \( \exists \forall V \in \tau_1 \) \( x \in V \land V \subseteq f(U) \) by auto
hence \( \forall x \in f(U) \) \( \exists V \in \tau_1 \) \( x \in V \land V \subseteq f(U) \) by auto
with \text{assms}(3) have \( f(U) \in \tau_1 \) using \text{topology0.open_neigh_open topology0_def}
by auto

hence \( \forall U \in \tau_2 \) \( f(U) \in \tau_1 \) by auto
then show thesis using \text{IsContinuous_def by auto}
qed

66.3 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.

\text{lemma (in two_top_spaces0) cont_imp_filter_conver_preserved:}
\text{assumes} \( \exists \{\text{is a filter on}\} X_1 \) \( f \{\text{is continuous}\} \to f \{\text{in}\} \tau_1 \)
\text{shows} \( (f \times X_2) \to f \{\text{in}\} \tau_2 \)
\text{proof -}
from \text{assms}(1) \text{assms}(3) have \( x \in X_1 \)
using \text{topology0.FilterConverges_def topol_cntxs_valid(1) X1_def by auto}
have \text{topology0}(\tau_2) \text{using topol_cntxs_valid(2) by simp}
moreover from \text{assms}(1) have \( (f \times X_2) \{\text{is a filter on}\} \) \( (\bigcup \tau_2) \) \( \{\text{is a base filter}\} \{\text{in}\} \tau_2 \)
using \text{base_image_filter fmapAssum X1_def X2_def by auto}
moreover have \( \forall U \in \text{Pow}((\bigcup \tau_2)) \) \( (fx) \in \text{Interior}(U, \tau_2) \to \exists D \in \{\text{fb .B} \in \exists \} \) \( D \subseteq U \)
\text{proof -}
{ fix \( U \)
assume \( U \in \text{Pow}(X_2) \) \( (fx) \in \text{Interior}(U, \tau_2) \)
with \( x \in X_1 \) have \( x \in f \text{-(Interior}(U, \tau_2)) \) \( \text{and} \ f \text{-(Interior}(U, \tau_2)) \in \text{Pow}(X_1) \)
using \text{func1_1_L6A fmapAssum func1_1_L15 fmapAssum by auto}
note sub
moreover
have \text{Interior}(U, \tau_2) \in \tau_2 \text{using topology0.Top_2_L2 topol_cntxs_valid(2) by auto}
with \text{assms}(2) have \( f \text{-(Interior}(U, \tau_2)) \in \tau_1 \) unfolding \text{isContinuous_def}
\text{IsContinuous_def by auto}
with \( x \in \text{Interior}(f \text{-(Interior}(U, \tau_2))), \tau_1 \)
using topology0.Top_2_L3 topol_cntxs_valid(1) by auto
moreover from assms(1) assms(3) have \( \{ U \in \text{Pow}(X_1) . x \in \text{Interior}(U, \tau_1) \} \subseteq \mathcal{N} \)

using topology0.FilterConverges_def topol_cntxs_valid(1) X1_def
by auto
ultimately have \( f^{-1}(\text{Interior}(U, \tau_2)) \in \mathcal{F} \) by auto
moreover have \( f(f^{-1}(\text{Interior}(U, \tau_2))) \subseteq \text{Interior}(U, \tau_2) \)
using topology0.Top_2_L1 topol_cntxs_valid(2) by auto
ultimately have \( \exists D \in \{ f(B) . B \subseteq \mathcal{N} \} . D \subseteq U \) 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 \( \forall x \in \bigcup \tau_1 . \forall \mathcal{N} . ((\mathcal{N} \text{ is a filter on } X_1) \land (\mathcal{N} \rightarrow F x \{\text{in} \} \tau_1)) \rightarrow ((f[\mathcal{N}]..X_2) \rightarrow F (fx) \{\text{in} \} \tau_2) \)
shows \( f \) is continuous
proof-
\{
fix x
assume as2: \( x \in \bigcup \tau_1 \)
with assms have reg:
\( \forall \mathcal{N} . ((\mathcal{N} \text{ is a filter on } X_1) \land (\mathcal{N} \rightarrow F x \{\text{in} \} \tau_1)) \rightarrow ((f[\mathcal{N}]..X_2) \rightarrow F (fx) \{\text{in} \} \tau_2) \)
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 \( \rightarrow F x \{\text{in} \} \tau_1 \)
\( \bigcup \tau_1 \)
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 \( f[\text{Neig}]..X_2 \rightarrow F (fx) \{\text{in} \} \tau_2 \) by auto
moreover have \( (f[\text{Neig}]..X_2) \text{ is a filter on } X_2 \)
using base_image_filter(2) NFil fmapAssum by auto
ultimately have \( U \in \{ f[\text{Neig}]..X_2 \} \)
using topology0.FilterConverges_def topol_cntxs_valid(2) unfold-
X1_def X2_def
    by auto
moreover
from fmapAssum NFil have \{FB .B∈Neig\} \{is a base filter\} (f[Neig]..X2)
    using base_image_filter(1) X1_def X2_def by auto
ultimately have \exists V\in\{FB .B∈Neig\}. V\subseteq U using basic_element_filter
by blast
then obtain B where B∈Neig fB⊆U by auto
moreover
have \text{Interior}(B, τ_1)⊆B using topology0.Top_2_L1 topol_cntxs_valid(1)
by auto
hence \text{fInterior}(B, τ_1)⊆ f(B) by auto
moreover have \text{Interior}(B, τ_1)∈τ_1
    using topology0.Top_2_L2 topol_cntxs_valid(1) by auto
ultimately have \exists V∈τ_1. x∈V ∧ fV⊆U by force
} hence \forall U∈τ_2. fx∈U \rightarrow (\exists V∈τ_1. x∈V ∧ fV⊆U) by auto
} hence \forall x∈\bigcup \tau_1. \forall U∈τ_2. fx∈U \rightarrow (\exists V∈τ_1. x∈V ∧ fV⊆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

67 Topology 7

theory Topology_ZF_7 imports Topology_ZF_5
begin

67.1 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\} ≡ \forall U. (U∈T ∧ (U \{is closed in\}T)) \rightarrow U=0 ∨ U=\bigcup T

lemma indiscrete_connected:
    shows \{0,X\} \{is connected\}
    unfolding IsConnected_def IsClosed_def by auto

The anti-property of connectedness is called total-diconnectedness.

882
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
    then have A ≲ 1 using eqpoll_imp_lepoll by auto
  }
  moreover
  { assume A = 0 then have A ≲ 1 using empty_lepollI[of 1] 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 connected} unfolding IsConnected_def by auto
    }
    moreover
    { assume ∪ T ≠ 0
      moreover
      from <A ≲ 1> ∪ T ≈ A have ∪ T ≲ 1 using eq_lepoll_trans by auto
    }
  }

883
ultimately
obtain $E$ where $\bigcup T = \{E\}$ 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 $T \subseteq \{0, \{E\}\}$ by `blast`
with $\langle T \text{ is a topology} \rangle$ have $\{0\} \subseteq T \subseteq \{0, \{E\}\}$ using `empty_open`
by `auto`
then have $T \subseteq \text{Pow}(\{0\})$ by `auto`
ultimately have $T \subseteq \{0, \{E\}\}$ by `auto`
then show $A$ is in the spectrum of $\text{IsConnected}$ unfolding `Spec_def` by `auto`
qed

The discrete space is a first example of totally-disconnected space.

**Lemma** `discrete_tot_dis`:
shows $\text{Pow}(X)$ is totally-disconnected
proof-

{ fix $A$ assume $A \subseteq \text{Pow}(X)$ and $\text{con}: (\text{Pow}(X)\text{restricted to }A)\{\text{is connected}\}$
  have $\text{res}: (\text{Pow}(X)\text{restricted to }A) = \text{Pow}(A)$ unfolding `RestrictedTo_def`
  using $\langle A \subseteq \text{Pow}(X) \rangle$
  by `blast`
  {
    assume $A = 0$
    then have $A \subseteq 1$ using `empty_lepollI[of 1]` by `auto`
    then have $A$ is in the spectrum of $\text{IsConnected}$ using `conn_spectrum`
  }
}
moreover
{
  assume $A \neq 0$
  then obtain $E$ where $E \in A$ by `blast`
  then have $\{E\} \subseteq \text{Pow}(A)$ by `auto`
  moreover
  have $A - \{E\} \subseteq \text{Pow}(A)$ by `auto`
  ultimately have $\{E\} \subseteq \text{Pow}(A) \wedge \{E\}$ is closed in $\text{Pow}(A)$ unfolding `IsClosed_def`
  by `auto`
  with $\text{con res have } \{E\} = A$ unfolding `IsConnected_def` by `auto`
  then have $A \approx 1$ using `singleton_eqpoll_1` by `auto`
  then have $A \subseteq 1$ using `eqpoll_imp_lepoll` by `auto`
  then have $A$ is in the spectrum of $\text{IsConnected}$ using `conn_spectrum`
  by `auto`
}
ultimately have $A$ is in the spectrum of $\text{IsConnected}$ by `auto`
}
then show thesis unfolding `IsTotDis_def` `antiProperty_def` by `auto`
An space is hyperconnected iff every two non-empty open sets meet.

**definition** IsHConnected (\(\text{is hyperconnected}\))

where \(T\{\text{is hyperconnected}\} \equiv \forall U \ W. U \in T \land W \in T \land U \cap W = 0 \rightarrow U = 0 \lor W = 0\)

Every hyperconnected space is connected.

**lemma** HConn_imp_Conn:

assumes \(T\{\text{is hyperconnected}\}\)

shows \(T\{\text{is connected}\}\)

**proof**

fix U assume U \(\in\) \(\cup\)\(\cap\)\(\cap\) \(\cap\)\(\subseteq\) \(\Rightarrow\) \(\lor\) \(\Rightarrow\)

moreover have \((\cup-U)\cap U = 0\)

moreover note asms

ultimately

have \(U = 0 \lor (\cup-U) = 0\) using IsHConnected_def by auto

with \(<U \in T>\) have \(U = 0 \lor U = \cup\) by auto

then show thesis using IsConnected_def by auto

qed

**lemma** Indiscrete_HConn:

shows \(\{\emptyset, X\}\{\text{is hyperconnected}\}\)

**proof**

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 \(\neg (X < \text{nat})\)

shows \((\text{CoFinite } X)\{\text{is hyperconnected}\}\)

**proof**

fix U V

assume \(U \in (\text{CoFinite } X) \cup (\text{CoFinite } X) \cap \emptyset = 0\)

then have eq: \((X-U) < \text{nat} \land U = 0 \land (X-V) < \text{nat} \land V = 0\)

unfolding Cofinite_def CoCardinal_def by auto

from \(<U \cap V = 0>\) have un: \((X-U) \cup (X-V) = X\)

by auto

assume AS: \((X-U) < \text{nat} \land (X-V) < \text{nat}\)

from un have X < nat using less_less_imp_un_less[OF AS InfCard_nat]

by auto

then have False using asms 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 \lessq 1
proof
  assume A is in the spectrum of)IsHConnected
  then have \forall T. (T is a topology) \union 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 \union (Pow(A))=A by auto
  then have \union (Pow(A)) \lessq A by auto
  ultimately have HC_Pow: Pow(A) is hyperconnected by auto
  { assume A=0
    then have A \lessq 1 using empty_lepollI by auto
  }
  moreover
  { assume A\neq 0
    then obtain e where e\in A by blast
    then have {e} \in Pow(A) by auto
    moreover have A-{e} \in Pow(A) by auto
    moreover have {e}\cap (A-{e})=0 by auto
    moreover note HC_Pow
    ultimately have A-{e}=0 unfolding IsHConnected_def by blast
    with <e\in A> have A={e} by auto
    then have A \lessq 1 using singleton_eqpoll_1 by auto
    then have A \lessq 1 using eqpoll_imp_lepoll by auto
  }
  ultimately show A \lessq 1 by auto
next
  assume A \lessq 1
  { fix T
    assume T is a topology \union T=A
    { assume \union 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 \( \bigcup T \neq 0 \)
moreover
from \( \leq T \leq A \) have \( \bigcup T \leq 1 \) using eq_lepoll_trans by auto
ultimately
obtain \( E \) where \( \bigcup T = \{ E \} \) using lepoll_1_is_sing by blast
moreover
have \( T \subseteq \mathcal{P}(\bigcup T) \) by auto
ultimately have \( T \subseteq \mathcal{P}(\{ E \}) \) by auto
then have \( T \subseteq \{ 0, \{ E \} \} \) 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 \( \text{IsHConnected} \) 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 \( x \in \bigcup T \)
shows \( (T(\text{restricted to}\{ x \},T)) \) is hyperconnected

**proof**

from assms have sub: \( \text{Closure}(\{ x \},T) \subseteq \bigcup T \) using Top_3_L11 by auto
then have tot: \( \bigcup (T(\text{restricted to}\{ x \},T)) = \text{Closure}(\{ x \},T) \)
unfolding RestrictedTo_def by auto

fix \( A \) \( B \)
assume \( A : A \in (T(\text{restricted to}\{ x \},T)B \in (T(\text{restricted to}\{ x \},T))A \cap B = 0 \)
then have \( B \subseteq \bigcup (T(\text{restricted to}\{ x \},T))A \subseteq \bigcup (T(\text{restricted to}\{ x \},T)) \)
by auto

with tot have \( B \subseteq \text{Closure}(\{ x \},T)A \subseteq \text{Closure}(\{ x \},T) \) by auto
from AS(1,2) obtain UA UB where UAUUB:UAB\in TUB\in TA=UA\cap \text{Closure}(\{ x \},T)B=UB\cap \text{Closure}(\{ x \},T)
unfolding RestrictedTo_def by auto
then have Closure(\{ x \},T)-A=Closure(\{ x \},T)-(UA\cap \text{Closure}(\{ x \},T)) Closure(\{ x \},T)-B=Closure(\{ x \},T)-(UB)
by auto
then have Closure(\{ x \},T)-A=Closure(\{ x \},T)-(UA) Closure(\{ x \},T)-B=Closure(\{ x \},T)-(UB)
by auto
with sub have Closure(\{ x \},T)-A=Closure(\{ x \},T) \cap (\bigcup T-UA) Closure(\{ x \},T)-B=Closure(\{ x \},T) \cap (\bigcup T-UB)
A consequence is that every totally-disconnected space is $T_1$. 

**Lemma (in topology0) tot_dis_imp_T1:**

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 $(T\{\text{restricted to}\} \text{Closure}(\{x\}, T))${is hyperconnected} using cl_point_imp_HConn by auto
  then have $(T\{\text{restricted to}\} \text{Closure}(\{x\}, T))${is connected} using HConn_imp_Conn by auto
  moreover from $<x \in \bigcup T>$ have $\text{Closure}(\{x\}, T) \subseteq \bigcup T$ using Top_3_L11(1) by auto
  moreover note assms
  ultimately have $\text{Closure}(\{x\}, T)${is in the spectrum of}IsConnected unfolding IsTotDis_def antiProperty_def by auto
  then have $\text{Closure}(\{x\}, T) \leq 1$ using conn_spectrum by auto
  moreover from $<x \in \bigcup T>$ have $x \in \text{Closure}(\{x\}, T)$ using cl_contains_set by auto
  ultimately have $\text{Closure}(\{x\}, T) = \{x\}$ using lepoll_1_is_sing[of $\text{Closure}(\{x\}, T)$ $x$] by auto

888
then have \{x\} \subseteq \text{closed in } T using Top.3_L8 \langle x \in \bigcup T \rangle by auto
then have \bigcup T - \{x\} \in T unfolding IsClosed_def by auto
moreover
from \langle y \in \bigcup T \rangle \langle y \neq x \rangle have y \in \bigcup T - \{x\} \wedge x \notin \bigcup T - \{x\} by auto
ultimately have \exists U \in T. y \in U \wedge x \notin U by force
\}
then show thesis unfolding 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 IsSober (_{is sober}90)
where T{is sober} \equiv \forall A \in \text{Pow}(\bigcup T)-\{0\}. (A{is closed in} T \wedge ((T{restricted to}A){is hyperconnected})) \rightarrow (\exists x \in \bigcup T. A=\text{Closure}(\{x\},T) \wedge (\forall y \in \bigcup T. A=\text{Closure}(\{y\},T) \rightarrow y=x))

Being sober is weaker than being anti-hyperconnected.

theorem (in topology0) anti_HConn_imp_sober:
assumes T{is anti-}IsHConnected
shows T{is sober}
proof-
{ 
  fix A assume A \in \text{Pow}(\bigcup T)-\{0\}.A{is closed in} T(T{restricted to}A){is hyperconnected)
  with assms have A{is in the spectrum of}IsHConnected unfolding antiProperty_def by auto
  then have A \subseteq 1 using 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 lepoll_1_is_sing by auto
  with \langle A{is closed in} T\rangle have \{x\} \{is closed in\} T 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 Closure(\{x\},T)=\{x\} unfolding Closure_def ClosedCovers_def by auto
  with \langle A={x}\rangle have A=\text{Closure}(\{x\},T) by auto
  moreover
  \{ 
  fix y assume y \in \bigcup T=\text{ Closure}(\{y\},T)
  then have \{y\} \subseteq \text{Closure}(\{y\},T) using cl_contains_set by auto
  with \langle A=\text{Closure}(\{y\},T)\rangle have y \in A by auto
  with \langle A={x}\rangle have y=x by auto
  \}
  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

889
then show thesis using IsSober_def by auto
qed

Every sober space is $T_0$.

**Lemma (in topology0) sober_imp_T0:**

- Assumes $T$ is sober
- Shows $T$ is $T_0$

**Proof:**

- Fix $x$ and $y$.
- Assume $x \in \bigcup T \land y \notin \bigcup T$. $x \in U \iff y \in U$.
- From $x \in \bigcup T$ have $\text{clx}: \text{Closure}({x},T)$ {is closed in} $T$ using cl_is_closed by auto.
- With $x \in \bigcup T$ have $(\bigcup T - \text{Closure}({x},T)) \in T$ using Top_3_L11(1) unfolding IsClosed_def by auto.
- Moreover from $x \in \bigcup T$ have $x \in \text{Closure}({x},T)$ using cl_contains_set by auto.
- Note $AS(1,4)$.
- Ultimately have $y \notin (\bigcup T - \text{Closure}({x},T))$ by auto.
- With $clx$ have $\text{ineq1}: \text{Closure}({x},T) \subseteq \text{Closure}({x},T)$ using Top_3_L13 by auto.
- From $y \in \bigcup T$ have $\text{cly}: \text{Closure}({y},T)$ {is closed in} $T$ using cl_is_closed by auto.
- With $y \in \bigcup T$ have $(\bigcup T - \text{Closure}({y},T)) \in T$ using Top_3_L11(1) unfolding IsClosed_def by auto.
- Moreover from $y \in \bigcup T$ have $y \in \text{Closure}({y},T)$ using cl_contains_set by auto.
- Note $AS(2,4)$.
- Ultimately have $x \notin (\bigcup T - \text{Closure}({y},T))$ by auto.
- With $AS(1)$ have $x \in \text{Closure}({y},T)$ by auto.
- With $cly$ have $\text{Closure}({x},T) \subseteq \text{Closure}({y},T)$ using Top_3_L13 by auto.
- With $\text{ineq1}$ have $\text{eq}: \text{Closure}({x},T) = \text{Closure}({y},T)$ by auto.
- Have $\text{Closure}({x},T) \in \text{Pow}((\bigcup T) - \emptyset)$ using Top_3_L11(1) $x \in \bigcup T \land x \in \text{Closure}({x},T)$ by auto.
- Moreover note $\text{assms clx}$.
- Ultimately have $\exists t \in \bigcup T. (\text{Closure}({x},T) = \text{Closure}({t},T) \land (\forall y \in \bigcup T. \text{Closure}({x},T) = \text{Closure}({y},T) \rightarrow y = t))$.
- Unfolding IsSober_def using cl_point_imp_HConn[OF $x \in \bigcup T$] by auto.
- Then obtain $t$ where $t \in \bigcup T$.
- $\text{Closure}({x},T) = \text{Closure}({y},T) \rightarrow y = t$ by blast.
- With $\text{eq}$ have $y = t$ using $y \in \bigcup T$ by auto.
- Moreover from $t \in \bigcup T$ have $x = t$ by blast.
- Ultimately have $y = x$ by auto.
- With $x \neq y$ have False by auto.

890
Every $T_2$ space is anti-hyperconnected.

**Theorem (in topology0) T2_imp_anti_HConn:**

assumes $T$ is $T_2$.

shows $T$ is anti-IsHConnected

**Proof:**

```plaintext
{ fix TT
 assume TT is a topology } TT is hyperconnected \( \forall x, y. x \in \bigcup T \land y \in \bigcup T \land x \neq y \rightarrow (\exists U \in T. (x \in U \land y \notin U) \lor (y \in U \land x \notin U)) \)

by auto

then show thesis using isT0_def by auto

qed
```

891
then show thesis using assms topSpaceAssum by auto
qed

Every anti-hyperconnected space is $T_1$.

**Theorem anti_HConn_imp_T1:**

- Assumes $T$ is anti-IsHConnected
- Shows $T$ is $T_1$

**Proof**:

- Fix $x, y$
- Assume $x \in \bigcup Ty \in \bigcup Tx \neq y$
- Assume $AS: \forall U \in T. x \notin U \lor y \in U$
- From $x \in \bigcup T \Rightarrow y \in \bigcup T$ have $\{x, y\} \in \text{Pow}(\bigcup T)$ by auto
- Then have $\text{sub}:(T\{\text{restricted to}\}\{x, y\}) \subseteq \text{Pow}(\{x, y\})$ using RestrictedTo_def
- By auto
  - Fix $U, V$
  - Assume $H: U \in T\{\text{restricted to}\}\{x, y\} \lor V \in (T\{\text{restricted to}\}\{x, y\}) \lor V=0$
  - With $AS$ have $x \in U \rightarrow y \in U \rightarrow y \in V$ unfolding RestrictedTo_def by auto
  - With $H(1,2)$ sub have $x \in U \rightarrow (x, y) x \in V \rightarrow V=\{x, y\}$ by auto
  - With $H$ sub have $x \in U \rightarrow (U=\{x, y\} \lor V=0) x \in V \rightarrow (V=\{x, y\} \lor U=0)$ by auto
  - Moreover
    - From sub $H$ have $(x \notin U \land x \notin V) \rightarrow (U=0 \lor V=0)$ by blast
    - Ultimately have $U=0 \lor V=0$ by auto
  - Then have $(T\{\text{restricted to}\}\{x, y\})\{\text{is hyperconnected}\}$ unfolding IsHConnected_def
  - By auto
    - With assms-$\{x, y\} \in \text{Pow}(\bigcup T)$ have $\{x, y\}\{\text{is in the spectrum of}\}$IsHConnected
    - Unfolding antiProperty_def
    - By auto
      - Then have $\{x, y\} \preceq 1$ using HConn_spectrum by auto
      - Moreover
        - Have $x \in \{x, y\}$ by auto
        - Ultimately have $\{x, y\} \equiv \{x\} \equiv \{x, y\}$ using lepoll_1_is_sing[of $\{x, y\}$] by auto
        - Moreover
          - Have $y \in \{x, y\}$ by auto
          - Ultimately have $y \in \{x\}$ by auto
          - Then have $y=x$ by auto
          - With $x \neq y$ have False by auto
      - Then have $\exists U \in T. x \in U \land y \notin U$ by auto
    - Then show thesis using isT1_def by auto
qed

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.

**Lemma Cofinite_not_anti_HConn:**

\[
\text{shows } \neg ((\text{CoFinite } \mathbb{N})\{\text{is anti-}\})\text{IsHConnected} \quad \text{and} \quad (\text{CoFinite } \mathbb{N})\{\text{is T}_1}\]

**Proof:**

\[
\begin{align*}
\text{assume } (\text{CoFinite } \mathbb{N})\{\text{is anti-}\}\text{IsHConnected} \\
\text{moreover } \text{have } \bigcup (\text{CoFinite } \mathbb{N})=\mathbb{N} \quad \text{unfolding } \text{Cofinite_def} \quad \text{using } \text{union_cocardinal} \\
\text{by auto} \\
\text{moreover } \text{have } (\text{CoFinite } \mathbb{N})\{\text{restricted to}\}\mathbb{N}=(\text{CoFinite } \mathbb{N}) \quad \text{using } \text{subspace_cocardinal} \\
\text{unfolding } \text{Cofinite_def} \quad \text{by auto} \\
\text{moreover } \text{have } \neg (\mathbb{N} \prec \mathbb{N}) \quad \text{by auto} \\
\text{then have } (\text{CoFinite } \mathbb{N})\{\text{is hyperconnected}\} \quad \text{using } \text{Cofinite_nat_HConn[of } \mathbb{N}\text{]} \quad \text{by auto} \\
\text{ultimately have } \mathbb{N}\{\text{is in the spectrum of}\}\text{IsHConnected} \quad \text{unfolding } \text{antiProperty_def} \quad \text{by auto} \\
\text{then show } \neg ((\text{CoFinite } \mathbb{N})\{\text{is anti-}\})\text{IsHConnected} \quad \text{by auto}
\end{align*}
\]

**Next:**

\[
\begin{align*}
\text{show } (\text{CoFinite } \mathbb{N})\{\text{T}_1\} \quad \text{using } \text{cocardinal_is_T1 InfCard_nat} \quad \text{unfolding } \text{Cofinite_def} \quad \text{by auto}
\end{align*}
\]

**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:**

\[
\text{shows } (\text{joinT } \{\text{CoFinite } \mathbb{N},\text{ExcludedSet}(\mathbb{N},\{0,1\})\})=(\text{CoFinite } \mathbb{N}) \cup \text{ExcludedSet}(\mathbb{N},\{0,1\})
\]

**Proof:**

\[
\begin{align*}
\text{have coftop:(CoFinite } \mathbb{N})\{\text{is a topology}\} \quad \text{unfolding } \text{Cofinite_def} \quad \text{using } \text{CoCar_is_topology InfCard_nat} \quad \text{by auto} \\
\text{moreover } \text{have } \text{ExcludedSet}(\mathbb{N},\{0,1\})\{\text{is a topology}\} \quad \text{using } \text{excludedset_is_topology} \quad \text{by auto} \\
\text{moreover } \text{have } \text{exuni:(ExcludedSet}(\mathbb{N},\{0,1\})=\mathbb{N} \quad \text{using } \text{union_excludedset} \quad \text{by auto} \\
\text{moreover } \text{have } \text{exuni:(ExcludedSet}(\mathbb{N},\{0,1\})=\mathbb{N} \quad \text{using } \text{union_excludedset} \quad \text{by auto}
\end{align*}
\]
ultimately have \( \bigcup \) (CoFinite nat) = nat using union_cardinal unfolding Cofinite_def by auto

ultimately have (joinT {CoFinite nat, ExcludedSet(nat, \{0,1\})}) = (THE T. (CoFinite nat) \cup \text{ExcludedSet(nat, \{0,1\})} is a subbase for T)
using joinT_def by auto

moreover
have \( \bigcup \) (CoFinite nat) \in CoFinite nat using CoCar_is_topology[OF InfCard_nat]
unfolding Cofinite_def IsATopology_def by auto

with cofuni have n:nat \in CoFinite nat by auto

ultimately have (joinT {CoFinite nat, ExcludedSet(nat, \{0,1\})}) = (THE T. (CoFinite nat) \cup \text{ExcludedSet(nat, \{0,1\})} is a subbase for T)
using same_subbase_same_top[OF _ Pa] the_equality
where a = \( \bigcup \) A. A \in Pow((CoFinite nat) \cup \text{ExcludedSet(nat, \{0,1\}))
and P = \lambda T. ((CoFinite nat) \cup \text{ExcludedSet(nat, \{0,1\})} is a subbase for T)

ultimately have equal = (joinT {CoFinite nat, ExcludedSet(nat, \{0,1\})}) = (\bigcup A. A \in Pow((CoFinite nat) \cup \text{ExcludedSet(nat, \{0,1\}))
by auto

\{ 
fix U assume U \in (\bigcup A. A \in Pow((CoFinite nat) \cup \text{ExcludedSet(nat, \{0,1\}))
then obtain AU where U = \bigcup AU and base: AU \in Pow((CoFinite nat) \cup \text{ExcludedSet(nat, \{0,1\}))
by auto

have (CoFinite nat) \subseteq Pow((CoFinite nat) \cup \text{ExcludedSet(nat, \{0,1\}))) by auto
moreover
have ExcludedSet(nat, \{0,1\}) \subseteq Pow((CoFinite nat) \cup \text{ExcludedSet(nat, \{0,1\}))) by auto
moreover
note cofuni exuni
ultimately have sub: (CoFinite nat) \subseteq Pow((CoFinite nat) \cup \text{ExcludedSet(nat, \{0,1\}))) by auto
from base have \( \forall S \subseteq AU. S \subseteq (\bigcap B. B \in \text{FinPow((CoFinite nat) \cup \text{ExcludedSet(nat, \{0,1\}))
by blast
then have \( \forall S \subseteq AU. \exists B \in \text{FinPow((CoFinite nat) \cup \text{ExcludedSet(nat, \{0,1\})
by blast
then have eq: \( \forall S \subseteq AU. \exists B \in \text{FinPow((CoFinite nat) \cup \text{ExcludedSet(nat, \{0,1\}))
by blast

\{ 
fix S assume S \subseteq AU
with eq obtain B where B \in \text{FinPow((CoFinite nat) \cup \text{ExcludedSet(nat, \{0,1\})\)
by auto

with sub have B \in \text{Pow(nat)} by auto

\{ 
fix x assume x \in B
then have \( \forall N \in B. x \in N \neq 0 \)
with \langle B \in \text{Pow(nat)}\rangle have x \in nat by blast

894
\[ \{ \text{then have } \forall S \in \mathbb{U}, S \in \text{Pow}(\mathbb{n}) \text{ by blast} \} \]
\[
\begin{align*}
&\text{with } S \subseteq B \text{ have } S \in \text{Pow}(\mathbb{n}) \text{ by auto} \\
&\text{then have } \forall S \in A, S \in \text{Pow}(\mathbb{n}) \text{ by blast} \\
&\{ \\
&\text{assume } 0 \in U \land 1 \in U \\
&\text{with } S \subseteq \bigcup A \text{ obtain } S \text{ where } S \subseteq A \cup S \subseteq B \text{ by auto} \\
&\text{with base obtain } BS \text{ where } S \subseteq BS \text{ and } bsbase:BS \in \text{FinPow}(\text{CoFinite nat} \cup \text{ExcludedSet}(\mathbb{n}, \{0, 1\})) \text{ by auto} \\
&\text{then have } \forall M \in BS. 0 \in M \lor 1 \in M \text{ by auto} \\
&\text{ultimately have } BS \in \text{FinPow}(\text{CoFinite nat}) \text{ unfolding } \text{FinPow_def by auto} \\
&\text{moreover} \\
&\text{note bsbase } n \text{ unfolding } \text{ExcludedPoint_def} \text{ by auto} \\
&\text{moreover} \\
&\text{from } S \subseteq \bigcup A \text{ have } S \neq 0 \text{ by auto} \\
&\text{with } S \subseteq BS \text{ have } S \neq 0 \text{ by auto} \\
&\text{moreover} \\
&\text{ultimately have } \bigcap BS \subseteq \text{CoFinite nat} \text{ using } \text{topology0.fin_inter_open_open[OF topology0_CoCardinal[OF InfCard_nat]]} \\
&\text{unfolding } \text{Cofinite_def by auto} \\
&\text{with } S \subseteq BS \text{ have } S \subseteq \text{CoFinite nat} \text{ by auto} \\
&\text{with } 0 \in S \subseteq S \text{ unfolding } \text{Cofinite_def CoCardinal_def by auto} \\
&\text{moreover} \\
&\text{from } U = \bigcup A \text{ have } S \subseteq U \text{ by auto} \\
&\text{then have } U \subseteq \text{nat} \subseteq S \text{ by auto} \\
&\text{then have } U \subseteq \text{nat} \subseteq S \text{ using } \text{subset_imp_lepoll by auto} \\
&\text{ultimately} \\
&\text{have } U \subseteq \text{nat} \text{ using } \text{lesspoll_trans1 by auto} \\
&\text{with } U \subseteq \text{Pow(\mathbb{n}) have } U \subseteq \text{CoFinite nat unfolding } \text{Cofinite_def CoCardinal_def by auto} \\
&\text{with } U \subseteq \text{Pow(\mathbb{n}) have } U \subseteq (\text{CoFinite nat} \cup \text{ExcludedSet}(\mathbb{n}, \{0, 1\})) \text{ by auto} \\
&\text{moreover} \\
&\text{with } U \subseteq \text{Pow(\mathbb{n}) have } U \subseteq (\text{CoFinite nat} \cup \text{ExcludedSet}(\mathbb{n}, \{0, 1\})) \text{ unfolding } \text{ExcludedSet_def by blast} \\
&\text{then have } (\bigcup A \cdot A \in \text{Pow}(\bigcap B \cdot B \in \text{FinPow}(\text{CoFinite nat} \cup \text{ExcludedSet}(\mathbb{n}, \{0, 1\})))) \subseteq (\text{CoFinite nat} \cup \text{ExcludedSet}(\mathbb{n}, \{0, 1\})) \text{ by blast} \\
&\text{moreover} \\
&\text{fix } U \\
&\text{assume } U \subseteq (\text{CoFinite nat} \cup \text{ExcludedSet}(\mathbb{n}, \{0, 1\}))
\end{align*}
\]
then have \(\{U\} \in \text{FinPow}(\text{CoFinite nat} \cup \text{ExcludedSet}(\text{nat},\{0,1\}))\) unfolding FinPow_def by auto 
then have \(\{U\} \in \text{FinPow}(\text{CoFinite nat} \cup \text{ExcludedSet}(\text{nat},\{0,1\}))\) by blast 
moreover 
have \(U=\bigcup\{\{U\}\}\) by auto 
ultimately have \(U \in \{\bigcup A . A \in \text{FinPow}(\text{CoFinite nat} \cup \text{ExcludedSet}(\text{nat},\{0,1\}))\}\) by blast 
moreover 
have \(U=\bigcup\{\{U\}\}\) by auto 
ultimately have \(\text{(CoFinite nat)} \cup \text{ExcludedSet}(\text{nat},\{0,1\}) = \{\bigcup A . A \in \text{FinPow}(\text{CoFinite nat} \cup \text{ExcludedSet}(\text{nat},\{0,1\}))\}\) by blast 
moreover 
have \(U=\bigcup\{\{U\}\}\) by auto 
ultimately have \(\text{(CoFinite nat)} \cup \text{ExcludedSet}(\text{nat},\{0,1\}) = \{\bigcup A . A \in \text{FinPow}(\text{CoFinite nat} \cup \text{ExcludedSet}(\text{nat},\{0,1\}))\}\) by blast 
moreover 
have \(U=\bigcup\{\{U\}\}\) by auto 
ultimately have \(\text{IsHConnected}\) 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 
\(\neg((\text{joinT } \{\text{CoFinite nat}, \text{ExcludedSet}(\text{nat},\{0,1\})\})\text{is }T_2)\) and 
\((\text{joinT } \{\text{CoFinite nat},\text{ExcludedSet}(\text{nat},\{0,1\})\})\text{is anti-} \text{IsHConnected}\) 
proof- 
have \((\text{CoFinite nat}) \subseteq (\text{CoFinite nat}) \cup \text{ExcludedSet}(\text{nat},\{0,1\})\) by auto 
have \(\bigcup((\text{CoFinite nat}) \cup \text{ExcludedSet}(\text{nat},\{0,1\}))=(\bigcup(\text{CoFinite nat})\cup \text{ExcludedSet}(\text{nat},\{0,1\}))\) by auto 
moreover 
have ...=nat unfolding Cofinite_def using union_cocardinal union_excludedset by auto 
ultimately have tot: \(\bigcup((\text{CoFinite nat}) \cup \text{ExcludedSet}(\text{nat},\{0,1\}))\)=nat by auto 
\{ 
assume \((\text{joinT } \{\text{CoFinite nat},\text{ExcludedSet}(\text{nat},\{0,1\})\})\text{ is }T_2\) 
then have t2: \((\text{CoFinite nat}) \cup \text{ExcludedSet}(\text{nat},\{0,1\}))\text{ is }T_2\) using join_top_cofinite_excluded_set by auto 
with tot have \(\exists U\in((\text{CoFinite nat}) \cup \text{ExcludedSet}(\text{nat},\{0,1\}))\). \(\exists V\in((\text{CoFinite nat}) \cup \text{ExcludedSet}(\text{nat},\{0,1\}))\). \(0\in U\land 1\in V\land V=0\) using isT2_def by auto 
then obtain U V where U \(\in (\text{CoFinite nat}) \lor (0 \notin U\land 1\notin V)\) using Cofinite_nat_HConn isHConnected_def unfolding ExcludedSet_def by auto 
then have U \(\in (\text{CoFinite nat})\) by auto 
with \(0\in U\land 1\in V\) have \(U\cap V\neq 0\) using Cofinite_nat_HConn IsHConnected_def by auto 
with \(U\cap V\neq 0\) have False by auto 
\} 
then show \(\neg((\text{joinT } \{\text{CoFinite nat},\text{ExcludedSet}(\text{nat},\{0,1\})\})\text{ is }T_2)\) by
auto
{
  fix A assume AS:A \in \bigcup((\text{CoFinite } \mathbb{N}) \cup \text{ExcludedSet}(\mathbb{N},\{0,1\})) \cup \text{ExcludedSet}(\mathbb{N},\{0,1\})\{\text{restricted to} A\}\{\text{is hyperconnected}\}
  with tot have A \in \text{Pow}(\mathbb{N}) by auto
  then have sub:A \cap \mathbb{N} = A by auto
  have ((\text{CoFinite } \mathbb{N}) \cup \text{ExcludedSet}(\mathbb{N},\{0,1\}))\{\text{restricted to} A\} = ((\text{CoFinite } \mathbb{N})\{\text{restricted to} A\} \cup (\text{ExcludedSet}(\mathbb{N},\{0,1\})\{\text{restricted to} A\})
    unfolding \text{RestrictedTo_def} by auto
  also from sub have ...=(\text{CoFinite } A)\cup\text{ExcludedSet}(A,\{0,1\}) using \text{subspace_excludedset[of nat nat A]} unfolding \text{Cofinite_def}
    by auto
  finally have ((\text{CoFinite } \mathbb{N})\cup\text{ExcludedSet}(\mathbb{N},\{0,1\}))\{\text{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\}))\{\text{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 \text{empty_excludedset[of \{0,1\} A]} unfolding \text{Cofinite_def CoCardinal_def}
    by auto
  with eq have \text{Pow}(A)\{\text{is hyperconnected}\} by auto
    then have \text{Pow}(A)\{\text{is connected}\} using \text{HConn_imp_Conn} by auto
    moreover have \text{Pow}(A)\{\text{is anti-}Is\text{Connected}\} using \text{discrete_tot_dis unfolding IsTotDis_def by auto}
      moreover have \bigcup(\text{Pow}(A)) \subseteq \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}\}Is\text{Connected unfolding \text{antiProperty_def}}
          by auto
            then have A\{\text{is in the spectrum of}\}Is\text{Connected} by auto
              then have A \subseteq 1 using \text{conn_spectrum by auto}
                then have A\{\text{is in the spectrum of}\}Is\text{HConnected} using \text{HConn_spectrum}
                  by auto
  }
moreover
{
  assume AS:\{0,1\} \cap A \neq 0
  
  assume A = \{0\} \cup A = \{1\}
    then have A \approx 1 using \text{singleton_eqpoll_1 by auto}
      then have A \subseteq 1 using \text{eqpoll_imp_lepoll by auto}
        then have A\{\text{is in the spectrum of}\}Is\text{HConnected} using \text{HConn_spectrum}
          by auto
}
moreover
{
  assume AS2:～(A={0}∨A={1})
  
  assume AS3:A⊆{0,1}
  with AS AS2 have A_def:A={0,1} by blast
  then have ExcludedSet(A,{0,1})=ExcludedSet(A,A) by auto
  moreover have ExcludedSet(A,A)={0,A} unfolding ExcludedSet_def
  by blast
  ultimately have ExcludedSet(A,{0,1})={0,A} by auto
  moreover
  have 0∈(CoFinite A) using empty_open[of CoFinite A]
  CoCar_is_topology[OF InfCard_nat,of A] unfolding Cofinite_def
  by auto
  moreover
  have ∪(CoFinite A)=A using union_cocardinal unfolding Cofinite_def
  by auto
  then have A∈(CoFinite A) using CoCar_is_topology[OF InfCard_nat,of A] unfolding Cofinite_def
  IsATopology_def by auto
  ultimately have (CoFinite A)∪ExcludedSet(A,{0,1})=(CoFinite A) by auto
  with eq have (CoFinite A){is hyperconnected} by auto
  with A_def have hyp:(CoFinite {0,1}){is hyperconnected} by auto
  have {0}≈1(1)≈1 using singleton_eqpoll_1 by auto
  moreover
  have 1<nat using n_lesspoll_nat by auto
  ultimately have {0}~{nat}~{1}~{nat} using eq_lesspoll_trans by auto
  moreover
  have {0,1}~{1}~{0}~{1}~{0}~{1} by auto
  ultimately have {1}∈(CoFinite {0,1})∈(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∈A t≠0 t≠1 by auto
then have {t}∈ExcludedSet(A,{0,1}) unfolding ExcludedSet_def
by auto
moreover
{
  have {t}≈1 using singleton_eqpoll_1 by auto
  moreover
  have t<nat using n_lesspoll_nat by auto
  ultimately have {t}~{nat}~{t}~{nat} using eq_lesspoll_trans by auto
  moreover
  with t∈A have A~(A~{t})={t} by auto
  ultimately have A~{t}∈(CoFinite A) unfolding Cofinite_def CoCardinal_def
  by auto
}
898
ultimately have \( \{ t \} \in ((\text{CoFinite } A) \cup \text{ExcludedSet}(A, \{0,1\})) \setminus \{ t \} \in ((\text{CoFinite } A) \cup \text{ExcludedSet}(A, \{0,1\})) \setminus \{(A \setminus \{ t \}) = 0 \) by auto

with eq have \( A \setminus \{ t \} = 0 \) unfolding IsHConnected_def by auto

with \( t \in A \) have \( A \setminus \{ t \} = 0 \) by auto

then have \( A \approx 1 \) using singleton_eqpoll_1 by auto

then have \( A \subseteq 1 \) using eqpoll_imp_lepoll by auto

then have \( A \) is in the spectrum of \( \text{IsHConnected} \) using HConn_spectrum by auto

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 \( ((\text{CoFinite } \mathbb{N}) \cup \text{ExcludedSet}(\mathbb{N}, \{0,1\})) \) is anti-\( \text{IsHConnected} \)

unfolding antiProperty_def by auto

then show \( (\text{joinT } \{ \text{CoFinite } \mathbb{N}, \text{ExcludedSet}(\mathbb{N}, \{0,1\}) \}) \) 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.

\begin{theorem}[in topology0] T1_sober_imp_anti_HConn:
  assumes \( T \) is \( T_1 \) and \( T \) is sober
  shows \( T \) is anti-\( \text{IsHConnected} \)
\end{theorem}

\begin{proof}
  \{ fix \( A \) assume \( A \in \text{Pow}(\bigcup T) \) \( (T \text{ restricted to } A) \) is hyperconnected \}
  \{ assume \( A = 0 \)
    then have \( A \subseteq 1 \) using empty_lepollI by auto
    then have \( A \) is in the spectrum of \( \text{IsHConnected} \) using HConn_spectrum by auto
  \}
  moreover
  \{ assume \( A \neq 0 \)
    then obtain \( x \) where \( x \in A \) by blast
  \}
  assume \( \neg ((T \text{ restricted to } \text{Closure}(A,T)) \) is hyperconnected
  then obtain \( U \) \( V \) where \( UV \text{ def } U \in (T \text{ restricted to } \text{Closure}(A,T)) \) \( V \in (T \text{ restricted to } \text{Closure}(A,T)) \)
  \( U \cap V = 0 \neq 0 \neq V \) using IsHConnected_def by auto

989
then obtain \( UCA \cap VCA \) where \( UCA \in TVCA \) \( \cap \) \( TU = UCA \cap \text{Closure}(A, T) \)

unfolding \( \text{RestrictedTo}_\text{def} \) by auto

from \( <A \in \text{Pow}(\bigcup T)> \) have \( A \subseteq \text{Closure}(A, T) \) using \( \text{cl\_contains\_set} \) by auto

then have \( UCA \cap A \subseteq UCA \cap \text{Closure}(A, T) \) \( \cap \) \( VCA \cap A \subseteq VCA \cap \text{Closure}(A, T) \) by auto

with \( <U = UCA \cap \text{Closure}(A, T) > <V = VCA \cap \text{Closure}(A, T) > \) \( \cup V = 0 \) have \( (UCA \cap A) \cap (VCA \cap A) = 0 \) by auto

moreover from \( <UCA \in T > <VCA \in T > \) have \( UCA \cap A \subseteq (T \{ \text{restricted to} \} \cap A) \) \( \cap VCA \cap A \subseteq (T \{ \text{restricted to} \} \cap A) \) by auto

unfolding \( \text{RestrictedTo}_\text{def} \) by auto

moreover note \( \text{AS(2)} \)

ultimately have \( UCA \cap A = 0 \) \( \vee VCA \cap A = 0 \) using \( \text{IsHConnected}_\text{def} \) by auto

with \( <A \subseteq \text{Closure}(A, T) > \) have \( A \subseteq \text{Closure}(A, T) - UCA \cap A \subseteq \text{Closure}(A, T) - VCA \) by auto

moreover

\[
\begin{align*}
\text{have } & \text{Closure}(A, T) - UCA = \text{Closure}(A, T) \cap (\bigcup T - UCA) \\
\text{Closure}(A, T) - VCA = \text{Closure}(A, T) \cap (\bigcup T - VCA) \\
\text{using } & \text{Top}_3\_L11(1) \text{ AS(1)} \text{ by auto} \\
\text{moreover } & \text{with } <UCA \in T > <VCA \in T > \text{ have } (\bigcup T - UCA) \{ \text{is closed in} \} T (\bigcup T - VCA) \{ \text{is closed in} \} T \\
\text{using } & \text{Top}_3\_L9 \text{ cl\_is\_closed AS(1) by auto} \\
\text{ultimately have } & (\text{Closure}(A, T) - UCA) \{ \text{is closed in} \} T (\text{Closure}(A, T) - VCA) \{ \text{is closed in} \} T \\
\text{using } & \text{Top}_3\_L5(1) \text{ by auto} \\
\end{align*}
\]

ultimately have \( \text{Closure}(A, T) \subseteq \text{Closure}(A, T) - UCA \cap \text{Closure}(A, T) - VCA \)

using \( \text{Top}_3\_L13 \)

by auto

then have \( UCA \cap \text{Closure}(A, T) = 0 \) \( \vee VCA \cap \text{Closure}(A, T) = 0 \) by auto

with \( <U = UCA \cap \text{Closure}(A, T) > <V = VCA \cap \text{Closure}(A, T) > \) have \( U = 0 \) \( \vee V = 0 \) by auto

with \( <U \neq 0 > <V \neq 0 > \) have False by auto

} then have \( (T \{ \text{restricted to} \} \cap \text{Closure}(A, T)) \{ \text{is hyperconnected} \} \) by auto

moreover have \( \text{Closure}(A, T) \{ \text{is closed in} \} T \) using \( \text{cl\_is\_closed AS(1)} \) by auto

moreover from \( <x \in A > \) have \( \text{Closure}(A, T) \neq 0 \) using \( \text{cl\_contains\_set AS(1)} \) by auto

moreover from \( \text{AS(1)} \) have \( \text{Closure}(A, T) \subseteq \bigcup T \) using \( \text{Top}_3\_L11(1) \) by auto

ultimately have \( \text{Closure}(A, T) \subseteq \text{Pow}(\bigcup T) - \{0\} \{ \text{restricted to} \} \cap \text{Closure}(A, T) \{ \text{is hyperconnected} \} \)

\( \text{Closure}(A, T) \{ \text{is closed in} \} T \)

by auto
moreover note assms(2)
ultimately have $\exists x \in \bigcup T. \ (\text{Closure}(A,T) = \text{Closure}(\{x\},T) \land (\forall y \in \bigcup T. \ \text{Closure}(A,T) = \text{Closure}(\{y\},T) \implies y = x))$
unfolding IsSober_def
by auto
then obtain $y$ where $y \in \bigcup TClosure(A,T) = \text{Closure}(\{y\},T)$ by auto
moreover
{ 
fix $z$ 
assume $z \in (\bigcup T) - \{y\}$ 
with assms(1) $\langle y \in \bigcup T \rangle$ obtain $U$ where $U \in T \ z \in U \ y \not\in U$ using isT1_def
by blast 
then have $U \in T \ z \in U \ U \subseteq (\bigcup T) - \{y\}$ by auto 
then have $\forall z \in (\bigcup T) - \{y\}. \ \exists U \in T. \ z \in U \land U \subseteq (\bigcup T) - \{y\}$ by auto 
} 
then have $\forall z \in (\bigcup T) - \{y\}. \ \exists U \in T. \ z \in U \land U \subseteq (\bigcup T) - \{y\}$ by auto 
then have $\exists U \in T. \ z \in U \land U \subseteq (\bigcup T) - \{y\}$ by auto 
then have $\{y\}$ using cl_contains_set[of $A$] by auto 
with $\langle A \not= 0 \rangle$ have $A = \{y\}$ 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 $A \{\text{is in the spectrum of}\} \text{IsHConnected}$ using HConn_spectrum 
by auto 
} 
ultimately have $A \{\text{is in the spectrum of}\} \text{IsHConnected}$ by blast 
then show thesis using antiProperty_def by auto 
qed 

theorem (in topology0) anti_HConn_iff_T1_sober: 
shows $\langle T \{\text{is anti-}\} \text{IsHConnected} \iff (T \{\text{is sober}\} \land T \{\text{T1}\}) \rangle$
using T1_sober_imp_anti_HConn anti_HConn_imp_sober 
by auto 

A space is ultraconnected iff every two non-empty closed sets meet.

definition IsUConnected (_{is ultraconnected}80)
where $T \{\text{is ultraconnected} \equiv \forall A \ B. \ A \{\text{is closed in}\} T \land B \{\text{is closed in}\} T \implies A \cap B = 0 \implies A = 0 \lor B = 0 \rangle$

Every ultraconnected space is trivially normal.

lemma (in topology0) UConn_imp_normal: 
assumes $T \{\text{is ultraconnected}\}$ 
shows $T \{\text{is normal}\}$ 
proof- 
{ 
fix $A \ B$ 
assume $A \{\text{is closed in}\} T \ B \{\text{is closed in}\} T \ A \cap B = 0$ 
with assms have $A = 0 \lor B = 0$ using IsUConnected_def by auto 

901
with $AS(1,2)$ have $(A \subseteq 0 \land B \subseteq \bigcup T) \lor (A \subseteq \bigcup T \land B \subseteq 0)$ unfolding IsClosed_def by auto
moreover
have $0 \in T$ using empty_open topSpaceAssum by auto
moreover
have $\bigcup T \in T$ using topSpaceAssum unfolding IsATopology_def by auto
ultimately have $\exists U. \exists V. A \subseteq U \land B \subseteq V \land U \cap V = 0$ by auto
} then show thesis unfolding IsNormal_def by auto qed

Every ultraconnected space is connected.

lemma UConn_imp_Conn:
assumes $T$ {is ultraconnected}
shows $T$ {is connected}
proof -
{ fix $U$ $V$
assume $U \in T$ {is closed in} $T$
then have $\bigcup T - (\bigcup T - U) = U$ by auto
with $<U \in T>$ have $(\bigcup T - U)$ {is closed in} $T$ unfolding IsClosed_def by auto
with $<U$ {is closed in} $T>$ assms have $U = 0 \lor \bigcup T - U = 0$ unfolding IsUConnected_def by auto
with $<U \in T>$ have $U = 0 \lor U = \bigcup T$ by auto
} then show thesis unfolding IsConnected_def by auto qed

lemma UConn_spectrum:
shows $(A$ {is in the spectrum of} $IsUConnected) \leftrightarrow A \leq 1$
proof
assume $A$ _spec: $(A$ {is in the spectrum of} $IsUConnected)
{ assume $A = 0$
then have $A \leq 1$ using empty_lepollI by auto
}
moreover
{ assume $A \neq 0$
from $A$ _spec have $\forall T. (T$ {is a topology} $\land \bigcup T = A) \rightarrow (T$ {is ultraconnected}) unfolding 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 $\bigcup Pow(A) = A$ by auto
ultimately have $ult:Pow(A)$ {is ultraconnected} by auto
moreover

902
from $A \neq \emptyset$ obtain $b$ where $b \in A$ by auto
then have $\{b\}$ is closed in $\mathrm{Pow}(A)$ unfolding \texttt{IsClosed_def} by auto
{  
  fix $c$
  assume $c \in A \neq b$
  then have $\{c\}$ is closed in $\mathrm{Pow}(A) \{c\} \cap \{b\} = \emptyset$ unfolding \texttt{IsClosed_def}
by auto
  with ult $\{b\}$ is closed in $\mathrm{Pow}(A)$ have False using \texttt{IsUConnected_def}
by auto
}
with $\{b\}$ is closed in $\mathrm{Pow}(A)$ have $A = \{b\}$ by auto
then have $A \approx 1$ using \texttt{singleton_eqpoll_1} by auto
then have $A \leq 1$ using \texttt{eqpoll_imp_lepoll} by auto
ultimately show $A \leq 1$ by auto
next
assume $A \leq 1$
{  
  fix $T$
  assume $T$ is a topology $\bigcup T = A$
  {  
    assume $\bigcup T = \emptyset$
    with $\{T\}$ is a topology $\emptyset$ have $T = \emptyset$ using \texttt{empty_open} by auto
    then have $T$ is ultraconnected unfolding \texttt{IsUConnected_def} \texttt{IsClosed_def}
by auto
  }
  moreover
  {  
    assume $\bigcup T \neq \emptyset$
    moreover
    from $A \leq 1 \land \bigcup T = A$ have $\bigcup T \leq 1$ using \texttt{eq_lepoll_trans} by auto
    ultimately
    obtain $E$ where $eq : \bigcup T = \{E\}$ using \texttt{lepoll_1_is_sing} by blast
    moreover
    have $T \subseteq \mathrm{Pow}(\bigcup T)$ by auto
    ultimately have $T \subseteq \mathrm{Pow}(\{E\})$ by auto
    then have $T \subseteq \{\emptyset, \{E\}\}$ by blast
    with $\{T\}$ is a topology $\emptyset \subseteq T \subseteq \{0, \{E\}\}$ using \texttt{empty_open}
by auto
    then have $T$ is ultraconnected unfolding \texttt{IsUConnected_def} \texttt{IsClosed_def}
by (simp only: eq, safe, force)
    }  
  ultimately have $T$ is ultraconnected by auto
  }
then show $A$ is in the spectrum of $\text{IsUConnected}$ unfolding \texttt{Spec_def} by auto
qed

This time, anti-ultraconnected is an old property.
theorem (in topology0) anti_UConn:
shows (T(is anti-)IsUConnected) ⟷ T(is T₁)
proof
  assume T(is T₁)
  { fix TT
    { assume TT(is a topology)TT(is T₁)TT(is ultraconnected)
      { assume ∪TT=0
        then have ∪TT≤1 using empty_lepollI by auto
        then have ((∪TT)(is in the spectrum of)IsUConnected) using UConn_spectrum
        by auto }
    moreover
    { assume ∪TT≠0
      then obtain t where t∈∪TT by blast
      { fix x
        assume p:x∈∪TT
        { fix y assume y∈(∪TT)-{x}
          with <TT(is T₁)> p obtain U where U∈TT y∈U x∉U using isT1_def
          by blast
          then have U∈TT y∈U U≤(∪TT)-{x} by auto
          then have ∃U∈TT. y∈U ∧ U≤(∪TT)-{x} by auto
          then have ∀ y∈(∪TT)-{x}. ∃U∈TT. y∈U ∧ U≤(∪TT)-{x} by auto
          unfolding topology0_def by auto
          with p have {x} {is closed in}TT using IsClosed_def by auto }
      then have reg:∀ x∈∪TT. {x}{is closed in}TT by auto
      with <t∈∪TT> have t_cl:{t}{is closed in}TT by auto
      { fix y
        assume y∈∪TT
        with reg have {y}{is closed in}TT by auto
        with <TT(is ultraconnected)> t_cl have y=t unfolding IsUConnected_def
        by auto }
      with <t∈∪TT> have ∪TT={t} by blast
      then have ∪TT≤1 using singleton_eqpoll_1 by auto
      then have ∪TT≤1 using eqpoll_imp_lepoll by auto
      then have (∪TT){is in the spectrum of}IsUConnected using UConn_spectrum
      by auto }
    ultimately have (∪TT){is in the spectrum of}IsUConnected by blast
  }
then have \((\bigcup \mathcal{T})\text{is a topology}\wedge (\mathcal{T}\text{is }T_1)\wedge (\mathcal{T}\text{is ultraconnected})) \rightarrow ((\bigcup \mathcal{T})\text{is in the spectrum of }\text{IsUConnected})
  \text{by auto}\)

\}

then have \(\forall \mathcal{T}. (\mathcal{T}\text{is a topology}\wedge (\mathcal{T}\text{is }T_1)\wedge (\mathcal{T}\text{is ultraconnected})) \rightarrow ((\bigcup \mathcal{T})\text{is in the spectrum of }\text{IsUConnected})\)
  \text{by auto}\)

moreover
note here_T1
ultimately have \(\forall T. T\text{is a topology} \rightarrow ((T\text{is }T_1) \rightarrow (T\text{is anti-IsUConnected}))\)
using Q_P_imp_Spec[where Q=isT1 and P=IsUConnected]
by auto
with topSpaceAssum have \((T\text{is }T_1) \rightarrow (T\text{is anti-IsUConnected})\) by auto
next
assume ASS:T\text{is anti-IsUConnected}

{ fix x y
  assume \(x\in\bigcup \mathcal{T}y\in\bigcup \mathcal{T}x\neq y\)
  then have tot:\((\bigcup (T\text{restricted to}\{x,y\}))=\{x,y\}\) unfolding RestrictedTo_def
  by auto
  \}
  assume AS:\(\forall U\in T. x\in U \rightarrow y\in U\)
  { assume \(\{y\}\text{is closed in}(T\text{restricted to}\{x,y\})\)
    moreover
    from \(<x\neq y>\) have \(\{x,y\}-\{y\}=\{x\}\) by auto
    ultimately have \(\{x\}\in (T\text{restricted to}\{x,y\})\) unfolding IsClosed_def
    by (simp only:tot)
    then obtain \(U\) where \(U\in T\{x,y\}\cap U\) unfolding RestrictedTo_def
    by auto
    moreover
    with \(<x\neq y>\) have \(y\notin \{x\} y\in \{x,y\}\) by (blast+)
    with \(<x>\{x,y\}\cap U\) have \(y\notin U\) by auto
    moreover have \(x\in \{x\}\) by auto
    with \(<x>\{x,y\}\cap U\) have \(x\in U\) by auto
    ultimately have \(x\in U y \notin U\in T\) by auto
    with AS have False by auto
  }
then have \(\neg (\{y\}\text{is closed in}(T\text{restricted to}\{x,y\}))\) 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\cap B=0
  with tot have \(A\subseteq \{x,y\}B\subseteq \{x,y\}A\cap B=0\) unfolding IsClosed_def
  by auto
then have \(x\in A \rightarrow x\notin B\in A \rightarrow y\notin B\subseteq \{x,y\}B\subseteq \{x,y\}\) by auto
assume \( x \in A \) with \(<x \in A \rightarrow x \notin B< \subseteq (x,y)>\) have \( B \subseteq \{y\}\) by auto
then have \( B=0 \lor B=\{y\}\) by auto
with \( y_{\text{no cl}} \ cl(2)\) have \( B=0\) by auto
\}
moreover
\{
assume \( x \notin A \) with \(<A \subseteq (x,y)>\) have \( A \subseteq \{y\}\) by auto
then have \( A=0 \lor A=\{y\}\) by auto
with \( y_{\text{no cl}} \ cl(1)\) have \( A=0\) by auto
\}
ultimately have \( A=0 \lor B=0\) by auto
with \( y_{\text{no cl}} \ cl(2)\) have \( B=0\) by auto

then have \( (T\{\text{restricted to}\}(x,y))\{\text{is ultraconnected}\}\) unfolding \( \text{IsUConnected}_\text{def}\) by auto
with \( ASS <x \in \bigcup T-y \in \bigcup T>\) have \( \{x,y\}\{\text{is in the spectrum of}\} \text{IsUConnected}\)
unfolding \( \text{antiProperty}_\text{def}\) by auto
then have \( (x,y) \leq 1\) using \( \text{UConn}_\text{spectrum}\) by auto
moreover have \( x \in \{x,y\}\) by auto
ultimately have \( \{x\}=\{x,y\}\) using \( \text{lepoll}_\text{1_is_sing}\{\text{of }\{x,y\}x\}\) 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 \( \text{False}\) using \(<x \neq y>\) by auto
\}
then have \( \exists U \in T. x \in U \land y \notin U\) by auto
\}
then show \( T\{\text{is}\ T_1}\) unfolding \( \text{isT1}_\text{def}\) by auto
qed

It is 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.

\textbf{Lemma} \( \text{sober}_\text{spectrum}: \)
shows \( (A\{\text{is in the spectrum of}\} \text{IsSober}) \iff A \leq 1\)
\textbf{proof}
assume \( AS:A\{\text{is in the spectrum of}\} \text{IsSober}\)
\{
assume \( A=0\)
then have \( A \leq 1\) using \( \text{empty_lepoll}\) by auto
\}
moreover
\{
assume \( A \neq 0\)
note AS
moreover
have top:{0,A}{is a topology} unfolding IsATopology_def by auto
moreover
have ∪{0,A}=A by auto
then have ∪{0,A}=A by auto
ultimately have {0,A}{is sober} using Spec_def by auto
moreover
have {0,A}{is hyperconnected} using Indiscrete_HConn by auto
moreover
have {0,A}{restricted to}A={0,A} unfolding RestrictedTo_def by auto
moreover
have A{is closed in}{0,A} unfolding IsClosed_def by auto
moreover
note <A≠0>
ultimately have ∃x∈A. A=Closure({x},{0,A})∧ (∀y∈∪{0, A}. A = Closure({y},
{0, A}) → y = x) unfolding IsSober_def by auto
then obtain x where x∈A A=Closure({x},{0,A}) and reg:∀y∈A. A = Closure({y},
{0, A}) → y = x by auto
{
fix y assume y∈A
with top have Closure({y},{0,A}){is closed in}{0,A} using topology0.cl_is_closed
moreover
from <y∈A> top have y∈Closure({y},{0,A}) using topology0.cl_contains_set
moreover
ultimately have A-Closure({y},{0,A})∈{0,A}Closure({y},{0,A})∩A≠0
unfolding IsClosed_def
by auto
then have A-Closure({y},{0,A})=AVA-Closure({y},{0,A})=0
by auto
moreover
from <y∈A> y∈Closure({y},{0,A})> have y∈Ay∉A-Closure({y},{0,A})
by auto
ultimately have A-Closure({y},{0,A})=0 by (cases A-Closure({y},{0,A})=A,
simp, auto)
moreover
from <y∈A> top have Closure({y},{0,A})⊆A using topology0_def topology0.Top_3_L11(1)
by blast
then have A-(A-Closure({y},{0,A}))=Closure({y},{0,A}) by auto
ultimately have A=Closure({y},{0,A}) by auto
}
with reg have ∀y∈A. x=y by auto
with <x∈A> have A={x} by blast
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

907
assume $A \leq 1$
{
fix T assume $T$ is a topology $\bigcup T \approx A$
{
assume $\bigcup T = 0$
then have $T$ is sober unfolding IsSober_def by auto
}
moreover
{
assume $\bigcup T \neq 0$
then obtain $x$ where $x \in \bigcup T$ by blast
moreover
from $\bigcup T \approx A < A \leq 1$ have $\bigcup T \leq 1$ using eq_lepoll_trans by auto
ultimately have $\bigcup T = \{x\}$ using lepoll_1_is_sing by auto
moreover
have $T \subseteq \text{Pow}(\bigcup T)$ by auto
ultimately have $T \subseteq \{0, \{x\}\}$ by blast
moreover
from $\langle T \text{ is a topology} \rangle$ have $0 \in T$ using empty_open by auto
moreover
from $\langle T \text{ is a topology} \rangle$ have $\bigcup T \in T$ unfolding IsATopology_def by auto
with $\langle \bigcup T = \{x\}\rangle$ have $\{x\} \in T$ by auto
ultimately have $T$ is sober unfolding IsSober_def by auto

908
ultimately have $T$ is sober by blast

then show $A$ is in the spectrum of $\text{IsSober}$ unfolding Spec_def by auto qed

\textbf{theorem (in topology0) anti_sober:}
shows $(T$ is anti-$\text{IsSober}) \iff T = \{0, \bigcup T\}$
\textbf{proof}
assume $T = \{0, \bigcup T\}$
  \{ fix $A$ assume $A \in \text{Pow}(\bigcup T)$ $(T \text{ restricted to } A)$ is sober \}
  \{ assume $A = 0$
     then have $A \subseteq 1$ using empty_lepollI by auto
     then have $A$ is in the spectrum of $\text{IsSober}$ using sober_spectrum
     by auto \}
moreover
  \{ assume $A \neq 0$
     have $\bigcup T \subseteq \{0, \bigcup T\}$ 0 $\in \{0, \bigcup T\}$ by auto
     with $\langle T = \{0, \bigcup T\} \rangle$ have $(\bigcup T) \in T$ 0 $\in T$ by auto
     with $\langle A \in \text{Pow}(\bigcup T) \rangle$ have $\{0, A\} \subseteq (T \text{ restricted to } A)$ unfolding RestrictedTo_def
     by auto
     moreover
     have $\forall B \in \{0, \bigcup T\}$. $B = 0 \lor B = \bigcup T$ by auto
     with $\langle T = \{0, \bigcup T\} \rangle$ have $\forall B \in T$. $B = 0 \lor B = \bigcup T$ by auto
     with $\langle A \in \text{Pow}(\bigcup T) \rangle$ have $T \text{ restricted to } A \subseteq \{0, A\}$ unfolding RestrictedTo_def
     by auto
     ultimately have $\top_\text{def}: T \text{ restricted to } A \subseteq \{0, A\}$ by auto
     moreover
     have $A$ is closed in $\{0, A\}$ unfolding IsClosed_def by auto
     moreover
     have $\{0, A\}$ is hyperconnected using Indiscrete_HConn by auto
     moreover
     from $\langle A \in \text{Pow}(\bigcup T) \rangle$ have $(T \text{ restricted to } A) \{\text{restricted to } A\} = T \{\text{restricted to } A\}$ unfolding subspace_of_subspace[of AAT]
     by auto
     moreover
     note $\langle A \neq 0 \rangle$. $\langle A \in \text{Pow}(\bigcup T) \rangle$
     ultimately have $A \in \text{Pow}(\bigcup (T \text{ restricted to } A)) \{-0\} A \text{ is closed in } (T \text{ restricted to } A) \!(\!(T \text{ restricted to } A) \{\text{restricted to } A\}(\!(T \text{ restricted to } A) \{\text{restricted to } A\} A \{\text{is hyperconnected}\}$
     by auto
     with $\langle (T \text{ restricted to } A) \{\text{is sober}\} \rangle$ have $\exists x \in \bigcup (T \text{ restricted to } A)$. $A = \text{Closure}(\{x\}, (T \text{ restricted to } A)) \land (\forall y \in \bigcup (T \text{ restricted to } A). A = \text{Closure}(\{y\}, (T \text{ restricted to } A)) \implies y = x)$ unfolding IsSober_def by auto
     with top_def have $\exists x \in A$. $A = \text{Closure}(\{x\}, \{0, A\}) \land (\forall y \in A. A = \text{Closure}(\{y\}, \{0, A\}) \implies y = x)$ by auto

909
then obtain $x$ where $x \in A = \text{Closure}(\{x\}, \{0, A\})$ and $\forall y \in A. A = \text{Closure}(\{y\}, \{0, A\})$  

$\rightarrow y = x$ by auto

\{
  \begin{align*}
    \text{fix } y & \text{ assume } y \in A \\
    \text{from } & <A \neq 0> \text{ have top:} \{0, A\} \text{is a topology} \text{ using indiscrete_ptopology[of A]}
    \text{indiscrete_partition[of A] Ptopology_is_a_topology(1)of } \{A\} \text{A} \\
    \text{by auto} \\
    \text{with } & <y \in A> \text{ have } \text{Closure}(\{y\}, \{0, A\}) \text{is closed in} \{0, A\} \text{ using topology0.cl_is_closed} \\
    \text{topology0_def by auto} \\
    \text{moreover} \\
    \text{from } & <y \in A> \text{ top have } y \in \text{Closure}(\{y\}, \{0, A\}) \text{ using topology0.cl_contains_set} \\
    \text{topology0_def by auto} \\
    \text{ultimately have } & A = \text{Closure}(\{y\}, \{0, A\}) \cap A \neq 0 \\
    \text{unfolding IsClosed_def by auto} \\
    \text{then have } & A = \text{Closure}(\{y\}, \{0, A\}) = A \text{ by auto} \\
    \text{simp, auto) \\
    \text{moreover} \\
    \text{from } & <y \in A> \text{ have } y \in \text{Closure}(\{y\}, \{0, A\}) \text{ by auto} \\
    \text{topology0.def by auto} \\
    \text{ultimately have } & A = \text{Closure}(\{y\}, \{0, A\}) \text{ by auto} \\
    \text{by auto} \\
    \text{ultimately have } & A = \text{Closure}(\{y\}, \{0, A\}) = 0 \text{ by (cases A = \text{Closure}(\{y\}, \{0, A\}) = A, simp, auto)} \\
    \text{simp, auto) \\
    \text{moreover} \\
    \text{from } & <y \in A> \text{ top have } \text{Closure}(\{y\}, \{0, A\}) \subseteq A \text{ using topology0_def} \\
    \text{topology0.Top_3_L11(1) by blast} \\
    \text{then have } & A = \text{Closure}(\{y\}, \{0, A\}) \text{ by auto} \\
    \text{ultimately have } & A = \text{Closure}(\{y\}, \{0, A\}) \text{ by auto} \\
    \text{by auto} \\
    \text{ultimately have } & A = \text{is in the spectrum of} IsSober \text{ using sober_spectrum by auto} \\
    \text{by auto} \\
    \text{ultimately have } & A = \text{is in the spectrum of} IsSober \text{ by auto} \\
    \text{by auto} \\
    \text{then show } & T = \text{is anti-} IsSober \text{ using antiProperty_def by auto} \\
    \text{next} \\
    \text{assume } & T = \text{is anti-} IsSober \\
    \text{by auto} \\
    \text{fix } A \\
    \text{assume } & A \in T \neq 0A \neq \bigcup T \\
    \text{then obtain } & x y \text{ where } x \in A \cup \bigcup T \neq x \neq y \text{ by blast} \\
    \text{then have } & (x) = (x, y) \cap A \text{ by auto} \\
    \text{with } & <A \in T> \text{ have } (x) \in \bigcup T \text{restricted to}\{x, y\} \text{ unfolding RestrictedTo_def by auto} \\
    \text{by auto} \\
    \text{assume } & \{y\} \subseteq \bigcup T \text{restricted to}\{x, y\} \\
    \text{from } & <y \in \bigcup T \neq A> \text{ have } x \in A \text{ and } \bigcup T \text{restricted to}\{x, y\} = \{x, y\} \\
\}
unfolding RestrictedTo_def
  by auto
  with \(x \neq y\), \(\{y\} \subseteq T\{\text{restricted to}\}{x,y}\), \(\{x\} \subseteq T\{\text{restricted to}\}{x,y}\)
  have \(T\{\text{restricted to}\}{x,y}\) is \(T_2\)
  unfolding isT2_def by auto
  then have \(T\{\text{restricted to}\}{x,y}\) is sober
  using topology0.T2_imp_anti_HConn[of T\{restricted to\}{x,y}]
  Top_1_L4 topology0_def topology0.anti_HConn_iff_T1_sober[of T\{restricted to\}{x,y}]
  by auto
moreover
  { assume \(\{y\} \notin T\{\text{restricted to}\}{x,y}\)
  moreover
  from \(\cup T - A\) have \(T\{\text{restricted to}\}{x,y}\) is Pow({x,y})
  unfolding RestrictedTo_def by auto
  then have \(T\{\text{restricted to}\}{x,y}\) is \(0,\{x\},\{y\},\{x,y\}\)
  by blast
  moreover
  note \(\{x\} \subseteq T\{\text{restricted to}\}{x,y}\)
  empty_open[0F Top_1_L4[of \{x,y\}]]
  moreover
  from \(\cup T - A\) have \(\{x\},\{y\}\) empty_open
  unfolding RestrictedTo_def by auto
  from Top_1_L4[of \{x,y\}]
  have \(\cup (T\{\text{restricted to}\}{x,y})\) in T\{restricted to\}{x,y}
  unfolding isATopology_def
  by auto
  with \(\{x\} \subseteq T\{\text{restricted to}\}{x,y}\)
  have \(\{x\} \subseteq T\{\text{restricted to}\}{x,y}\)
  unfolding closure_by_open by auto
  ultimately have \(\{x\} \subseteq T\{\text{restricted to}\}{x,y}\)
  by auto
  { fix B assume \(B \subseteq \{x,y\}\)
  with \(\{x\} \subseteq T\{\text{restricted to}\}{x,y}\)
  have \(\cup (T\{\text{restricted to}\}{x,y})\) is closed in \(T\{\text{restricted to}\}{x,y}\)
  unfolding isClosed_def by simp
  moreover
  have \(\{x\} \subseteq T\{\text{restricted to}\}{x,y}\)
  unfolding closure_by_open by auto
  ultimately have \(\{x\} \subseteq T\{\text{restricted to}\}{x,y}\)
  by auto
  { assume \(\{x\} \subseteq T\{\text{restricted to}\}{x,y}\)
  then have \(\{x\} \subseteq T\{\text{restricted to}\}{x,y}\)
  unfolding isClosed_def
  using \(\{x\} \subseteq T\{\text{restricted to}\}{x,y}\)
  by auto
  moreover
  from \(x \neq y\)
  have \(\{x\} \subseteq T\{\text{restricted to}\}{x,y}\)
  by auto
ultimately have \(\{y\} \subseteq T\{\text{restricted to}\}{x,y}\)
  by auto

911
then have False using \( \{y\} \not\in (T{\text{restricted to}}{\{x,y\}}) \) by auto

then have \( \neg(\{x\}{\text{is closed in}}(T{\text{restricted to}}{\{x,y\}})) \) by auto

moreover
from tot have (Closure({x},T{\text{restricted to}}{\{x,y\}}){\text{is closed in}}(T{\text{restricted to}}{\{x,y\}})

using topology0.cl_is_closed unfolding topology0_def using Top_1_L4[of {x,y}]

tot by auto

ultimately have \( \neg(\text{Closure}({\{x\},T{\text{restricted to}}{\{x,y\}}})=\{x\}) \) by auto

moreover note xin topology0.Top_3_L11(1)[of T{\text{restricted to}}{\{x,y\}}]
tot

ultimately have cl_x:Closure({x},T{\text{restricted to}}{\{x,y\}})=\{x,y\}

unfolding topology0_def

using Top_1_L4[of \{x,y\}] by auto

have \{y\}{\text{is closed in}}(T{\text{restricted to}}{\{x,y\}}) unfolding IsClosed_def

using tot

tot_d_def \( \langle x \neq y \rangle \) by auto

then have cl_y:Closure({y},T{\text{restricted to}}{\{x,y\}})=\{y\} using topology0.Top_3_L8[of T{\text{restricted to}}{\{x,y\}}]

unfolding topology0_def using Top_1_L4[of \{x,y\}] tot by auto

\{ assume \( \langle x,y\rangle - B = 0 \)

with \( \langle B \in \text{Pow}(\{x,y\}) - \{0\} \rangle \) have B:{x,y}=B by auto

{ fix m
assume dis:m\(\in\){x,y} and B_def:B=Closure({m},T{\text{restricted to}}{\{x,y\}})

\{ assume m=y

with B_def have B=Closure({y},T{\text{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 \( (\forall m \in \{x,y\}.\ B = \text{Closure}(\{m\},T{\text{restricted to}}\{x,y\}) \rightarrow m = x) \) by auto

moreover

have B=Closure({x},T{\text{restricted to}}{\{x,y\}}) using cl_x B by auto

ultimately have \( \exists t \in \{x,y\}.\ B = \text{Closure}(\{t\},T{\text{restricted to}}\{x,y\}) \)

\( \wedge (\forall m \in \{x,y\}.\ B = \text{Closure}(\{m\},T{\text{restricted to}}\{x,y\}) \rightarrow m = t) \) by auto

912
moreover

{  
  assume \{x,y\}-B \neq 0
  with \{x,y\}-B \in \{0,\{x\},\{x,y\}\}  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 \(B\in\{\{x\},\{y\},\{x,y\}\}\) \(x\neq y\) 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 \(x\neq y\) have False by auto
        
      }  
      with dis have m=y by auto
      
      {  
        moreover
        have B=Closure(\{y\},T\{restricted to\}\{x,y\}) using cl_y B by auto
        ultimately have \(\exists t\in\{x,y\}. \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\}) \rightarrow m=t )
        by auto
      }  
      moreover
      
      {  
        assume \{x,y\}-B \neq \{x\}
        with or have \{x,y\}-B=\{x,y\} by auto
        then have \(x\in\{x,y\}\)-By\in\(\{x,y\}\)-B by auto
        with \(B\in\{\{x\},\{y\},\{x,y\}\}\) \(x\neq y\) have False 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\}) \rightarrow m=t )
      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\}) \rightarrow m=t )
    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\}) \rightarrow m=t )
  by auto
}

913
then have \((T\{restricted \to \{x,y\}\}\{is \ sober\})\) unfolding \mbox{IsSober\_def}
using tot by auto 
}
ultimately have \((T\{restricted \to \{x,y\}\}\{is \ sober\})\) by auto
with \(<T\{is \ anti-\}IsSober>\) have \{x,y\}\{is \ in \ the \ spectrum \ of\}\mbox{IsSober}
unfolding \mbox{antiProperty\_def}
using \(<x \in A \Rightarrow A \in T \Rightarrow y \in \bigcup T-A>\) by auto
then have \(\{x,y\}\leq 1\) using \mbox{sober\_spectrum} by auto
moreover
have \(x \in \{x,y\}\) by auto
ultimately have \(\{x,y\}=\{x\}\) using \mbox{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 \(\mbox{False}\) using \(<x \neq y>\) by auto
}
then have \(T \subseteq \{0,\bigcup T\}\) by auto
with \mbox{empty\_open} OF \mbox{topSpaceAssum} \mbox{topSpaceAssum} show \(T=\{0,\bigcup T\}\) unfolding \mbox{IsATopology\_def}
by auto
qed

end

68 Topology 8

theory TopologyZF\_8 imports TopologyZF\_6 EquivClass1
begin

This theory deals with quotient topologies.

68.1 Definition of quotient topology

Given a surjective function \(f: X \to Y\) and a topology \(\tau\) in \(X\), it is possible to consider a special topology in \(Y\). \(f\) is called quotient function.

definition (in topology0)

QuotientTop (\{quotient topology in\}_{by}_ 80)
where \(f\in\mbox{surj}(\bigcup T,Y) \Rightarrow \{\mbox{quotient topology in}\}Y\{by\}f\equiv\{U\in\mbox{Pow}(Y). \ f-\in U\}

abbreviation QuotientTopTop (\{quotient topology in\}_{by}.\{from\}_)
where QuotientTopTop(Y,f,T) \equiv \mbox{topology0}.QuotientTop(T,Y,f)

The quotient topology is indeed a topology.

definition (in topology0)

QuotientTop_is_top:
assumes \(f\in\mbox{surj}(\bigcup T,Y)
shows (\{\mbox{quotient topology in}\} Y \{by\} f) \{is \ a \ topology\}
proof-
have \((\text{quotient topology in} \ Y \ \{\text{by} \ f\}) = \{U \in \text{Pow}(Y) . f - U \in T\}\) using QuotientTop_def assms
  by auto
moreover
{ fix M \times B assume M:M \subseteq \{U \in \text{Pow}(Y) . f - U \in T\}
then have \(\bigcup M \subseteq Y\) by blast
moreover
have A1:\(f - (\bigcup M) = \bigcup \{y \in \bigcup M . f - \{y\}\}\) using vimage_eq_UN by blast
}
fix A assume A\in M
with M have A\inPow(Y) f - A\in T by auto
have f - A = \(\bigcup \{y \in A . f - \{y\}\}\) using vimage_eq_UN by blast
}
then have \((\bigcup A \in M . f - A) = \bigcup \{y \in A . f - \{y\}\}\) by auto
then have \((\bigcup A \in M . f - A) = \bigcup \{y \in M . f - \{y\}\}\) by auto
with A1 have A2:\(f - (\bigcup M) = \bigcup \{f - A . A \in M\}\) by auto
{ fix A assume A\in M
with M have f - A\in T by auto
}
then have \(\forall A \in M . f - A \in T\) by auto
then have \(\{f - A . A \in M\} \subseteq T\) by auto
then have \((\bigcup \{f - A . A \in M\}) \in T\) using topSpaceAssum unfolding IsATopology_def
by auto
with A2 have \((f - (\bigcup M)) \in T\) by auto
ultimately have \(\bigcup M \in \{U \in \text{Pow}(Y) . f - U \in T\}\) by auto
moreover
{ fix U V assume U\in\{U \in \text{Pow}(Y) . f - U \in T\} V\in\{U \in \text{Pow}(Y) . f - U \in T\}
then have U\inPow(Y)V\inPow(Y)f - U \in Tf - V \in T by auto
then have \((f - U) \cap (f - V) \in T\) using topSpaceAssum unfolding IsATopology_def
by auto
then have f - (U \cap V) \in T using invim_inter_inter_invim assms unfolding surj_def
by auto
with \(<U \in \text{Pow}(Y)>.<V \in \text{Pow}(Y)>. U \cap V \in \{U \in \text{Pow}(Y) . f - U \in T\}\> by auto
}
ultimately show thesis using IsATopology_def by auto
qed

The quotient function is continuous.

lemma (in topology0) quotient_func_cont:
  assumes f\in surj(\bigcup T,Y)
  shows IsContinuous(T,\{\text{quotient topology in} \ Y \ \{\text{by} \ f\}\),f)
  unfolding 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.

915
Theorem (in two_top_spaces0) cont_quotient_top:
  assumes h:surj(∪τ₁,Y) g:Y→∪τ₂ IsContinuous(τ₁,τ₂,g 0 h)
  shows IsContinuous({quotient topology in} Y {by} h {from} τ₁),τ₂,g)
proof-
  { fix U assume U∈τ₂
    with assms(3) have (g 0 h)-(U)∈τ₁ unfolding IsContinuous_def by auto
    then have h-(g-(U))∈τ₁ using vimage_comp by auto
    then have g-(U)∈({quotient topology in} Y {by} h {from} τ₁) using
    topology0.QuotientTop_def
      tau1_is_top assms(1) using func1_1_L3 assms(2) unfolding topology0_def
    by auto
  } then show thesis unfolding IsContinuous_def by auto
qed

68.2 Quotient topologies from equivalence relations

In this section we will show that the quotient topologies come from an
equivalence relation.

First, some lemmas for relations.

Lemma (in topology0) total_quo_func:
  assumes f:surj(∪T,Y)
  shows (∪({quotient topology in}Y{by}f))=Y
proof-
  from assms have f-Y=∪T using func1_1_L4 unfolding surj_def by auto
  moreover have ∪T∈T using topSpaceAssum unfolding IsATopology_def by auto ultimately
  have Y∈({quotient topology in}Y{by}f{from}T) using QuotientTop_def
    assms by auto
  then show thesis using QuotientTop_def assms by auto
qed

916
with \( A(1) \) have \( \exists y \in A. \{ (b, r\{b\}). b \in A \} yy = y \) by auto

with quotient_proj_fun show thesis unfolding surj_def by auto qed

lemma preim_equi_proj:
  assumes \( U \subseteq A // r \) equiv\( (A, r) \)
  shows \( \{ (b, r\{b\}). b \in A \} - U = \bigcup U \)
proof
  
  fix y assume \( y \in \bigcup U \)
  then obtain \( V \) where \( V: y \in V \subseteq U \) by auto
  moreover from \( U \subseteq (A // r) \) \( V \) have \( r\{y\} = V \) using EquivClass_1_L2 assms(2) by auto
  moreover note \( V(2) \) ultimately have \( y \in \{ x \in A. r\{x\} \subseteq U \} \) by auto
  then have \( y \in \{ (b, r\{b\}). b \in A \} - U \) by auto
  
  then show \( \bigcup U \subseteq \{ (b, r\{b\}). b \in A \} - U \) by blast
qed

Now we define what a quotient topology from an equivalence relation is:

definition (in topology0)
  EquivQuo (_quotient by_) 70
  where equiv\( (\bigcup T, r) \) = (\( \{ \{ \text{quotient by} \} \) r \) \( \equiv \) \{ quotient topology in\} (\( \bigcup T \))//r\{by\}\{ (b, r\{b\}). b \in \bigcup T \}

abbreviation
  EquivQuoTop (_quotient by_) 60
  where EquivQuoTop(T, r)\equiv topology0.EquivQuo(T, r)

First, another description of the topology (more intuitive):

theorem (in topology0) quotient_equiv_rel:
  assumes \( \text{equiv}(\bigcup T, r) \)\( \Rightarrow \) \( (\{ \text{quotient by} \} r) \equiv \{ \text{quotient topology in} \} (\bigcup T) // r\{by\}\{ (b, r\{b\}). b \in \bigcup T \}
proof
  have \( (\{ \text{quotient topology in} \} (\bigcup T) // r\{by\}\{ (b, r\{b\}). b \in \bigcup T \}) = U \subseteq \text{Pow}( (\bigcup T) // r) \) \( \cup U \subseteq T \)
  using QuotientTop_def quotient_proj_surj by auto
  moreover have \( U \subseteq \text{Pow}( (\bigcup T) // r) \) \( \{ (b, r\{b\}). b \in \bigcup T \} - U \subseteq T \)
  have \( U \subseteq \text{Pow}( (\bigcup T) // r) \) \( \{ (b, r\{b\}). b \in \bigcup T \} - U \subseteq T \)
  \( \cup U \subseteq T \)

917
proof
{ fix U assume U∈Pow(∪T)//r). {b,r{b}). b∈∪T}-U∈T
then have U∈Pow(∪T)//r). ∪U∈T} using preim_equi_proj assms
by auto
} then show {U∈Pow(∪T)//r). {b,r{b}). b∈∪T}-U∈T}⊆U∈Pow(∪T)//r).
∪U∈T by auto
{ fix U assume U∈Pow(∪T)//r). ∪U∈T
then have U∈Pow(∪T)//r). {b,r{b}). b∈∪T}-U∈T using preim_equi_proj
assms by auto
} then show {U∈Pow(∪T)//r). ∪U∈T}⊆U∈Pow(∪T)//r). {b,r{b}). b∈∪T}-U∈T} by auto
qed
ultimately show thesis using EquivQuo_def assms by auto
qed

We apply previous results to this topology.

theorem (in topology0) total_quo_equi:
assumes equiv(∪T,r)
shows ∪({quotient by}r)=(∪T)//r
using total_quo_func quotient_proj_surj EquivQuo_def assms by auto

theorem (in topology0) equiv_quo_is_top:
assumes equiv(∪T,r)
shows ({quotient by}r){is a topology}
using quotientTop_is_top quotient_proj_surj EquivQuo_def assms by auto

MAIN RESULT: 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∈surj(∪T,Y)
defines r≡{(x,y)∈∪T×∪T. f(x)=f(y)}
defines g≡{y,f-{y}). y∈Y}
sshows equiv(∪T,r) and IsAhomeomorphism(({quotient topology in}Y{by}f),({quotient
by}r),g)
proof-
have ff:f:[∪T→Y using assms(1) unfolding surj_def by auto
show B:equiv(∪T,r) unfolding equiv_def refl_def sym_def trans_def
unfolding r_def by auto
have gg:g:Y→(∪T)//r)
proof-
{ fix B assume B∈g
then obtain y where Y:y∈Y B=⟨y,f-{y}⟩ unfolding g_def by auto

918
then have \(f^{-\{y\}} \subseteq \bigcup T\) using \(\text{func1}_1\_L3\) \(ff\) by blast
then have \(eq: f^{-\{y\}} = \{x \in \bigcup T. (x, y) \in f\}\) using \(\text{vimage\_iff}\) by auto
from \(Y\) obtain \(A\) where \(A1: A \in \bigcup T\) \(fA = y\) using \(\text{assms(1)}\) unfolding \(\text{surj\_def}\)
by blast
with \(eq\) have \(A: A \in f^{-\{y\}}\) using \(\text{apply\_Pair[OF ff]}\) by auto

\(\{\)
  \(\text{fix } t\) assume \(t \in f^{-\{y\}}\)
  with \(A\) have \(t \in \bigcup T\) \(t, y \in f\) using \(eq\) by auto
  then have \(ft = fA\) using \(\text{apply\_equality assms(1)}\) unfolding \(\text{surj\_def}\)
\(\}\)

by auto
with \(<t \in \bigcup T, A \in \bigcup T>\) have \((A, t) \in r\) using \(\text{r\_def}\) by auto
then have \(t \in r\{A\}\) using \(\text{image\_iff}\) by auto
\(\{\)
  \(\text{fix } t\) assume \(t \in r\{A\}\)
  then have \(f^{-\{y\}} \subseteq r\{A\}\) by auto
\(\}\)
moreover
from \(un\) have \((t, ft) \in f\) using \(\text{apply\_Pair[OF ff]}\) by auto
with \(eq2\) \(A1\) have \((t, y) \in f\) by auto
\(\{\)
  \(\text{fix } t\) assume \(t \in r\{A\}\)
  then have \(A, t) \in r\) using \(\text{image\_iff}\) by auto
  then have \(un: t \in \bigcup T\) \(A \in \bigcup T\) \((A, t) \in r\) using \(\text{eq2, assms(1)}\) unfolding \(\text{surj\_def}\)
moreover
from \(un\) have \((t, ft) \in f\) using \(\text{apply\_Pair[OF ff]}\) by auto
with \(eq2\) \(A1\) have \((t, y) \in f\) by auto
\(\{\)
  \(\text{fix } t\) assume \(t \in r\{A\}\)
  then have \(f^{-\{y\}} \subseteq r\{A\}\) by auto
\(\}\)
ultimately have \(f^{-\{y\}} = r\{A\}\) by auto
then have \(f^{-\{y\}} \in (\bigcup T) / r\) using \(A1(1)\) unfolding \(\text{quotient\_def}\)
then have \(\bigcup T / r \subseteq Y \times (\bigcup T) / r\) by auto
\(\{\)
  \(\text{fix } s\) assume \(S: s \in (\{\text{quotient topology in} Y\} \{by\} f)\)
  then have \(s \in \text{Pow}(Y)\) and \(P: s \in T\) using \(\text{QuotientTop\_def topSpaceAssum assms(1)}\)
by auto
qed
then have \(gg2: g: Y \to (\bigcup \{\text{quotient by} r\})\) using \(\text{total\_quo\_equi B}\) by auto
\(\{\)
  \(\text{fix } s\) assume \(S: s \in (\{\text{quotient topology in} Y\} \{by\} f)\)
  then have \(s \in \text{Pow}(Y)\) and \(P: s \in T\) using \(\text{QuotientTop\_def topSpaceAssum assms(1)}\)
by auto
have \(f-s=(\bigcup y \in s. f^{-\{y\}})\) using \(\text{vimage\_eq\_UN}\) by blast moreover
from \(<s \in \text{Pow}(Y)>\) have \(\forall y \in s. (y, f^{-\{y\}}) \in g\) unfolding \(g\_def\) by auto
then have \(\forall y \in s. gy = f^{-\{y\}}\) using \(\text{apply\_equality gg}\) by auto ultimately
have \(f-s=(\bigcup y \in s. gy)\) by auto
with \(P\) have \((\bigcup y \in s. gy) \in T\) by auto moreover
from \(<s \in \text{Pow}(Y)>\) have \(\forall y \in s. gy \in (\bigcup T) / r\) using \(\text{apply\_type gg}\) by auto
ultimately have \(\{gy. y \in s\} \in (\{\text{quotient by} r\})\) using \(\text{quotient\_equiv\_rel}\)
B by auto
with \(<s \in \text{Pow}(Y)>\) have \(gs \in (\{\text{quotient by} r\})\) using \(\text{func\_imagedef gg}\) by
919
then have gopen:∀s∈({quotient topology in}Y(by)f). gs∈(T{quotient 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 bY:fb∈Y using apply_funtype ff b by auto
  with b have com:∃(g 0 f)b=g(fb) using comp_fun_apply ff by auto
  from bY have pg:g(fb)=f-(fb) using apply_equality gg by auto
  with com have comeq:∃(g O f)b=f-(fb) by auto
  from b have A:f{b}={fb} {b}⊆⋃T using func_imagedef ff by auto
  from A(2) have b∈f - (f {b}) using func1_1_L9 ff by blast
  then have b∈f-({fb}) using A(1) by auto moreover
  from pg have f-({fb})∈(⋃T)//r using gg by auto
  ultimately have r(b)=f-({fb}) using EquivClass_1_L2 B by auto
  then have (g 0 f)b=r{b} using comeq by auto moreover
  from b have ⟨b,r{b}⟩∈{⟨b,r{b}⟩. b∈⋃T} by auto
  with pr_fun have {⟨b,r{b}⟩. b∈⋃T}b=r{b} using apply_equality by auto
ultimately have (g 0 f)b→(b,r{b}). b∈⋃Tb by auto
} then have reg:∀b∈⋃T. (g 0 f)b→(b,r{b}). b∈⋃Tb by auto moreover
have compost:∀f∈⋃T→(⋃T)//r using comp_fun ff gg by auto
have feq:(g O f)=f using fun_extension[OF compost pr_fun]
reg by auto
then have IsContinuous(T, {quotient by}r, (g 0 f)) using quotient_func_cont
quotient_proj_surj
EquivQuo_def topSpaceAssum B by auto moreover
have (g 0 f):⋃T→(⋃T)//r using comp_fun ff gg2 by auto
ultimately have gcont:IsContinuous(⋃(T{quotient by}r), g)
using two_top_spaces0.quotient_top assm(1) gg2 unfolding two_top_spaces0_def
using topSpaceAssum equiv_quo_is_top B by auto
{ fix x y assume T:x∈Yy∈Ygx=gy
  then have f-⟨x⟩=f-⟨y⟩ using apply_equality gg unfolding g_def by auto
  then have f(f-⟨x⟩)=f(f-⟨y⟩) by auto
  with T(1,2) have {x}={y} using surj_image_vimage assm(1) by auto
  then have x=y by auto
} with gg2 have g∈inj(Y,⋃(quotient by)r)) unfolding inj_def by auto
moreover
have (g 0 f):surj(⋃T, (⋃T)//r) using eqf quotient_proj_surj by auto
then have g∈surj(Y,⋃T//r) using comp_mem_surjD1 ff gg by auto
then have g∈surj(Y,⋃(T{quotient by}r)) using total_quo_equi B by auto

920
ultimately have \( g \in \text{bij}(\bigcup\{(\text{quotient topology in } Y \text{ by } f), \bigcup\{(\text{quotient by } r)\}) \) unfolding bij_def using total_quo_func assms(1) by auto

with gcont gopen show \( \text{IsAhomeomorphism}(\bigcup\{(\text{quotient topology in } Y \text{ by } f), \bigcup\{(\text{quotient by } r)\}, g) \)

using bij_cont_open_homeo by auto

qed

lemma product_equiv_rel_fun:
  shows \( \{\langle\langle b, c \rangle, \langle r\{b\}, r\{c\}\rangle\rangle. \langle b, c \rangle \in \bigcup T \times \bigcup T\} : (\bigcup (\bigcup T) \times (\bigcup T)) \rightarrow (\bigcup (\bigcup T) \times (\bigcup T)) \)

proof

  have \( \{\langle b, r\{b\}\rangle. b \in \bigcup T\} : (\bigcup T) \rightarrow (\bigcup T) \)

  unfolding quotient_def using quot_proj_fun by auto

  moreover have \( \forall A \in \bigcup T. \langle A, r\{A\}\rangle \in \{\langle b, r\{b\}\rangle. b \in \bigcup T\} \)

  unfolding quotient_def using quot_proj_fun by auto

  ultimately have \( \forall A \in \bigcup T. \{\langle b, r\{b\}\rangle. b \in \bigcup T\} A = r\{A\} \)

  using apply_equality by auto

  then have \( \bigcup T \times \bigcup T \) using surj_def unfolding Top_1_4_T1(3) topSpaceAssum by auto

qed

lemma (in topology0) product_quo_fun:

  shows \( \{\langle\langle b, c \rangle, \langle r\{b\}, r\{c\}\rangle\rangle. \langle b, c \rangle \in \bigcup T \times \bigcup T\} : (\bigcup (\bigcup T) \times (\bigcup T)) \rightarrow (\bigcup (\bigcup T) \times (\bigcup T)) \)

proof

  have \( \{\langle b, r\{b\}\rangle. b \in \bigcup T\} : (\bigcup T) \rightarrow (\bigcup T) \)

  by force

  then show thesis using prod_fun quotient_proj_fun by auto

qed
assumes equiv(⋃T,r)
shows IsContinuous(ProductTopology(T,T),ProductTopology({quotient by}r,({quotient by}r)),{(b,c),(r{b},r{c})}. (b,c)∈⋃T×⋃T)

proof-

have {⟨b,r{b}⟩. b∈⋃T}. b∈⋃T: U→(⋃T)//r using quotient_proj_fun by auto
moreover
have ∀A∈⋃T. {(b,r{b}). b∈⋃T}∈(quotient by)r using quotient_proj_surj by auto
ultimately
have ∀A∈⋃T. {⟨b,r{b}⟩. b∈⋃T}∈{quotient by}r using apply_equality by auto
then have IN: {⟨⟨b,c⟩,⟨r{b},r{c}⟩⟩. ⟨b,c⟩∈⋃T×⋃T}= {⟨⟨x,y⟩,⟨b,c⟩⟩. ⟨b,c⟩∈⋃T}. ⟨x,y⟩∈⋃T×⋃T}
by force
have cont:IsContinuous(T,({quotient by}r,({quotient by}r))) using quotient_func_cont quotient_proj_surj
EquivQuo_def assms by auto
have tot: ⋃(T{quotient by}r) = (⋃T)//r and top:({quotient by}r) is a topology using total_quo_equi equiv_quo_is_top assms by auto
then have fun:{⟨b,r{b}⟩. b∈⋃T}: ⋃T→(⋃(T{quotient by}r)) using quotient_proj_fun by auto
then have two:two_top_spaces0(T,({quotient by}r,({quotient by}r))) unfolding two_top_spaces0_def using topology0.topSpaceAssum assms by auto
show thesis using two_top_spaces0.product_cont_functions two fun fun cont cont top topSpaceAssum IN 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 equiv(⋃T,r)
∀A∈⋃T. {(b,r{b}). b∈⋃T}∈(quotient by)r
shows (ProductTopology({quotient by}r,({quotient by}r))) = (quotient topology in)(((⋃T)//r)×((⋃T)//r)) by {⟨⟨b,c⟩,⟨r{b},r{c}⟩⟩. ⟨b,c⟩∈⋃T×⋃T}}{from}(ProductTopology(T,T))

proof
fix A assume A:A∈ProductTopology({quotient by}r,({quotient by}r)
from assms have IsContinuous(ProductTopology(T,T),ProductTopology({quotient by}r,({quotient by}r)),{(b,c),(r{b},r{c})}. (b,c)∈⋃T×⋃T) using product_quo_fun by auto
with A have {{⟨b,c⟩,⟨r{b},r{c}⟩}}. (b,c)∈⋃T×⋃T-A∈ProductTopology(T,T)
unfolding IsContinuous_def by auto
moreover
from A have A∈⋃ProductTopology(T,T)={quotient by}r,T(quotient by)r by auto

ultimately have A∈(quotient topology in)(((⋃T)//r)×((⋃T)//r)) by {⟨⟨b,c⟩,⟨r{b},r{c}⟩⟩} by {⟨⟨b,c⟩,⟨r{b},r{c}⟩⟩} (b,c)∈⋃T×⋃T, from Topology0.ProductTopology(T,T)
using topology0.QuotientTop_def Top_1_4_T1(1) topSpaceAssum prod_equiv_rel_surj
assms(1) unfolding topology0_def by auto
} then show ProductTopology(T{quotient by}r,T{quotient by}r)≤{(quotient topology in}((∪T)//r)×((∪T)//r)}{b}{((b,c),(r{b},r{c}))}. (b,c)∈∪T×∪T{from}(ProductTopology(T,T))
by auto
{ fix A assume A∈{(quotient topology in}((∪T)//r)×((∪T)//r)}{b}{((b,c),(r{b},r{c}))}. (b,c)∈∪T×∪T{from}(ProductTopology(T,T))
then have A:A≤((∪T)//r)×((∪T)//r) {((b,c),(r{b},r{c}))}. (b,c)∈∪T×∪T-A∈ProductTopology(T,T)
using topology0.quotient_top_def Top_1_4.T1(1) topSpaceAssum prod_equiv_rel_surj
assms(1) unfolding topology0_def by auto
{ fix CC assume CC∈A
with A(1) obtain C1 C2 where CC:CC=(C1,C2) C1∈((∪T)//r)C2∈((∪T)//r)
by auto
then obtain c1 c2 where CC1:c1∈∪TCc2∈∪T and CC2:C1=r{c1}C2=r{c2}
unfolding quotient_def
by auto
then have ⟨c1,c2⟩∈∪T×∪T by auto
then have ⟨⟨c1,c2⟩,(r{c1},r{c2})⟩∈{(b,c),(r{b},r{c}}). (b,c)∈∪T×∪T
by auto
with CC2 CC have ⟨⟨c1,c2⟩,CC⟩∈{(b,c),(r{b},r{c}}). (b,c)∈∪T×∪T
by auto
with CC∈A have ⟨c1,c2⟩∈{(b,c),(r{b},r{c}}). (b,c)∈∪T×∪T-A
using vimage_iff by auto
with A(2) have ∃V W. V ∈ T ∧ W ∈ T ∧ V × W ⊆ {((b,c),(r{b},r{c}}).
(b,c)∈∪T×∪T-A ∧ ⟨c1,c2⟩ ∈ V × W
using prod_top_point_neighb topSpaceAssum by blast
then obtain V W where VW:V∈T∧V T × W ⊆ {((b,c),(r{b},r{c}}). (b,c)∈∪T×∪T-Ac1∈Vc2∈W
by auto
with assms(2) have {(b,r{b})}. b∈∪T)V∈T{quotient by}r}{b,r{b}}.
b∈∪T)c∈∪T{quotient by}r by auto
then have P:⟨b,r{b})}. b∈∪T)V∈∪T{quotient by}r) ProductTopology(T{quotient by}r,T{quotient by}r) using prod_open_open_prod equiv_quo_is_top
assms(1) by auto

{ fix S assume SC∈{(b,r{b})}. b∈∪T)V∈∪T{quotient by}r}{b,r{b}}.
b∈∪T}W by blast
then obtain t1 t2 where T:{t1,s1}∈{(b,r{b})}. b∈∪T)(t2,s2)∈{(b,r{b}).
b∈∪T)t1∈∪Tv∈∪T using image_iff by auto
then have ⟨t1,t2⟩∈V×W by auto
with VW(3) have ⟨t1,t2⟩∈{(b,c),(r{b},r{c}}). (b,c)∈∪T×∪T-A
by auto
then have ∃SS∈A. ⟨⟨t1,t2⟩,SS⟩∈{(b,c),(r{b},r{c}}). (b,c)∈∪T×∪T
using vimage_iff by auto
then obtain SS where SS∈A⟨t1,t2⟩,SS⟩∈{(b,c),(r{b},r{c}}). (b,c)∈∪T×∪T
by auto moreover
from T VW(1,2) have ⟨t1,t2⟩∈∪T×∪T{s1,s2}=⟨r{t1},r{t2}} by auto
with $S(1)$ have $\langle t_1, t_2 \rangle, S(1) \in \{\langle b, c \rangle, \langle r\{b\}, r\{c\} \rangle\}$. $b, c \in \cup T \times \cup T$

ultimately have $S \in A$ using product_equiv_rel_fun unfolding $\Pi$ def

function_def

by auto

$\}$

then have sub: $\langle b, r\{b\} \rangle, b \in \cup T \cup W \subseteq A$ by blast

have $\langle c_1, C_1 \rangle \in \{\langle b, r\{b\} \rangle, b \in \cup T \cup C_2 \in \{\langle b, r\{b\} \rangle, b \in \cup T\} \text{ using } CC$  

by auto

with $<c_1 \in V, c_2 \in W>$ have $C_1 \in \{\langle b, r\{b\} \rangle, b \in \cup T \cup V \times \{\langle b, r\{b\} \rangle, b \in \cup T \cup W \text{ using } CC$ by auto

with sub $P$ have $\exists OO \in ProductTopology(T\{quotient by\}r, T\{quotient by\}r)$. $CC \subseteq A$  

using $\prod$ where $x=\{\langle b, r\{b\} \rangle, b \in \cup T \cup V \times \{\langle b, r\{b\} \rangle, b \in \cup T \cup W \text{ using } CC$ by auto

$\}$

then have $\forall C \in A. \exists OO \in ProductTopology(T\{quotient by\}r, T\{quotient by\}r)$. $CC \subseteq A$ by auto

then have $A \in ProductTopology(T\{quotient by\}r, T\{quotient by\}r)$ using $\prod$. $OO \subseteq A$ by auto

$\}$

then show $(\{quotient topology in\}(\cup T)/r) \times (\cup T)/r) \{by\}(\{b, c\}, \{r\{b\}, r\{c\}\})$.  

$\langle b, c \rangle \in \cup T \times \cup T\{from\}(ProductTopology(T, T)) \subseteq ProductTopology(T\{quotient by\}r, T\{quotient by\}r)$ by auto

qed

end

69 Topology 9

theory Topology_ZF_9

imports Topology_ZF_2 Group_ZF_2 Topology_ZF_7 Topology_ZF_8

begin

69.1 Group of homeomorphisms

This theory file deals with the fact that the set of homeomorphisms of a topological space into itself forms a group.

First, we define the set of homeomorphisms.

definition
The homeomorphisms are closed by composition.

**Lemma (in topology0)** homeo_composition:

- assumes \( f \in \text{HomeoG}(T) \) \( g \in \text{HomeoG}(T) \)
- shows \( \text{Composition}(\bigcup T)(f, g) \in \text{HomeoG}(T) \)

**Proof**

- from \( \text{assms} \) have \( \text{fun} : \bigcup T \to \bigcup T \) and \( \text{homeo:} \text{IsAhomeomorphism}(T, T, f) \text{IsAhomeomorphism}(T, T, g) \)
- unfolding \( \text{HomeoG_def} \) by \( \text{auto} \)
- from \( \text{fun} \) have \( \text{f \ O \ g} \in \bigcup T \to \bigcup T \) using \( \text{comp_fun} \) by \( \text{auto} \)
- moreover from \( \text{homeo} \) have \( \text{bij:} \text{bij}(\bigcup T, \bigcup T), g\text{bij}(\bigcup T, \bigcup T) \) and \( \text{cont:} \text{IsContinuous}(T, T, f) \text{IsContinuous}(T, T, g) \)
- and \( \text{contconv:} \text{IsContinuous}(T, T, \text{converse}(f)), \text{IsContinuous}(T, T, \text{converse}(g)) \)
- unfolding \( \text{IsAhomeomorphism_def} \) by \( \text{auto} \)
- from \( \text{bij} \) have \( \text{f \ O \ g} \in \text{bij}(\bigcup T, \bigcup T) \) using \( \text{comp_bij} \) by \( \text{auto} \)
- moreover from \( \text{cont} \) have \( \text{IsContinuous}(T, T, f \ O \ g) \)
- and \( \text{contconv:} \text{IsContinuous}(T, T, \text{converse}(f \ O \ g)) \)
- unfolding \( \text{HomeoG_def} \) \( \text{IsAhomeomorphism_def} \) by \( \text{auto} \)
- ultimately have \( \text{f \ O \ g} \in \text{HomeoG}(T) \)
- unfolding \( \text{HomeoG_def} \) \( \text{IsAhomeomorphism_def} \) by \( \text{auto} \)
- with \( \text{contconv} \) have \( \text{f \ O \ g} = \text{converse}(g) \text{O} \text{converse}(f) \)
- unfolding \( \text{converse_comp} \) by \( \text{auto} \)
- with \( \text{cont} \) have \( \text{IsContinuous}(T, T, \text{converse}(f \ O \ g)) \)
- unfolding \( \text{HomeoG_def} \) \( \text{IsAhomeomorphism_def} \) by \( \text{auto} \)
- ultimately have \( \text{f \ O \ g} \in \text{HomeoG}(T) \)
- unfolding \( \text{func_ZF_5_L2} \) \( \text{fun} \) by \( \text{auto} \)

**Theorem (in topology0)** homeo_submonoid:

- shows \( \text{IsAmonoid(HomeoG(T), restrict(Composition(\bigcup T), HomeoG(T) \times HomeoG(T)))} \)

- TheNeutralElement(HomeoG(T), restrict(Composition(\bigcup T), HomeoG(T) \times HomeoG(T))) = \text{id}(\bigcup T) \)

**Proof**

- have \( \text{cl:HomeoG(T) \ {is closed under} Composition(\bigcup T) \ unfolding} \text{IsOpClosed_def using} \text{homeo_composition by auto} \)
- moreover have \( \text{sub:HomeoG(T) \subseteq} \bigcup T \to \bigcup T \ unfolding} \text{HomeoG_def by auto} \)

925
The homeomorphisms form a group, with the composition.

**Theorem (in topology0)** homeo_group:
shows IsAgroup(HomeoG(T),restrict(Composition(∪T),HomeoG(T)×HomeoG(T)))

**Proof**
{  
  fix x assume AS:x∈HomeoG(T)
  then have surj:x∈surj(∪T,∪T) and bij:x∈bij(∪T,∪T) unfolding HomeoG_def IsAhomeomorphism_def bij_def by auto
  from bij have converse(x)∈bij(∪T,∪T) using bij_converse_bij by auto
  with bij have conx_fun:converse(x)∈∪T→∪Tx∈∪T→∪T unfolding bij_def inj_def by auto
  from surj have id:x O converse(x)=id(∪T) using right_comp_inverse by auto
  from conx_fun have Composition(∪T){x,converse(x)}=x O 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
}

92.2 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
    IsAhomeomorphism_def
      using union_cocardinal by auto
  { fix f assume a:f∈bij(X,X)
    then have converse(f)∈bij(X,X) unfolding biject_bij by auto
    then have cinj:converse(f)∈inj(X,X) unfolding biject_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
        using union_cocardinal assms n CoCar_is_topology by auto
  } fix N assume N{is closed in}(CoCardinal(X,Q))
  then have N_def:N=X ∨ (N∈Pow(X) ∧ N≺Q) using closed_sets_cocardinal
  n by auto
  then have restrict(converse(f),N)∈bij(N,converse(f)N) unfolding cinj
    restrict_bij by auto
  then have N=N{is closed in}(CoCardinal(X,Q)) by auto
  with fun have f-N{is closed in}(CoCardinal(X,Q)) using func1_1_L3
    func1_1_L4 by auto
  then have ∀f∈bij(X,X). IsContinuous((CoCardinal(X,Q)),(CoCardinal(X,Q)),f)
    unfolding IsAhomeomorphism_def
    using n union_cocardinal by auto
  then show bij(X,X)⊆HomeoG((CoCardinal(X,Q))) unfolding HomeoG_def bij_def
    inj_def using n union_cocardinal
    by auto
qed
The group of homeomorphism of the excluded set is a direct product of the
bijections on \( X \setminus T \) and the bijections on \( X \cap T \).

**Theorem** \texttt{homeo\_excluded}:

\( \text{shows HomeoG(ExcludedSet(X,T))} = \{ f \in \text{bij}(X,X). f(X-T) = (X-T) \} \)

**Proof**

\( \text{have sub1:} X \setminus T \subseteq X \text{ by auto} \)

\[
\begin{align*}
\text{fix } g & \text{ assume } g \in \text{HomeoG(ExcludedSet(X,T))} \\
\text{then have } & \text{fun:} g : X \rightarrow X \text{ and } \text{bij:} g \in \text{bij}(X,X) \text{ and } \text{hom:} \text{IsAhomeomorphism}((\text{ExcludedSet(X,T)}), (\text{ExcludedSet(X,T)})) \text{ (using } \text{union\_excludedset}\text{ un}\text{folding } \text{IsAhomeomorphism}\text{\_def}\text{ by auto}) \\
& \text{have } \text{rfun:} \text{restrict}(g,X-T) : X \rightarrow X \text{ using } \text{fun restrict\_fun sub1 by auto} \\
& \text{moreover } \text{from } A \text{ fun have } \{ gaa. \text{ aa} \in X-T = X \text{ using } \text{func\_imagedef sub1 by auto} \\
& \text{then have } \forall x \in X. \text{ x} \in \{ gaa. \text{ aa} \in X-T \} \text{ by auto} \\
& \text{then have } \forall x \in X. \exists aa \in X-T. x = gaa \text{ by auto} \\
& \text{with } A \text{ have } \text{surj:} \text{restrict}(g,X-T) \subseteq \text{surj}(X-T,X) \text{ using } \text{rfun unfolding surj\_def by auto} \\
& \text{from } B \text{ obtain } d \text{ where } d \in X \setminus T \text{ by auto} \\
& \text{with } \text{bij have } gd \in X \text{ using } \text{apply\_funttype unfolding bij\_def inj\_def by auto} \\
& \text{then obtain s where } \text{restrict}(g,X-T)s = gds \in X-T \text{ using surj unfolding surj\_def by blast} \\
& \text{then have } gs = gd \text{ by auto} \\
& \text{with } <d \in X-s \subseteq X-T> \text{ have s=d using bij unfolding bij\_def inj\_def by auto} \\
& \text{then have False using } <s \in X-T> <d \in T> \text{ by auto} \\
& \text{then have } g(X-T) = X \rightarrow X \setminus T = 0 \text{ by auto} \\
& \text{then have } \text{reg:} g(X-T) = X \rightarrow X-T = X \text{ by auto} \\
& \text{then have } g(X-T) = X \rightarrow g(X-T) = X-T \text{ by auto} \\
& \text{then have } g(X-T) = X \rightarrow g \in \{ f \in \text{bij}(X,X). f(X-T) = (X-T) \} \text{ using bij by auto} \\
& \text{moreover } \text{fix gg} \\
& \text{assume } A: g(g(X-T)) \neq X \text{ and } \text{hom2:} \text{IsAhomeomorphism}((\text{ExcludedSet(X,T)}), (\text{ExcludedSet(X,T)}), g) \text{ (using } \text{union\_excludedset}\text{ by auto} \\
& \text{from } \text{hom2 have } \text{fun:} gg \in X \rightarrow X \text{ and } \text{bij:} gg \in \text{bij}(X,X) \text{ unfolding } \text{IsAhomeomorphism}\text{\_def bij\_def inj\_def using } \text{union\_excludedset}\text{ by auto} \\
& \text{have sub:} X-T \subseteq \bigcup (\text{ExcludedSet(X,T)}) \text{ using } \text{union\_excludedset}\text{ by auto} \\
& \text{with } \text{hom2 have } gg(\text{Interior}(X-T, (\text{ExcludedSet(X,T)}))) = \text{Interior}(gg(X-T), (\text{ExcludedSet(X,T)})) \text{ (using } \text{int\_top\_invariant}\text{ by auto} \text{ auto}) \\
& \text{from } \text{sub1 have } \text{Interior}(X-T, (\text{ExcludedSet(X,T)})) = X-T \text{ using } \text{interior\_set\_excludedset}\text{ by auto} \\
& \text{ultimately have } gg(X-T) = \text{Interior}(gg(X-T), (\text{ExcludedSet(X,T)})) \text{ by auto} \\
& \text{moreover } \text{have ss:} g(g(X-T)) \subseteq X \text{ using } \text{fun\_cl_L6(2)} \text{ by auto} \\
& \text{then have } \text{Interior}(gg(X-T), (\text{ExcludedSet(X,T)})) = (gg(X-T)) \cap \text{us-}
ing interior_set_excludedset A 

by auto 

ultimately have eq:gg(X-T)=(gg(X-T))-T by auto 

{ 
assume (gg(X-T))∩T≠0 
then obtain t where t∈T and im:t∈gg(X-T) by blast 
then have t∉(gg(X-T))-T by auto 
then have False using eq im by auto 
} 

then have (gg(X-T))/T=0 by auto 
then have gg(X-T)⊆X-T using ss by blast 

}


then have ∀gg. gg(X-T)≠X ∧ IsAhomeomorphism(ExcludedSet(X,T), ExcludedSet(X,T),gg)→ gg(X-T)⊆X-T by auto moreover 

from bij have conbij:converse(g)∈bij(X,X) using bij_converse_bij 

by auto 

then have confun:converse(g)∈X→X unfolding bij_def inj_def by auto 

{ 
assume A:converse(g)(X-T)=X and B:X∩T≠0 
have rfun:restrict(converse(g),X-T):X-T→X using confun restrict_fun 
sub1 by auto moreover 
from A confun have {converse(g)aa. aa∈X-T}=X using func_imagedef 

sub1 by auto 

then have ∀x∈X. x∈{converse(g)aa. aa∈X-T} by auto 
then have ∀x∈X. ∃aa∈X-T. x=converse(g)aa by auto 
then have ∀x∈X. ∃aa∈X-T. x=restrict(converse(g),X-T)aa by auto 
with A have surj:restrict(converse(g),X-T)∈surj(X-T,X) using rfun 
unfolding surj_def by auto 

from B obtain d where d∈Xd∈T by auto 
with conbij have converse(g)d∈X using apply_funtype unfolding bij_def inj_def by auto 

then obtain s where restrict(converse(g),X-T)s=converse(g)ds∈X-T using surj unfolding surj_def by blast 
then have converse(g)s=converse(g)d by auto 
with <d∈X>-<s∈X-T> have s=d using conbij unfolding bij_def inj_def by auto 

then have False using <s∈X-T> <d∈T> by auto 

} 

then have converse(g)(X-T)=X → X∩T=0 by auto 
then have converse(g)(X-T)=X → X-T=X by auto 
then have converse(g)(X-T)=X → g-(X-T)=(X-T) unfolding vimage_def by auto 

then have G:converse(g)(X-T)=X → g(g-(X-T))=g(X-T) by auto 
have GG:g(g-(X-T))=(X-T) using sub1 surj_image_vimage bij unfolding bij_def by auto 

with G have converse(g)(X-T)=X → g(X-T)=X-T by auto 
then have converse(g)(X-T)=X → g∈{f∈bij(X,X). f(X-T)=(X-T)} unfolding bij by auto moreover 

from hom have IsAhomeomorphism(ExcludedSet(X,T), ExcludedSet(X,T),

929
converse(g)) using homeo_inv by auto

moreover note hom ultimately have g∈{f∈bij(X,X). f(X-T)=(X-T)} \lor
(g(X-T)⊆X-T \land converse(g)(X-T)⊆X-T)

by force

then have g∈{f∈bij(X,X). f(X-T)=(X-T)} \lor (g(X-T)⊆X-T \land g-(X-T)⊆X-T)

unfolding vimage_def by auto moreover

have g(X-T)⊆X-T → g(g(X-T))⊆g(X-T) using func1_1_L8 by auto

with GG have g-(X-T)⊆X-T → (X-T)⊆g(X-T) by force

ultimately have g∈{f∈bij(X,X). f(X-T)=(X-T)} \lor (g(X-T)⊆X-T \land (X-T)⊆g(X-T))

by auto

then have g∈{f∈bij(X,X). f(X-T)=(X-T)} using bij by auto

} then show HomeoG(ExcludedSet(X,T))⊆{f∈bij(X,X). f(X-T)=(X-T)} by auto

{ fix g assume as:g∈bij(X,X)g(X-T)=X-T

then have inj:g∈inj(X,X) and im:g-(g(X-T))=g-(X-T) unfolding bij_def

by auto

from inj have g-(g(X-T))=X-T using inj_vimage_image sub1 by force

with im have as_3:g-(X-T)=X-T by auto

} fix A assume A∈(ExcludedSet(X,T))

then have A=X\setminus T=0 A⊆X unfolding ExcludedSet_def by auto

then have A⊆X-T \setminus A=X by auto moreover

{ assume A=X

with as(1) have gA=X using surj_range_image_domain unfolding bij_def

by auto

} moreover

{ assume A⊆X-T

then have gA⊆g(X-T) using func1_1_L8 by auto

then have gA⊆(X-T) using as(2) by auto

}

ultimately have gA⊆(X-T) \lor gA=X by auto

then have gA∈(ExcludedSet(X,T)) unfolding ExcludedSet_def by auto

} then have ∀A∈(ExcludedSet(X,T)). gA∈(ExcludedSet(X,T)) by auto moreover

{ fix A assume A∈(ExcludedSet(X,T))

then have A=X\setminus T=0 A⊆X unfolding ExcludedSet_def by auto

then have A⊆X-T \setminus A=X by auto moreover

{ assume A=X

with as(1) have g-A=X using func1_1_L4 unfolding bij_def inj_def

by auto

} }
moreover
{ 
  assume \( A \subseteq X - T \)
  then have \( g - A \subseteq g - (X - T) \) using func1_1_L8 by auto
  then have \( g - A \subseteq (X - T) \) using as_3 by auto
}
ultimately have \( g - A \subseteq (X - T) \) using as_3 unfolding ExcludedSet_def by auto

then 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

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(\text{IncludedSet}(X, T), \text{IncludedSet}(X, T), f) \( f: X \rightarrow X \) \( T \subseteq X \)
  shows IsContinuous(\text{ExcludedSet}(X, T), \text{ExcludedSet}(X, T), f)
proof-
  from assms(2,3) have two:two_top_spaces0(\text{IncludedSet}(X, T), \text{IncludedSet}(X, T), f)
  using union_includedset includedset_is_topology
  unfolding two_top_spaces0_def by auto

  \{ 
  fix \( A \) assume \( A \in (\text{ExcludedSet}(X, T)) \)
  then have \( A \cap T = 0 \lor A = X A \subseteq X \) unfolding ExcludedSet_def by auto
  then have \( A \{\text{is closed in}\}(\text{IncludedSet}(X, T)) \) using closed_sets_includedset
  assms by auto
  then have \( f - A \{\text{is closed in}\}(\text{IncludedSet}(X, T)) \) using two_top_spaces0.TopZF_2_1_L1
  assms(1)
  two assms includedset_is_topology by auto
  then have \( (f - A) \cap T = 0 \lor f - A = X f - A \subseteq X \) using closed_sets_includedset assms(1,3)
  by auto
  then have \( f - A \in (\text{ExcludedSet}(X, T)) \) unfolding ExcludedSet_def by auto
  \}
  then show IsContinuous(\text{ExcludedSet}(X, T), \text{ExcludedSet}(X, T), f) unfolding IsContinuous_def by auto
qed

lemma cont_ex_cont_in:
  assumes IsContinuous(\text{ExcludedSet}(X, T), \text{ExcludedSet}(X, T), f) \( f: X \rightarrow X \) \( T \subseteq X \)
  shows IsContinuous(\text{IncludedSet}(X, T), \text{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=Ø⊆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=Ø⊆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(ExcludedSet(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 HomeoG(IncludedSet(X,T))={f ∈ bij(X,X) . f (X - T) = X - T} proof-
  { fix f assume f∈HomeoG(IncludedSet(X,T)) then have hom:IsAhomeomorphism(IncludedSet(X,T),IncludedSet(X,T),f) and fun:f∈bij(X,X) unfolding HomeoG_def IsAhomeomorphism_def using union_includedset assms by auto then have cont:IsContinuous(IncludedSet(X,T),IncludedSet(X,T),f) unfolding IsAhomeomorphism_def by auto moreover
    { from hom have cont1:IsContinuous(IncludedSet(X,T),IncludedSet(X,T),converse(f)) unfolding IsAhomeomorphism_def by auto moreover have converse(f):X→X using bij_converse_bij bij unfolding bij_def inj_def by auto moreover note assms ultimately have IsContinuous(ExcludedSet(X,T),ExcludedSet(X,T),converse(f)) using cont_in_cont_ex fun assms by auto } then have IsContinuous(ExcludedSet(X,T),ExcludedSet(X,T),converse(f)) by auto moreover note bij ultimately have IsAhomeomorphism(ExcludedSet(X,T),ExcludedSet(X,T),f) unfolding IsAhomeomorphism_def by auto moreover
}
using union_excludedset by auto
with fun have f∈HomeoG(ExcludedSet(X,T)) unfolding HomeoG_def using union_excludedset by auto
}
then have HomeoG(IncludedSet(X,T))⊆HomeoG(ExcludedSet(X,T)) by auto
moreover
{ fix f assume f∈HomeoG(ExcludedSet(X,T))
then have hom:IsAhomeomorphism(ExcludedSet(X,T),ExcludedSet(X,T),f)
and fun:f∈X→X and
bij:f∈bij(X,X) unfolding HomeoG_def IsAhomeomorphism_def using union_excludedset assms by auto
then have cont:IsContinuous(ExcludedSet(X,T),ExcludedSet(X,T),f)
unfolding IsAhomeomorphism_def by auto
then have IsContinuous(IncludedSet(X,T),IncludedSet(X,T),f) using cont_ex_cont_in assms
by auto
moreover
from hom have cont1:IsContinuous(ExcludedSet(X,T),ExcludedSet(X,T),converse(f))
unfolding IsAhomeomorphism_def by auto moreover
have converse(f):X→X using bij_converse_bij bij unfolding bij_def inj_def by auto moreover
note assms ultimately
have IsContinuous(IncludedSet(X,T),IncludedSet(X,T),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

933
have 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 \in Xd \in X 
  \fix x assume AA:x \in X \neq d \in X \in x \in X 
  \then have \( (fc,fx) \in r (fx,fd) \in r \) 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 \neq fc \neq fd by auto moreover 
  have fx \in X using bij unfolding bij_def inj_def using apply_type AA(1) by auto 
  ultimately have fx \in \text{IntervalX}(X,r,fc,fd) unfolding IntervalX_def by auto 
  \then have \( \{fx. x \in \text{IntervalX}(X,r,c,d)\} \subseteq \text{IntervalX}(X,r,fc,fd) \) unfolding IntervalX_def Interval_def by auto 
  moreover 
  \{ 
    \fix y assume y \in \text{IntervalX}(X,r,fc,fd) 
    \then have y:y \in X \neq fcy \neq fd(yc,yc) \in r \text{ unfolding IntervalX_def} Interval_def by auto 
    \then obtain s where s:s \in Xy=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 s(2) y(2,3) by auto 
    \} 
    \then have s \neq cs \neq d by auto moreover 
    have \( (c,s) \in r (s,d) \in r \) using y(4,5) s ord A(2,1) by auto moreover 
    note s(1) ultimately have s:s \in \text{IntervalX}(X,r,c,d) unfolding IntervalX_def interval_def by auto 
    \then have y \in \( \{fx. x \in \text{IntervalX}(X,r,c,d)\} \) using s(2) by auto 
  \} 
  ultimately have \( \{fx. x \in \text{IntervalX}(X,r,c,d)\}=\text{IntervalX}(X,r,fc,fd) \) by auto moreover 
  have \text{IntervalX}(X,r,c,d) \subseteq X unfolding IntervalX_def by auto moreover 
  have f:X \rightarrow X using bij unfolding bij_def surj_def by auto ultimately have f:intervalX(X,r,c,d)=IntervalX(X,r,fc,fd) using func_imagedef by auto 
  \} 

934
∀ \ c \in X. ∀ \ d \in X. fIntervalX(X,r,c,d)=IntervalX(X,r,fc,fd) ∧ fc\in X ∧ fd\in X using bij

unfolding bij_def inj_def by auto

{ fix c assume A:c\in X
{ fix x assume AA:x\in X \neq c(x)\in r
then have \ (fc,fx)\in 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\neq fc by auto moreover
have fx\in X using bij unfolding bij_def inj_def by auto
AA(1)
ultimately have fx\in RightRayX(X,r,fc) unfolding RightRayX_def by auto
}
then have \ (fx. x\in RightRayX(X,r,c))\subseteq RightRayX(X,r,fc) unfolding RightRayX_def
by auto
moreover
{ fix y assume y\in RightRayX(X,r,fc)
then have \ y:y\in X \neq fc(y)\in r unfolding RightRayX_def by auto
then obtain s where s:s\in X\Rightarrow fs using bij unfolding bij_def surj_def
by auto
{ assume s=c
then have fs=fc by auto
then have False using s(2) y(2) by auto
}
then have s\neq c by auto moreover
have \ (c,s)\in r using y(3) s ord A by auto moreover
note s(1) ultimately have s\in RightRayX(X,r,c) unfolding RightRayX_def
by auto
then have y \in \ {fx. x\in RightRayX(X,r,c)} using s(2) by auto
}
ultimately have \ {fx. x\in RightRayX(X,r,c)}=RightRayX(X,r,fc) by auto
moreover
have RightRayX(X,r,c)\subseteq X unfolding RightRayX_def by auto moreover
have f:X\Rightarrow X using bij unfolding bij_def surj_def by auto ultimately
have fRightRayX(X,r,c)=RightRayX(X,r,fc) using func_imagedef by auto
}
then have \ rr\Rightarrow \ c\in X. fRightRayX(X,r,c)=RightRayX(X,r,fc) ∧ fc\in X using bij
unfolding bij_def inj_def by auto
\{ \text{fix } c \text{ assume } A : c \in X \} \\
\{ \text{fix } x \text{ assume } A A : x \in X \neq c(x,c) \in r \}
\text{then have } (fx, fc) \in r \text{ using } A \ ord \text{ by auto moreover} \\
\{ \text{assume } fx = fc \}
\text{then have } x = c \text{ using bij unfolding bij_def inj_def using } A AA(1) \text{ by auto} \\
\text{then have } False \text{ 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} \\
\text{ultimately have } fx \in \text{LeftRayX}(X, r, fc) \text{ unfolding LeftRayX_def by auto} \\
\text{then have } \{ fx \cdot x \in \text{LeftRayX}(X, r, c) \} \subseteq \text{LeftRayX}(X, r, fc) \text{ unfolding LeftRayX_def by auto} \\
\text{moreover} \\
\{ \text{fix } y \text{ assume } y \in \text{LeftRayX}(X, r, fc) \}
\text{then have } y : y \in X \neq fc(y, fc) \in r \text{ unfolding LeftRayX_def by auto} \\
\text{then obtain } s \text{ where } s : s \in Xy = 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 \text{ using } s(2) \text{ by auto} \\
\text{then have } s \neq c \text{ by auto moreover} \\
\text{have } (s, c) \in r \text{ using } y(3) \text{ s ord } A \text{ by auto moreover} \\
\text{note } s(1) \text{ ultimately have } s \in \text{LeftRayX}(X, r, c) \text{ unfolding LeftRayX_def by auto} \\
\text{then have } y \in \{ fx \cdot x \in \text{LeftRayX}(X, r, c) \} \text{ using } s(2) \text{ by auto} \\
\text{ultimately have } \{ fx \cdot x \in \text{LeftRayX}(X, r, c) \} = \text{LeftRayX}(X, r, fc) \text{ by auto} \\
\text{moreover} \\
\text{have } \text{LeftRayX}(X, r, c) \subseteq X \text{ unfolding LeftRayX_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{LeftRayX}(X, r, fc) = \text{LeftRayX}(X, r, fc) \text{ using func_imagedef by auto} \\
\text{then have } lray : \forall c \in X. f \text{LeftRayX}(X, r, c) = \text{LeftRayX}(X, r, fc) \wedge fc \in X \text{ using bij unfolding bij_def inj_def by auto} \\
\text{have } r1 : \forall U \in \{ \text{IntervalX}(X, r, b, c) \cdot (b, c) \in X \times X \} \cup \{ \text{LeftRayX}(X, r, b) \cdot b \in X \} \cup \\
\{ \text{RightRayX}(X, r, b) \cdot b \in X \}. fU \in \{ \text{IntervalX}(X, r, b, c) \cdot (b, c) \in X \times X \} \cup \{ \text{LeftRayX}(X, r, b) \cdot b \in X \} \cup \\
\{ \text{RightRayX}(X, r, b) \cdot b \in X \} \text{ apply safe prefer } 3 \text{ using rray apply}
blast prefer 2 using lray apply blast
  using inter apply auto
proof-
  fix xa y assume xa∈Xy∈X
  then have fxa∈Xfya∈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} ∪ {RightRayX(X, r, b) . b ∈ X} with r1 have fU∈(OrdTopology X r) by blast
  then have f_open:∀ U∈(OrdTopology X r). fU∈(OrdTopology X r) using two_top_spaces0.base_image_open[OF twoSpac Ordtopology_is_a_topology(2)[OF assms(1)]]
  by auto
  { fix c d assume A:c∈Xd∈X
  then obtain cc dd where pre:fc=cf=dcc∈Xdd∈X using bij unfolding bij_def surj_def by blast
  with inter have f IntervalX(X, r, cc, dd) = IntervalX(X, r, c, d) by auto
  then have f-(fIntervalX(X, r, cc, dd)) = f-(IntervalX(X, r, c, d)) by auto
  moreover have IntervalX(X, r, cc, dd)⊆X unfolding IntervalX_def by auto moreover
  have f∈inj(X,X) using bij unfolding bij_def by auto ultimately
  have IntervalX(X, r, cc, dd)=f-IntervalX(X, r, c, d) using inj_vimage_image by auto
  moreover from pre(3,4) have IntervalX(X, r, cc, dd)∈{IntervalX(X,r,e1,e2). (e1,e2)∈X×X} by auto
  ultimately have f-IntervalX(X, r, c, d)∈(OrdTopology X r) using base_sets_open[OF Ordtopology_is_a_topology(2)[OF assms(1)]]
  by auto
  } then have inter:∀c∈X. ∀d∈X. f-IntervalX(X, r, c, d)∈(OrdTopology X r) by auto

937
\[
\{\text{fix } c \text{ assume } A : c \in X \\
\text{then obtain } cc \text{ where } \text{pre: } fcc = ccc \in X \text{ using bij unfolding bij_def surj_def} \\
\text{by blast}
\}
\]

with rray have \( f \text{ RightRayX}(X, r, cc) = \text{RightRayX}(X, r, c) \) by auto
then have \( f-(f\text{RightRayX}(X, r, cc)) = f-(\text{RightRayX}(X, r, c)) \) by auto

moreover
have \( \text{RightRayX}(X, r, cc) \subseteq X \) unfolding RightRayX_def by auto
moreover
have \( f \in \text{inj}(X, X) \) using bij unfolding bij_def by auto
ultimately
have \( \text{RightRayX}(X, r, cc) = f-\text{RightRayX}(X, r, c) \) using inj_vimage_image by auto

moreover
from pre(2) have \( \text{RightRayX}(X, r, cc) \in \{\text{RightRayX}(X, r, e2). e2 \in X\} \) by auto
ultimately have \( f-\text{RightRayX}(X, r, c) \in (\text{OrdTopology X r}) \) using base_sets_open[OF Ordtopology_is_a_topology(2)[OF assms(1)]] by auto

then have rray: \( \forall c \in X. f-\text{RightRayX}(X, r, c) \in (\text{OrdTopology X r}) \) by auto

\[
\{\text{fix } c \text{ assume } A : c \in X \\
\text{then obtain } cc \text{ where } \text{pre: } fcc = ccc \in X \text{ using bij unfolding bij_def surj_def} \\
\text{by blast}
\}
\]

with lray have \( f \text{ LeftRayX}(X, r, cc) = \text{LeftRayX}(X, r, c) \) by auto
then have \( f-(f\text{LeftRayX}(X, r, cc)) = f-(\text{LeftRayX}(X, r, c)) \) by auto

moreover
have \( \text{LeftRayX}(X, r, cc) \subseteq X \) unfolding LeftRayX_def by auto
moreover
have \( f \in \text{inj}(X, X) \) using bij unfolding bij_def by auto
ultimately
have \( \text{LeftRayX}(X, r, cc) = f-\text{LeftRayX}(X, r, c) \) using inj_vimage_image by auto

moreover
from pre(2) have \( \text{LeftRayX}(X, r, cc) \in \{\text{LeftRayX}(X, r, e2). e2 \in X\} \) by auto
ultimately have \( f-\text{LeftRayX}(X, r, c) \in (\text{OrdTopology X r}) \) using base_sets_open[OF Ordtopology_is_a_topology(2)[OF assms(1)]] by auto

then have lray: \( \forall c \in X. f-\text{LeftRayX}(X, r, c) \in (\text{OrdTopology X r}) \) by auto

\[
\{\text{fix } U \text{ assume } U \in \{\text{IntervalX}(X, r, b, c). \langle b, c \rangle \in X \times X \} \cup \{\text{LeftRayX}(X, r, b) . b \in X\} \cup \{\text{RightRayX}(X, r, b) . b \in X\} \\
\text{with lray inter rray have } f-U \in (\text{OrdTopology X r}) \text{ by auto}
\}
\]

then have \( \forall U \in \{\text{IntervalX}(X, r, b, c). \langle b, c \rangle \in X \times X \} \cup \{\text{LeftRayX}(X, r, b) . b \in X\} \cup \{\text{RightRayX}(X, r, b) . b \in X\}. \\
f-U \in (\text{OrdTopology X r}) \text{ by blast} \)
then have \( f \text{cont} : \text{IsContinuous}(\text{OrdTopology} X r, \text{OrdTopology} X r, f) \) using two_top_spaces0.Top_ZF_2_1_L5[OF twoSpac \text{Ordtypeopology\_is\_a\_topology}(2)[OF assms(1)]] by auto
from \( f \text{cont}\) \( f\_open\) \( \text{bij} \) have \( \text{IsAhomeomorphism}(\text{OrdTopology} X r, \text{OrdTopology} X r, f) \) using \( \text{bij\_cont\_open\_homeo} \)
union_ordtopology[OF assms] by auto
then show \( f \in \text{HomeoG}(\text{OrdTopology} X r) \) unfolding \( \text{HomeoG\_def} \) using \( \text{bij} \) union_ordtopology[OF assms] unfolding bij_def inj_def by auto
qed

This last example shows that order isomorphic sets give homeomorphic topological spaces.

### 69.3 Properties preserved by functions

The continuous image of a connected space is connected.

**Theorem (in two_top_spaces0) cont_image_conn:**
assumes \( \text{IsContinuous}(\tau_1, \tau_2, f) \) \( f \in \text{surj}(X_1, X_2) \) \( \tau_1 \{\text{is connected}\} \)
shows \( \tau_2 \{\text{is connected}\} \)
**Proof:**

- \( \{ \text{fix } U \}
  - \( \text{assume } Uop: U \in \tau_2 \) and \( Ucl: U \{\text{is closed in}\} \tau_2 \)
  - \( \text{from } Uop \) \( \text{assms(1)} \) have \( f-U \in \tau_1 \) unfolding \( \text{IsContinuous\_def} \) by auto
- moreover
  - from \( Ucl \) \( \text{assms(1)} \) have \( f-U \{\text{is closed in}\} \tau_1 \) using TopZF_2_1_L1 by auto ultimately
  - have \( \text{disj}: f-U=0 \lor f-U=\bigcup \tau_1 \) unfolding \( \text{assms(3)} \) unfolding \( \text{IsConnected\_def} \) by auto moreover
    - \( \{ \text{assume as: } f-U \neq 0 \}
      - \( \text{then have } U \neq 0 \) using func1_1_L13 by auto
      - from \( \text{as disj} \) have \( f-U=\bigcup \tau_1 \) by auto
      - \( \text{then have } f(f-U)=f(\bigcup \tau_1) \) by auto moreover
      - have \( U \subseteq \bigcup \tau_2 \) using \( Uop \) by blast ultimately
      - have \( U=f(\bigcup \tau_1) \) using surj_image_vimage \( \text{assms(2)} \) \( Uop \) by force
      - \( \text{then have } \bigcup \tau_2=U \) using surj_range_image_domain \( \text{assms(2)} \) by auto
    - moreover
      - \( \{ \text{assume as: } U \neq 0 \}
        - from \( Uop \) have \( s: U \subseteq \bigcup \tau_2 \) by auto
        - with \( \text{as obtain } u \) where \( uU: u \in U \) by auto
        - with \( s \) have \( u \in \bigcup \tau_2 \) by auto
      - with \( \text{assms(2)} \) obtain \( w \) where \( fw=u \in \bigcup \tau_1 \) unfolding surj_def X1_def X2_def by blast
      - with \( uU \) have \( w \in f-U \) using func1_1_L15 \( \text{assms(2)} \) unfolding surj_def by auto

939
then have \( f \cdot U \neq 0 \) by auto

ultimately have \( U = 0 \lor U = \bigcup \tau_2 \) by auto

then show thesis unfolding IsConnected_def by auto

qed

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, if 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:\( X_1 \to X_2 \) \( X_1 \neq 0 \)
- shows \( \exists q \in X_2, \forall w \in X_1. f(w) = q \)

proof-
- from assms(4) have surj:\( f \in \text{surj}(X_1, \text{range}(f)) \) using fun_is_surj by auto
- have sub:range(\( f \)) \( \subseteq X_2 \) using func1_1_L5B assms(4) by auto
- from assms(1) have cont:IsContinuous(\( \tau_1, \tau_2 \) restricted to)range(\( f \)), \( f \) unfolding restr_image_cont range_image_domain
- assms(4) by auto
- have union:\( \bigcup \tau_2 \) restricted to)range(\( f \)) = range(\( f \)) unfolding RestrictedTo_def
- using sub by auto
- then have two_top_spaces0(\( \tau_1, \tau_2 \) restricted to)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:range(\( f \)) \( \{ \text{is connected} \) using two_top_spaces0.cont_image_conn
- surj assms(2) cont
- union by auto
- then have range(\( f \)) \( \{ \text{is in the spectrum of} \)IsConnected using assms(3)
- sub unfolding IsTotDis_def antiProperty_def
- using union by auto
- then have range(\( f \)) \( \leq 1 \) 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
- { fix \( w \) assume \( w \in X_1 \)
- then have \( f \in \text{range}(f) \) using func1_1_L5A(2) assms(4) by auto
- with uniq have \( f \in q \) by auto
- }
- then have \( \forall w \in X_1. f \in q \) 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($\tau_1, \tau_2, f$) $f$ ∈ surj($X_1, X_2)$ $X_1$ is compact of cardinal) K( in $\tau_1$
- Shows $X_2$ is compact of cardinal) K(in $\tau_2$

**Proof:**

- Have $X_2 \subseteq \bigcup \tau_2$ by auto Moreover

  - 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 $f^{-1}X_2 \subseteq f^{-1}(\bigcup U)$ by blast
  - Then have $f^{-1}X_2 \subseteq \text{converse}(f)(\bigcup U)$ unfolding vimage_def by auto

- Ultimately
  - Have $f^{-1}X_2 \subseteq (\bigcup \{f-V. V \in U\}) \subseteq \tau_2$
  - Then have $f^{-1}X_2 \subseteq \text{converse}(f)(\bigcup U)$ unfolding vimage_def by auto
  - Have $\{f-V. V \in U\} \subseteq \bigcup \tau_2$ using assms(2) unfolding surj_def by force

- With P assms(3) have $\exists N \in \text{Pow}\{f-V. V \in U\}$. $X_1 \subseteq \bigcup N \wedge N < K$ unfolding IsCompactOfCard_def by auto

- Then obtain $N$ where $N \in \text{Pow}\{f-V. V \in U\}$. $X_1 \subseteq \bigcup N$ $N < K$ by auto

- Then have fin: $N < K$ and sub: $N \subseteq \{f-V. V \in U\}$ and cov: $X_1 \subseteq \bigcup N$ unfolding FinPow_def by auto

- From sub have $\{fR. R \in N\} \subseteq \{f-V. V \in U\}$ by auto Moreover

  - Have $\forall V \in U$. $V \subseteq \tau_2$ using as(2) by auto Ultimately

  - Have $\{fR. R \in N\} \subseteq U$ using surj_image_vimage assms(2) by auto Moreover

- Let $FN=\{R, fR. R \in N\}$
  - Have $FN: FN: N \rightarrow \{fR. R \in N\}$ unfolding Pi_def function_def domain_def by auto

  - Fix S assume $S \in \{fR. R \in N\}$
  - Then obtain $R$ where $R \in \text{fin}_N$. $R \in N$. $R \in S$ by auto

- Then have $(R, fR) \in FN$ by auto

- Then have $fNR=fR$ using FN apply equality by auto

- Then have $\exists R \in N$. $fNR=S$ using R_def by auto

- Then have surj: $FN \in \text{surj}\{N, \{fR. R \in N\}\}$ unfolding surj_def using FN by force

  - From fin have $N \in \text{fin}_N$. $N \leq K$. Ord$K$ unfolding IsCompactOfCard_def

  - Using Card_is_Ord by auto

  - Then have $\{fR. R \in N\} \subseteq N$ using surj_fun_inv_2 surj by auto

  - Then have $\{fR. R \in N\} < K$ using fin lesspoll_trans1 by blast

- Moreover

  - Have $\bigcup \{fR. R \in N\} = \{f\{U\} \mid U\}$ using image_UN by auto

  - Then have $fX_1 \subseteq \bigcup \{fR. R \in N\}$ using cov by blast

941
then have $X_2 \subseteq \bigcup \{fR. R \in \mathbb{N}\}$ using assms(2) surj_range_image_domain by auto
ultimately have $\exists NN \in \text{Pow}(U). X_2 \subseteq \bigcup NN$ using assms(2) surj_range_image_domain by auto
ultimately show thesis using assms(3) unfolding IsCompactOfCard_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$(\tau_1, \tau_2, f) \quad X_1\{\text{is compact in}\} \tau_1 \quad \tau_2\{\text{is anti-compact}\} 
shows \text{Finite}(\text{range}(f))$ and $\text{range}(f) \neq 0$
proof-
  from assms(4) have surj:$f \in \text{surj}(X_1, \text{range}(f))$ using fun_is_surj by auto
  have sub:$\text{range}(f) \subseteq X_2$ using func1_1_L5B assms(4) by auto
  from assms(1) have cont:IsContinuous$(\tau_1, \tau_2\{\text{restricted to}\} \text{range}(f), f)$ 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 range$(f)\{\text{is compact in}\}(\tau_2\{\text{restricted to}\} \text{range}(f))$ using surj two_top_spaces0.cont_image_com cont union assms(2) Compact_is_card_nat by force
  then have range$(f)\{\text{is in the spectrum of}\}(\lambda T. \bigcup T) \{\text{is compact in}\} T$ using assms(3) sub unfolding IsAntiComp_def antiProperty_def using union by auto
  then show Finite$(\text{range}(f))$ using compact_spectrum by auto moreover from assms(5) have $fX_1 \neq 0$ using func1_1_L15A assms(4) by auto
  then show $\text{range}(f) \neq 0$ using range_image_domain 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:$\{b, r\{b\}. b \in \bigcup T \in \text{surj}(\bigcup T, (\bigcup T) / r) $ using quotient_proj_surj by auto

942
moreover have \( \text{tot} : \bigcup \{ \text{quotient by} r \} = ( \bigcup T ) \div r \) using total_quo_equi assms(2) by auto
ultimately have \( \text{cont} : \text{IsContinuous}(T, \{ \text{quotient by} r \}, \{ (b, r \{ b \}) \mid b \in \bigcup T \}) \)
using quotient_func_cont
EquivQuo_def assms(2) by auto
from surj tot have two_top_spaces0(T, \{ \text{quotient by} r \}, \{ (b, r \{ b \}) \mid b \in \bigcup 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_com
Compact_is_card_nat by force
qed

corollary (in topology0) ConnQuot:
assumes \( T \{ \text{is connected} \} \)
equiv(\( \bigcup T \), r)
shows \( \{ \text{quotient by} r \} \{ \text{is connected} \} \)
proof-
  have surj: \( \{ (b, r \{ b \}) \mid b \in \bigcup T \} \subseteq \text{surj}(\( \bigcup T \), (\( \bigcup T \)) \div r) \)
using quotient_proj_surj
by auto
moreover have \( \text{tot} : \bigcup \{ \text{quotient by} r \} = ( \bigcup T ) \div r \) using total_quo_equi assms(2) by auto
ultimately have \( \text{cont} : \text{IsContinuous}(T, \{ \text{quotient by} r \}, \{ (b, r \{ b \}) \mid b \in \bigcup T \}) \)
using quotient_func_cont
EquivQuo_def assms(2) by auto
from surj tot have two_top_spaces0(T, \{ \text{quotient by} r \}, \{ (b, r \{ b \}) \mid b \in \bigcup 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

70 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.

70.1 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) \( A \subseteq \bigcup T \) \( B \subseteq \bigcup S \)
shows \( \text{Closure}(A \times B, \text{ProductTopology}(T, S)) = \text{Closure}(A, T) \times \text{Closure}(B, S) \)

\textbf{proof}

have \( A \times B \subseteq \bigcup T \times \bigcup S \) using \text{assms}(3, 4) by auto
then have \( \text{sub}(A \times B \subseteq \bigcup \text{ProductTopology}(T, S)) \) using \text{Top_1_4_T1}(3) \text{assms}(1, 2)
by auto
have \( \text{top} \) : \( \text{ProductTopology}(T, S) \) is a topology
using \text{Top_1_4_T1}(1) \text{assms}(1, 2)
by auto

\begin{align*}
\{ & \text{fix } x \text{ assume } \text{asx}: x \in \text{Closure}(A \times B, \text{ProductTopology}(T, S)) \\
& \text{then have } \text{reg}: \forall U \in \text{ProductTopology}(T, S). x \in U \rightarrow U \cap (A \times B) \neq 0 \text{ using topology0.cl_inter_neigh sub top unfolding topology0_def by blast} \\
& \text{from asx have } x \in \bigcup \text{ProductTopology}(T, S) \text{ using topology0.Top_3_L11(1)} \\
& \text{top unfolding topology0_def using sub by blast} \\
& \text{then have } x \Sigma : x \in \bigcup \text{ProductTopology}(T, S) \text{ using Top_1_4_T1(3) assms(1, 2) by auto} \\
& \text{then have } xT : \text{fst}(x) \in \bigcup T \text{ and } xS : \text{snd}(x) \in \bigcup S \text{ by auto} \\
& \{ & \text{fix } U \text{ V assume as: } U \in T \text{ fst}(x) \in U \\
& \text{have } \bigcup S C U \text{ using assms(2) unfolding IsATopology_def by auto} \\
& \text{with as have } U \times (\bigcup S) \subseteq \text{ProductCollection}(T, S) \text{ unfolding ProductCollection_def by auto} \\
& \text{then have } P : U \times (\bigcup S) \subseteq \text{ProductTopology}(T, S) \text{ using Top_1_4_T1(2) assms(1, 2)} \\
& \text{base_sets_open by blast with xS as(2) have } \langle \text{fst}(x), \text{snd}(x) \rangle \in U \times (\bigcup S) \text{ by auto} \\
& \text{then have } x \in U \times (\bigcup S) \text{ using Pair_fst_snd_eq xSigma by auto} \\
& \text{with P reg have } U \times (\bigcup S) \cap A \times B \neq 0 \text{ by auto} \\
& \text{then have noEm: } U \cap A \neq 0 \text{ by auto} \\
& \} \\
& \text{then have } \forall U \subseteq T. \text{ fst}(x) \in U \rightarrow U \cap A \neq 0 \text{ by auto moreover} \\
& \{ & \text{fix U V assume as: } U \subseteq S \text{ snd}(x) \subseteq U \\
& \text{have } \bigcup T \subseteq U \text{ using assms(1) unfolding IsATopology_def by auto} \\
& \text{with as have } (\bigcup T) \times U \subseteq \text{ProductCollection}(T, S) \text{ unfolding ProductCollection_def by auto} \\
& \text{then have } P : (\bigcup T) \times U \subseteq \text{ProductTopology}(T, S) \text{ using Top_1_4_T1(2) assms(1, 2)} \\
& \text{base_sets_open by blast with xT as(2) have } \langle \text{fst}(x), \text{snd}(x) \rangle \in (\bigcup T) \times U \text{ by auto} \\
& \text{then have } x \in (\bigcup T) \times U \text{ using Pair_fst_snd_eq xSigma by auto} \\
& \text{with P reg have } (\bigcup T) \times U \cap A \times B \neq 0 \text{ by auto} \\
& \text{then have noEm: } U \cap B \neq 0 \text{ by auto} \\
& \} \\
& \text{then have } \forall U \subseteq S. \text{ snd}(x) \subseteq U \rightarrow U \cap B \neq 0 \text{ by auto} \\
\text{ultimately have } \text{fst}(x) \subseteq \text{Closure}(A, T) \text{ snd}(x) \subseteq \text{Closure}(B, S) \text{ using topology0.inter_neigh_cl assms(3, 4) unfolding topology0_def using assms(1, 2)} \\
\text{xT xS by auto} \\
& \text{then have } \langle \text{fst}(x), \text{snd}(x) \rangle \in \text{Closure}(A, T) \times \text{Closure}(B, S) \text{ by auto} \\
\end{align*}
with xSigma have x∈Closure(A,T)×Closure(B,S) by auto
}
then show Closure(A×B,ProductTopology(T,S))⊆Closure(A,T)×Closure(B,S)
by auto
{
  fix x assume x:x∈Closure(A,T)×Closure(B,S)
  then have xcl:fst(x)∈Closure(A,T) snd(x)∈Closure(B,S) by auto
from xcl(1) have regT:∀U∈T. fst(x)∈U −→ U∩A≠0 using topology0.cl_inter_neigh
unfolding topology0_def using assms(1,3) by blast
from xcl(2) have regS:∀U∈S. snd(x)∈U −→ U∩B≠0 using topology0.cl_inter_neigh
unfolding topology0_def using assms(2,4) by blast
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 x tot:x∈⋃ProductTopology(T,S) using Top_1_4_T1(3) assms(1,2) by auto
{
  fix PO assume as:PO∈ProductTopology(T,S) x∈PO
  then obtain POB where base:POB∈ProductCollection(T,S) x∈POB POB⊆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 POB=VT×VS unfolding ProductCollection_def
  by auto
  from V(3) have x:fst(x)∈VT snd(x)∈VS by auto
  from V(1) regT x(1) have VT∩A≠0 by auto moreover
  from V(2) regS x(2) have VS∩B≠0 by auto ultimately
  have VT×VS∩A×B≠0 by auto
  with V(4) base(3) have PO∩A×B≠0 by blast
}
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
unfolding topology0_def using top sub xtot by auto
}
then show Closure(A,T)×Closure(B,S)⊆Closure(A×B,ProductTopology(T,S))
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}S
  shows (A×B) {is closed in}ProductTopology(T,S)
proof-
  from assms(3,4) have sub:A⊆⋃TB⊆⋃S unfolding IsClosed_def by auto
  then have A×B⊆⋃T×⋃S by auto
  then have sub1:A×B⊆⋃ProductTopology(T,S) using Top_1_4_T1(3) assms(1,2)
  by auto
  from sub assms have Closure(A,T)=AClosure(B,S)=B using topology0.Top_3_L8
unfolding topology0_def by auto
then have Closure(\(A \times B, \text{ProductTopology}(T, S)) = A \times B\) 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

70.2 Separation properties in product space

The product of \(T_0\) spaces is \(T_0\).

**Theorem T0_product:**

assumes \(T\{\text{is a topology}\}\), \(S\{\text{is a topology}\}\), \(T\{\text{is T}_0\}\), \(S\{\text{is T}_0\}\)

shows \(\text{ProductTopology}(T, S)\{\text{is T}_0\}\)

**Proof:**

{ fix \(x, y\) assume \(x \in \bigcup \text{ProductTopology}(T, S)\) \(y \in \bigcup \text{ProductTopology}(T, S)\) \(x \neq y\)
  then have \(\text{tot}: x \in \bigcup T \times \bigcup S\) \(y \in \bigcup T \times \bigcup S\) 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 \(\text{disj:} \text{fst}(x) \neq \text{fst}(y)\) \(\text{snd}(x) \neq \text{snd}(y)\)
  using Pair_fst_snd_eq by auto
  then have \(\text{T:} \text{fst}(x) \in \bigcup T\) \(\text{snd}(x) \in \bigcup S\) \(\text{and:} \text{fst}(x) \neq \text{fst}(y)\) \(\text{snd}(x) \neq \text{snd}(y)\)
  by auto
  moreover have \(\text{p:} x \in \bigcup T \times \bigcup S\) \(y \notin \bigcup T \times \bigcup S\) \(\text{or:} y \in \bigcup T \times \bigcup S\) \(x \notin \bigcup T \times \bigcup S\)
  using Pair_fst_snd_eq tot(1,2) by auto
  ultimately have \(\exists V \in \text{ProductTopology}(T, S). (x \in V \land y \notin V) \lor (y \in V \land x \notin V)\) proof qed
  moreover
  { assume \(\text{snd}(x) \neq \text{snd}(y)\)
    with \(S\) assms(4) have \(\exists U \in \bigcup T \times \bigcup S. (\text{snd}(x) \in U \land \text{snd}(y) \notin U) \lor (\text{snd}(y) \in U \land \text{snd}(x) \notin U)\)
    unfolding isT0_def by auto
    then obtain \(U\) where \(U \in \bigcup T \times \bigcup S\) \(x \in U\) \(\land\) \(y \notin U\)
    using prod_open_open_prod assms(1,2) by auto
    ultimately have \(\exists V \in \text{ProductTopology}(T, S). (x \in V \land y \notin V) \lor (y \in V \land x \notin V)\) proof qed
  }
}

946
by auto
   with \( T \) have \((\text{fst}(x), \text{snd}(x)) \in (\bigcup T) \times U \land \langle \text{fst}(y), \text{snd}(y) \rangle \not\in (\bigcup T) \times U \lor \langle \text{fst}(y), \text{snd}(y) \rangle \in (\bigcup T) \times U \land \langle \text{fst}(x), \text{snd}(x) \rangle \not\in (\bigcup T) \times U) \)
   by auto
   then have \( x \in (\bigcup T) \times U \land y \not\in (\bigcup T) \times U \lor \langle \text{fst}(y), \text{snd}(y) \rangle \not\in (\bigcup T) \times U \) 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 \( U \in S \) have \((\bigcup T) \times U \in \text{ProductTopology}(T, S)\) using prod_open_open_prod assms(1,2) by auto
ultimately have \( \exists V \in \text{ProductTopology}(T, S). (x \in V \land y \not\in V) \lor (y \in V \land x \not\in V) \) proof qed
}
}
then show thesis unfolding isT0_def by auto
qed

The product of \( T_1 \) spaces is \( T_1 \).

theorem \( T_1 \)-product:
  assumes \( T \{\text{is a topology}\} \)\( S \{\text{is a topology}\} \)\( T \{\text{is T}_1\} \)\( S \{\text{is T}_1\} \)
  shows \( \text{ProductTopology}(T, S) \{\text{is T}_1\} \)
proof-
  {  
    \begin{align*}
      \text{fix } & x \ y \text{ assume } x \in \bigcup \text{ProductTopology}(T, S) \text{ } y \in \bigcup \text{ProductTopology}(T, S) x \neq y \\
      \text{then have } & \text{tot: } x \in \bigcup T \times U \text{ } y \in \bigcup T \times U \text{ } x \neq y \text{ using } \text{Top}_1_4_\text{T1}(3) \text{ } \text{assms}(1,2) \\
      \text{by auto} & \\
      \text{then have } & \langle \text{fst}(x), \text{snd}(x) \rangle \in (\bigcup T) \times U \times (\bigcup S) \text{ } \langle \text{fst}(y), \text{snd}(y) \rangle \in (\bigcup T) \times U \times (\bigcup S) \text{ } \text{and disj: } \text{fst}(x) \neq \text{fst}(y) \lor \text{snd}(x) \neq \text{snd}(y) \\
      \text{using } & \text{Pair_fst_snd_eq by auto} \\
      \text{then have } & T : \text{fst}(x) \in \bigcup T \text{ } \text{y: } \text{fst}(y) \in \bigcup T \text{ } \text{and } S : \text{snd}(y) \in \bigcup S \text{ } \text{and } p : \text{fst}(x) \neq \text{fst}(y) \lor \text{snd}(x) \neq \text{snd}(y) \\
      \text{by auto} & \\
    \end{align*}
}

\{  
  \begin{align*}
    & \text{assume } \text{fst}(x) \neq \text{fst}(y) \\
    \text{with } & T \text{ } \text{assms}(3) \text{ have } (\exists U \in T. \ \langle \text{fst}(x), \text{snd}(x) \rangle \in U \times (\bigcup S) \lor \langle \text{fst}(y), \text{snd}(y) \rangle \in U \times (\bigcup S) ) \text{ unfolding } \text{isT1_def by auto} \\
    \text{then obtain } & U \text{ where } U \in T \text{ } \text{fst: } \text{fst}(x) \in U \land \text{fst}(y) \not\in U \text{ by auto} \\
    \text{with } & S \text{ have } ( \langle \text{fst}(x), \text{snd}(x) \rangle \in U \times (\bigcup S) \land \langle \text{fst}(y), \text{snd}(y) \rangle \not\in U \times (\bigcup S) ) \\
    \text{by auto} & \\
    \text{then have } & (x \in U \times (\bigcup S) \land y \not\in U \times (\bigcup S) ) \text{ using } \text{Pair_fst_snd_eq tot(1,2) by auto} \\
    \text{by auto} & \\
    \text{moreover have } & (\bigcup S) \in S \text{ using } \text{assms}(2) \text{ unfolding } \text{IsATopology_def} \\
    \text{by auto} & \\
    \text{with } & U \times (\bigcup S) \in \text{ProductTopology}(T, S) \text{ using } \text{prod_open_open_prod} \text{ } \text{assms}(1,2) \text{ by auto} \\
    \text{ultimately} & 
  \end{align*}
\}
have ∃V∈ProductTopology(T,S). (x∈V ∧ y∉V) proof qed

moreover
{
assume snd(x)≠snd(y)
with S assms(4) have (∃U∈S. (snd(x)∈U ∧ snd(y)∉U)) unfolding isT1_def by auto
then obtain U where U∈S (snd(x)∈U ∧ snd(y)∉U) by auto
with T have ((fst(x),snd(x))∈(⋃T×U ∧ (fst(y),snd(y))∉(⋃T×U)
by auto
then have (x∈(⋃T)×U ∧ y∉(⋃T)×U) using Pair_fst_snd_eq tot(1,2)
by auto
moreover have (⋃T)∈T using assms(1) unfolding IsATopology_def
by auto
ultimately have ∃V∈ProductTopology(T,S). (x∈V ∧ y∉V) proof qed
moreover
note disj
ultimately have ∃V∈ProductTopology(T,S). (x∈V ∧ y∉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-
{
fix x y assume x∈⋃T ProductTopology(T,S)y∈⋃T ProductTopology(T,S)x≠y
then have tot:x∈⋃T×U y∈⋃T×U x≠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))∈⋃T and S:snd(y)∈⋃S snd(x)∈⋃S and
p:fst(x)≠fst(y) and 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))∈U×(⋃S) (fst(y),snd(y))∈V×(⋃S) and
  disjoint:(U×(⋃S))∩(V×(⋃S))=0 by auto
then have x∈U×(⋃S)y∈V×(⋃S) using Pair_fst_snd_eq tot(1,2) by auto
}
moreover have \((\bigcup S)\in S\) using assms(2) unfolding IsATopology_def
by auto
with \(<U\in T, V\in T>\) have \(P=U\times(\bigcup S)\in ProductTopology(T,S)\ V=(\bigcup S)\in ProductTopology(T,S)\)
using prod_open_open_prod assms(1,2) by auto
note disjoint ultimately
have \(x\in U\times(\bigcup S) \land y\in V\times(\bigcup S) \land (U\times(\bigcup S))\cap(V\times(\bigcup S))=0\) by auto
with \(P(2)\) have \(\exists U\in ProductTopology(T,S)\ (x\in U\times(\bigcup S) \land y\in U\land (U\times(\bigcup S))\cap U=0)\)
using exI[where \(x=V\times(\bigcup S)\ and \ P=\lambda t. t\in ProductTopology(T,S)\ ^ (x\in U\times(\bigcup S) \land y\in t \land (U\times(\bigcup S))\cap t=0)\)] by auto
with \(P(1)\) have \(\exists V\in ProductTopology(T,S)\ (y\in V\times(\bigcup S) \land (\exists U\in ProductTopology(T,S). \ (x\in t \land y\in U\land (t)\cap U=0)))\) by auto
} moreover
{ assume \(\text{snd}(x)\neq\text{snd}(y)\)
with \(S\) assms(4) have \(\exists U\in S\. \ (x\in U\land \text{snd}(y)\in V) \land U\cap V=0\) unfolding
isT2_def by auto
then obtain \(U, V\) where \(U\in S\ V\in S\ \text{snd}(x)\in U\ \text{snd}(y)\in V\ U\cap V=0\) by auto
with \(T\) have \(\langle \text{fst}(x), \text{snd}(x)\rangle\in (\bigcup T)\times U\ \langle \text{fst}(y), \text{snd}(y)\rangle\in (\bigcup T)\times V\) and
disjoint:\((\bigcup T)\times U\cap((\bigcup T)\times V)=0\) by auto
then have \(x\in (\bigcup T)\times U\cap V\) 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 \(<U\in S, V\in S>\) have \(P=:(\bigcup T)\times U\in ProductTopology(T,S)\ (\bigcup T)\times V\in ProductTopology(T,S)\)
using prod_open_open_prod assms(1,2) by auto
note disjoint ultimately
have \(x\in (\bigcup T)\times U\land y\in (\bigcup T)\times V\land ((\bigcup T)\times U)\cap ((\bigcup T)\times V)=0\) by auto
with \(P(2)\) have \(\exists U\in ProductTopology(T,S)\ (x\in (\bigcup T)\times U\land y\in U\land ((\bigcup T)\times U)\cap U=0)\)
using exI[where \(x=\bigcup T\times V\ and \ P=\lambda t. t\in ProductTopology(T,S)\ ^ (x\in (\bigcup T)\times U\land y\in t \land ((\bigcup T)\times U)\cap t=0)\)] by auto
with \(P(1)\) have \(\exists V\in ProductTopology(T,S)\ (y\in V\times (\bigcup T)\times U\land (\exists U\in ProductTopology(T,S). \ (x\in t \land y\in U\land (t)\cap U=0)))\) by auto
} moreover
note disjoint ultimately have \(\exists V\in ProductTopology(T,S)\ (\exists U\in ProductTopology(T,S). \ x\in V\land y\in U\land V\cap U=0\) by auto
} then show 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 ProductTopology(T,S){is regular}
proof-
{  fix x U assume x∈⋃ ProductTopology(T,S) U∈ProductTopology(T,S) x∈U
  then obtain V W where VW:VW∈T VW∈S V×W⊆U and x:x∈V×W using prod_top_point_neighb

  assms(1,2) by blast
  then have p:fst(x)∈V snd(x)∈W by auto
  from p(1) V∈T. obtain VV where VV:fst(x)∈VV Closure(VV,T)⊆V VV∈T

  using assms(1,3) topology0.regular_imp_exist_clos_neig unfolding topology0_def
  by force
  moreover from p(2) W∈S. obtain WW where WW:snd(x)∈WW Closure(WW,S)⊆WW WW∈S

  using assms(2,4) topology0.regular_imp_exist_clos_neig unfolding topology0_def
  by force ultimately
  have x∈VV×WW using x by auto
  moreover from Closure(VV,T)⊆V Closure(WW,S)⊆W have Closure(VV,T)×Closure(WW,S) ⊆ V×W
  by auto
  moreover from VV(3) WW(3) have VV∪TVV∪S by auto
  ultimately have x∈VV×WW Closure(VV×WW,ProductTopology(T,S)) ⊆ V×W

  using cl_product assms(1,2)
  by auto
  moreover have VV×WW∈ProductTopology(T,S) using prod_open_open_prod

  using assms(1,2) VV(3) WW(3) by auto
  ultimately have ∃Z∈ProductTopology(T,S). x∈Z ∧ Closure(Z,ProductTopology(T,S))⊆V×W
  by auto with WW(3) have ∃Z∈ProductTopology(T,S). x∈Z ∧ Closure(Z,ProductTopology(T,S))⊆U
  by auto
  } then have ∀x∈⋃ ProductTopology(T,S). ∀U∈ProductTopology(T,S).x∈U →
  (∃Z∈ProductTopology(T,S). x∈Z ∧ Closure(Z,ProductTopology(T,S))⊆U)

  by auto
  then show thesis using topology0.exist_clos_neig_imp_regular unfolding topology0_def

  using assms(1,2) Top_1_4_T1(1) by auto
qend

70.3 Connection properties in product space

First, we prove that the projection functions are open.

lemma projection_open:
assumes T{is a topology}S{is a topology}B∈ProductTopology(T,S)
shows {y∈⋃ T. ∃x∈⋃ S. (y,x)∈B}∈T
proof-
\[
\{ \text{fix } z \text{ assume } z \in \{ y \in U. \ \exists x \in S. \ (y, x) \in B \} \text{ then obtain } x \text{ where } x : x \in U \text{ and } z : z \in U \text{ and } p : (z, x) \in B \text{ by auto} \text{ then have } z \in \{ y \in U. \ \{ y, x \} \in B \} \ \{ y \in U. \ \{ y, x \} \in B \} \leq \{ y \in U. \ \exists x \in S. \ (y, x) \in B \} \text{ by auto moreover } \}
\]
from x have \{ y \in U. \ (y, x) \in B \} \in T \text{ using prod_sec_open2 assms by auto ultimately have } \exists V \in T. \ z \in V \wedge \forall \\in \{ y \in U. \ \exists x \in S. \ (y, x) \in B \} \text{ unfolding Bex_def by auto} \}
then show \{ y \in U. \ \exists x \in S. \ (y, x) \in B \} \in T \text{ using topology0.open_neigh_open unfolding topology0_def using assms(1) by blast qed}\\

lemma projection_open2:
assumes T\{is a topology\} S\{is a topology\} B \in ProductTopology(T, S)
shows \{ y \in U. \ \exists x \in U. \ (y, x) \in B \} \in S
proof-\\
{ \text{fix } z \text{ assume } z \in \{ y \in U. \ \exists x \in U. \ (y, x) \in B \} \text{ then obtain } x \text{ where } x : x \in U \text{ and } z : z \in U \text{ and } p : (x, z) \in B \text{ by auto then have } z \in \{ y \in U. \ \exists x \in U. \ (y, x) \in B \} \ \{ y \in U. \ \exists x \in U. \ (y, x) \in B \} \leq \{ y \in U. \ \exists x \in U. \ (y, x) \in B \} \text{ by auto moreover } \}
from x have \{ y \in U. \ (y, x) \in B \} \in S \text{ using prod_sec_open1 assms by auto ultimately have } \exists V \in S. \ z \in V \wedge \forall \\in \{ y \in U. \ \exists x \in U. \ (y, x) \in B \} \text{ unfolding Bex_def by auto} \}
then show \{ y \in U. \ \exists x \in U. \ (y, x) \in B \} \in S \text{ using topology0.open_neigh_open unfolding topology0_def 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) \in ProductTopology(T, S)
proof-\\
{ \text{fix } U \text{ assume } U : U \in ProductTopology(T, S) \ U \{is closed in\} ProductTopology(T, S) \text{ then have } P : U \in ProductTopology(T, S) \ ProductTopology(T, S) - U \in ProductTopology(T, S) \text{ unfolding IsClosed_def by auto} \}
{ \text{fix } s \text{ assume } s : s \in U \text{ with } P(1) \text{ have } p : \{ x \in U. \ (x, s) \in U \} \in T \text{ using prod_sec_open2 assms(1,2) by auto} \}
from s P(2) have oop : \{ y \in U. \ (y, s) \in (\cup \ ProductTopology(T, S) - U) \} \in T \text{ using prod_sec_open2 assms(1,2) by blast then have } \cup T - (\cup T - \{ y \in U. \ (y, s) \in (\cup ProductTopology(T, S) - U) \}) = \{ y \in U. \}.
\[ \langle y, s \rangle \in (\bigcup \text{ProductTopology}(T, S) - U) \] 
by auto

with oop have cl: \( \bigcup T - \{ y \in T. \ (y, s) \in (\bigcup \text{ProductTopology}(T, S) - U) \} \) 
{is closed in}T unfolding IsClosed_def by auto

fix t assume t \in \bigcup T - \{ y \in \bigcup T. \ (y, s) \in (\bigcup \text{ProductTopology}(T, S) - U) \} 
then have tt: t \in \bigcup T \ (t, s) \notin (\bigcup \text{ProductTopology}(T, S) - U) 
by auto

then have \( (t, s) \notin (\bigcup \text{ProductTopology}(T, S) - U) \) 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

then have \( (t, s) \notin (\bigcup \text{ProductTopology}(T, S) - U) \) by auto

then have \( t \notin \{ y \in \bigcup T. \ (y, s) \in (\bigcup \text{ProductTopology}(T, S) - U) \} \) by auto

with tt(1) s have \( (t, s) \notin (\bigcup \text{ProductTopology}(T, S) - U) \) by auto

then have \( (t, s) \notin (\bigcup \text{ProductTopology}(T, S) - U) \) by auto

ultimately have \( \{ x \in \bigcup T. \ (x, s) \in U \} = \bigcup T - \{ y \in \bigcup T. \ (y, s) \in (\bigcup \text{ProductTopology}(T, S) - U) \} \) by blast

with cl have \( \{ x \in \bigcup T. \ (x, s) \in U \} \subseteq \bigcup T - \{ y \in \bigcup T. \ (y, s) \in (\bigcup \text{ProductTopology}(T, S) - U) \} \) by auto

with p assms(3) have \( \{ x \in \bigcup T. \ (x, s) \in U \} = \bigcup T \) unfolding IsConnected_def by auto

moreover

assume \( \{ x \in \bigcup T. \ (x, s) \in U \} = \bigcup T \)

then have \( \forall x \in \bigcup T. \ (x, s) \notin U \) by auto

moreover

assume AA: \( \{ x \in \bigcup T. \ (x, s) \in U \} = \bigcup T \)

fix x assume x \in \bigcup T

with AA have \( x \in \{ x \in \bigcup T. \ (x, s) \in U \} \) by auto

then have \( (x, s) \in U \) by auto

then have \( \forall x \in \bigcup T. \ (x, s) \in U \) by auto

ultimately have \( (\forall x \in \bigcup T. \ (x, s) \notin U) \lor (\forall x \in \bigcup T. \ (x, s) \in U) \) by blast

then have reg: \( \forall s \in \bigcup S. \ (\forall x \in \bigcup T. \ (x, s) \notin U) \lor (\forall x \in \bigcup T. \ (x, s) \in U) \) by auto

fix q assume qU: q \in \bigcup T \times \{ \text{snd}(qq) \}. \ qq \in U

then obtain t u where t: t \in \bigcup T \ u \in U \ q = (t, \text{snd}(u)) by auto

with U(1) have \( u \in \bigcup \text{ProductTopology}(T, S) \) by auto

952
then have \( u \in \bigcup T \times \bigcup S \) using \( \text{Top}_1.4.\_T1(3) \) assms(1,2) by auto moreover

then have \( uu \colon u = (\text{fst}(u), \text{snd}(u)) \) using \( \text{Pair}_{\text{fst, snd}}_{\text{eq}} \) by auto ultimately

have \( fu \colon \text{fst}(u) \in \bigcup T \) \( \text{snd}(u) \in \bigcup S \) by (safe,auto)

with reg have \( (\forall tt \in T. \langle tt, \text{snd}(u)\rangle \not\in U) \lor (\forall tt \in T. \langle tt, \text{snd}(u)\rangle \in U) \) by auto

moreover

ultimately have \( fu \colon \text{fst}(u) \in \bigcup T \) \( \text{snd}(u) \in \bigcup S \) by (safe,auto)

with t(1,3) have \( q \in U \) by auto

fix \( t \) assume \( t \colon t \in \bigcup T \) with \( \text{P}(1) \)

have \( p \colon \{ x \in \bigcup S. \langle t, x\rangle \in U \} \in S \) using \( \text{prod}_{\text{sec, open}}1 \) assms(1,2) by auto

from \( t \) \( \text{P}(2) \) have \( oop \colon \{ x \in \bigcup S. \langle t, x\rangle \in (\bigcup \text{ProductTopology}(T, S) - U) \} \in S \) using \( \text{prod}_{\text{sec, open}}1 \) assms(1,2) by blast

then have \( \bigcup S - (\bigcup S - \{ y \in \bigcup S. \langle t, y\rangle \in (\bigcup \text{ProductTopology}(T, S) - U) \}) \subseteq \{ y \in \bigcup S. \langle t, y\rangle \in (\bigcup \text{ProductTopology}(T, S) - U) \} \) by (force,auto)

with \( oop \) have \( \{ x \in \bigcup S. \langle t, x\rangle \in \bigcup \text{ProductTopology}(T, S) - U) \} \subseteq \bigcup S \) unfolding \( \text{IsClosed}_{\text{def}} \) by (force,auto)

moreover

ultimately have \( \{ x \in \bigcup S. \langle t, x\rangle \in \bigcup \text{ProductTopology}(T, S) - U) \} \subseteq \bigcup S \) by (force,auto)

with \( oop \) have \( \{ x \in \bigcup S. \langle t, x\rangle \in \bigcup \text{ProductTopology}(T, S) - U) \} \subseteq \bigcup S \) unfolding \( \text{IsConnected}_{\text{def}} \) by (force,auto)

moreover

ultimately have \( \{ x \in \bigcup S. \langle t, x\rangle \in \bigcup \text{ProductTopology}(T, S) - U) \} \subseteq \bigcup S \) by (force,auto)

with \( oop \) have \( \{ x \in \bigcup S. \langle t, x\rangle \in \bigcup \text{ProductTopology}(T, S) - U) \} \subseteq \bigcup S \) unfolding \( \text{IsConnected}_{\text{def}} \) by (force,auto)

moreover

ultimately have \( \{ x \in \bigcup S. \langle t, x\rangle \in \bigcup \text{ProductTopology}(T, S) - U) \} \subseteq \bigcup S \) by (force,auto)

with \( oop \) have \( \{ x \in \bigcup S. \langle t, x\rangle \in \bigcup \text{ProductTopology}(T, S) - U) \} \subseteq \bigcup S \) unfolding \( \text{IsConnected}_{\text{def}} \) by (force,auto)

moreover

ultimately have \( \{ x \in \bigcup S. \langle t, x\rangle \in \bigcup \text{ProductTopology}(T, S) - U) \} \subseteq \bigcup S \) by (force,auto)

with \( oop \) have \( \{ x \in \bigcup S. \langle t, x\rangle \in \bigcup \text{ProductTopology}(T, S) - U) \} \subseteq \bigcup S \) unfolding \( \text{IsConnected}_{\text{def}} \) by (force,auto)

moreover

ultimately have \( \{ x \in \bigcup S. \langle t, x\rangle \in \bigcup \text{ProductTopology}(T, S) - U) \} \subseteq \bigcup S \) by (force,auto)

with \( oop \) have \( \{ x \in \bigcup S. \langle t, x\rangle \in \bigcup \text{ProductTopology}(T, S) - U) \} \subseteq \bigcup S \) unfolding \( \text{IsConnected}_{\text{def}} \) by (force,auto)
then have $\forall x \in \bigcup S. \langle t, x \rangle \notin U$ by auto

moreover

\{  
  assume $AA : \{ x \in \bigcup S. \langle t, x \rangle \in U \} = \bigcup S$
  \{
    fix $x$ assume $x \in \bigcup S$
    with $AA$ have $x \in \{ x \in \bigcup S. \langle t, x \rangle \in U \}$ by auto 
    then have $\langle t, x \rangle \in U$ by auto 
  \}
  then have $\forall x \in \bigcup S. \langle t, x \rangle \in U$ by auto
\}

ultimately have $(\forall x \in \bigcup S. \langle t, x \rangle \notin U) \lor (\forall x \in \bigcup S. \langle t, x \rangle \in U)$ by blast

then have reg : $\forall s \in \bigcup T. (\forall x \in \bigcup S. \langle s, x \rangle \notin U) \lor (\forall x \in \bigcup S. \langle s, x \rangle \in U)$ by auto

\{
  fix $q$ assume $qU : q \in \{ \text{fst}(qq). \langle s, x \rangle \notin U \} \times \bigcup S$
  then obtain $qq$ s where $t : q = (\text{fst}(qq), s)$ $qq \in U$ $s \in \bigcup S$ by auto 
  with $U(1)$ have $qq \in \bigcup \text{ProductTopology}(T, S)$ by auto
  then have $qq \in \bigcup T \times \bigcup S$ using $\text{Top}_1.4.\text{T1}(3)$ assms(1,2) by auto
moreover then have $qq : qq = (\text{fst}(qq), \text{snd}(qq))$ using $\text{Pair}_f\text{st}_\text{snd}_\text{eq}$ by auto
ultimately have $fq : fq : fq \in \bigcup T \text{snd}(qq) \in \bigcup S$ by (safe,auto) 
  from $fq(1)$ reg have $(\forall t \in \bigcup S. \langle \text{fst}(qq), tt \rangle \notin U) \lor (\forall tt \in \bigcup S. \langle \text{fst}(qq), tt \rangle \in U)$ by auto
moreover have $<qq \in U>$ $fq(2)$ have $\forall tt \in \bigcup S. \langle \text{fst}(qq), tt \rangle \in U$ by force 
  with $t(1,3)$ have $q \in U$ by auto
\}

then have $U2 : \{ \text{fst}(qq). \langle s, x \rangle \in U \} \times \bigcup S \subseteq U$ by blast

\{
  assume $U \neq 0$
  then obtain $u$ where $u : u \in U$ by auto

  \{
    fix $aa$ assume $aa \in \bigcup T \times \bigcup S$
    then obtain $t s$ where $t \in \bigcup T s \in \bigcup S a a = \langle t, s \rangle$ by auto 
    with $u$ have $\langle t, \text{snd}(u) \rangle \in \bigcup T \times \{ \text{snd}(qq) \} \in \bigcup U$ by auto
    with $U(1)$ have $\langle t, \text{snd}(u) \rangle \in U$ by auto
    moreover have $t = \text{fst}(\langle t, \text{snd}(u) \rangle)$ by auto
moreover note $<s \in U>$ ultimately
have $\langle t, s \rangle \in \{ \text{fst}(qq). \langle s, x \rangle \in U \} \times \bigcup S$ by blast 
  with $U2$ have $\langle t, s \rangle \in U$ by auto
  with $<aa = \langle t, s \rangle>$ have $aa \in U$ by auto
\}

then have $\bigcup T \times \bigcup S \subseteq U$ by auto moreover
with $U(1)$ have $U \subseteq \bigcup \text{ProductTopology}(T, S)$ by auto ultimately 
have $\bigcup T \times \{ s \} = U$ using $\text{Top}_1.4.\text{T1}(3)$ assms(1,2) by auto
\}

then have $(U = 0) \lor (U \in \bigcup T \times \bigcup S)$ by auto

954
71 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.

71.1 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.

71.2 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 (OrdTopology X r) \{is T_2\}
proof-
{
  fix x y assume A1:x\in \bigcup (OrdTopology X r)y\in \bigcup (OrdTopology X r)x\neq y
  then have AS:x\in Xy \in Xx \neq y using union_ordtopology[OF assms(1) assms(2)]
  by auto
  moreover
  \{ assume A2: \exists z \in X-\{x,y\}. (x,y) \in r \longrightarrow (x,z) \in r \land (z,y) \in r) \land ((y,x) \in r \longrightarrow (y,z) \in r \land (z,x) \in r)
  from AS(1,2) assms(1) have \langle x,y \rangle \in r \lor \langle y,x \rangle \in r unfolding IsLinOrder_def
  IsTotal_def by auto moreover
  \{ assume \langle x,y \rangle \in r
  with A2 obtain z where z: (x,z) \in r (z,y) \in rz \in X z \neq x \neq y by auto
  with A2(1,2) have x \in LeftRayX(X,r,z)y \in RightRayX(X,r,z) unfolding LeftRayX_def RightRayX_def
  by auto moreover

955
have LeftRayX(X,r,z) ∩ RightRayX(X,r,z) = 0 using inter_lray_rray[OF 
  z(3) z(3) assms(1)] unfolding IntervalX_def using Order_ZF_2_L4[OF total_is_refl 
  _ z(3)] assms(1) unfolding IsLinOrder_def by auto moreover 
  have LeftRayX(X,r,z) ∈ (OrdTopology X r) RightRayX(X,r,z) ∈ (OrdTopology X r) 
  using z(3) base_sets_open[OF Ordtopology_is_a_topology(2)[OF 
  assms(1)]] by auto 
ultimately have ∃ U ∈ (OrdTopology X r) ∃ V ∈ (OrdTopology X r) x ∈ U ∧ y ∈ V ∧ U ∩ V = 0 by auto 
} 
moreover 
  assume ⟨y,x⟩ ∈ r 
  with AS A2 obtain z where z:(y,z) ∈ (z,x) ∈ rz: x ≠ x ≠ y by auto 
  with A2(1,2) have y: LeftRayX(X,r,z) x ∈ RightRayX(X,r,z) unfolding LeftRayX_def RightRayX_def 
  by auto moreover 
  have LeftRayX(X,r,z) ∩ RightRayX(X,r,z) = 0 using inter_lray_rray[OF 
  z(3) z(3) assms(1)] unfolding IntervalX_def using Order_ZF_2_L4[OF total_is_refl 
  _ z(3)] assms(1) unfolding IsLinOrder_def by auto 
moreover 
  have LeftRayX(X,r,z) ∈ (OrdTopology X r) RightRayX(X,r,z) ∈ (OrdTopology X r) 
  using z(3) base_sets_open[OF Ordtopology_is_a_topology(2)[OF 
  assms(1)]] by auto 
ultimately have ∃ U ∈ (OrdTopology X r) ∃ V ∈ (OrdTopology X r) x ∈ U ∧ y ∈ V ∧ U ∩ V = 0 by auto 
} 
ultimately have ∃ U ∈ (OrdTopology X r) ∃ V ∈ (OrdTopology X r) x ∈ U ∧ y ∈ V ∧ U ∩ V = 0 by auto 
} 
moreover 
  assume A2: ∀ z ∈ X - {x, y}. ((x, y) ∈ r ∧ (x, z) /∈ r ∨ (y, z) /∈ r) 
  ∨ ((y, x) ∈ r ∧ (y, z) /∈ r ∨ (z, x) /∈ r) 
  from AS(1,2) assms(1) have disj: (x,y) ∈ r ∨ (y,x) ∈ r unfolding IsLinOrder_def 
  IsTotal_def by auto moreover 
  assume TT: ⟨x,y⟩ ∈ r 
  with AS assms(1) have T: (y,x) /∈ r unfolding IsLinOrder_def antisym_def 
  by auto 
  from TT AS(1-3) have x ∈ LeftRayX(X,r,y) y ∈ RightRayX(X,r,x) unfolding 
  LeftRayX_def RightRayX_def by auto moreover 
  { fix z assume z ∈ LeftRayX(X,r,y) ∩ RightRayX(X,r,x) }
then have \((z,y) \in r(x,z) \in r \in X - \{x, y\}\) unfolding \(\text{RightRayX \_ def} \) \(\text{LeftRayX \_ def}\)
by auto
with \(A2\) \(T\) have False by auto
\}
then have \(\text{LeftRayX}(X, r, y) \cap \text{RightRayX}(X, r, x) = 0\) by auto moreover
have \(\text{LeftRayX}(X, r, y) \in (\text{OrdTopology} \ X \ r) \text{RightRayX}(X, r, x) \in (\text{OrdTopology} \ X \ r)\)
using base_sets_open\([\text{OF Ordtopology \_ is \_ a \_ topology}(2)[\text{OF assms}(1)]\]\)
\(\text{AS by auto}\)
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
\}
moreover
\{
assume \(\text{TT}: (y, x) \in r\)
with \(\text{AS assms}(1)\) have \(T: (x, y) \notin r\) unfolding \(\text{IsLinOrder \_ def} \) \(\text{antisym \_ def}\)
by auto
from \(\text{TT} \ \text{AS}(1-3)\) have \(y \in \text{LeftRayX}(X, r, x) x \in \text{RightRayX}(X, r, y)\) unfolding \(\text{LeftRayX \_ def} \) \(\text{RightRayX \_ def}\)
by auto
moreover
\{
fix \(z\) assume \(z \in \text{LeftRayX}(X, r, x) \cap \text{RightRayX}(X, r, y)\)
then have \((z, x) \in r(y, z) \in r \in X - \{x, y\}\) unfolding \(\text{RightRayX \_ def} \) \(\text{LeftRayX \_ def}\)
by auto
with \(A2\) \(T\) have False by auto
\}
then have \(\text{LeftRayX}(X, r, x) \cap \text{RightRayX}(X, r, y) = 0\) by auto moreover
have \(\text{LeftRayX}(X, r, x) \in (\text{OrdTopology} \ X \ r) \text{RightRayX}(X, r, y) \in (\text{OrdTopology} \ X \ r)\)
using base_sets_open\([\text{OF Ordtopology \_ is \_ a \_ topology}(2)[\text{OF assms}(1)]\]\)
\(\text{AS by auto}\)
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
\}
then show thesis unfolding \(\text{isT2 \_ def}\) 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.
71.3 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
\[ A \{is dense in\}X{\text{ with respect to}}r \equiv \forall x \in X. \forall y \in X. (x,y) \in r \land x \neq y \rightarrow (\exists z \in A-\{x,y\}. (x,z) \in r \land (z,y) \in r) \]

**definition**

IsDenseUnp (_{is not-properly dense in}_{with respect to}_) where
\[ A \{is not-properly dense in\}X{\text{ with respect to}}r \equiv \forall x \in X. \forall y \in X. (x,y) \in r \land x \neq y \rightarrow (\exists z \in A. (x,z) \in r \land (z,y) \in r) \]

**definition**

IsWeaklyDenseSub (_{is weakly dense in}_{with respect to}_) where
\[ A \{is weakly dense in\}X{\text{ with respect to}}r \equiv \forall x \in X. \forall y \in X. (x,y) \in r \land x \neq y \rightarrow ((\exists z \in A-\{x,y\}. (x,z) \in r \land (z,y) \in r) \lor \text{ IntervalX}(X,r,x,y)=0) \]

**definition**

IsDense (_{is dense with respect to}_) where
\[ X \{is dense with respect to\}r \equiv \forall x \in X. \forall y \in X. (x,y) \in r \land x \neq y \rightarrow (\exists z \in X-\{x,y\}. (x,z) \in r \land (z,y) \in r) \]

**lemma**

dense_sub: shows \((X \{is dense with respect to\}r) \iff (X \{is dense in\}X{\text{ with respect to}}r)\)

**lemma**

not_prop_dense_sub: shows \((A \{is dense in\}X{\text{ with respect to}}r) \rightarrow (A \{is not-properly dense in\}X{\text{ with respect to}}r)\)

In densely ordered sets, intervals are infinite.

**theorem**

dense_order_inf_intervals:

assumes \(\text{IsLinOrder}(X,r) \land \text{IntervalX}(X, r, b, c) \neq 0\)

shows \(\neg \text{Finite}(\text{IntervalX}(X, r, b, c))\)

**proof**

assume \(\text{fin} : \text{Finite}(\text{IntervalX}(X, r, b, c))\)

have \(\text{sub} : \text{IntervalX}(X, r, b, c) \subseteq X\)

have \(\text{p} : \text{Minimum}(r, \text{IntervalX}(X, r, b, c)) \in \text{IntervalX}(X, r, b, c)\)

using Finite_ZF_1_T2(2)[\text{OF assms}(1) \text{ Finite_Fin}[\text{OF fin sub}] \text{ assms}(2)]

by auto

then have \((b, \text{Minimum}(r, \text{IntervalX}(X, r, b, c))) \notin b\)
unfolding IntervalX_def using Order_ZF_2_L1 by auto
with assms(3,5) sub p obtain z1 where z1:z1∈Xz1≠bd1≠Minimum(r,IntervalX(X, r, b, c))(b,z1)∈r(z1,Minimum(r,IntervalX(X, r, b, c)))∈r
unfolding IsDense_def by blast
from p have B:(Minimum(r,IntervalX(X, r, b, c)),c)∈r unfolding IntervalX_def using Order_ZF_2_L1 by auto moreover
have trans(r) using assms(1) unfolding IsLinOrder_def by auto moreover
note z1(5) ultimately have z1a:(z1,c)∈r unfolding trans_def by fast
{
  assume z1=c
  with B have (Minimum(r,IntervalX(X, r, b, c)),z1)∈r by auto
  with z1(5) have z1=Minimum(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≠c by auto
with z1(1,2,4) z1a have z1∈IntervalX(X, r, b, c) unfolding IntervalX_def using Order_ZF_2_L1 by auto
then have (Minimum(r,IntervalX(X, r, b, c)),z1)∈r using Finite_ZF_1_T2(4)[OF assms(1) Finite_Pin[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)]
  end
  fix d assume d∈IntervalX(X, r, b, c)
  then have (d,b)∈r(d,c)∈r(b≠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
Right rays are infinite.

**Theorem dense_order_inf_rays:**

Assumes IsLinOrder\((X,r)\) RightRay\(X(X,r,b)\)\(\neq 0\)\(\in X\) \(X\) (is dense with respect to)\(r\)

Shows \(\neg\)Finite\(\left(\text{RightRay}\_X(X,r,b)\right)\)

**Proof:**

- From assms(2) obtain \(c\) where \(c\in X\) \(\langle b,c\rangle \in r\) \(\neq c\) unfolding RightRay\_X_def
  - With assms(3) obtain \(z\) where \(z\in X\) \(\langle b,c\rangle \in r\) \(\langle z,c\rangle \in r\) using assms(4)
  - Unfolding IsDense_def by auto
  - Then have Interval\(X(X,r,b,c)\)\(\neq 0\) unfolding Interval\_X_def using Order\_ZF\_2\_L1
    - By auto
    - Then have \(\text{nFIN}\) : \(\neg\)Finite\(\left(\text{Interval}\_X(X,r,b,c)\right)\) using dense_order_inf_intervals[of assms(1)_ assms(3)_ assms(4)]
    - \(<c\in X>\) by auto
  - Fix \(d\) assume \(d\in\text{Interval}\_X(X,r,b,c)\)
    - Then have \(\langle b,d\rangle \in r\) \(\langle d,c\rangle \in r\) \(\langle d\neq c\rangle \in r\)
      - Unfolding Interval\_X_def using Order\_ZF\_2\_L1
    - By auto
    - Then have \(d\in\text{RightRay}\_X(X,r,b)\) unfolding RightRay\_X_def by auto
  - Then have Interval\(X(X,r,b,c)\subseteq\text{RightRay}\_X(X,r,b)\) 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)\(r\)

\(\exists x,y. x\neq y\wedge x\in X\wedge y\in X\)

Shows \(\neg(X\prec\text{nat})\)

**Proof:**

- From assms(3) obtain \(b\) \(c\) where \(B:b\in X\) \(c\in X\) \(b\neq c\) by auto
  - Assume \(\langle b,c\rangle \notin r\)
    - With assms(1) have \(\langle c,b\rangle \in r\) unfolding IsLinOrder\_def IsTotal\_def using \(\langle b\in X\rangle\langle c\in X\rangle\) by auto
    - With assms(2) \(B\) obtain \(z\) where \(z\in X\) \(\langle b,c\rangle \in r\) \(\langle z,b\rangle \in r\) unfolding IsDense\_def by auto
    - Then have Interval\(X(X,r,b,c)\)\(\neq 0\) unfolding Interval\_X_def using Order\_ZF\_2\_L1
      - By auto
    - Then have \(\neg(\text{Finite}(\text{Interval}\_X(X,r,b,c)))\) using dense_order_inf_intervals[of assms(1)_ \(<c\in X\rangle\langle b\in X\rangle\) assms(2)]
      - By auto furthermore
        - Have Interval\(X(X,r,b,c)\subseteq X\) unfolding Interval\_X_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

qed
moreover
{
  assume \((b,c) \in r\)
  with assms(2) obtain \(z\) where \(z \in X \setminus \{b,c\}\) \(\in r\) \((z,c) \in r\) unfolding IsDense_def by auto
  then have \(\text{IntervalX}(X,r,b,c) \neq 0\) unfolding IntervalX_def using Order_ZF_2_L1 by auto
  then have \(\neg(\text{Finite}(\text{IntervalX}(X,r,b,c)))\) unfolding IsComplete_def by force
  ultimately have \(\neg(\text{Finite}(X))\) unfolding lesspoll_nat_is_Finite by auto
}
utimately show thesis by auto
qed

If an order topology is connected, then the order is complete. It is equivalent to assume that \(r \subseteq X \times X\) or prove that \(r \cap X \times X\) is complete.

theorem conn_imp_complete:
assumes IsLinOrder(X,r)
\(\exists x \ y. x \neq y \wedge x \in X \wedge y \in X \wedge r \subseteq X \times X\)
shows \(r\) is complete
proof-
{
  assume \(\neg(r\) is complete\))
  then obtain \(A\) where \(A: A \neq 0\) IsBoundedAbove(A,r) \(\neg(\text{HasAminimum}(r, \bigcap b \in A)\)
  unfolding IsComplete_def by auto
  from A(3) have r1: \(\forall m \in \bigcap b \in A. \ r \{b\}. \ \exists x \in \bigcap b \in A. \ r \{b\}. \ (m,x) \notin r\) unfolding HasAminimum_def
  by force
  from A(1,2) obtain \(b\) where \(r2: \forall x \in A. \ (x, b) \in r\) unfolding IsBoundedAbove_def
  by auto
  with assms(3) A(1) have \(A \subseteq X b \in X\) by auto
  with assms(3) have r3: \(\forall c \subseteq A. \ r \{c\} \subseteq X\) using image_iff by auto
  from r2 have \(\forall x \in A. \ b \in r(x)\) using image_iff by auto
  then have noE: \(b \notin \bigcap b \in A. \ r \{b\}\) using A(1) by auto
  { fix \(x\) assume \(x \in \bigcap b \in A. \ r \{b\}\)
    then have \(\forall c \subseteq A. \ x \in r(c)\) by auto
    with A(1) obtain \(c\) where \(c \subseteq A\) \(x \in r(c)\) by auto
    with r3 have \(x \in X\) by auto
  }
  then have sub: \(\bigcap b \in A. \ r \{b\} \subseteq X\) by auto
  { fix \(x\) assume \(x: x \in \bigcap b \in A. \ r \{b\}\)
    with r1 have \(\exists z \in \bigcap b \in A. \ r \{b\}. \ (x,z) \notin r\) by auto
    then obtain \(z\) where \(z: z \in \bigcap b \in A. \ r \{b\}\) \((x,z) \notin r\) by auto
  }
}
961
from $x \in z(1)$ sub have $x \in X$ by auto
with $z(2)$ have $(z,x) \in r$ using assms(1) unfolding IsLinOrder_def IsTotal_def by auto
then have $x : x \in \text{RightRayX}(X,r,z)$ unfolding RightRayX_def using $x \in X$ ($x,z) \notin r$
assms(1) unfolding IsLinOrder_def using total_is_refl unfolding refl_def by auto
{
  fix $m$ assume $m : m \in \text{RightRayX}(X,r,z)$
  then have $m : m \in X - \{z\} (z,m) \in r$ unfolding RightRayX_def by auto
  with $m(2)$ have $(c,m) \in r$ using assms(1) unfolding IsLinOrder_def
trans_def by fast
  then have $m : m \in \{c\}$ using image_iff by auto
  with $A(1)$ have $m : m \in (\bigcap b \in A. r \{b\})$ by auto
  then have $X : X - (\bigcap b \in A. r \{b\})$ unfolding $X \in \text{RightRayX}(X,r,z)$ using
  base_sets_open[OF Ordtopology_is_a_topology(2) [OF assms(1)]] $z \in X$
by auto
  with $X : x \in X - (\bigcap b \in A. r \{b\})$ unfolding $X \in \text{RightRayX}(X,r,z)$ using
  $x \in X$ ($x,z) \notin r$
assms(1) unfolding refl_def by auto
}
then have$sub_1 : \text{RightRayX}(X,r,z) \subseteq (\bigcap b \in A. r \{b\})$
by auto
moreover
{
  fix $x$ assume $x : x \in X - (\bigcap b \in A. r \{b\})$
  then have $x : x \in X - (\bigcap b \in A. r \{b\})$ by auto
  with $A(1)$ obtain $b$ where $x \notin r(b) b \in A$ by auto
  then have $(b,x) \notin r$ using image_iff by auto
  with $A \subseteq X$ $b \in A$ $x \in X$ have $(x,b) \in r$ using assms(1) unfolding IsLinOrder_def
  IsTotal_def by auto
  then have $x : x \in \text{LeftRayX}(X,r,b)$ unfolding LeftRayX_def using $x \in X$ $(b,x) \notin r$
assms(1) unfolding IsLinOrder_def using total_is_refl unfolding refl_def by auto
  
  fix $y$ assume $y : y \in \text{LeftRayX}(X,r,b)$ unfolding $X - (\bigcap b \in A. r \{b\})$
  then have $y : y \in X - (\bigcap b \in A. r \{b\})$ unfolding LeftRayX_def by auto
  then have $y : y \in X - (\bigcap b \in A. r \{b\})$ using image_iff by auto
  with $b \in A$ have $y = b$ using assms(1) unfolding IsLinOrder_def antisym_def
by auto
  then have False using $y \in X - (\bigcap b \in A. r \{b\})$ unfolding LeftRayX_def
by auto
}
then have $sub_1 : \text{LeftRayX}(X,r,b) \subseteq X - (\bigcap b \in A. r \{b\})$ unfolding LeftRayX_def
by auto

962
have LeftRayX(X,r,b) ∈ (OrdTopology X r) using
  base_sets_open[OF Ordtopology_is_a_topology[OF assms(1)]] <b∈A>¬A⊆x
by blast
with sub1 xx have ∃U∈(OrdTopology X r). x∈U ∧ U⊆¬(⋂b∈A. r {b})
by auto

then have X - (⋂b∈A. r {b})∈(OrdTopology X r) using topology0.open_neigh_open[OF topology0_ordtopology[OF assms(1)]]
by auto

then have ∪(OrdTopology X r)-{⋂b∈A. r {b}}∈(OrdTopology X r) unfolding union_ordtopology[OF assms(1,2)]
by auto

then have (⋂b∈A. r {b})is closed in(OrdTopology X r) unfolding IsClosed_def using union_ordtopology[OF assms(1,2)]
by auto

then have (∩b∈A. r {b}){is closed in}(OrdTopology X r) unfolding IsClosed_def using union_ordtopology[OF assms(1,2)]
by auto

moreover note assms(4) ultimately
have (∩b∈A. r {b})=X using union_ordtopology[OF assms(1,2)] unfolding IsConnected_def
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 using <A⊂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

with A(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 using <A⊂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 x1∈Xx2∈X|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
⟨x∈X⟩ ⟨x2∈X⟩
then have ⟨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 x1:x∈X-{x2}|x1,x)∈r unfolding LeftRayX_def by auto
from x1 xr x(5) have False by auto
}
with xr(1) 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(3)] unfolding IsConnected_def by auto
with x(1,3,4) have LeftRayX(X,r,x2)=X unfolding LeftRayX_def by auto
then have x2∈LeftRayX(X,r,x2) using x(2) 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(\( A, r \)) \( A \neq \emptyset \ r \subseteq X \times X \)
  shows HasAmaximum(\( r, \bigcap c \in A. \ r \{c\} \))
proof-
  let \( M = \bigcap c \in A. \ r \{-c\} \)
  from assms(3) obtain \( t \) where \( A : t \in A \) by auto
  { fix \( m \) assume \( m \in M \)
    with \( A \) have \( m \in r \{-t\} \) by auto
    then have \( \langle m, t \rangle \in r \) by auto
  }
  then have \( (\forall x \in \bigcap c \in A. \ r - \{c\}. \ (x, t) \in r) \) by auto
  then have IsBoundedAbove(\( M, r \)) unfolding IsBoundedAbove_def by auto
  moreover
  from assms(2,3) obtain \( l \) where \( \forall x \in A. \ (l, x) \in r \) unfolding IsBoundedBelow_def by auto
  then have \( \forall x \in A. \ l \in r - \{x\} \) using vimage_iff by auto
  with assms(3) have \( l \in M \) by auto
  then have \( M \neq \emptyset \) by auto
  moreover note assms(1)
  ultimately have HasAminimum(\( r, \bigcap c \in M. \ r \ \{c\} \)) unfolding IsComplete_def by auto
  then obtain \( rr \) where \( \forall x \in A. \ (l, x) \in r \)
    unfolding HasAminimum_def by auto
    { fix \( aa \) assume \( A : aa \in A \)
      { fix \( c \) assume \( M : c \in M \)
        with \( A \) have \( \langle c, aa \rangle \in r \) by auto
        then have \( aa \in r \{c\} \) by auto
      }
      then have \( aa \in (\bigcap c \in M. \ r \ \{c\}) \) using \( rr(1) \) by auto
    }
  then have \( A \subseteq (\bigcap c \in M. \ r \ \{c\}) \) by auto
  with \( rr(2) \) have \( \forall s \in A. \ (rr, s) \in r \) by auto
  then have \( rr \in M \) using assms(3) by auto
  moreover
  { fix \( m \) assume \( m \in M \)
    then have \( rr \in r \{m\} \) using \( rr(1) \) by auto
    then have \( \langle m, rr \rangle \in r \) by auto
  }
  then have \( \forall m \in M. \ (m, rr) \in r \) by auto
  ultimately show thesis unfolding HasAmaximum_def by auto

965
theorem comp_dense_imp_conn:
  assumes IsLinOrder(X,r) \( \exists x \ y. \ x \neq y \land x \in X \land y \in X \ \subseteq X \times X \)
  \( X \) is dense with respect to \( r \) \( r \) is complete
  shows \( \text{(OrdTopology } X \ r) \) is connected
proof-
  \{ 
  assume \( \neg(\text{(OrdTopology } X \ r) \text{ is connected}) \)
  then obtain U where U:U \neq 0 \cup U \in (\text{OrdTopology } X \ r) \text{ is closed in } \text{(OrdTopology } X \ r) \)
    unfolding IsConnected_def using union_orptopology[OF assms(1,2)] by auto
  from U(4) have A:X-U \subseteq X unfolding IsClosed_def using union_orptopology[OF assms(1,2)] by auto
  from A(2) U(1,2) have X-U \neq 0 by auto
  then obtain v where v \in X-U by auto
  with u \in U \cup X\cup v \in U using assms(1) unfolding IsLinOrder_def
  IsTotal_def by auto
  \{ 
  assume \( \langle u,v \rangle \in r \)
    have LeftRayX(X,r,v) \in (\text{OrdTopology } X \ r) \text{ using base_sets_open[OF Ortopology_is_a_topology(2)[OF assms(1)]} 
      \langle v \in X-U \rangle \text{ by auto } 
    then have U \cap LeftRayX(X,r,v) \in (\text{OrdTopology } X \ r) \text{ using U(3) using Ortopology_is_a_topology(1)} 
      [OF assms(1)] unfolding IsATopology_def by auto
      \{ 
      fix b assume b \in (U \cap LeftRayX(X,r,v) \text{ using base_sets_open[OF Ortopology_is_a_topology(2)[OF assms(1)]} 
        \langle v \in X-U \rangle \text{ by auto } 
      then have \( \langle b,v \rangle \in r \) unfolding LeftRayX_def by auto
      \} 
    then have bound:isBoundedAbove(U \cap LeftRayX(X,r,v),r) \text{ unfolding IsBoundedAbove_def by auto moreover } 
      with \( \langle u,v \rangle \in r \cup u \cup v \in X \cup v \in X-U \rangle \text{ have nE:U \cap LeftRayX(X,r,v) \neq 0 unfolding LeftRayX_def by auto } 
    ultimately have Hmin:HasAminimum(r,\bigcap c \in U \cap LeftRayX(X,r,v). r(c)) using assms(5) unfolding IsComplete_def 
      by auto 
    let min=Supremum(r,U \cap LeftRayX(X,r,v)) 
      \{ 
      fix c assume c \in U \cap LeftRayX(X,r,v) 
        then have \( \langle c,v \rangle \in r \) unfolding LeftRayX_def by auto
      \} 
    then have a1: \( \langle min,v \rangle \in r \) using Order_ZF_5_L3[OF _ nE Hmin] assms(1) unfolding IsLinOrder_def 
      by auto 
  \}
assume $(a,m) \in X$
then obtain $V$ where $V:=(a,m) \in X$

$x \in \{\text{Interval}(X,r,b,c). (b,c) \subseteq X \times X) \cup \{\text{LeftRay}(X,r,b), b \in X\} \cup \{\text{RightRay}(X,r,b), b \in X\}$ using point_open_base_neigh

[OF Ordtopology_is_a_topologiel(2)][OF assms(1)] <U)=(OrdTopol

X r> ass] by blast

{ assume $V \in \{\text{RightRay}(X,r,b), b \in X\}$
then obtain $b$ where $b:b \in X = \text{RightRay}(X,r,b)$ by auto
note a1 moreover
from V(1) b(2) have a2: $(b,\min) \in r_{\min} \neq b$ unfolding RightRayX_def
by auto
ultimately have $(b,v) \in r$ using assms(1) unfolding IsLinOrder_def
trans_def by blast moreover

{ assume $b=v$
with a1 a2(1) have $b=\min$ using assms(1) unfolding IsLinOrder_def
antisym_def by auto
with a2(2) have False by auto
}
ultimately have False using V(2) b(2) unfolding RightRayX_def
using $<v \in X-U>$ by auto
}
moreover

{ assume $V \in \{\text{LeftRay}(X,r,b), b \in X\}$
then obtain $b$ where $b:V=\text{LeftRay}(X,r,b)$ by auto

{ assume $(v,b) \in r$
then have $b=v \vee v \in \text{LeftRay}(X,r,b)$ unfolding LeftRayX_def using $<v \in X-U>$ by auto
then have $b=v$ using b(1) V(2) $<v \in X-U>$ by auto
}
then have bv: $(v,b) \in r$ using assms(1) unfolding IsLinOrder_def
IsTotal_def using b(2)
$<v \in X-U>$ by auto
from b(1) V(1) have $(\min,b) \in r_{\min} \neq b$ unfolding LeftRayX_def by auto
with assms(4) obtain $z$ where $z:(\min,z) \in r_{\min} \neq b$ unfolding LeftRayX_def by auto
unfolding IsDense_def
using b(2) V(1,2) $<U \subseteq X>$ by blast
then have rayb: $z \in \text{LeftRay}(X,r,b)$ unfolding LeftRayX_def by auto
from z(2) bv have $(z,v) \in r$ using assms(1) unfolding IsLinOrder_def
trans_def by fast
moreover

{ assume $z=v$
with bv have $(b,z) \in r$ by auto

967
with \( z(2) \) have \( b = z \) using assms(1) unfolding IsLinOrder_def
antisym_def by auto
then have False using \( z(3) \) by auto }
ultimately have \( z \in \text{LeftRayX}(X,r,v) \) unfolding LeftRayX_def using \( z(3) \) by auto
with rayb have \( z \in U \cap \text{LeftRayX}(X,r,v) \) using \( V(2) \) \( b(1) \) 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 \( z(1,3) \) have False using assms(1) unfolding IsLinOrder_def
antisym_def by auto }
moreover }
assume \( V \in \{\text{IntervalX}(X,r,b,c) \mid \langle b,c \rangle \in X \times X\} \)
then obtain \( b \ c \) where \( b: V = \text{IntervalX}(X,r,b,c) \ b \in X \ c \in X \) by auto
from \( b \ V(1) \) have \( m: \langle \min , c \rangle \in r \langle b, \min \rangle \in r \min \neq b \ min \neq c \) unfolding IntervalX_def Interval_def by auto
\{ assume \( A: \langle c,v \rangle \in r \)
from \( m \) obtain \( z \) where \( z: \langle z,c \rangle \in r \langle \min , z \rangle \in r \langle z, \min \rangle \in r \langle z, \min \rangle \in X - \{c, \min \} \) using assms(4)
unfolding IsDense_def
using \( b(3) \ V(1,2) \ U \subseteq X \) by blast
from \( z(2) \) have \( \langle b,z \rangle \in r \) using \( m(2) \) assms(1)
unfolding IsLinOrder_def
trans_def by fast
with \( z(1) \) have \( z \in \text{IntervalX}(X,r,b,c) \setminus z = b \) using \( z(3) \) unfolding IntervalX_def
Interval_def by auto
then have \( z \in \text{IntervalX}(X,r,b,c) \) 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 \( \langle z,v \rangle \in r \) 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 \( z \in U \cap \text{LeftRayX}(X,r,v) \) 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 \( vc: \langle v,c \rangle \in r \) using assms(1)
unfolding IsLinOrder_def
968
IsTotal_def using \( <v \in X - U> \)

\[ b(3) \text{ by auto} \]

\{ 
  \text{assume } \min = v 
  \text{ with } V(2,1) <v \in X - U> \text{ have False by auto} 
\}

then have \( \min \neq v \) by auto

with a1 obtain \( z \) where \( \langle \min, z \rangle \in r \)

\text{using assms(4) unfolding IsDense_def}

using \( V(1,2) <U \subseteq X> <v \in X - U> \) by blast

from z(2) vc(1) have \( zc : \langle z, c \rangle \in r \) 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_L1B by auto moreover

\{ 
  \text{assume } z = c 
  \text{ 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

\{ 
  \text{assume } z = b 
  \text{ then have } z = \min \text{ using m(2) z(1) using assms(1) unfolding IsLinOrder_def} 
\}

IsLinOrder_def

antisym_def by auto

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 \( z \in \text{Interval}(X, r, b, c) \) unfolding 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 \( z \in \text{LeftRay}(X, r, v) \) unfolding LeftRayX_def by auto ultimately

have \( z \in U \cap \text{LeftRay}(X, r, v) \) 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 \( z(1,3) \) have False using assms(1) unfolding IsLinOrder_def

antisym_def by auto

\}

ultimately have False using \( V(3) \) by auto

\}

then have \text{ass:min} \( \in X - U \) using a1 assms(3) by auto

969
then obtain $V$ where $V : \min \in V \subseteq X - U$

$V \in \{\text{Interval}(X,r,b,c). \langle b,c \rangle \in X \times X\} \cup \{\text{RightRay}(X,r,b). b \in X\}$ using point_open_base_neigh

$[\text{OF Ortdtopology_is_a_topology}(2) \text{OF assms}(1)] \text{OF assms}(1)$

$X - U \in (\text{OrdTopology} X r)$

by blast

\begin{verbatim}
{ assume $V \in \{\text{Interval}(X,r,b,c). \langle b,c \rangle \in X \times X\}$
  then obtain $b$ $c$ where $b : V = \text{Interval}(X,r,b,c)$ $b \in X$ $c \in X$
  by auto

  from $b$ $V(1)$ have $m : (\min, c) \in r$ $b, \min \in X$
  unfolding IntervalX_def
  by auto

  { fix $x$
    assume $A : x \in U \cap \text{LeftRay}(X,r,x)$
    then have $\langle x,v \rangle \in r$ $x \in U$
      unfolding LeftRayX_def
    by auto
    then have $x / \in V$ using $V(2)$
      by auto
    then have $x / \in \text{Interval}(r, b, c) \cap X$
      $\lor x = b$ $\lor x = c$
      unfolding Interval_def
    by auto
    then have $\langle x,b \rangle \in r$ $\lor \langle c,x \rangle \in r$
      $\lor x = b$ $\lor x = c$
      using $\text{assms}(1)$
      unfolding IsLinOrder_def
    by auto
    ultimately have $\langle x,b \rangle \in r$ $\lor \langle c,x \rangle \in r$
      using $\text{assms}(1)$
      unfolding IsLinOrder_def
    by auto
    with $m(1)$ have $\langle x,b \rangle \in r$ $\lor \langle c, \min \rangle \in r$
      using $\text{assms}(1)$
      unfolding IsLinOrder_def
    by auto
    with $m(4)$ have $\langle x,b \rangle \in r$
      by auto
  }

  then have $\langle x,b \rangle \in r$ using $\text{Order}_2.L_1$ $\text{assms}(1)$
  unfolding Supremum_def
  by auto

  with $m(2,3)$ have False using $\text{assms}(1)$ unfolding IsLinOrder_def
  antisym_def
  by auto
}

moreover

\begin{verbatim}
{ assume $V \in \{\text{RightRay}(X,r,b). b \in X\}$
  then obtain $b$ where $b : V = \text{RightRay}(X,r,b)$ $b \in X$
  by auto

  from $b$ $V(1)$ have $m : (b, \min) \in r$ $\min \neq b$
    unfolding RightRayX_def
  by auto

  { fix $x$
    assume $A : x \in U \cup \text{LeftRay}(X,r,x)$
    then have $\langle x,v \rangle \in r$ $x \in U$
      unfolding LeftRayX_def
    by auto
    then have $x / \in V$ using $V(2)$
      by auto
  }

  then have $\langle \min, b \rangle \in r$
    using $\text{Order}_2.L_3$ $\text{assms}(1)$
    unfolding IsLinOrder_def
  by auto

  with $m(2,3)$ have False using $\text{assms}(1)$ unfolding IsLinOrder_def
  antisym_def
  by auto
}
\end{verbatim}

970
then have $x \in \text{RightRay}_X(X, r, b)$ using $b(1)$ by auto
then have $(\langle b, x \rangle, x) \in r \vee x = b \in X$ unfolding RightRayX_def using $\langle x \in U \rangle < U \subseteq X$ by auto
then have $(x, b) \in r$ using assms(1) unfolding IsLinOrder_def using total_is_refl unfolding refl_def unfolding IsTotal_def using $b(2)$ by auto
then have $(\langle b, x \rangle, x) \in r \vee x = b \in X$ unfolding RightRayX_def by auto
with $m(2,1)$ have False using assms(1) unfolding IsLinOrder_def antisym_def by auto
moreover
\begin{enumerate}
\item assume $V \in \{\text{LeftRay}_X(X, r, b) \mid b \in X\}$
then obtain $b$ where $b : V = \text{LeftRay}_X(X, r, b) \subseteq X$ by auto
from $b$ $V(1)$ have $m : (\langle \text{min}, b \rangle, \text{min} \neq b) \subseteq \text{LeftRay}_X(X, r, b)$ unfolding LeftRayX_def by auto
\item fix $x$ assume $A : x \in U \cap \text{LeftRay}_X(X, r, v)$
then have $(x, v) \in r \subseteq U$ unfolding LeftRayX_def by auto
then have $x \in V$ using $V(2)$ by auto
then have $(\langle x, v \rangle, x) \in r \subseteq U$ unfolding LeftRayX_def using $\langle x \in U \rangle < U \subseteq X$ by auto
then have $(\langle b, x \rangle, x) \in r \vee x = b \in X$ unfolding RightRayX_def by auto
with $m(1)$ have $(\langle \text{min}, x \rangle, \text{min}) \in r$ using assms(1) unfolding IsLinOrder_def trans_def by fast
moreover
\item assume $V \in \{\text{LeftRay}_X(X, r, b) \mid b \in X\}$
then obtain $b$ where $b : V = \text{LeftRay}_X(X, r, b) \subseteq X$ by auto
from $b$ $V(1)$ have $m : (\langle \text{min}, b \rangle, \text{min} \neq b) \subseteq \text{LeftRay}_X(X, r, b)$ unfolding LeftRayX_def by auto
\item fix $x$ assume $A : x \in U \cap \text{LeftRay}_X(X, r, v)$
then have $(x, v) \in r \subseteq U$ unfolding LeftRayX_def by auto
then have $x \in V$ using $V(2)$ by auto
then have $(\langle x, v \rangle, x) \in r \subseteq U$ unfolding LeftRayX_def using $\langle x \in U \rangle < U \subseteq X$ by auto
then have $(\langle b, x \rangle, x) \in r \vee x = b \in X$ unfolding RightRayX_def by auto
with $m(1)$ have $(\langle \text{min}, x \rangle, \text{min}) \in r$ using assms(1) unfolding IsLinOrder_def trans_def by fast
moreover
\end{enumerate}
ultimately have $x = \text{min}$ using assms(1) unfolding IsLinOrder_def antisym_def by auto
then have $U \subseteq \text{LeftRay}_X(X, r, v) \subseteq \{\text{min}\}$ by auto
moreover
\begin{enumerate}
\item assume $\text{min} \in U \subseteq \text{LeftRay}_X(X, r, v)$
then have $\text{min} \in U$ by auto
\end{enumerate}
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 \langle v, u \rangle \in r unfolding IsLinOrder_def IsTotal_def using
<br \subseteq X>
<br \in X-U> by auto

have RightRayX(X, r, v) \in (OrdTopology X r) using base_sets_open[OF Ordtopology_is_a_topology assms(2)] unfolding IsATopology_def by auto
<br \subseteq X>
<br \in X-U> by auto

then have U \cap RightRayX(X, r, v) \in (OrdTopology X r) using assms(5) unfolding IsATopology_def by auto
<br \subseteq X>
<br \in X-U>

let max = Infimum(r, U \cap RightRayX(X, r, v))
<br \subseteq X>
<br \in X-U> unfolding IsLinOrder_def by auto

{ assume max: max \in U then obtain V where V:max \in V \subseteq U V \in \{IntervalX(X, r, b, c). \langle b, c \rangle \in X \times X\} \cup \{LeftRayX(X, r, b). b \in X\} \cup \{RightRayX(X, r, b). b \in X\} using point_open_base_neigh[of Ordtopology_is_a_topology assms(1)] \in (OrdTopology X r) assms(5) unfolding IsLinOrder_def by blast

assume V\in\{RightRayX(X, r, b). b\in X\}
then obtain b where b\in X V=RightRayX(X, r, b) by auto
from V(1) b(2) have a2:\langle b, max \rangle \in r \neq b unfolding RightRayX_def by auto

assume \langle b, v \rangle \in r
then have b=v \forall v\in RightRayX(X, r, b) unfolding RightRayX_def using \langle v \in X-U \rangle by auto
then have $b=v$ using $b(2) \cdot v \in X - U$ by auto

then have $bv : \langle v, b \rangle \in r$ using assms(1) unfolding IsLinOrder_def IsTotal_def using $b(1) - v \in X - U$ by auto

from $a2$ assms(4) obtain $z : \langle b, z \rangle \in r \cdot \langle z, max \rangle \in rz \cdot X - \{b, max\}$ unfolding IsDense_def using $b(1) \cdot V(1, 2) - U \subseteq X$ by blast
then have rayb : $z \in \text{RightRayX}(X, r, b)$ unfolding RightRayX_def by auto
from $z(1)$ bv have $\langle v, z \rangle \in r$ using assms(1) unfolding IsLinOrder_def trans_def by fast
moreover

\begin{itemize}
  \item assume $z=v$
  with bv have $\langle z, b \rangle \in r$ 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
\end{itemize}

ultimately have $z \in \text{RightRayX}(X, r, v)$ unfolding RightRayX_def using $z(3)$ by auto

with rayb have $z \in U - \text{RightRayX}(X, r, v)$ using $V(2) \cdot b(2)$ by auto
then have $max \in r - \{z\}$ using Order_ZF_4_L3(1)[OF _ Hmax] assms(1) unfolding infimum_def IsLinOrder_def by auto
then have $\langle max, z \rangle \in r$ by auto
with $z(2, 3)$ have False using assms(1) unfolding IsLinOrder_def antisym_def by auto

moreover

\begin{itemize}
  \item assume $\forall \in \{\text{LeftRayX}(X, r, b) \cdot b \in X\}$
  then obtain $b$ where $b : \forall \subseteq \text{LeftRayX}(X, r, b) \cdot b \in X$ by auto
  note $a1$ moreover
  from $V(1) \cdot b(1)$ have $a2 : \langle max, b \rangle \in \text{max} \neq b$ unfolding LeftRayX_def by auto
  ultimately have $\langle v, b \rangle \in r$ using assms(1) unfolding IsLinOrder_def trans_def by blast
moreover

\begin{itemize}
  \item assume $b=v$
  with $a1$ a2(1) have $b = max$ using assms(1) unfolding IsLinOrder_def antisym_def by auto
  with $a2(2)$ have False by auto
\end{itemize}

ultimately have False using $V(2) \cdot b(1)$ unfolding LeftRayX_def using $\langle v \in X - U\rangle$ by auto

moreover

\begin{itemize}
\end{itemize}

973
assume \( V \in \{ \text{IntervalX}(X,r,b,c) \} \). \( \langle b,c \rangle \in X \times X \) 
then obtain \( b \), \( c \) where \( b : V = \text{IntervalX}(X,r,b,c) \) \( b \in X \) \( c \in X \) by auto 
from \( b \) \( V(1) \) have \( m : (\max,c) \in r(b,\max) \in r \max \neq b \max \neq c \) unfolding \( \text{IntervalX_def} \) 
\( \text{Interval_def} \) by auto 
\{ 
  assume \( A : (v,b) \in r \) 
  from \( m \) obtain \( z \) where \( z : (\max,z) \in r \) \( b,z \in X \) \( \{ b,max \} \) using \( \text{assms(4)} \) unfolding \( \text{IsDense_def} \) 
  using \( b(2) \) \( V(1,2) \) \( \cup X \) by blast 
  from \( z(1) \) have \( (z,c) \in r \) using \( m(1) \) \( \text{assms(1)} \) unfolding \( \text{IsLinOrder_def} \) 
  by fast 
  with \( z(2) \) have \( z \in \text{IntervalX}(X,r,b,c) \) \( v = c \) using \( \text{z(3)} \) unfolding \( \text{IntervalX_def} \) 
  \( \text{Interval_def} \) by auto 
  then have \( z \in \text{IntervalX}(X,r,b,c) \) using \( m(1) \) \( z(1,3) \) using \( \text{assms(1)} \) unfolding \( \text{IsLinOrder_def} \) 
  \( \text{antisym_def} \) 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 \( \text{assms(1)} \) unfolding \( \text{IsLinOrder_def} \) 
  \( \text{trans_def} \) by fast 
  moreover have \( z \neq v \) using \( A \) \( z(2,3) \) \( \text{assms(1)} \) unfolding \( \text{IsLinOrder_def} \) 
  \( \text{antisym_def} \) by auto 
  ultimately have \( z \in U \) \( \cup \text{RightRayX}(X,r,v) \) unfolding \( \text{RightRayX_def} \) 
  using \( z(3) \) by auto 
  then have \( \max \in r - \{ z \} \) using \( \text{Order_ZF_4_L3(1)[OF _ Hmax]} \) \( \text{assms(1)} \) unfolding \( \text{Infimum_def} \) 
  \( \text{IsLinOrder_def} \) by auto 
  then have \( (\max,z) \in r \) by auto 
  with \( z(1,3) \) have \( \text{False} \) using \( \text{assms(1)} \) unfolding \( \text{IsLinOrder_def} \) 
  \( \text{antisym_def} \) by auto 
  \} 
  then have \( vc : \langle b,v \rangle \in rv \neq b \) using \( \text{assms(1)} \) unfolding \( \text{IsLinOrder_def} \) 
  \( \text{IsTotal_def} \) using \( \langle v \in X - U \rangle \) 
  \( b(2) \) by auto 
  \{ 
  assume \( v = \max \) 
  with \( V(2,1) \) \( \langle v \in X - U \rangle \) have \( \text{False} \) by auto 
  \} 
  then have \( v \neq \max \) by auto moreover 
  \( \text{note a1 moreover} \) 
  have \( \max \in X \) using \( V(1,2) \) \( \cup X \) by auto 
  moreover have \( v \in X \) using \( \langle v \in X - U \rangle \) by auto 
  ultimately obtain \( z \) where \( z : (v,z) \in r(z,\max) \in rz \in X - \{ v,\max \} \) using \( \text{assms(4)} \) unfolding \( \text{IsDense_def} \) 
  by auto 
  from \( z(1) \) \( vc(1) \) have \( zc : \langle b,z \rangle \in r \) using \( \text{assms(1)} \) unfolding \( \text{IsLinOrder_def} \) 
  \( \text{trans_def} \) by fast moreover
from \( m(1) \) \( z(2) \) have \( (z, c) \in r \) using \( \text{assms(1)} \) unfolding \( \text{IsLinOrder_def} \)

\[
\text{trans_def} \\
\text{by fast ultimately have } z \in \text{Interval}(r, b, c) \text{ using } \text{Order_ZF_2_L1B} \text{ by auto moreover} \\
\{
\text{assume } z = b \\
\text{then have False using } z(1) \text{ vc using } \text{assms(1)} \text{ unfolding } \text{IsLinOrder_def}
\}
\text{antisym_def} \\
\text{by fast}
\}
\text{then have } z \neq b \text{ by auto moreover} \\
\{
\text{assume } z = c \\
\text{then have } z = \text{max using } m(1) \text{ } z(2) \text{ using } \text{assms(1)} \text{ unfolding } \text{IsLinOrder_def}
\text{antisym_def by auto} \\
\text{with } z(3) \text{ have False by auto}
\}
\text{then have } 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{IntervalX}(X, r, b, c) \text{ unfolding } \text{IntervalX_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{RightRayX}(X, r, v) \text{ unfolding } \text{RightRayX_def} \text{ by auto ultimately} \\
\text{have } z \in U \cap \text{RightRayX}(X, r, v) \text{ by auto} \\
\text{then have } \text{max} \in r \setminus \{z\} \text{ using } \text{Order_ZF_4_L3(1)[OF } \_\text{Hmax]} \text{ } \text{assms(1)} \\
\text{unfolding } \text{Infimum_def IsLinOrder_def} \text{ by auto} \\
\text{then have } (\text{max}, z) \in r \text{ by auto} \\
\text{with } z(2, 3) \text{ have False using } \text{assms(1)} \text{ unfolding } \text{IsLinOrder_def} \text{ antisym_def by auto}
\}
\text{ultimately have False using } V(3) \text{ by auto}
\}
\text{then have } \text{ass: } \text{max} \in X \setminus U \text{ using } a1 \text{ assms(3) by auto} \\
\text{then obtain } V \text{ where } V : \text{max} \in V \subseteq X \setminus 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\} \text{ using } \text{point_open_base_neigh} \\
\text{[OF Ordtopology_is_a_topology(2)[OF } \text{assms(1)] <X-U}(\text{OrdTopology}\ X \ r)> \text{ ass]} \text{ by blast}
\{
\text{assume } V \in \{\text{IntervalX}(X, r, b, c). \ (b, c) \in X \times X\} \\
\text{then obtain } b \text{ c where } b: \text{IntervalX}(X, r, b, c) b \in X c \in X \text{ by auto} \\
\text{from } b \text{ V(1) have } m: (\text{max}, c) \in r \text{ } (b, \text{max}) \in r \text{max} \neq b \text{ max} \neq c \text{ unfolding } \text{IntervalX_def} \text{ by auto}
\}
\text{fix } x \text{ assume } A: x \in U \cap \text{RightRayX}(X, r, v) \\
\text{then have } (v, x) \in r x \in U \text{ unfolding } \text{RightRayX_def by auto} \\
\text{then have } x \notin V \text{ using } V(2) \text{ by auto}

975
then have \( x \in \text{Interval}(r, b, c) \cap X \) using \( \text{b(1)} \) unfolding \( \text{IntervalX_def} \) by auto
then have \( (b, x) \not\in r \lor (x, c) \not\in r \) using \( \text{OrderZF_2_L1B} \) by auto
then have \( (x, b) \in r \lor (x, c) \in r \) using \( \text{assms(1)} \) unfolding \( \text{IsLinOrder_def} \)
using \( \text{total_is_refl} \) unfolding \( \text{refl_def} \) by auto
moreover from \( A \) have \( (\max, x) \in r \) using \( \text{Order_ZF_4_L3(1)[OF _ Hmax]} \) \( \text{assms(1)} \)
ultimately have \( (\max, b) \in r \) using \( \text{assms(1)} \) unfolding \( \text{IsLinOrder_def} \)
trans_def by fast
with \( \text{m(2)} \) have \( (\max, b) = b \) using \( \text{IsLinOrder_def} \)
antisym_def by auto
with \( \text{m(3)} \) have \( (c, x) \in r \) by auto
\}
then have \( (c, \max) \in r \) using \( \text{Order_ZF_5_L4[OF _ nE Hmax]} \) \( \text{assms(1)} \)
with \( \text{m(1,4)} \) have \( \text{False} \) using \( \text{assms(1)} \) unfolding \( \text{IsLinOrder_def} \)
antisym_def by auto
moreover
\{ assume \( V \in \{\text{RightRay}^1(X, r, b) \} \)
then obtain \( b \) where \( b : V = \text{RightRay}^1(X, r, b) \) \( b \in X \) by auto
from \( b \) \( V(1) \) have \( m : (b, \max) \neq b \) unfolding \( \text{RightRayX_def} \) by auto
\{
fix \( x \) assume \( A : x \in U \cap \text{RightRay}^1(X, r, v) \)
then have \( (v, x) \in U \) unfolding \( \text{RightRayX_def} \) by auto
then have \( x \in V \) using \( V(2) \) by auto
then have \( x \in \text{RightRay}^1(X, r, b) \) unfolding \( \text{RightRayX_def} \) using \( \text{<U>U} \subseteq X \) by auto
\{
then have \( (b, x) \in r \) using \( \text{assms(1)} \) unfolding \( \text{IsLinOrder_def} \)
using \( \text{total_is_refl} \) unfolding \( \text{refl_def} \) by auto
moreover from \( A \) have \( (\max, x) \in r \) using \( \text{OrderZF_4_L3(1)[OF _ Hmax]} \) \( \text{assms(1)} \)
ultimately have \( (\max, b) \in r \) using \( \text{assms(1)} \) unfolding \( \text{IsLinOrder_def} \)
trans_def by fast
with \( m \) have \( \text{False} \) using \( \text{assms(1)} \) unfolding \( \text{IsLinOrder_def} \)
antisym_def by auto
\}
then have \( \text{False} \) using \( nE \) by auto
976
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 \( m : (\max,b) \in r \max \neq b \) unfolding LeftRayX_def by auto

with \( nE \) have \( b \in (\bigcap c \in U \cap \text{RightRayX}(X,r,v). r-\{c\}) \) by auto
then have \( (b,max) \in r \) unfolding Infimum_def using Order_ZF_4_L3[OF _ Hmax assms(1)] unfolding IsLinOrder_def  using total_is_refl unfolding refl_def unfolding IsTotal_def using assms(1) unfolding IsLinOrder_def by auto

moreover note \( V(3) \)
ultimately have \( False \) using assms(1) unfolding IsLinOrder_def antisym_def by auto

then show thesis by auto
qed

71.4 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 \( r1 \) \( r2 \) assume \( r1 \in Xr2 \in Xr1 \neq r2 \) \( \langle r1,r2 \rangle \in r \)
then have IntervalX(X,r,r1,r2) \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 \} \) by auto
then have P:IntervalX(X,r,r1,r2) \in (OrdTopology X r) using base_sets_open[OF Ordtopology_is_a_topology(2)[OF assms(1)]]
by auto

977
have $\text{Interval}(X,r,r_1,r_2) \subseteq X$ unfolding $\text{Interval}_X$ def by auto

then have $\text{int} : \text{Closure}(A, \text{OrdTopology } X \ r) \cap \text{Interval}(X,r,r_1,r_2) = \text{Interval}(X,r,r_1,r_2)$ using assms(2) by auto

\begin{itemize}
    \item assume $\text{Interval}(X,r,r_1,r_2) \neq 0$
    \item then have $A \cap (\text{Interval}(X,r,r_1,r_2)) \neq 0$ using topology0.cl_inter_neigh[OF topology0_ordtopology[OF assms(1)] _ P , of A]
    \item using assms(3) union_ordtopology[OF assms(1,4)] int by auto
\end{itemize}

then have $(\exists z \in A \setminus \{r_1,r_2\}. \langle r_1, z \rangle \in r \land \langle z, r_2 \rangle \in r) \lor \text{Interval}(X,r,r_1,r_2) = 0$ unfolding $\text{Interval}_X$ def $\text{Interval}_X$ def by auto

then show thesis unfolding $\text{IsWeaklyDenseSub}$ 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 $\{\text{min, max}\}$ is open: $A$ is topologically dense in $X \setminus \{\text{min, max}\}$, where $\text{min}$ is the minimum in $X$ and $\text{max}$ is the maximum in $X$.
- If $X$ has a maximum, $\{\text{max}\}$ is open and $X$ has no minimum or $\{\text{min}\}$ isn’t open: $A$ is topologically dense in $X \setminus \{\text{max}\}$, where $\text{max}$ is the maximum in $X$.
- If $X$ has a minimum, $\{\text{min}\}$ is open and $X$ has no maximum or $\{\text{max}\}$ isn’t open $A$ is topologically dense in $X \setminus \{\text{min}\}$, where $\text{min}$ is the minimum in $X$.
- If $X$ has no minimum or maximum, or $\{\text{min, max}\}$ has no proper open sets: $A$ is topologically dense in $X$.

**Lemma:** $\text{Wdense Ord Imp Dense Top}$:

assumes $\text{IsLinOrder}(X,r)$ $A$ (is weakly dense in) $X$ (with respect to) $r \subseteq X$

$\exists x \ y. \ x \neq y \land x \in X \land y \in X$

$\text{HasMinimum}(r,X) \implies \{\text{Minimum}(r,X)\} \in (\text{OrdTopology } X \ r) \implies \text{Minimum}(r,X) \in A$

$\text{HasMaximum}(r,X) \implies \{\text{Maximum}(r,X)\} \in (\text{OrdTopology } X \ r) \implies \text{Maximum}(r,X) \in A$

shows $\text{Closure}(A, \text{OrdTopology } X \ r) = X$

**Proof:**

\begin{itemize}
    \item fix $x$ assume $x \in X$
    \item fix $U$ assume ass:$x \in U \in (\text{OrdTopology } X \ r)$
\end{itemize}
then have $\exists V \in \{\text{Interval}_X(X, r, b, c) \cdot (b, c) \in X \times X\} \cup \{\text{LeftRay}_X(X, r, b) \cdot b \in X\} \cup \{\text{RightRay}_X(X, r, b) \cdot b \in X\} \cup U \cup x \in V$

by auto

then obtain $V$ where $V \in \{\text{Interval}_X(X, r, b, c) \cdot (b, c) \in X \times X\} \cup \{\text{LeftRay}_X(X, r, b) \cdot b \in X\} \cup \{\text{RightRay}_X(X, r, b) \cdot b \in X\} \cup U \cup x \in V$

by blast

note $V(1)$ moreover

\{ assume $V \in \{\text{Interval}_X(X, r, b, c) \cdot (b, c) \in X \times X\}$

then obtain $b \ c$ where $b \ b \in X \text{Interval}_X(X, r, b, c) \choose X \times X$ by auto

with $V(3)$ have $x \in X \times X \text{Interval}_X(X, r, b, c) \choose X \times X$ unfolding $\text{Interval}_X$ by auto

\{ assume $V \in \{\text{LeftRay}_X(X, r, b) \cdot b \in X\}$

then obtain $b$ where $b \ b \in X \text{LeftRay}_X(X, r, b)$ by auto

with $V(3)$ have $x \in X \times X \text{LeftRay}_X(X, r, b) \choose X \times X$ unfolding $\text{LeftRay}_X$ by auto

moreover have $x(1-3)$ have $b \neq c$ using $\text{assms}(1)$ unfolding $\text{IsLinOrder}_X$ by fast

moreover note $\text{assms}(2)$

ultimately have $\exists z \in A - \{b, c\}. (b, z) \in r \wedge (z, c) \in r$ unfolding $\text{IsWeaklyDenseSub}_X$ by auto

\{ assume $B : \text{Interval}_X(X, r, b, x) = 0$

\{ assume $y \in X. (x, y) \in r \wedge x \neq y$

then obtain $y$ where $y \ y \in X \times X \text{Interval}_X(X, r, b, y) \choose X \times X$ by auto

with $x$ have $x \in X \times X \text{Interval}_X(X, r, b, y) \choose X \times X$ unfolding $\text{Interval}_X$ by auto

moreover have $\text{assms}(1)$ unfolding $\text{IsLinOrder}_X$ by fast

moreover have $b \neq y$ using $y(2, 3)$ unfolding $\text{IsLinOrder}_X$ by fast

antism_def by fast

ultimately have $\exists z \in A - \{b, c\}. (b, z) \in r \wedge (z, c) \in r$ unfolding $\text{IsWeaklyDenseSub}_X$ by auto
ultimately have \((\exists z \in A - \{b, y\}. (b, z) \in r \land (z, y) \in r)\) using assms(2) unfolding IsWeaklyDenseSub_def
  using y(1) b(1) by auto
  then obtain \(z\) where \(z \in A - \{b\} \land (b, z) \in r\) by auto
  then have \(z \in A \cap V\) using b(2) unfolding RightRayX_def using assms(3) by auto
  then have \(z \in A \cap U\) using V(2) by auto
  then have \(A \cap U \neq 0\) by auto

  moreover
  { assume \(R:\forall y \in X. (x, y) \in r \rightarrow x = y\)
    { fix \(y\) assume \(y \in \text{RightRayX}(X, r, b)\)
      then have \(\langle b, y \rangle \in r\) \(y \in X - \{b\}\) unfolding RightRayX_def by auto
      { assume \(A: y \neq x\)
        then have \(\langle x, y \rangle \notin r\) using \(R\) y(2) by auto
        then have \(\langle y, x \rangle \in r\) using assms(1) unfolding IsLinOrder_def IsTotal_def
          using \(x \in X\) by auto
        with \(A\) have \(y \in \text{IntervalX}(X, r, b, x)\) unfolding IntervalX_def by auto
        then have \(False\) using B by auto
      }
      moreover
      { assume \(t \neq x\)
        then have \(\langle x, t \rangle \notin r\) using \(R\) T by auto
        then have \(\langle t, x \rangle \in r\) using assms(1) unfolding IsLinOrder_def IsTotal_def
          using \(T < x \in X\) by auto
      }
    }
    ultimately have \(\langle t, x \rangle \in r\) by auto
  }
  with \(x \in X\) have \(\text{HM:HasAmaximum}(r, X)\) unfolding HasAmaximum_def
by auto
  then have $\text{Maximum}(r, X) \in X \forall t \in X. \langle t, \text{Maximum}(r, X) \rangle \in r$ using Order_ZF_4_L3
assms(1) unfolding IsLinOrder_def
  by auto
  with $R \prec x \in X$ have $x = \text{Maximum}(r, X)$ 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 base_sets_open[OF Ordtopology_is_a_topology(2) [OF assms(1)]]
V(1) by auto
with HM have $\text{Maximum}(r, X) \in A$ using assms(6) by auto
moreover have $x \in A$ by auto
with V(2,3) have $A \cap U \neq \emptyset$ by auto
moreover
  { assume $\text{IntervalX}(X, r, b, x) \neq 0$
    with disj have $\exists z \in A - \{b, x\}. \langle b, z \rangle \in r \land \langle z, x \rangle \in r$ by auto
    then obtain $z$ where $z \in A \land b \neq z \in r$ by auto
    then have $z \in A \land \text{RightRayX}(X, r, b)$ unfolding RightRayX_def using assms(3) by auto
    then have $z \in A \land U$ using V(2) b(2) by auto
    then have $A \cap U \neq \emptyset$ by auto
  }
ultimately have $A \cap U \neq \emptyset$ by auto
moreover
  { assume $V \in \{\text{LeftRayX}(X, r, b) . b \in X\}$
    then obtain $b$ where $b : b \in X = \text{LeftRayX}(X, r, b)$ by auto
    with V(3) have $x : \langle x, b \rangle \in r \land b \neq x$ unfolding LeftRayX_def by auto
moreover
  note assms(2) moreover
  have $U \subseteq \bigcup (\text{OrdTopology} X r)$ using ass(2) by auto
  then have $U \subseteq X$ using union_ordtopology[OF assms(1,4)] by auto
  then have $x \in X$ using ass(1) by auto
moreover
  note assms(2) ultimately
  have disj: $\exists z \in A - \{b, x\}. \langle x, z \rangle \in r \land \langle z, b \rangle \in r \lor \text{IntervalX}(X, r, x, b)$ = 0 unfolding IsWeaklyDenseSub_def by auto
  
  { assume $\text{IntervalX}(X, r, x, b) = 0$
    
    { assume $\exists y \in X. \langle y, x \rangle \in r \land x \neq y$
      then obtain $y$ where $y : y \in X \langle y, x \rangle \in r \land x \neq y$ by auto
      with $x$ have $x \in \text{IntervalX}(X, r, y, b)$ unfolding IntervalX_def interval_def using $\prec x \in X$ by auto
moreover
  have $\langle y, b \rangle \in r$ using y(2) x(1) assms(1) unfolding IsLinOrder_def

981
trans_def by fast
moreover have \( b \neq y \) using \((2,3)\) \( x(1) \) assms(1) unfolding IsLinOrder_def
antisym_def by fast
ultimately
have \((\exists z \in A - \{b,y\}. (y,z) \in r \wedge (z,b) \in r)\) using assms(2) unfolding IsWeaklyDenseSub_def
using \( y(1) \) \( b(1) \) by auto
then obtain \( z \) where \( z \in A \cap V \) using \( b(2) \) unfolding LeftRayX_def using assms(3)
by auto
then have \( z \in A \cap U \) using \( V(2) \) by auto
then have \( A \cap U \neq 0 \) by auto
moreover
\{ assume R: \( \forall y \in X. (y,x) \in r \rightarrow x=y \)
\{ fix \( y \) assume \( y \in \text{LeftRayX}(X,r,b) \)
then have \( y : (y,b) \in r \ y \in X - \{b\} \) unfolding LeftRayX_def by auto
\{ assume A: \( y \neq x \)
then have \( (y,x) \notin r \) using R \( y(2) \) by auto
then have \( (x,y) \in r \) using assms(1) unfolding 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 IntervalX_def
interval_def
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
\{ fix \( t \) assume T: \( t \in X \)
\{ assume \( t = x \)
then have \( \langle x,t \rangle \in r \) using assms(1) unfolding IsLinOrder_def
using Order_ZF_1_L1 T by auto
\}
moreover
\{ assume \( t \neq x \)
then have \( \langle t,x \rangle \notin r \) using R T by auto
then have \( \langle x,t \rangle \in r \) using assms(1) unfolding IsLinOrder_def
IsTotal_def
using \( T < x \in X \) by auto
\}
982
ultimately have \( (x,t) \in r \) by auto

with \( \langle x \in X \rangle \) have \( \text{HM:HasMinimum}(r,X) \) unfolding \( \text{HasMinimum_def} \)

by auto
then have \( \text{Minimum}(r,X) \in X \) \( \forall t \in X. \langle \text{Minimum}(r,X),t \rangle \in r \) using \( \text{Order_ZF_4_L4} \)

unfolding \( \text{IsLinOrder_def} \)
by auto

with \( R < x \in X \rangle \) 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 \( \text{HM have Minimum}(r,X) \in A \) 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
\{ assume \( \text{IntervalX}(X, r, x, b) \neq 0 \)
with \( \text{disj have } \exists z \in A - \{ b, x \}. \langle x, z \rangle \in r \land \langle z, b \rangle \in r \) by auto
then obtain \( z \) where \( z \in A z \neq b \langle z, b \rangle \in r \) by auto
then have \( z \in A z \in \text{LeftRayX}(X,r,b) \) unfolding \( \text{LeftRayX_def} \) using \( \text{assms}(3) \)
by auto
then have \( z \in A \cap U \) using \( V(2) \) \( b(2) \) by auto
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 \( \langle x \in X \rangle \) moreover
note \( \text{assms}(3) \) \( \text{topology0.inter_neigh_cl[OF \text{topology0_ordtopology}[OF \text{assms}(1)]]} \)
\( \text{union_ordtopology}[OF \text{assms}(1,4)] \) 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 \( \text{topology0.Top_3_L11(1)[OF \text{topology0_ordtopology}[OF \text{assms}(1)]]} \)
\( \text{assms}(3) \) \( \text{union_ordtopology}[OF \text{assms}(1,4)] \) 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 ∈ X ∧ y ∈ X
IsLinOrder(X,r)
shows ∃ A ∈ Pow(X). A ≺ Q ∧ (A is weakly dense in) X (with respect to) r
proof-
from assms obtain U where U ∈ Pow(∪ (OrdTopology X r)) Closure(U, OrdTopology X r) = X ∪ Q unfolding IsSeparableOfCard_def by auto
then have U ∈ Pow(X) Closure(U, OrdTopology X r) = X ∪ Q using union_ordtopology[OF assms(3,2)]
by auto
with dense_top_imp_Wdense_ord[OF assms(3) _ _ assms(2)] show thesis
by auto
qed

theorem wdense_imp_separable:
assumes ∃ x y. x ≠ y ∧ x ∈ X ∧ y ∈ X (A is weakly dense in) X (with respect to) r
IsLinOrder(X,r) A ≺ Q InfCard(Q) A ⊆ X
shows (OrdTopology X r) (is separable of cardinal) Q
proof-
{ assume Hmin: HasAmaximum(r,X)
then have MaxX: Maximum(r,X) ∈ X using Order_ZF_4_L3(1) assms(3) unfolding IsLinOrder_def by auto

{ assume HMax: HasAminimum(r,X)
then have MinX: Minimum(r,X) ∈ X using Order_ZF_4_L4(1) assms(3) unfolding IsLinOrder_def by auto

let A = A ∪ {Maximum(r,X), Minimum(r,X)}
have Finite({Maximum(r,X), Minimum(r,X)}) by auto
then have (Maximum(r,X), Minimum(r,X)) ≺ nat using n_lesspoll_nat unfolding Finite_def using eq_lesspoll_trans by auto
moreover from assms(5) have nat ≺ Q ∨ 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 {Maximum(r,X), Minimum(r,X)} ≺ Q using lesspoll_trans by auto
with assms(4,5) have C: A ≺ 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 MaxX MinX assms(6) have S: A ⊆ X by auto
then have Closure(A, OrdTopology X r) = X using Wdense_ord_imp_dense_top[of assms(3) WeakDense _ _ assms(1)] by auto
then have thesis unfolding IsSeparableOfCard_def using union_ordtopology[OF

984
moreover 
{ 
assume nmin:\neg HasAminimum(r,X) 
let A=A \cup \{Maximum(r,X)\} 
have Finite({Maximum(r,X)}) by auto 
then have {Maximum(r,X)}\prec nat using n_lesspoll_nat 
  unfolding Finite_def using eq_lesspoll_trans by auto 
moreover 
from assms(5) have nat\prec Q\lor 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 {Maximum(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 MaxX 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 by auto 
then have thesis unfolding IsSeparableGfCard_def using union_ordtopology[of 
  assms(3,1)] S C by auto 
} 
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 nat using n_lesspoll_nat 
  unfolding Finite_def using eq_lesspoll_trans by auto 
moreover 
from assms(5) have nat\prec Q\lor 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 
} 
}
unfolding
  IsWeaklyDenseSub_def by auto
from MinX assms(6) have S:A⊆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:¬HasAminimum(r,X)
  let A=A
  from assms(4,5) have C:A≺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⊆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
qed

end

72 Properties in topology 2

theory Topology_ZF_properties_2 imports Topology_ZF_7 Topology_ZF_1b
  Finite_ZF_1 Topology_ZF_11
begin

72.1 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 @term"P" iff every point has a neighbourhood basis of subsets that have the property @term"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)))

72.2 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

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}(_{is locally}_ 90. (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}XA] by auto

ultimately have ∃c∈Pow(b). x∈Interior(c,IncludedSet(X,A))∧ Finite(c)

using rr by blast

}
then have \( \forall x \in \bigcup (\text{IncludedSet}(X,A)) \). \( \forall b \in (\text{IncludedSet}(X,A)) \). if \( x \in b \rightarrow (\exists c \in \text{Pow}(b)). x \in \text{Interior}(c,\text{IncludedSet}(X,A)) \land \text{Finite}(c) \) by auto
then show thesis using IsLocally_def includedset_is_topology by auto qed

72.3 Local compactness

definition
IsLocallyComp (_\{is locally-compact\} 70)
where T\{is locally\}(\lambda B. \lambda T. 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 \subseteq 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 \subseteq \bigcup T unfolding IsCompactOfCard_def by auto
then have AA:A \subseteq \bigcup (T\{restricted to\}B) using assms(2) unfolding RestrictedTo_def
by auto moreover
{
fix M assume M\in(Pow(T\{restricted to\}B)) A\subseteq\bigcup M
let M=\{S\in T. B\subseteq S\}
from \langle M\in(Pow(T\{restricted to\}B)) \rangle have \bigcup M\subseteq \bigcup M unfolding RestrictedTo_def
by auto
with \langle A\subseteq \bigcup M \rangle have A\subseteq \bigcup M\subseteq Pow(T) by auto
with assms have \exists N\in(Pow(M)). A\subseteq \bigcup N\land N\prec K unfolding IsCompactOfCard_def
by auto
then obtain N where N\in(Pow(M)) A\subseteq \bigcup N\land N\prec K by auto
then have N\{restricted to\}B\subseteq M unfolding RestrictedTo_def FinPow_def
by auto
moreover
let f=\{\langle B,B\cap B \rangle. B\in N\}
have f:N\rightarrow \bigcup (N\{restricted to\}B) unfolding Pi_def function_def domain_def
RestrictedTo_def by auto
then have f\in(surj(N,N\{restricted to\}B)) unfolding surj_def RestrictedTo_def
using apply_equal
by auto
from \langle N\prec K \rangle have N\subseteq K unfolding lesspoll_def by auto
with \langle f\in(surj(N,N\{restricted to\}B)) \rangle have N\{restricted to\}B\subseteq N using surj_fun_inv_2 Card_is_Ord C by auto
with \langle N\prec K \rangle have N\{restricted to\}B\prec K unfolding lesspoll_trans1 by auto
moreover from \langle A\subseteq \bigcup N \rangle have A\subseteq \bigcup (N\{restricted to\}B) using assms(2)
unfolding RestrictedTo_def by auto
ultimately have \exists N\in(Pow(M)). A\subseteq \bigcup N \land N\prec K by auto

988
with AA show $A \subseteq \bigcup (T \{\text{restricted to} B\}) \land (\forall M \in \text{Pow}(T \{\text{restricted to} B\}). A \subseteq \bigcup M \land M \neq K))$ 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 $M \in \text{Pow}(T)$ $A \subseteq \bigcup M$
  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 $A \subseteq \bigcup M$
  have $A \subseteq \bigcup M$
  unfolding RestrictedTo_def
  using assms(2)
  by auto
  with assms
  obtain $N$ where $N \in \text{FinPow}(M)$
  unfolding IsCompact_def
  by blast
  ultimately
  obtain $f$ where $AA \in \text{Pi}(N, \lambda t. (\forall w \in N. \{S \in M. B \cap S = w\} \ t) \neq 0) \land (\forall t \in N. f \ t \in (\lambda w \in N. \{S \in M. B \cap S = w\} \ t)))$
  using finite_choice
  unfolding AxiomCardinalChoiceGen_def
  by blast
  ultimately
  obtain $f$ where $AA \in \text{Pi}(N, \lambda t. (\forall w \in N. \{S \in M. B \cap S = w\} \ t) \neq 0) \land (\forall t \in N. f \ t \in (\lambda w \in N. \{S \in M. B \cap S = w\} \ t)))$
  using finite_choice
  unfolding AxiomCardinalChoiceGen_def
  by blast
  ultimately
  obtain $f$ where $AA \in \text{Pi}(N, \lambda t. (\forall w \in N. \{S \in M. B \cap S = w\} \ t) \neq 0) \land (\forall t \in N. f \ t \in (\lambda w \in N. \{S \in M. B \cap S = w\} \ t)))$
  using finite_choice
  unfolding AxiomCardinalChoiceGen_def
  by blast

989
fix t assume t ∈ N
    with ss have ft ∈ {S ∈ M. B ∩ S ∈ N} by auto
} with A(1) have FF: f: N → {S ∈ M. B ∩ S ∈ N} unfolding Pi_def Sigma_def using beta_if by auto moreover
{ fix aa bb assume AAA: aa ∈ N bb ∈ N faa = fbb
    from AAA(1) ss have B ∩ (faa) = aa by auto
    with AAA(3) have B ∩ (fbb) = aa by auto
    with ss AAA(2) have aa = bb by auto
} ultimately have f ∈ inj(N, {S ∈ M. B ∩ S ∈ N}) unfolding inj_def by auto
then have f ∈ bij(N, range(f)) using inj_bij_range by auto
then have f ∈ bij(N, fN) using range_image_domain FF by auto
then have N ≈ {ft. t ∈ N} unfolding eqpoll_def by auto
with <N≈> have {ft. t ∈ N} ≈ n using eqpoll_sym eqpoll_trans by blast
with <n∈nat> have Finite({ft. t ∈ N}) unfolding Finite_def by auto
moreover
{ fix aa assume aa ∈ A
    with <A⊆N> obtain b where b ∈ N and aa ∈ b by auto
    with ss have B ∩ (fb) = b by auto
    with <aa∈b> have aa ∈ B ∩ (fb) by auto
    then have aa ∈ fb by auto
    with <b∈N> have aa ∈ {ft. t ∈ N} by auto
} then have A ⊆ {ft. t ∈ N} by auto ultimately
have ∃ R ∈ FinPow(M). A ⊆ ⋃ R by auto
} then show ∀ M ∈ Pow(T). A ⊆ ⋃ M −→ (∃ N ∈ FinPow(M). A ⊆ ⋃ 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** Kcompact_subspace_imp_Kcompact:
assumes A{is compact of cardinal}Q{in}(T{restricted to}B) A ⊆ B ({the axiom of} Q {choice holds})
s shows A{is compact of cardinal}Q{in}T
proof -
from assms(1) have a1: Card(Q) unfolding IsCompactOfCard_def RestrictedTo_def by auto
from assms(1) have a2: A ⊆ T unfolding IsCompactOfCard_def RestrictedTo_def by auto
{ fix M assume M ∈ Pow(T). A ⊆ ⋃ M
    let M = M{restricted to}B

990
from \(<M \in \text{Pow}(T)\) have \(M \in \text{Pow}(T\text{restricted to}B)\) unfolding RestrictedTo_def by auto
from \(<A \subseteq \bigcup M\) have \(A \subseteq \bigcup M\) unfolding RestrictedTo_def using assms(2) by auto

with assms \(<M \in \text{Pow}(T\text{restricted to}B)\) obtain \(N\) where \(N : N \in \text{Pow}(M)\)
\(A \subseteq \bigcup N\) \(N \setminus Q\) unfolding IsCompactOfCard_def by blast
from \(N(3)\) have \(N \setminus Q\) using lesspoll_imp_lepoll by auto moreover

\{ fix \(BB\) assume \(BB \in N\)
with \(<N \in \text{Pow}(N)\) have \(BB \in M\) unfolding FinPow_def by auto
then obtain \(S\) where \(S \in M\) and \(BB = B \cap S\) unfolding RestrictedTo_def by auto
then have \(S \in \{S \in M. B \cap S = BB\}\) by auto
then obtain \(\{S \in M. B \cap S = BB\}\) \(\neq 0\) by auto
\}
then have \(\forall BB \in N. ((\lambda \in N. \{S \in M. B \cap S = W\}) \text{BB}\) \(\neq 0\) by auto moreover
have \(N \subseteq Q \land (\forall t \in N. (\lambda \in N. \{S \in M. B \cap S = W\}) t \neq 0) \rightarrow (\exists f \in \Pi(N, \lambda t. (\lambda \in N. \{S \in M. B \cap S = W\}) t) \land (\forall t \in N. f t \in (\lambda \in N. \{S \in M. B \cap S = W\}) t)))\)

using assms(3) unfolding AxiomCardinalChoiceGen_def by blast
ultimately
obtain \(f\) where \(AA : f \in \Pi(N, \lambda t. (\lambda \in N. \{S \in M. B \cap S = W\}) t) \land (\forall t \in N. f t \in (\lambda \in N. \{S \in M. B \cap S = W\}) t))\)
with \(AA(2)\) have \(ss : \forall t \in N. f t \in \{S \in M. B \cap S = t\}\) using beta_if by auto
then have \(\{f t. t \in N\} \subseteq M\) by auto

\{ fix \(t\) assume \(t \in N\)
with \(ss\) have \(f t \in \{S \in M. B \cap S = t\}\) by auto
\}
with \(AA(1)\) have \(FF : f : N \rightarrow \{S \in M. B \cap S = N\}\) unfolding Pi_def Sigma_def using beta_if by auto moreover

\{ fix \(aa\) \(bb\) assume \(AAA : aa \in N\) \(bb \in N\) \(faa = fbb\)
from \(AAA(1)\) \(ss\) have \(B \cap (faa) = aa\) by auto
with \(AAA(3)\) have \(B \cap (fbb) = aa\) by auto
with \(ss\) \(AAA(2)\) have \(aa = bb\) by auto
\}
ultimately have \(f \in \text{inj}(N, \{S \in M. B \cap S = N\})\) unfolding inj_def by auto
then have \(f \in \text{bij}(N, \text{range}(f))\) unfolding inj_bij_range by auto
then have \(f \in \text{bij}(N, fN)\) unfolding range_image_domain FF by auto
then have \(f \in \text{bij}(N, \{f t. t \in N\})\) unfolding func_imagedef FF by auto
then have \(N \approx \{f t. t \in N\}\) unfolding eqpoll_def by auto
with \(<N < Q\) have \(\{f t. t \in N\} < Q\) unfolding eqpoll_sym eq_lesspoll_trans by blast moreover
with \(ss\) have \(\{f t. t \in N\} \in \text{Pow}(N)\) unfolding FinPow_def by auto moreover

\{ fix \(aa\) assume \(aa \in A\)
with \(<A \subseteq M\) obtain \(b\) where \(b \in N\) and \(aa \in B\) by auto

991
with ss have B ∩ (fb) = b by auto
with ⟨aa ∈ b⟩ have aa ∈ B ∩ (fb) by auto
then have aa ∈ fb by auto
with ⟨b ∈ N⟩ have aa ∈ ∪ {ft. t ∈ N} by auto
}
then have A ⊆ ∪ {ft. t ∈ N} by auto ultimately
have ∃ R ∈ Pow(M). A ⊆ ∪ R ∧ R ⊆ Q by auto
}
then show thesis using a1 a2 unfolding IsCompactOfCard_def by auto
qed

Every set, with the cofinite topology is compact.

lemma cofinite_compact:
shows X {is compact in} (CoFinite X) unfolding IsCompact_def
proof
show X ⊆ ∪ (CoFinite X) using union_cocardinal unfolding Cofinite_def
by auto
next
{
fix M
assume M ∈ Pow (CoFinite X) X ⊆ ∪ M
{
}
assume M ≠ 0 ∨ M ≠ {0}
then have M ∈ FinPow (M) unfolding FinPow_def by auto
with ⟨X ⊆ ∪ M⟩ have ∃ N ∈ FinPow (M). X ⊆ ∪ N by auto
}
moreover
{
assume M ≠ 0 ∨ M ≠ {0}
then obtain U where U ∈ M ≠ 0 by auto
with ⟨M ∈ Pow (CoFinite X)⟩ have U ∈ CoFinite X by auto
with ⟨U ≠ 0⟩ have U ⊆ X (X − U) ⊆ nat unfolding Cofinite_def CoCardinal_def
by auto
then have Finite (X − U) using lesspoll_nat_is_Finite by auto
then have (X − U) {is in the spectrum of} (λ T. (∪ T) {is compact in} T) using compact_spectrum
by auto
then have ( (∪ (CoFinite (X − U))) ≈ X − U) → ((∪ (CoFinite (X − U))) {is compact in} (CoFinite (X − U))) unfolding Spec_def
using InfCard_nat CoCar_is_topology unfolding Cofinite_def CoCardinal_def
by auto
then have com: (X − U) {is compact in} (CoFinite (X − U)) using union_cocardinal unfolding Cofinite_def by auto
have (X − U) ⊆ X − U by auto
then have (CoFinite X) {restricted to} (X − U) = (CoFinite (X − U)) using subspace_cocardinal unfolding Cofinite_def by auto
with com have (X − U) {is compact in} (CoFinite X) using compact_subspace_imp_compact[of X − U CoFinite XX − U] by auto
moreover have X − U ⊆ M using ⟨X ⊆ ∪ M⟩ by auto
moreover note ⟨M ∈ Pow (CoFinite X)⟩
ultimately have \( \exists N \in \text{FinPow}(M). X \subseteq \bigcup N \) unfolding IsCompact_def by auto
then obtain \( N \) where \( N \subseteq M \) Finite(\( N \)) \( X \subseteq \bigcup (N \cup \{U\}) \) by auto
with \( \langle U \in M \rangle \) have \( N \subseteq M \) Finite(\( N \cup \{U\} \)) \( X \subseteq \bigcup (N \cup \{U\}) \) by auto
ultimately
have \( \exists N \in \text{FinPow}(M). X \subseteq \bigcup N \) unfolding 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 cof:topology0(\( \text{CoFinite } X \)) and cof1:(\( \text{CoFinite } X \)\{\text{is a topology}\})
using CoCar_is_topology InfCard_nat Cofinite_def unfolding topology0_def by auto
{
fix \( x, B \) assume \( x \in \bigcup (\text{CoFinite } X) \) \( B \in (\text{CoFinite } X) \) \( x \in B \)
then have \( x \in \text{Interior}(B, \text{CoFinite } X) \) using topology0.Top_2_L3[OF cof]
by auto
moreover
from \( \langle B \in (\text{CoFinite } X) \rangle \) have \( B \subseteq X \) unfolding Cofinite_def CoCardinal_def 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 \( \exists c \in \text{Pow}(B). x \in \text{Interior}(c, \text{CoFinite } X) \land c \{\text{is compact in}\} (\text{CoFinite } X) \) by auto
}
then have \( (\forall x \in \bigcup (\text{CoFinite } X). \forall b \in (\text{CoFinite } X). x \in b \rightarrow (\exists c \in \text{Pow}(b). x \in \text{Interior}(c, \text{CoFinite } X) \land c \{\text{is compact in}\} (\text{CoFinite } X)) \) 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 $T$ is locally-compact.
shows $\forall x \in \bigcup T. \exists A \in \Pow(\bigcup T). A$ (is compact in) $T \land x \in \text{int}(A)$

proof-
{
  fix $x$ assume $x \in \bigcup T$ moreover
  then have $\bigcup T \neq 0$ by auto
  have $\bigcup T \in T$ using union_open topSpaceAssum by auto
  ultimately have $\exists c \in \Pow(\bigcup T). x \in \text{int}(c) \land c$ (is compact in) $T$
  using assms
  IsLocally_def topSpaceAssum unfolding IsLocallyComp_def by auto
  then have $\exists c \in \Pow(\bigcup T). c$ (is compact in) $T \land x \in \text{int}(c)$ 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 $\forall x \in \bigcup T. \exists A \in \Pow(\bigcup T). x \in \text{int}(A) \land A$ (is compact in) $T$ $T$ (is T$_2$)
  shows $T$ is locally-compact
proof-
{
  fix $x$ assume $x \in \bigcup T$
  with assms(1) obtain $A$ where $A \in \Pow(\bigcup T) \land x \in \text{int}(A)$ and $A$ (is compact in) $T$
  by blast
  then have $A \subseteq \bigcup T$ using in_t2_compact_is_cl assms(2)
  by auto
  then have $A \subseteq \bigcup T$ unfolding IsClosed_def by auto
  { fix $U$ assume $U \in T \land x \in U$
    let $V = \text{int}(A \cap U)$
    from $x \in U \land x \in \text{int}(A)$ have $x \in U \cap \text{int}(A)$ by auto
    moreover from $U \in T$ have $U \cap \text{int}(A) \subseteq T$ using Top_2_L2 topSpaceAssum
    unfolding IsATopology_def by auto
    moreover have $U \cap \text{int}(A) \subseteq A \cap U$ using Top_2_L1 by auto
    ultimately have $x \in V$ using Top_2_L5 by blast
    have $V \subseteq A$ using Top_2_L1 by auto
    then have $c(V) \subseteq A$ using Ac1 Top_2_L13 by auto
    then have $A \cap c(V) = c(V)$ by auto moreover
    have $c(V) \subseteq \text{int}(A)$ using $c(V)$ (is closed in) $T$ using cl_is_closed $\forall A \in T$ by auto
    ultimately have $\text{comp}:c(V)$ (is compact in) $T$ using Acomp compact_closed[of $\text{AnatT}(V)$] Compact_is_card_nat
    by auto
    { then have $c(V)$ (is compact in) $(T \text{ restricted to } c(V))$ using compact_imp_compact_sub $c(V)$ nat $T$] Compact_is_card_nat
      by auto moreover
      have $\bigcup (T \text{ restricted to } c(V)) = c(V)$ unfolding RestrictedTo_def
      994
using clcl unfolding IsClosed_def by auto moreover
ultimately have \( (\bigcup (T \{\text{restricted to}\} \text{cl}(V))) \{\text{is compact in}\} (T \{\text{restricted to}\} \text{cl}(V)) \) by auto
\}
then have \( (\bigcup (T \{\text{restricted to}\} \text{cl}(V))) \{\text{is compact in}\} (T \{\text{restricted to}\} \text{cl}(V)) \) by auto
have \( (T \{\text{restricted to}\} \text{cl}(V)) \{\text{is } T_2\} \) using assms(2) T2_here clcl
unfolding IsClosed_def by auto
ultimately have \( (T \{\text{restricted to}\} \text{cl}(V)) \{\text{is } T_4\} \) using topology0.T2_compact_is_normal
unfolding topology0_def
using Top_1_L4 unfolding isT4_def using T2_is_T1
then have clvreg: \( (T \{\text{restricted to}\} \text{cl}(V)) \{\text{is regular}\} \) using topology0.T4_is_T3
unfolding topology0_def isT3_def using Top_1_L4
ultimately have \( \text{cl}(V) \{\text{is regular}\} \) using topology0.regular_imp_exist_clos_neig
unfolding topology0_def
then have \( V \subseteq \text{cl}(V) \) using topology0.cl_contains_set unfolding topology0_def using Top_1_L4 by auto
moreover from \( V \subseteq A \) \( A \subseteq \bigcup T \) have \( V \subseteq \bigcup T \) by auto
then have \( V \subseteq \text{cl}(V) \) \( \text{cl}(V) \subseteq T \) using \( V \subseteq \text{cl}(V) \) Top_3_L11(1) by auto
then have \( (T \{\text{restricted to}\} \text{cl}(V)) \{\text{restricted to}\} V = (T \{\text{restricted to}\} \text{cl}(V)) \) using subpace_of_subspace by auto
ultimately have \( \text{cl}(V) \{\text{is compact in}\} (T \{\text{restricted to}\} \text{cl}(V)) \) by auto
then obtain \( U \) where \( W \subseteq V \) using topology0.regular_imp_exist_clos_neig
unfolding Top_1_L4 clvreg
by blast
from clcont W have \( W \subseteq V \) using topology0.cl_contains_set unfolding Top_1_L4 by auto
moreover from \( V \subseteq A \) \( A \subseteq \bigcup T \) have \( V \subseteq \bigcup T \) by auto
then have \( V \subseteq \text{cl}(V) \) \( \text{cl}(V) \subseteq T \) using \( V \subseteq \text{cl}(V) \) Top_3_L11(1) by auto
then have \( (T \{\text{restricted to}\} \text{cl}(V)) \{\text{restricted to}\} V = (T \{\text{restricted to}\} \text{cl}(V)) \) using subpace_of_subspace by auto
ultimately have \( \text{cl}(V) \{\text{is compact in}\} (T \{\text{restricted to}\} \text{cl}(V)) \) by auto
then have \( W \subseteq T \) using Top_2_L2 topSpaceAssum unfolding IsATopology_def by auto
moreover from \( V \subseteq A \) \( A \subseteq \bigcup T \) have \( V \subseteq \bigcup T \) by auto
then have \( V \subseteq \text{cl}(V) \) \( \text{cl}(V) \subseteq T \) using \( V \subseteq \text{cl}(V) \) Top_3_L11(1) by auto
then have \( (T \{\text{restricted to}\} \text{cl}(V)) \{\text{restricted to}\} V = (T \{\text{restricted to}\} \text{cl}(V)) \) using subpace_of_subspace by auto
ultimately have \( W \subseteq T \) by auto
then obtain \( U \) where \( W \subseteq T \) using topology0.cl_is_closed
ultimately have \( A_1 : x \in \text{int}(\text{cl}(V)) \) using Top_2_L6 by auto
from clcont W have \( A_2 : \text{cl}(V) \subseteq U \) using Top_2_L1 by auto
have clcl: \( \text{cl}(V) \{\text{is closed in}\} (T \{\text{restricted to}\} \text{cl}(V)) \) using topology0.cl_is_closed
by auto
from comp have \( \text{cl}(V) \{\text{is compact in}\} (T \{\text{restricted to}\} \text{cl}(V)) \) using compact_imp_compact_subspace[of cl(V) natT] Compact_is_card_nat
by auto

995
with clwcl have \((\text{cl}(V) \cap \text{Closure}(W,(T\{\text{restricted to} \text{cl}(V)))))\{\text{is compact in}\}(T\{\text{restricted to} \text{cl}(V))\)

using compact_closed Compact_is_card_nat by auto

moreover from clcont have cont:\((\text{Closure}(W,(T\{\text{restricted to} \text{cl}(V)))))\subseteq \text{cl}(V)\)

using cl_contains_set \(<V\subseteq A>\rightarrow A\subseteq \bigcup T>\)

by blast

then have \((\text{cl}(V) \cap \text{Closure}(W,(T\{\text{restricted to} \text{cl}(V)))))\subseteq \text{Closure}(W,(T\{\text{restricted to} \text{cl}(V)))))\)

by auto

ultimately have \((\text{cl}(V) \cap \text{Closure}(W,(T\{\text{restricted to} \text{cl}(V)))))\{\text{is compact in}\}(T\{\text{restricted to} \text{cl}(V))\)

by auto

then have \((\text{Closure}(W,(T\{\text{restricted to} \text{cl}(V)))))\{\text{is compact in}\}(T\{\text{restricted to} \text{cl}(V))\)

using compact_subspace_imp_compact[of \(\text{Closure}(W,(T\{\text{restricted to} \text{cl}(V)))))\]

cont by auto

with \(A1 \ A2\) have \(\exists c\in \text{Pow}(U). x\in \text{int}(c) \land c\{\text{is compact in}\}T\) by auto

} then have \(\forall U\in T. x\in U \rightarrow (\exists c\in \text{Pow}(U). x\in \text{int}(c) \land c\{\text{is compact in}\}T)\)

by auto

} then show thesis unfolding IsLocally_def[OF topSpaceAssum] IsLocallyComp_def by auto

qed

72.4 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

\(\text{OPCompactification}\)\(\{\text{one-point compactification of}_90\)\)

where \(\{\text{one-point compactification of}_90\)\)\(T\equiv T \cup \{\bigcup T\} \cup ((\bigcup T) - K). K\in\{B\in \text{Pow}(\bigcup T). B\{\text{is compact in}\}T \land B\{\text{is closed in}\}T}\)\)

Firstly, we check that what we defined is indeed a topology.

theorem (in topology0) op_comp_is_top:

shows \((\text{one-point compactification of}_90)\{\text{is a topology}\) unfolding IsATopology_def proof (safe)

fix \(M\) assume \(M\subseteq \{\text{one-point compactification of}_90\)\)\(T\equiv T \cup \{\bigcup T\} \cup ((\bigcup T) - K). K\in\{B\in \text{Pow}(\bigcup T). B\{\text{is compact in}\}T \land B\{\text{is closed in}\}T}\)\)

\(\forall B\in \text{Pow}(\bigcup T). B\{\text{is compact in}\}T \land B\{\text{is closed in}\}T)\}

unfolding OPCompactification_def by auto

let \(MT\{\{A\in M. A\in T\)

have \(MT\subseteq T\) by auto

then have \(\text{cl}(\bigcup MT)\)\(\in T\) using topSpaceAssum unfolding IsATopology_def by auto

let \(MK\{\{A\in M. A\not\in T\)

have \(\bigcup MK \subseteq \bigcup MT\) by auto

from disj have \(MK\subseteq \{A\in M. A\in \{\bigcup T\} \cup ((\bigcup T) - K). K\in\{B\in \text{Pow}(\bigcup T). B\{\text{is compact in}\}T \land B\{\text{is closed in}\}T)\})\)

by auto

moreover have \(N\in \bigcup T\}\)\(\notin (\bigcup T)\) using mem_not_refl by auto


fix B assume B∈\{∪T,(∪T)−K\}. K∈B∈Pow(∪T). B\{is compact in\}T ∧ B\{is closed in\}T
  then obtain K where K∈Pow(∪T) B=∪T,(∪T)−K by auto
  with \(N\) have \(∪T∈B\) by auto
  with \(N\) have \(B\not∈T\) by auto
  with \(<B∈M>\) have \(B∈MK\) by auto
}\}
then have \{A∈M. A∈\{∪T,(∪T)−K\}. K∈B∈Pow(∪T). B\{is compact in\}T ∧ B\{is closed in\}T\}⊂MK by auto
ultimately have MK_def:MK={A∈M. A∈\{∪T,(∪T)−K\}. K∈B∈Pow(∪T). B\{is compact in\}T ∧ B\{is closed in\}T} by auto
let KK={K∈Pow(∪T). {∪T,(∪T)−K}\{is compact in\}T} using Compact_is_card_nat
K1{is compact in}T KK{is compact in}T KK{is closed in}T using auto
ultimately have KK{is compact in}T K1{is compact in}T using MK_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 closed in}T K1{is compact in}T using MK_def by auto
  with \(<A∈MK>\) have \(K1\subseteq K\) by auto
from \(<A∈MK>\) A∈\{∪T,(∪T)−K1\> K1∈Pow(∪T)> have KK≠0 by blast
\} fix K assume K∈KK
then have \(∪T,(∪T)−K\)∈MK K⊆∪T by auto
then obtain KK where A=∪T,(∪T)−K=∪T,(∪T)−KK KK⊆∪T
K1{is compact in}T KK{is closed in}T using MK_def by auto
note A(1) moreover
have \(∪T,(∪T)−K\subseteq∪T,(∪T)−K\) (∪T)−KK⊆∪T,(∪T)−KK by auto
ultimately have \(∪T,(∪T)−K\subseteq∪T,(∪T)−KK\) (∪T)−KK⊆∪T,(∪T)−KK by auto
by auto moreover
from \(N\) have \(∪T∉(∪T)−K\) \(∪T∉(∪T)−KK\) by auto ultimately
have \(∪T−K\subseteq∪T−K\) (∪T)−KK⊆∪T−KK by auto
then have \(∪T−K\)=(∪T)−KK by auto moreover
from \(<K∪∪T−K\> have K=(∪T)−(∪T−K) by auto ultimately
have K=(∪T)−(∪T−KK) by auto
with \(<KK∪∪T−K\> have K=KK by auto
with A(4) have K{is closed in}T by auto
}\}
then have \(∀K∈KK\). K{is closed in}T by auto
with \(<KK≠0>\) have \(∪KK\)\{is closed in\}T using Top_3_L4 by auto
with \(<K1\{is compact in\}T>\) have \(K1\cap∪KK\)\{is compact in\}T using Compact_is_card_nat
compact_closed[of K1\cap∪KK] by auto moreover

997
from \( \bigcap KK \subseteq K1 \) have \( K1 \cap (\bigcap KK) = (\bigcap KK) \) by auto ultimately
have \( (\bigcap KK) \{ \text{is compact in} \} T \) by auto
with \( \bigcap KK \{ \text{is closed in} \} T \) <\( \bigcap KK \subseteq K1 \) <\( K1 \in \text{Pow}(T) \) have \( \{ \{ U \} \cup (\bigcup (U \setminus (\bigcap KK))) \} \) in \{ \text{one-point compactification of} \} T
unfolding \text{OPCompactification_def} \text{ by blast}
have \( t \in \{ MK = \bigcup \{ A \in M. A \in \{ \{ U \} \cup (\bigcup (U \setminus K)\} K \} \in \{ \text{Pow}(U) \} B \{ \text{is compact in} \} T \} \wedge B \{ \text{is closed in} \} T \} \)
using \text{MK_def} \text{ by auto}

\{  
fix \( x \) assume \( x \in \cup MK \) with \( t \) have \( x \in \{ A \in M. A \in \{ \{ U \} \cup (\bigcup (U \setminus K)\} K \} \in \{ \text{Pow}(U) \} B \{ \text{is compact in} \} T \} \)
then have \( \exists AA \in \{ A \in M. A \in \{ \{ U \} \cup (\bigcup (U \setminus K)\} K \} \in \{ \text{Pow}(U) \} B \{ \text{is compact in} \} T \} \)
\( x \in AA \)
using \text{Union_iff} \text{ by auto}
then obtain \( AA \) where \( \text{AAp}(A) = \{ A \in \{ A \in M. A \in \{ \{ U \} \cup (\bigcup (U \setminus K)\} K \} \in \{ \text{Pow}(U) \} B \{ \text{is compact in} \} T \} \)
B \{ \text{is compact in} \} T \wedge B \{ \text{is closed in} \} T \}
\( x \in AA \) by auto
then obtain \( K2 \) where \( AA = \{ \{ U \} \cup (\bigcup (U \setminus K2)\} K2 \in \text{Pow}(U) T \} \)
\( B \{ \text{is compact in} \} T \wedge B \{ \text{is closed in} \} T \) by auto
with \( x \in AA \) have \( x \in \bigcup T \)
from \( x \in \bigcup T \) <\( x \in \bigcup T \wedge x \notin K2 \) by auto
by auto
then have \( \bigcap (K \subseteq K2) \) by auto
with \( \bigcap (K \subseteq K2) \) \( x \in \bigcup T \wedge x \notin \bigcap (K \subseteq K2) \) have \( x \in \bigcup TV (x \in \bigcup T \wedge x \notin \bigcap (K \subseteq K2) \) by auto

\{  
fix \( x \) assume \( x \in \{ \{ U \} \cup (\bigcup (U \setminus (\bigcap KK)\} K \) \) then have \( x \in \bigcup TV (x \in \bigcup T \wedge x \notin (\bigcap KK) \) by auto
with \( x \in (\bigcap KK) \) obtain \( K2 \) where \( K2 \in \text{MK} \) \( x \in \bigcup TV (x \in \bigcup T \wedge x \notin K2) \) by auto
then have \( (\bigcup (U \setminus K2) \subseteq MK \) by auto
with \( x \in (\bigcup TV (x \in \bigcup T \wedge x \notin K2) \) have \( x \in \bigcup MK \) by auto

\{  
then have \( (\bigcup (U \setminus (\bigcap KK)\} K \subseteq MK \) by (safe, auto)
ultimately have \( \bigcup MK = \{ \{ U \} \cup (\bigcup (U \setminus (\bigcap KK)\} K \) \subseteq MK \) by blast
from \( \bigcup MT \subseteq T \) have \( T \setminus (U \setminus MT) = \bigcup MT \) by auto
with \( \bigcup MT \subseteq T \) have \( (U \setminus \bigcup MT) \} \) is closed in \( T \) unfolding \text{IsClosed_def} \text{ by auto}
have \( (\bigcup (U \setminus (\bigcap KK)\} K \subseteq U \setminus MT \) \subseteq U \subseteq U \setminus MT \) \subseteq (\bigcup (U \setminus (\bigcap KK)\} K \subseteq MK \) by auto
then have \( (\bigcup MK = \{ \{ U \} \cup (\bigcup (U \setminus (\bigcap KK)\} K \) \subseteq MK \) by auto
with \( \bigcup MK = \{ \{ U \} \cup (\bigcup (U \setminus (\bigcap KK)\} K \) \) have \( \bigcup MK \subseteq \bigcup MT = \{ \{ U \} \cup (\bigcup (U \setminus (\bigcap KK)\} K \) \subseteq MK \) by auto
with \( \bigcup MK \subseteq \bigcup MT \) have \( \text{unM} = \bigcup MK = \{ \{ U \} \cup (\bigcup (U \setminus (\bigcap KK)\} K \) \subseteq MK \) by auto

998
have \( ((\bigcap KK) \cap (\bigcup T - \bigcup MT)) \) {is closed in} \( T \) using \( <(\bigcap KK) \{\text{is closed in}\} T > -(\bigcup T - (\bigcup MT)) \) {is closed in} \( T \).

\[ \text{Top_3_L5 by auto} \]
moreover note \( <(\bigcup T - (\bigcup MT)) \{\text{is closed in}\} T > -(\bigcap KK) \{\text{is compact in}\} T \)
then have \( ((\bigcap KK) \cap (\bigcup T - \bigcup MT)) \) {is compact of cardinal} \( \text{nat} \{\text{in}\} T \)
by auto
ultimately have \( \{\bigcup T\} \cup (\bigcup T - ((\bigcap KK) \cap (\bigcup T - \bigcup MT))) \) \( \in \{\text{one-point compactification of}\} T \)
unfolding \( \text{OPCompactification_def} \) \( \text{IsClosed_def} \) by auto
with \( \{\text{unM}\} \)
have \( \bigcup \{\text{one-point compactification of}\} T \) by auto

next
fix \( U \) \( V \)
assume \( U \in \{\text{one-point compactification of}\} T \) and \( V \in \{\text{one-point compactification of}\} T \)
then have \( A: U \cap V \in \{\text{one-point compactification of}\} T \)
unfolding \( \text{OPCompactification_def} \) by auto
have \( N: \bigcup T \notin (\bigcup T) \) using \( \text{mem_not_refl} \) by auto
(
assume \( U \in \bigcup T \)
then have \( U \cap V \in \{\text{one-point compactification of}\} T \) unfolding \( \text{OPCompactification_def} \) by auto
)
moreover
(
assume \( U \notin \bigcup T \)
then obtain \( KV \) where \( V:KV \{\text{is closed in}\} T \) \( KV \{\text{is compact in}\} T \)
using \( A(2) \) by auto
with \( N \) \( U \in \bigcup T \) have \( \bigcup T \notin U \) by auto
then have \( \bigcup T \notin V \) by auto
then have \( \bigcup T \notin (\bigcup T - KV) \) using \( V(3) \) by auto
moreover have \( \bigcup T - KV \in T \) using \( V(1) \) unfolding \( \text{IsClosed_def} \) by auto
with \( \bigcup T \) have \( \bigcup (\bigcup T - KV) \in T \) using \( \text{topSpaceAssum} \) unfolding \( \text{IsATopology_def} \) by auto
with \( \bigcup V = \bigcup (\bigcup T - KV) \) have \( U \cap V \in \{\text{one-point compactification of}\} T \) unfolding \( \text{OPCompactification_def} \) by auto
)
moreover

assume $U \in TV \in T$
then obtain $KV$ where $V:KV$ is closed in $TV$ using $A(1)$ by auto
with $N < V \in T$ have $T \not\in V$ by auto
then have $\bigcup T \not\in V$ by auto
then have $U \cap V = (\bigcup T-KV) \cap V$ using $V(3)$ by auto
moreover have $\bigcup T-KV \in T$ using $\text{topSpaceAssum}$ unfolding $\text{IsATopology_def}$ by auto
with $< V \in T$ have $\bigcup T / \in U$ by auto
then have $\bigcup T / \in U \cap V$ by auto
then have $U \cap V = (\bigcup T-KV) \cap V$ using $V(3)$ by auto
moreover have $\bigcup T-KV \in T$ using $V(1)$ unfolding $\text{IsClosed_def}$ by auto
then have $\bigcup T-KV \in T$ unfolding $\text{IsATopology_def}$ by auto
moreover have $\bigcup T-KV \in T$ unfolding $\text{IsClosed_def}$ by auto
moreover have $\bigcup T - (\bigcup T-(\bigcup T-KV) \cap (\bigcup T-KU)) = KV \cup KU$ unfolding $\text{IsClosed_def}$ by auto
with $V(2)$ have $\bigcup T-KV \cap (\bigcup T-KU) \in T$ unfolding $\text{IsClosed_def}$ by auto
ultimately have $U \cap V \in \{\text{one-point compactification of} T\}$ unfolding $\text{OPCompactification_def}$ 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-
unfolding $\text{OPCompactification_def}$ using $\text{topSpaceAssum}$ unfolding $\text{IsATopology_def}$ by auto

1000
have $T \subseteq \{\text{one-point compactification of } T\} \cup T$ unfolding OPCompactification_def RestrictedTo_def by auto
moreover
{ fix $A$ assume $A \in \{\text{one-point compactification of } T\} \cup T$
then obtain $R$ where $R \in \{\text{one-point compactification of } T\}$ $A = \cup T \cap R$
unfolding RestrictedTo_def by auto
then obtain $K$ where $K, R \in T$ $A = \cup T \cap K$ $K$ is closed in $T$
unfolding OPCompactification_def by auto
with $\langle A = \cup T \cap R, A = \cup T \cap K, K \text{ is closed in } T \rangle$ using mem_not_refl unfolding IsClosed_def by auto
with $K$ have $A \in T$ unfolding IsClosed_def by auto
}
ultimately
show $\{\text{one-point compactification of } T\} \cup T = T$ by auto
qed

We added only one new point to the space.

**Lemma** (in topology0) op_compact_total:
shows $\{\cup \text{one-point compactification of } T\} = \{\cup T\} \cup \{\cup T\}$

**Proof**

have $0$ is compact in $T$ unfolding IsCompact_def FinPow_def by auto
moreover note Top_3_L2 ultimately have $TT: 0 \in \{A \in \text{Pow}(\cup T). A \text{ is compact in } T \land A \text{ is closed in } T\}$ by auto
have $\cup \{\text{one-point compactification of } T\} = \{\cup T\} \cup \{\cup T\}$ unfolding OPCompactification_def by blast
also have $\ldots = \{\cup T\} \cup \{\cup T\}$ unfolding OPCompactification_def by auto
ultimately show $\cup \{\text{one-point compactification of } T\} = \{\cup T\} \cup \{\cup T\}$ by auto
qed

The one point compactification, gives indeed a compact topological space.

**Theorem** (in topology0) compact_op:
shows $\{\cup \text{one-point compactification of } T\} \cup \{\cup T\} \subseteq \{\cup T\} \cup \{\cup T\}$

**Proof** (safe)

have $0$ is compact in $T$ unfolding IsCompact_def FinPow_def by auto
moreover note Top_3_L2 ultimately have $0 \in \{A \in \text{Pow}(\cup T). A \text{ is compact in } T \land A \text{ is closed in } T\}$ by auto
then have $\cup \{\cup T\} \cup \{\cup T\} \subseteq \{\cup T\} \cup \{\cup T\}$ unfolding OPCompactification_def by auto
then show $\cup T \subseteq \{\cup \text{one-point compactification of } T\} \cup \{\cup T\}$ by auto
next
fix $x$ $B$ assume $x \in BB \in T$
then show $x \in \{\cup \text{one-point compactification of } T\}$ using open_subspace by auto
next

1001
The one point compactification is Hausdorff iff the original space is also Hausdorff and locally compact.

**Lemma (in topology0) op_compact_T2_1:**

assumes \( \{\text{one-point compactification of}\}T \{\text{is } T_2} \)

shows \(T\{\text{is } T_2} \)

using T2_here[OF assms, of \( \bigcup T \)] open_subspace by auto

**Lemma (in topology0) op_compact_T2_2:**

assumes \( \{\text{one-point compactification of}\}T \{\text{is } T_2} \)

shows \(T\{\text{is locally-compact}} \)

**proof**:

\[
\begin{align*}
\text{fix } M & \text{ assume } A:M \subseteq \{\text{one-point compactification of}\}T \{\bigcup T \} \cup T \subseteq \bigcup M \\
\text{then obtain } R & \text{ where } R \subseteq \bigcup M \text{ by auto} \\
\text{have } UT & \not\subseteq T \text{ using mem_not_refl by auto} \\
\text{with } R \subseteq T & \text{ obtain } K \text{ where } K:R=\{\bigcup T \} \cup (\bigcup T-K) \text{ is compact in } T \subseteq \bigcup M \text{ is closed in } T \\
\text{unfolding } \text{OPCompactification_def by auto} \\
\text{from } K(1,2) & \text{ have } B:\bigcup T \subseteq R \text{ using mem_not_refl by auto} \\
\text{with } A(1) & \text{ obtain } K:R=\{\bigcup T \} \cup (\bigcup T-K) \text{ is compact in } T \subseteq \bigcup M \\
\text{ultimately have } \exists N \in \text{FinPow}(M). K \subseteq \bigcup N \text{ unfolding } \text{IsCompact_def by auto} \\
\text{then obtain } N & \text{ where } N \subseteq \bigcup M \text{ by auto} \\
\text{with } B & \text{ show } \exists N \in \text{FinPow}(M). \bigcup T \cup \bigcup T \subseteq \bigcup N \text{ by autoqed} \\
\end{align*}
\]
folding OPCompactification_def
using op_compact_total by auto
with \langle U \cap V = 0 \rangle \ k have U \subseteq K \subseteq T unfolding IsClosed_def by auto
then have \langle \bigcup T \cap U = U \rangle by auto moreover
from UV(1) have \langle \bigcup T \cap U = \text{(one-point compactification of) T} \{\text{restricted to} \bigcup T \rangle
unfolding RestrictedTo_def by auto
ultimately have \bigcup T using open_subspace(2) by auto
with \langle \bigcup T \rangle have x \in \text{int}(K) using Top_2_L6 by auto
with \langle K \subseteq T \rangle \langle \{\text{is compact in} \rangle T \rangle \rangle have \exists A \in Pow(\bigcup T). x \in \text{int}(A) \land A\{\text{is compact in} \rangle T \rangle by auto
then have \forall x \in \bigcup T. \exists A \in Pow(\bigcup T). x \in \text{int}(A) \land A\{\text{is compact in} \rangle T \rangle by auto
then show thesis using op_compact_T2_1[OF assms] exist_compact_neig_T2_imp_locally_compact by auto
qed

lemma (in topology0) op_compact_T2_3:
assumes T\{is locally-compact} T\{is T_2\}
shows \{\text{(one-point compactification of) T} \{is T_2\}
proof-

{ fix x y assume x \neq y \in \bigcup \{\text{(one-point compactification of) T}\} y \in \bigcup \{\text{(one-point compactification of) T}\} then have S: x \in \bigcup T \cup \bigcup y \in \bigcup T \cup \bigcup T \cup \bigcup T using op_compact_total by auto
  
  assume x \in \bigcup T y \in \bigcup T
with \langle x \neq y \rangle have \exists U \in T. \exists V \in T. x \in U \wedge y \in V \wedge U \cap V = 0 using assms(2) unfolding isT2_def by auto
then have \exists U \in \{\text{(one-point compactification of) T}\}. \exists V \in \{\text{(one-point compactification of) T}\}. x \in U \wedge y \in V \wedge U \cap V = 0
unfolding OPCompactification_def by auto
}

moreover

{ assume x \in \bigcup T y \in \bigcup T
with S have x \in \bigcup T y \in \bigcup T by auto
with \langle x \neq y \rangle have (x \in \bigcup T \wedge y \neq \bigcup T) \vee (y \in \bigcup T \wedge x \neq \bigcup T) by auto
with S have (x \in \bigcup T \wedge y \in \bigcup T) \vee (y \in \bigcup T \wedge x \in \bigcup T) by auto
then obtain Ky Kx where x \in \bigcup T \wedge Ky\{is compact in} T \wedge y \in \text{int} (Ky) \wedge (y \in \bigcup T \wedge Kx\{is compact in} T \wedge x \in \text{int} (Kx))
  using assms(1) locally_compact_exist_compact_neig by blast
then have (x \in \bigcup T \wedge Ky\{is compact in} T \wedge Ky\{is closed in} T \wedge y \in \text{int} (Ky) \wedge (y \in \bigcup T \wedge Kx\{is compact in} T \wedge Kx\{is closed in} T \wedge x \in \text{int} (Kx))
  using in_t2compact_is_cl assms(2) by auto
then have (x \in \bigcup T \wedge (\bigcup T \wedge Ky\{is compact in} T \wedge Ky\{is closed in} T \wedge y \in \text{int} (Ky) \wedge (y \in \bigcup T \wedge Kx\{is compact in} T \wedge Kx\{is closed in} T))
}

1003
by auto moreover
{
  fix K
  assume A:K\{is closed in}\TK\{is compact in\}T
  then have K\subseteq\bigcup T unfolding IsClosed_def by auto
  moreover have \bigcup T\subseteq\bigcup T using mem_not_refl by auto
  ultimately have \{{\bigcup T}\cup(\bigcup T-K)\}\cap K=0 by auto
  then have \{{\bigcup T}\cup(\bigcup T-K)\}\cap \text{int}(K)=0 using Top_2_L1 by auto more-
over
  from A have \{{\bigcup T}\cup(\bigcup T-K)\}\subset\{{\one-point compactification of}T\}
  unfolding OPCompactification_def
  IsClosed_def by auto moreover
  have \text{int}(K)\in\{{\one-point compactification of}T\} using Top_2_L2
  unfolding OPCompactification_def
  by auto ultimately
  have \text{int}(K)\in\{{\one-point compactification of}T\}\land \{{\bigcup T}\cup(\bigcup T-K)\}\cap \text{int}(K)=0
  by auto
}

ultimately have \{{\bigcup T}\cup(\bigcup T-K)\}\subset\{{\one-point compactification of}T\}\land \text{int}(Ky)\in\{{\one-point compactification of}T\}\land \exists x \in \{{\bigcup T}\cup(\bigcup T-K)\}\land y \in \text{int}(Ky) \land \{{\bigcup T}\cup(\bigcup T-K)\}\cap \text{int}(Ky)=0 \lor

\{{\bigcup T}\cup(\bigcup T-Kx)\}\subset\{{\one-point compactification of}T\}\land \text{int}(Kx)\in\{{\one-point compactification of}T\}\land \exists y \in \{{\bigcup T}\cup(\bigcup T-Kx)\}\land x \in \text{int}(Kx) \land \{{\bigcup T}\cup(\bigcup T-Kx)\}\cap \text{int}(Kx)=0
by auto
moreover
{
  assume \{{\bigcup T}\cup(\bigcup T-Ky)\}\subset\{{\one-point compactification of}T\}\land \text{int}(Ky)\in\{{\one-point compactification of}T\}\land \exists x \in \{{\bigcup T}\cup(\bigcup T-Ky)\}\land y \in \text{int}(Ky) \land \{{\bigcup T}\cup(\bigcup T-Ky)\}\cap \text{int}(Ky)=0
  then have \exists U\in\{{\one-point compactification of}T\}. \exists V\in\{{\one-point compactification of}T\}. x\in\bigcup U \land y\in\bigcup V \land \emptyset=0 using exI[Of exI[of \_ \ \text{int}(Ky)], of \ \ \ \bigcup U. U\in\{{\one-point compactification of}T\}\land \exists V\in\{{\one-point compactification of}T\}\land x\in\bigcup U \land y\in\bigcup V \land \emptyset=0 by auto
}

moreover
{
  assume \{{\bigcup T}\cup(\bigcup T-Kx)\}\subset\{{\one-point compactification of}T\}\land \text{int}(Kx)\in\{{\one-point compactification of}T\}\land \exists y \in \{{\bigcup T}\cup(\bigcup T-Kx)\}\land x \in \text{int}(Kx) \land \{{\bigcup T}\cup(\bigcup T-Kx)\}\cap \text{int}(Kx)=0
  then have \exists U\in\{{\one-point compactification of}T\}. \exists V\in\{{\one-point compactification of}T\}. x\in\bigcup U \land y\in\bigcup V \land \emptyset=0 using exI[Of exI[of \_ \ \{{\bigcup T}\cup(\bigcup T-Kx)\}], of \ \ \ \bigcup U. U\in\{{\one-point compactification of}T\}\land \exists V\in\{{\one-point compactification of}T\}\land x\in\bigcup U \land y\in\bigcup V \land \emptyset=0 \text{int}(Kx) ]
by blast
}

ultimately have \exists U\in\{{\one-point compactification of}T\}. \exists V\in\{{\one-point compactification of}T\}. x\in\bigcup U \land y\in\bigcup V \land \emptyset=0 by auto

ultimately have \exists U\in\{{\one-point compactification of}T\}. \exists V\in\{{\one-point compactification of}T\}. x\in\bigcup U \land y\in\bigcup V \land \emptyset=0 by auto

1004
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{is locally-compact} T{is T2}
  shows T{is regular}
proof-
  from assms have ( {one-point compactification of}T) {is T2} using op_compact_T2_3
  by auto
  then have ( {one-point compactification of}T) {is T4} 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 ( {one-point compactification of}T) {is T3} using topology0.T4_is_T3
  op_comp_is_top unfolding topology0_def
  by auto
  then have ( {one-point compactification of}T) {is regular} using isT3_def
  by auto moreover
  have \bigcup T \subseteq \bigcup ( {one-point compactification of}T) using op_compact_total
  by auto
  ultimately have ( {one-point compactification of}T){restricted to}\bigcup T
  {is regular} using regular_here by auto
  then show T{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} \equiv T{is locally} (\lambda B \in T. B{is closed in}T)

lemma (in topology0) regular_locally_closed:
  shows T{is regular} \iff (T{is locally-closed})
proof
  assume T{is regular}
  then have a:\forall x \in \bigcup T. \forall U \in T. (x\in U) \implies (\exists V \in T. x \in V \land cl(V) \subseteq U) using regular_imp_exist_clos_neig by auto
  \{ fix x b assume x\in\bigcup T \in T x \in b
    with a obtain V where V \in T \in V cl(V) \subseteq b by blast
    note <cl(V) \subseteq b> moreover
    from <V \in T> have V \subseteq \bigcup T by auto
  \}

1005
then have $V \subseteq \text{cl}(V)$ using $\text{cl}_{\text{contains_set}}$ by auto
with $<x \in V, x \in V>$ have $x \in \text{int}(\text{cl}(V))$ using $\text{Top}_2.L6$ by auto moreover
from $<V \subseteq \bigcup T>$ have $\text{cl}(V)$ is closed in $T$ using $\text{cl}_{\text{is_closed}}$ by auto
ultimately have $x \in \text{int}(\text{cl}(V)) \subseteq \text{bcl}(V)$ is closed in $T$ by auto
then have $\exists K \in \text{Pow}(b). x \in \text{int}(K)$ is closed in $T$ by auto

then show $T$ is locally-closed unfolding $\text{IsLocally_def}[\text{OF topSpaceAssum}]$

next assume $T$ is locally-closed
then have $\forall x \in \bigcup T. \forall b \in T. x \in b \rightarrow (\exists K \in \text{Pow}(b). x \in \text{int}(K)$ is closed in $T)$ unfolding $\text{IsLocally_def}[\text{OF topSpaceAssum}]$

IsLocallyClosed_def by auto

fix $x$ $b$ assume $x \in \bigcup T b \in T x \in b$
with a obtain $K$ where $K : K \subseteq x \in \text{int}(K)$ is closed in $T$ by blast
have $\text{int}(K) \subseteq b$ using $\text{Top}_2.L1$ by auto
moreover have $\text{int}(K) \subseteq b$ by auto moreover
note $<x \in \text{int}(K) >$ ultimately have $\exists V \in T. x \in V \land \text{cl}(V) \subseteq b$ by auto

then have $\forall x \in \bigcup T. \forall b \in T. x \in b \rightarrow (\exists V \in T. x \in V \land \text{cl}(V) \subseteq b)$ by auto
then show $T$ is regular using $\text{exist_clos_neig_imp_regular}$ by auto
qed

72.5 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) $\text{her}_P$ is loc $P$:

assumes $\forall T. \forall B \in \text{Pow}(\bigcup TT). \forall A \in TT. TT$ is a topology $\land P(B, TT) \rightarrow P(B \land A, TT)$

shows $(T$ is locally $P)$ $\leftrightarrow (\forall x \in \bigcup T. \exists A \in TT. x \in A \land P(A, T))$

**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 $\text{IsLocally_def}[\text{OF topSpaceAssum}]$

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 $c \subseteq \bigcup T \land \text{int}(c) \subseteq P(c,T)$ by auto
have $P(\text{int}(c),T)$ by auto moreover
from $c(1,3)$ topSpaceAssum assms have $\forall A \in T. P(c \cap A,T)$ by auto
ultimately have $P(c \cap \text{int}(c),T)$ by auto
moreover from $c(1,3)$ topSpaceAssum assms have $\forall A \in T. P(c \cap A,T)$ by auto
ultimately have $P(\text{int}(c),T)$ by auto
then have $c \cap \text{int}(c) = \text{int}(c)$ by auto
ultimately have $P(\text{int}(c),T)$ 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 $x$ obtain $A$ where $A$ def: $A \in T \land x \in \text{AP}(A,T)$ by auto
from $A$ def(1,3) 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 $\text{int}(A \cap b) = A \cap b$ using Top_2_L3 by auto
ultimately have $x \in \text{int}(A \cap b) \land P(A \cap b,T)$ by auto
then have $\exists c \in \text{Pow}(b). x \in \text{int}(c) \land P(c,T)$ by auto
}
then 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 $(T\{\text{is locally-T_2}\}) \iff (\forall x \in \bigcup T. \exists A \in T. x \in A \land (T\{\text{restricted to}A\})\{\text{is T_2}\})$
proof-{
fix TT B A assume TT:TT{is a topology} (TT{restricted to}B){is T_2}
$A \in TT \land B \in \text{Pow}(\bigcup TT)$
then have $s : B \subseteq B \subseteq \bigcup TT$ by auto

1007
then have $(\text{TT}{\text{restricted to}}(B \cap A)) = (\text{TT}{\text{restricted to}}B){\text{restricted to}}(B \cap A)$ using subspace_of_subspace by auto moreover have $\bigcup (\text{TT}{\text{restricted to}}B) = B$ unfolding RestrictedTo_def using s(2) by auto

moreover have $B \cap A \subseteq \bigcup (\text{TT}{\text{restricted to}}B)$ using s(1) by auto moreover note $\text{TT}(2)$ ultimately have $(\text{TT}{\text{restricted to}}(B \cap A))$ is $T_2$ using $T_2$_here by auto

then have $\forall \text{TT}. \forall B \in \text{Pow}(\bigcup \text{TT}). \forall A \in \text{TT}. (\text{TT}{\text{restricted to}}B) \text{ is } T_2 \rightarrow (\text{TT}{\text{restricted to}}(B \cap A)) \text{ is } T_2$

by auto with her_P_is_loc_P[where $P=\lambda A. \lambda \text{TT}. (\text{TT}{\text{restricted to}}A) \text{ is } T_2$] 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{is hyperconnected} U \in T |
| shows (T{restricted to}U) \{is hyperconnected\} |

proof-

{ fix A B assume A \in (T{restricted to}U)B \in (T{restricted to}U)A \cap B = 0 then obtain AU BU where A = U \cap A \cup B \cap U \cup TBU \in T unfolding RestrictedTo_def by auto then have A \in TBC \in T using topSpaceAssum assms(2) unfolding IsATopology_def by auto with \langle A \cap B = 0 \rangle have A = 0 \cup 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{is locally-T_2} |
| shows T{is anti-}IsHConnected |

proof-

{ fix A assume A:A \in \text{Pow}(\bigcup T)(T{restricted to}A) \{is hyperconnected\} |
  { fix x assume x \in A |
    with A(1) have x \in \bigcup T by auto 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 (T \{\text{restricted to} c\} \{is } T_2\} using assms |

1008
unfolding IsLocallyT2_def IsLocally_def[OF topSpaceAssum] by auto then obtain c where \( c \in \text{Pow}(\bigcup T) \times \text{int}(c) \) \( (\text{restricted to} \ c) \) \{ is T_2 \} by auto

have \( \bigcup (T \{ \text{restricted to} \ c\}) = (\bigcup T) \cap c \) unfolding RestrictedTo_def by auto

with \( c \in \text{Pow}(\bigcup T) \cap c \) have \( \bigcup (T \{ \text{restricted to} \ c\}) = c \) by auto

have \( \text{int}(c) \in T \) 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 \( (T \{ \text{restricted to} \ A\}) \subseteq (A \cap (\text{int}(c))) \) \{ is hyperconnected \}

using topology0.open_subspace_hyperconn unfolding topology0_def using Top_1_L4 by auto

then have \( (T \{ \text{restricted to} \ A\}) \subseteq (A \cap (\text{int}(c))) \) \{ is T_2 \} using T2_here[OF c(3)] by auto

with sub have \( (T \{ \text{restricted to} \ A\}) \subseteq (A \cap (\text{int}(c))) \) \{ is T_2 \} using subspace_of_subspace[of A \cap (\text{int}(c))] unfolding RestrictedTo_def by auto

having \( A \cap (\text{int}(c)) \subseteq c \) using Top_2_L1 by auto

then have \( \text{sub} : A \cap (\text{int}(c)) \subseteq c \) by auto

then have \( A \cap (\text{int}(c)) \subseteq \bigcup (T \{ \text{restricted to} \ c\} \) using tot by auto

then have \( A \cap (\text{int}(c)) \subseteq \bigcup (T \{ \text{restricted to} \ c\} \) \{ is T_2 \} using T2_here[OF c(3)] by auto

ultimately have \( (T \{ \text{restricted to} \ (A \cap (\text{int}(c))) \} \subseteq \bigcup (T \{ \text{restricted to} \ (A \cap (\text{int}(c))) \) \} \subseteq 1 \) using HConn_spectrum by auto
then have \((A \cap (\text{int}(c))) = \{x\}\) using \text{lepoll} \_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
\{
  \begin{align*}
  \text{fix} x \ y & \text{ assume } x \neq y \in A y \in A \\
  \text{with pointOpen have } \{x\} \in (T\{\text{restricted to}\}A) \{y\} \in (T\{\text{restricted to}\}A) \{x\} \cap \{y\} = 0 \\
  \text{by auto} \\
  \text{with } A(2) \text{ have } \{x\} = 0 \lor \{y\} = 0 \text{ unfolding } \text{IsHConnected} \_\text{def} \text{ by auto} \\
  \text{then have False by auto}
  \end{align*}
\}
then have \(\text{uni} : \forall x \in A. \forall y \in A. x = y\) by auto
\{
  \begin{align*}
  \text{assume } A \neq 0 \\
  \text{then obtain } x \text{ where } x \in A \text{ by auto} \\
  \text{with uni have } A = \{x\} \text{ by auto} \\
  \text{then have } A \approx 1 \text{ using } \text{singleton_eqpoll} \_1 \text{ by auto} \\
  \text{then have } A \subseteq 1 \text{ using } \text{eqpoll} \_\text{imp_eqpoll} \text{ by auto}
  \end{align*}
\}
moreover
\{
  \begin{align*}
  \text{assume } A = 0 \\
  \text{then have } A \approx 0 \text{ by auto} \\
  \text{then have } A \subseteq 1 \text{ using } \text{empty_eqpoll} \_1 \text{ eq_lepoll_trans by auto}
  \end{align*}
\}
ultimately have \(A \subseteq 1\) by auto
then have \(A\{\text{is in the spectrum of}\} \text{IsHConnected}\) using \text{HConn} \_\text{spectrum} \text{ by auto}
\}
then show \(\text{thesis}\) unfolding \text{antiProperty} \_\text{def} \text{ 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, \(\text{where an infinite number of points don’t have finite sets as neighbourhoods.}\) 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 non-
empty open set. The question is if such a topology exists.

**theorem (in topology0)** \(\text{COF\_comp\_is\_top}:
\)
assumes \(T\{\text{is \ T}\_1\}-(\bigcup T<\text{nat})\)
shows \(((\{\text{one-point compactification of}(\text{CoFinite} (\bigcup T)))-(\{\bigcup T\})\cup T)\)
{is a topology}
**proof**
- have \(N:\bigcup T\notin (\bigcup T)\) using mem_not_refl by auto
  
  fix \(M\) assume \(M:M\subseteq((\{\text{one-point compactification of}(\text{CoFinite} (\bigcup T)))-(\{\bigcup T\})\cup T)\)
  
  let \(MT=\{A\in M. A\in T\}\)
  
  let \(MK=\{A\in M. A\notin T\}\)
  
  have \(MM:\bigcup MT\cup \bigcup MK=\bigcup M\) by auto
  
  have \(MN:\bigcup MT\subseteq(\{\text{one-point compactification of}(\text{CoFinite} (\bigcup T)))-(\{\bigcup T\})\)
  
  using \(M\) by auto
  then have \(MK\subseteq(\{\text{one-point compactification of}(\text{CoFinite} (\bigcup T)))\)
  
  by auto
  then have \(CO:\bigcup MK\subseteq((\{\text{one-point compactification of}(\text{CoFinite} (\bigcup T)))-(\{\bigcup T\})\)
  
  unfolding \text{Cofinite\_def}
  
  IsaTopology\_def by auto
  
  assume \(AS:\bigcup MK=\{\bigcup T\}\)
  moreore have \(\forall R\subseteq MK. R\subseteq MK\) by auto
  
  ultimately have \(\forall R\subseteq (\bigcup T)\) by auto
  then have \(\forall R\subseteq MK. R=\{\bigcup T\}\) by force moreover
  
  with \(MN\) have \(\forall R\subseteq MK. R=0\) by auto
  then have \(\bigcup MK=0\) by auto
  with \(AS\) have False by auto
  
  with \(CO\) have \(CO2:\bigcup MK\subseteq((\{\text{one-point compactification of}(\text{CoFinite} (\bigcup T)))-(\{\bigcup T\})\)
  
  by auto
  
  assume \(\bigcup MK\subseteq(\text{CoFinite} (\bigcup T))\)
  then have \(\bigcup MK\subseteq T\) using \(\text{assms(1)}\ T1\_cocardinal\_coarser\) by auto
  with \(MN\) have \(\{\bigcup MT, \bigcup MK\}\subseteq T\) by auto
  then have \(\bigcup MT\cup (\bigcup MK)\in T\) using \(\text{union\_open[OF topSpaceAssum, of} \{\bigcup MT, \bigcup MK\}\})\) by auto
  then have \(\bigcup M\subseteq T\) using \(\text{MM}\) by auto
  moreover
  
  assume \(\bigcup MK\notin(\text{CoFinite} (\bigcup T))\)
  
  with \(CO\) obtain \(B\) where \(B\{\text{is compact in}(\text{CoFinite} (\bigcup T))\}B\{\text{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)\}B\{\text{is closed in}(\text{CoFinite} (\bigcup T))\}
  
1011
using union_cocardinal unfolding Cofinite_def by auto
then have \( B:B \subseteq \bigcup T \) unfolding Cofinite_def by auto
\{  
  assume \( B=\bigcup T \)
  with \( MK \) have \( \bigcup MK=\{\bigcup T\} \) by auto
  then have False using CO2 by auto
\}
with \( B \) have \( B \subseteq \bigcup T \) and \( \text{nat} B: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 \( (\bigcup T)-(\bigcup MT)) \cap B \subseteq B \) unfolding Cofinite_def
then have \( ((\bigcup T)-(\bigcup MT)) \cap B) \{\text{is closed}\}(\text{Cofinite } (\bigcup T))\) using closed_sets_cocardinal
\( B(1) \) unfolding Cofinite_def by auto
then have \( \bigcup T-(\bigcup MT)\cap B \in (\text{Cofinite } (\bigcup T)) \) using IsClosed_def
using union_cocardinal unfolding Cofinite_def by auto
also have \( \bigcup T-(\bigcup MT)\cap B = (\bigcup T-(\bigcup MT))\cup (\bigcup T-B) \) by auto
also have \( \ldots = (\bigcup MT)\cup (\bigcup T-B) \) by auto
ultimately have \( P:(\bigcup MT)\cup (\bigcup T-B) \in (\text{Cofinite } (\bigcup T)) \) by auto
then have eq: \( \bigcup T-(\bigcup MT)\cup (\bigcup T-B)) = (\bigcup MT)\cup (\bigcup T-B) \) by auto
from \( P \) eq have \( \bigcup T-(\bigcup MT)\cup (\bigcup T-B)) \{\text{is closed}\}(\text{Cofinite } (\bigcup T)) \)
unfolding IsClosed_def
using union_cocardinal[of \( \text{nat} \bigcup T \) ] unfolding Cofinite_def by auto
moreover have \( \bigcup T-(\bigcup MT)\cup (\bigcup T-B)) \cap \bigcup T = \bigcup T-(\bigcup MT)\cup (\bigcup T-B)) \) by auto
then have \( (\text{Cofinite } (\bigcup T))\{\text{restricted to}\}(\bigcup T-(\bigcup MT)\cup (\bigcup T-B)) = \text{Cofinite } (\bigcup T-(\bigcup MT)\cup (\bigcup T-B)) \) using subspace_cocardinal unfolding Cofinite_def
by auto
then have \( \bigcup T-(\bigcup MT)\cup (\bigcup T-B)) \{\text{is compact}\}(\text{Cofinite } (\bigcup T)) \) using cofinite_compact
unifying compact_subspace_imp_compact by auto
ultimately have \( \{\bigcup T\} \cup (\bigcup T-(\bigcup MT)\cup (\bigcup T-B)) \in \{\text{one-point compactification of}\}(\text{Cofinite } (\bigcup T)) \)
unfolding one_point_compactification_def using union_cocardinal unfolding Cofinite_def by auto
with eq have \( \{\bigcup T\} \cup (\bigcup NT)\cup (\bigcup T-B) \in \{\text{one-point compactification of}\}(\text{Cofinite } (\bigcup T)) \) by auto
moreover have \( AA: (\bigcup T) \cup (\bigcup MT)\cup (\bigcup T-B) = (\bigcup MT)\cup (\bigcup MK) \) using \( MK(1) \)
by auto
ultimately have \( AA2: (\bigcup MT)\cup (\bigcup MK) \in \{\text{one-point compactification of}\}(\text{Cofinite } (\bigcup T)) \) 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 CO2 by auto

with AA2 have \(((\bigcup MT) \cup (\bigcup MK)) \in \{\text{one-point compactification of}\}(\text{CoFinite } (\bigcup T))-\{\{\bigcup T\}\}\) by auto

ultimately have \( \bigcup M \in \{\text{one-point compactification of}\}(\text{CoFinite } (\bigcup T))-\{\{\bigcup T\}\}\) by auto

then have \( \forall M \in \text{Pow}(\{\text{one-point compactification of}\}(\text{CoFinite } (\bigcup T))-\{\{\bigcup T\}\}\) \cup T\)
by auto moreover

fix \( U, V \) assume \( U \in \{\text{one-point compactification of}\}(\text{CoFinite } (\bigcup T))-\{\{\bigcup T\}\}\) \cup TV \in \{\text{one-point compactification of}\}(\text{CoFinite } (\bigcup T))-\{\{\bigcup T\}\}\) \cup T moreover

assume \( U \in TV \in \) topology0 using topSpaceAssum unfolding IsATopology_def by auto
then have \( U \in TV \in \) \{\text{one-point compactification of}\}(\text{CoFinite } (\bigcup T))-\{\{\bigcup T\}\}\) \cup T by auto

moreover

assume \( UV: U \in \{\text{one-point compactification of}\}(\text{CoFinite } (\bigcup T))-\{\{\bigcup T\}\}\) \cup V \in \{\text{one-point compactification of}\}(\text{CoFinite } (\bigcup T))-\{\{\bigcup T\}\}\) \cup T
using topology0.op_comp_is_top[OF topology0_CoCardinal[OF InfCard_nat]] unfolding Cofinite_def

IsATopology_def by auto
then have \( U \cap V \in \{\text{one-point compactification of}\}(\text{CoFinite } (\bigcup T))\) \cup T
unfolding RestrictedTo_def by auto

then have \( U \cap (U \cup V) \in \{\text{one-point compactification of}\}(\text{CoFinite } (\bigcup T))\) \cup T using topology0.open_subspace(2)[OF topology0_CoCardinal[OF InfCard_nat]] unfolding union_cocardinal

from \( UV \) have \( U \neq \{\bigcup T\} \neq \bigcup T \) \cup \( \bigcup W \in \{\text{one-point compactification of}\}(\text{CoFinite } (\bigcup T))\) \cup T \cup \( \bigcup W \in \{\text{one-point compactification of}\}(\text{CoFinite } (\bigcup T))\) \cup T

unfolding RestrictedTo_def by auto

1013
then have $R: U \neq \{T\} \cup \{T\}$, using topology0.open_subspace(2) [OF topology0_CoCardinal [OF InfCard_nat]]
  union_cocardinal unfolding Cofinite_def by auto

  from UV have $U \subseteq \bigcup \{\text{one-point compactification of}(\text{CoFinite } (\{T\}))\} \subseteq \bigcup \{\text{one-point compactification of}(\text{CoFinite } (\{T\}))\}$, by auto
  then have $U \subseteq \{T\} \cup \{T\} \subseteq \bigcup \{\text{one-point compactification of}(\text{CoFinite } (\{T\}))\}$, by auto

  then have $E: U = \bigcup \{T\} \cap U \cup \bigcup \{T\} \cap V \subseteq \{\bigcup \{T\} \cap U \bigcup \{\bigcup \{T\} \cap V\}\}$, by auto

  assume $Q: U \cap V = \{\bigcup \{T\}\}$
  then have $RR: \bigcup \{T\} \cap (U \cap V) = 0$, using N, by auto
  assume $\bigcup \{T\} \cap U = 0$
  with $E(1)$ have $U = \{T\} \cap U$, by auto
  also have $\ldots \subseteq \{T\}$, by auto
  ultimately have $U \subseteq \{T\}$, by auto
  then have $U = 0 \vee U = \{T\}$, by auto
  with $R(1)$ have $U = 0$, by auto
  then have $U \cap V = 0$, by auto
  then have False, using $Q$, by auto

  moreover
  { assume $\bigcup \{T\} \cap V = 0$
  with $E(2)$ have $V = \{T\} \cap V$, by auto
  also have $\ldots \subseteq \{T\}$, by auto
  ultimately have $V \subseteq \{T\}$, by auto
  then have $V = 0 \vee V = \{T\}$, 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 $\bigcup \{T\} \neq 0 \bigcup \{T\} \neq 0$
  with $R(3, 4)$ have $(\bigcup \{T\} \cap (\bigcup \{T\} \neq 0) using Cofinite_nat_HConn [OF assms(2)]$
  unfolding IsHConnected_def by auto
  then have $\bigcup \{T\} \cap (\bigcup \{T\} \neq 0)$, by auto
  then have False, using $RR$, by auto

  ultimately have False, by auto

  with 0 have $U \cap V = \{\text{one-point compactification of}(\text{CoFinite } (\{T\}))\} - \{\bigcup \{T\}\} \cup T$
  by auto

  moreover

1014
The previous construction preserves anti-hyperconnectedness.

**Theorem (in topology0)** COF_comp_antiHConn:

assumes \( T \) {is anti-}IsHConnected

shows \(((\{\text{one-point compactification of}\}(\text{CoFinite } (\bigcup T)))-\{\bigcup T\}) \cup T\) {is anti-}IsHConnected
proof-

have N:\bigcup T \notin (\bigcup T) using mem_not_refl by auto
from assms(1) have T1:T{is T1} using anti_HConn_imp_T1 by auto
have tot1:\bigcup ({one-point compactification of}(CoFinite (\bigcup T))) = \bigcup T
using topology0.op_compact_total[OF topology0_CoCardinal[OF InfCard_nat],
of \bigcup T]
union_cocardinal[of nat\bigcup T] unfolding Cofinite_def by auto

then have (\bigcup ({one-point compactification of}(CoFinite (\bigcup T)))) \bigcup \bigcup T = \bigcup T \bigcup T
by auto
ultimately have tot2: \bigcup ({one-point compactification of}(CoFinite (\bigcup T))) \bigcup T = \bigcup T \bigcup T
by auto

then have \bigcup T \neq 0 by auto
with assms(2) have \neg (0 < nat) by auto
then have False unfolding lesspoll_def using empty_lepollI eqpoll_0_is_0

\text{fix A assume AS:}\ A \subseteq \bigcup T \text{ (((\bigcup T \subseteq \{\bigcup T\}) \neg{\bigcup T}) \bigcup T)}
\text{by auto}

ultimately have TOT: \bigcup (\bigcup (\bigcup (\bigcup T \bigcup\bigcup T)) \bigcup T) \subseteq \bigcup (\bigcup T \bigcup T)
by auto

\text{fix A assume AS:}\ A \subseteq \bigcup T \text{ (((\bigcup T \subseteq \{\bigcup T\}) \neg{\bigcup T}) \bigcup T)}
\text{by auto}

ultimately have TOT: \bigcup (\bigcup (\bigcup (\bigcup T \bigcup\bigcup T)) \bigcup T) \subseteq \bigcup (\bigcup T \bigcup T)
by auto
\((\cup T)\)\ (((\{one-point compactification of\}(Cofinite \(\cup T\))) - \{\{\cup T\}\} \cup T\{restricted to\}A) \{is hyperconnected\}

from AS\((1,2)\) have e\(0:\(((\{one-point compactification of\}(Cofinite \(\cup T\))) - \{\{\cup T\}\} \cup T\{restricted to\}A\{using subspace of subspace of\}(A \cup T) (((\{one-point compactification of\}(Cofinite \(\cup T\))) - \{\{\cup T\}\} \cup T\{restricted to\}A \cup T\{restricted to\}A)\{is hyperconnected\}

have e\(1:\(((\{one-point compactification of\}(Cofinite \(\cup T\))) - \{\{\cup T\}\} \cup T\{restricted to\}(\cup T) = (((\{one-point compactification of\}(Cofinite \(\cup T\))) - \{\{\cup T\}\} \cup T\{restricted to\}(\cup T) \cup (T \{restricted to\} \cup T) \cup (T \{restricted to\} \cup T))\{restricted to\}A = (((\cup T) \cup (T \{restricted to\} \cup T)) \cup (T \{restricted to\} \cup T) \cup (T \{restricted to\} \cup T)) \cup (T \{restricted to\} \cup T) \cup (T \{restricted to\} \cup T)\{restricted to\}A).

unfolding RestrictedTo_def by auto

fix A assume A \in T \{restricted to\} \cup T
then obtain B where B \in TA = B \cap \cup T unfolding RestrictedTo_def by auto
then have A = B by auto
with \(B \in T\) have A \in T by auto

then have T \{restricted to\} \cup T \subseteq T by auto moreover

fix A assume A \in T
then have \(\cup T \cap A = A\) by auto
with \(A \in T\) have A \in T \{restricted to\} \cup T unfolding RestrictedTo_def by auto

ultimately have T \{restricted to\} \cup T = T by auto moreover

fix A assume A \in (((\{one-point compactification of\}(Cofinite \(\cup T\))) - \{\{\cup T\}\} \cup T\{restricted to\}A)\{is hyperconnected\}

unfolding RestrictedTo_def by auto

\fix A assume A \in (\cup T
then obtain B where B \in (\{one-point compactification of\}(Cofinite \(\cup T\))) - \{\{\cup T\}\} \cup T \cap B = A unfolding RestrictedTo_def by auto
then have B \in (\{one-point compactification of\}(Cofinite \(\cup T\))) \cup T \cap B = A by auto

then have A \in (\{one-point compactification of\}(Cofinite \(\cup T\))) \{restricted to\} \cup T unfolding RestrictedTo_def by auto
then have A \in (Cofinite \(\cup T\)) using topology0.open_subspace(2)[OF topology0_CoCardinal[OF InfCard_nat]]
union_cocardinal unfolding Cofinite_def by auto
with \(T_1\) have A \in T using \(T_1\)_cocardinal_coarser by auto

then have ((\{one-point compactification of\}(Cofinite \(\cup T\))) - \{\{\cup T\}\} \cup T) \{restricted to\} A \subseteq T by auto
moreover note e\(1\) ultimately have (((\{one-point compactification of\}(Cofinite \(\cup T\)) - \{\{\cup T\}\} \cup T) \{restricted to\} A = T by auto
with e\(0\) have (((\{one-point compactification of\}(Cofinite \(\cup T\)) - \{\{\cup T\}\} \cup T) \{restricted to\} A = T \{restricted to\} A by auto
with assms(1) A \in (\cup T) \{is in the spectrum of\} IsHConnected unfolding antiProperty_def by auto

then have \(\forall A \in \text{Pow}(\cup T). (((\{one-point compactification of\}(Cofinite \(\cup T\)))) - \{\{\cup T\}\} \cup T) \{restricted to\} A \subseteq T by auto.

1017
(\cup T)) - (\{\cup T\}) \cup T \{\text{restricted to} A\} \{\text{is hyperconnected}\} \rightarrow (A \{\text{is in the spectrum of} \text{IsHConnected}\}) \text{ by auto}

have \cup T \in T \text{ using topSpaceAssum unfolding IsATopology_def by auto}
then have P: \cup T \in ((({\text{one-point compactification of} (\text{CoFinite} (\cup T))) - (\{\cup T\}) \cup T) \{\text{restricted to} B\} \{\text{is hyperconnected}\}) \text{ by auto}

{\text{fix } B \text{ assume } \text{sub}: B \in \cup T \cup \{\cup T\} \text{ and hyphyppp} (((({\text{one-point compactification of} (\text{CoFinite} (\cup T))) - (\{\cup T\}) \cup T) \{\text{restricted to} B\} \{\text{is hyperconnected}\}) \text{ from } P \text{ have subop}: \cup T \cap B \in ((({\text{one-point compactification of} (\text{CoFinite} (\cup T))) - (\{\cup T\}) \cup T) \{\text{restricted to} B\} \{\text{is hyperconnected}\}) \text{ using topology0.open_subspace_hyperconn}}

have \cup T \subseteq B \text{ by auto}
then have \cup T \in \{x, \cup T\} \text{ by auto}

{x} \in \{x, \cup T\} \text{ by auto}

{x} \in \cup T \cup \{\cup T\} \text{ using sub by auto}
with y have y \in \cup T \cup B \lor y = \cup T \text{ by auto}
with sing have y = x \lor y = \cup T \text{ by auto}

then have \cup T \cup B \subseteq 1 \text{ using HConn_spectrum by auto}

{\text{fix } x \text{ assume } x \in \cup T \cup B \text{ by auto}

{\text{fix } y \text{ assume } y \in B \text{ using lepoll_1_is_sing by auto}

{\text{then have } y \in \cup T \cup \{\cup T\} \text{ using sub by auto}
with y have y \in \cup T \cup B \lor y = \cup T \text{ by auto}
with sing have y = x \lor y = \cup T \text{ by auto}

then have \cup T \subseteq B \text{ by auto}
with x have disj: B = \{x\} \lor B = \{x, \cup T\} \text{ by auto}

{\text{assume } \cup T \in B \text{ by auto}
with disj have B = \{x, \cup T\} \text{ by auto}
from sing subop have singUp: \{x\} \in ((({\text{one-point compactification of} (\text{CoFinite} (\cup T))) - (\{\cup T\}) \cup T) \{\text{restricted to} B\} \text{ by auto}

have \{x\} \{\text{is closed in} \text{CoFinite} (\cup T) \text{ using topology0.T1_iff_singleton_closed[of topology0.Cofinite[OF InfCard_nat]] cocardinal_is_T1[OF InfCard_nat]}}

x \cup T \cup B \subseteq \{x, \cup T\} \text{ by auto}

moreover
have Finite(\{x\}) \text{ by auto}
then have spec: \{x\} \{\text{is in the spectrum of} \lambda T. (\cup T) \{\text{is compact}}

1018
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]]
unfolding RestrictedTo_def Cofinite_def
  using x union_cocardinal by auto
with spec have {x}{is compact in}((CoFinite ∪ T){restricted to}{x})
  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)
  unfolding OPCompactification_def
    using union_cocardinal unfolding Cofinite_def by auto
moreover have A:{∪T}∪({∪T-{x}}){is compact in}({∪T})
  { assume y∈{∪T-{x}}
    then have y∈{∪T}∪({∪T-{x}}) by auto
    then have y=∪T using A by auto
    with x have False by auto
  }
then have ∪T-{x}=∅
  by auto
with x have ∪T={{}∪T}
  by auto
ultimately have {∪T}∈(({one-point compactification of}(CoFinite ∪ T))-{∪T})
    unfolding RestrictedTo_def
by auto
moreover have {∪T}∈(({one-point compactification of}(CoFinite ∪ T))-{∪T})
then have B∩{∪T}∪({∪T-{x}})∈((({one-point compactification of}(CoFinite ∪ T))-{∪T})∪T){restricted to}B
  unfolding RestrictedTo_def
by auto
moreover have B∩{∪T}∪({∪T-{x}})=∪T using B by auto
ultimately have {∪T}{is compact in}(CoFinite ∪ T)
  unfolding OPCompactification_def
by auto
with singOp hyp N x have False unfolding IsHConnected_def by auto
with disj have B={x} by auto
then have B≈1 using singleton_eqpoll_1 by auto
then have B≤1 using eqpoll_imp_lepoll by auto
then have ∪T∩B≠∅→B≤1 by blast
moreover
{ assume \( \bigcup T \cap B = 0 \)
with sub have \( B \subseteq \{ \bigcup T \} \) by auto
then have \( B \subseteq \{ \bigcup T \} \) using subset_imp_lepoll by auto
then have \( B \subseteq 1 \) using singleton_eqpoll_1 lepoll_eq_trans by auto
ultimately have \( B \approx 1 \) by auto
then have \( B \) is in the spectrum of \( \text{IsHConnected} \) using HConn_spectrum 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** \( T \text{\_def} \equiv (\text{OrdTopology} \ X \ r) \)

**Assumes** \( \text{IsLinOrder}(X, r) \) \( X \)\( \text{is dense with respect to} \) \( r \)

\[
\exists x \ y. x \neq y \land x \in X \land y \in X
\]

\[
U \subseteq U \cup (\{\text{one-point compactification of}\}(\text{CoFinite} \ (\bigcup T))) - \{\bigcup T\} \cup T
\]

\[
V \subseteq (\{\text{one-point compactification of}\}(\text{CoFinite} \ (\bigcup T))) - \{\bigcup T\} \cup T
\]

\( V \neq 0 \)

**Proof**

have \( N : \bigcup T \notin \{\bigcup T\} \) using mem_not_refl by auto
have \( \text{tot1} : (\{\text{one-point compactification of}\}(\text{CoFinite} \ (\bigcup T))) = \{\bigcup T\} \cup T \)
using topology0.op_compact_total[OF topology0_CoCardinal[OF InfCard_nat], of \( \bigcup T \)]
union_cocardinal[of nat \( \bigcup T \) ] unfolding Cofinite_def by auto
then have \( \{\bigcup T\} \cup (\{\text{one-point compactification of}\}(\text{CoFinite} \ (\bigcup T))) = \{\bigcup T\} \cup T \cup T \)
by auto moreover
have \( \{\bigcup T\} \cup (\{\text{one-point compactification of}\}(\text{CoFinite} \ (\bigcup T))) = \{\bigcup T\} \cup T \cup T \)
by auto
ultimately have \( \text{tot2} : (\{\text{one-point compactification of}\}(\text{CoFinite} \ (\bigcup T))) = \{\bigcup T\} \cup T \cup T \)
by auto
have \( \{\bigcup T\} \cup \bigcup T \subseteq (\{\text{one-point compactification of}\}(\text{CoFinite} \ (\bigcup T))) \)
using union_open[OF topology0.op_comp_is_top[OF topology0_CoCardinal[OF InfCard_nat], of \( \{\text{one-point compactification of}\}(\text{CoFinite} \ (\bigcup T))\)]]

\( \text{tot1} \) unfolding Cofinite_def by auto moreover

{ assume \( \bigcup T = 0 \)
then have \( X = 0 \) unfolding T_def using union_ordtopology[OF assms(2)]
}

1020
assms(4) by auto
  then have False using assms(4) by auto
}
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 \( \{\bigcup T\} \cup \bigcup T \in (\bigcup \{\text{(one-point compactification of) (CoFinite } \bigcup T)) \cup \{\bigcup T\} \cup \bigcup T \) by auto
  then have \( \{\bigcup T\} \cup \bigcup T \in (\bigcup \{\text{(one-point compactification of) (CoFinite } \bigcup T)) \cup \{\bigcup T\} \cup \bigcup T \) by auto moreover
  have \( (\bigcup \{\text{(one-point compactification of) (CoFinite } \bigcup T)) \cup \{\bigcup T\} \cup \bigcup T \) by auto
  then have \( (\bigcup \{\text{(one-point compactification of) (CoFinite } \bigcup T)) \cup \{\bigcup T\} \cup \bigcup T \) by auto
  with \( \text{tot2} \) have \( (\bigcup \{\text{(one-point compactification of) (CoFinite } \bigcup T)) \cup \{\bigcup T\} \cup \bigcup T \) by auto
  ultimately have \( \text{TOT: } \bigcup (\{\text{(one-point compactification of) (CoFinite } \bigcup T)) \cup \{\bigcup T\} \cup \bigcup T \) by auto

assume \( A : U \cap V = 0 \)
with assms(6) have \( \neg : \bigcup T \notin V \) by auto
with assms(7) have \( V \in \text{(CoFinite } \{\bigcup T\} \cup \bigcup T \) unfolding \text{OPCompactification_def using union_cocardinal unfolding Cofinite_def by auto}
  moreover have \( T \{\text{is } T_2\} \) unfolding \text{T_def using order_top_T2[OF assms(2)]}
assms(4) by auto
  then have \( T_1 : \{\text{is } T_1\} \) using \text{T2_is_T1 by auto}
ultimately have \( \text{VopT: } V \in T \) using \text{topology0.T1_cocardinal_coarser[OF topology0_ordtopology(1)][OF assms(2)]}
assms(4) by auto
  unfolding \text{T_def by auto}
from \( A \) assms(7) have \( V \in \{\text{(one-point compactification of) (CoFinite } \{\bigcup T\} \cup \bigcup T \) by auto
  then have \( V \in \{\text{(one-point compactification of) (CoFinite } \{\bigcup T\} \cup \bigcup T \) by auto
  then have \( V \in \{\text{(one-point compactification of) (CoFinite } \{\bigcup T\} \cup \bigcup T \) by auto
  from \( N \) have \( U \notin T \) using assms(6) by auto
  then have \( U \notin \text{(CoFinite } \{\bigcup T\} \cup \bigcup T \) using \text{T1 topology0.T1_cocardinal_coarser[OF topology0_ordtopology(1)[OF assms(2)]]
assms(4) by auto
  with assms(5,6) obtain \( B \) where \( U : U = \{\bigcup T\} \cup (\{\bigcup T\} - B) \) is closed in \( \text{(CoFinite } \{\bigcup T\} \cup \bigcup T \) by\( \neq \) by auto
  unfolding \text{OPCompactification_def using union_cocardinal unfolding}
Cofinite_def by auto
  then have \( U=\bigcup T \cup (\bigcup T-B) \) \( B=\bigcup T \) \( B\neq \bigcup T \) using closed_sets_cocardinal
  unfolding Cofinite_def
  by auto
  then have \( U=\bigcup T \cup (\bigcup T-B) \) \( B\neq \bigcup T \) using auto
  with \( N \) have \( \bigcup T=\bigcup T-B \) \( B\neq \bigcup T \) using closed_sets_cocardinal
  unfolding Cofinite_def
  by auto
  then have \( \bigcup T-U=\bigcup T-B \) \( B\neq \bigcup T \) using \( \bigcup T=\bigcup T-B \) using auto
  with \( N \) have \( \bigcup T-U=\bigcup T-B \) \( B\neq \bigcup T \) using \( \bigcup T=\bigcup T-B \) using auto
  from \( \text{assms}(8) \) obtain \( v \) where \( v \in V \) by auto
  with \( \text{assms}(8) \) obtain \( R \) where \( \text{assms}(8) \) by blast
  moreover
  { assume \( R \in \{ \text{IntervalX}(X, r, b, c) . \langle b,c \rangle \in X \times X \} \cup \{ \text{LeftRayX}(X, r, b) . b \in X \} \cup \{ \text{RightRayX}(X, r, b) . b \in X \} \) \( R \subseteq V \) \( v \in R \)
    by auto
    with \( \text{assms}(8) \) have \( \neg \text{Finite}(R) \) using dense_order_inf_intervals[OF assms(2)
    \( 2 \) [OF assms(2)]
    unfolding T_def by auto
    then obtain \( R \) where \( \text{assms}(8) \) by blast
    moreover
    { assume \( R \in \{ \text{IntervalX}(X, r, b, c) . \langle b,c \rangle \in X \times X \} \cup \{ \text{LeftRayX}(X, r, b) . b \in X \} \cup \{ \text{RightRayX}(X, r, b) . b \in X \} \)
      \( R \subseteq V \) \( v \in R \)
      by auto
      with \( \text{assms}(8) \) have \( \neg \text{Finite}(R) \) using dense_order_inf_intervals[OF assms(2)
      \( 2 \) [OF assms(2)]
      unfolding T_def by auto
      then obtain \( R \) where \( \text{assms}(8) \) by blast
      moreover
      { assume \( R \in \{ \text{IntervalX}(X, r, b, c) . \langle b,c \rangle \in X \times X \} \cup \{ \text{LeftRayX}(X, r, b) . b \in X \} \cup \{ \text{RightRayX}(X, r, b) . b \in X \} \)
        \( R \subseteq V \) \( v \in R \)
        by auto
        with \( \text{assms}(8) \) have \( \neg \text{Finite}(R) \) using dense_order_inf_intervals[OF assms(2)
        \( 2 \) [OF assms(2)]
        unfolding T_def by auto
        then obtain \( R \) where \( \text{assms}(8) \) by blast
        moreover
        { assume \( R \in \{ \text{IntervalX}(X, r, b, c) . \langle b,c \rangle \in X \times X \} \cup \{ \text{LeftRayX}(X, r, b) . b \in X \} \cup \{ \text{RightRayX}(X, r, b) . b \in X \} \)
          \( R \subseteq V \) \( v \in R \)
          by auto
          with \( \text{assms}(8) \) have \( \neg \text{Finite}(R) \) using dense_order_inf_intervals[OF assms(2)
          \( 2 \) [OF assms(2)]
          unfolding T_def by auto
          then obtain \( R \) where \( \text{assms}(8) \) by blast
          ultimately
          show False by auto
          qed

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 \( T \times r \)
defines \( T_{\text{def}} \equiv (\text{OrdTopology } X \ r) \)
assumes \( \text{IsLinOrder}(X, r) \) \( X \) is dense with respect to \( r \)
\( \exists x \ y. \ x \neq y \wedge x \in X \wedge y \in X \)
shows \( \neg((\{\text{one-point compactification of}\}(\text{CoFinite } (\text{\bigcup } T))) - \{\text{\bigcup } T}\} \cup T)\) is locally-\( T_2 \)

proof

have \( N: \text{\bigcup } T \notin (\text{\bigcup } T) \) using mem_not_refl by auto
have \( \text{tot1:} \text{\bigcup } (\{\text{one-point compactification of}\}(\text{CoFinite } (\text{\bigcup } T))) = (\text{\bigcup } T) \cup T \) using topology0.op_compact_total[OF topology0_CoCardinal[OF InfCard_nat], of \( \text{\bigcup } T \)]
union_cocardinal[of nat]\( \text{\bigcup } T \) unfolding Cofinite_def by auto
then have \( \text{\bigcup } (\{\text{one-point compactification of}\}(\text{CoFinite } (\text{\bigcup } T))) = (\text{\bigcup } T) \cup T \) by auto
ultimately have \( \text{tot2:} \text{\bigcup } (\{\text{one-point compactification of}\}(\text{CoFinite } (\text{\bigcup } T))) = (\text{\bigcup } T) \cup T \) by auto

moreover

have \( \text{\bigcup } T \subseteq (\text{\bigcup } T) \cup T \) by auto
ultimately have \( \text{\bigcup } T \subseteq (\text{\bigcup } T) \cup T \) by auto

moreover

have \( \text{\bigcup } T \subseteq (\text{\bigcup } T) \cup T \) by auto
ultimately have \( \text{\bigcup } T \subseteq (\text{\bigcup } T) \cup T \) by auto

moreover

have \( \text{\bigcup } (\{\text{one-point compactification of}\}(\text{CoFinite } (\text{\bigcup } T))) \cup T \subseteq (\text{\bigcup } T) \cup T \) by auto
ultimately have \( \text{\bigcup } (\{\text{one-point compactification of}\}(\text{CoFinite } (\text{\bigcup } T))) \cup T \subseteq (\text{\bigcup } T) \cup T \) by auto
with \( \text{tot2} \) have \( \text{\bigcup } (\{\text{one-point compactification of}\}(\text{CoFinite } (\text{\bigcup } T))) - \{\text{\bigcup } T\} \subseteq \{\text{\bigcup } T\} \cup T \) by auto

ultimately have $\text{T}: \bigcup (\{\text{cofinite}\{\bigcup T\}\} - \{\bigcup T\}) \cup T = \{\bigcup T\}$

by auto

have $T_1 : \{\text{is } T_1\}$ using order_top_T2[OF assms(2,4)] T2_is_T1 unfolding T_def by auto

moreover

from assms(4) obtain $b, c$ where $b \in X, c \in X, b \neq c$ by auto

{ assume $(b, c) \notin r$

  with assms(2) have $(b, c) \in r$ unfolding IsLinOrder_def IsTotal_def using

  $\langle b \in X, c \in X \rangle$ by auto

  with assms(3) B obtain $z$ where $z \in X - \{b, c\}, c \in r(z, b) \in r$ unfolding

  IsDense_def by auto

  then have $\langle c, z \rangle \in r$ unfolding IntervalX_def

  by auto

  with assms(3) B obtain $z$ where $z \in X - \{b, c\}$ unfolding

  IsDense_def by auto

  then have $IntervalX(X, r, c, b) \neq 0$ unfolding IntervalX_def using Order_ZF_2_L1

  by auto

  then have $\neg (\text{finite}(IntervalX(X, r, c, b)))$ using dense_order_inf_intervals[OF

  assms(2) _ \langle c \in X, b \in X \rangle asms(3)]

  by auto

  have $IntervalX(X, r, c, b) \subseteq X$ unfolding IntervalX_def by auto

  ultimately have $\neg (\text{finite}(X))$ using subset_Finite by auto

  then have $\neg (X \succ\text{nat})$ using lesspoll_nat_is_Finite by auto

} moreover

{ assume $(b, c) \in r$

  with assms(3) B obtain $z$ where $z \in X - \{b, c\}, c \in r(z, b) \in r$ unfolding

  IsDense_def by auto

  then have $\langle c, z \rangle \in r$ unfolding IntervalX_def using Order_ZF_2_L1

  by auto

  then have $\neg (\text{finite}(IntervalX(X, r, b, c)))$ using dense_order_inf_intervals[OF

  assms(2) _ \langle b \in X, c \in X \rangle asms(3)]

  by auto

  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 $\neg (X \succ\text{nat})$ using lesspoll_nat_is_Finite by auto

} ultimately have $\neg (X \succ\text{nat})$ by auto

with $T_1$ have top:(\{\text{one-point compactification of}\}(\text{cofinite}\{\bigcup T\}) - \{\bigcup T\}) \cup T\{\text{is a topology}\}$ using topology0.COF_comp_is_top[OF topology0_ordtypeppelinology[OF

assms(2)] unfolding T_def

using union_ordtopology[OF assms(2,4)] by auto

assume $(\{\text{one-point compactification of}\}(\text{cofinite}\{\bigcup T\}) - \{\bigcup T\}) \cup T\{\text{is locally-T}_2\}$ moreover

have $\bigcup T \in (\{\text{one-point compactification of}\}(\text{cofinite}\{\bigcup T\}) - \{\bigcup T\}) \cup T$

using TOT by auto

moreover have $\bigcup (\{\text{one-point compactification of}\}(\text{cofinite}\{\bigcup T\}) - \{\bigcup T\}) \in (\{\text{one-point compactification of}\}(\text{cofinite}\{\bigcup T\}) - \{\bigcup T\}) \cup T$

using top unfolding IsATopology_def by auto

ultimately have $\exists c \in \text{pow}(\bigcup (\{\text{one-point compactification of}\}(\text{cofinite}\{\bigcup T\}) - \{\bigcup T\}) \cup T). \bigcup T \in \text{interior}(c, (\{\text{one-point compactification of}\}(\text{cofinite}\{\bigcup T\}) - \{\bigcup T\} \cup T))$.
\( \top T) \cup \top T \setminus (\{\top T\}) \) unfolding IsLocallyT2_def IsLocally_def [OF top] by auto

then obtain \( C \) where \( C : C \subseteq \top (\{\top \text{point compactification of} \top (\{\top T\}) \setminus \{\top T\}) \cup \top T \subseteq \top T \) and \( T_2 : (\{\top \text{point compactification of} \top (\{\top T\}) \setminus \{\top T\}) \cup \top T \subseteq \top T \) unfolding IsLocally_def [OF top]

by auto

have \( \text{sub:Interior}(C, (\{\top \text{point compactification of} \top (\{\top T\}) \setminus \{\top T\}) \cup \top T) \subseteq C \) using topology0.Top_2_L1 unfolding topology0_def by auto

have \( ((\{\top \text{point compactification of} \top (\{\top T\}) \setminus \{\top T\}) \cup \top T) \cup \top T \subseteq \top T \) unfolding RestrictedTo_def [OF sub C(1)] by auto

moreover have \( (\top (\{\top \text{point compactification of} \top (\{\top T\}) \setminus \{\top T\}) \cup \top T) \cup \top T) = (\{\top T\}) \cup \top T \) unfolding RestrictedTo_def [OF T2 pp] by auto

ultimately have \( T_2_2 : (\{\top \text{point compactification of} \top (\{\top T\}) \setminus \{\top T\}) \cup \top T \subseteq \top T \) unfolding RestrictedTo_def [OF sub C(1)]

have \( \text{top2:} (\{\top \text{point compactification of} \top (\{\top T\}) \setminus \{\top T\}) \cup \top T \subseteq \top T \) unfolding topology0.Top_2_L2 unfolding topology0_def by auto

from \( C(2) \) pp have \( \text{pp:} \text{Interior}(C, (\{\top \text{point compactification of} \top (\{\top T\}) \setminus \{\top T\}) \cup \top T) \subseteq \top T \) unfolding topology0.Top_1_L4 unfolding topology0_def by auto

fix \( x \) assume \( x \notin \top T \subseteq \top T \) unfolding topology0_def by auto

with \( p_1 \) have \( \exists U : (\{\top \text{point compactification of} \top (\{\top T\}) \setminus \{\top T\}) \cup \top T \subseteq \top T \) unfolding topology0_Top_2_L2 have intOP: \( \text{interior}(C, (\{\top \text{point compactification of} \top (\{\top T\}) \setminus \{\top T\}) \cup \top T) \subseteq (\{\top T\}) \cup \top T \) unfolding topology0_def by auto

1025
\[ x \in U \cup \bigcup T \in U \cup V = 0 \text{ using } T_2.2 \text{ unfolding isT}_2 \text{ def by auto} \]

then obtain \( U, V \) where \( UV : U \in \left( \left( \left( \text{one-point compactification of} \bigcup T \right) - \{\{\cup T\}\} \right) \cup T \right) \text{ restricted to } \left( \text{Interior}(C, \left( \left( \text{one-point compactification of} \bigcup T \right) - \{\{\cup T\}\} \right) \cup T) \right) \)

\[ V \in \left( \left( \text{one-point compactification of} \bigcup T \right) - \{\{\cup T\}\} \right) \cup T \text{ restricted to } \left( \text{Interior}(C, \left( \left( \text{one-point compactification of} \bigcup T \right) - \{\{\cup T\}\} \right) \cup T) \right) \]

\[ U \neq 0 \text{ using } \bigcup T \in U \cup V = 0 \text{ by auto} \]

from UV(1) obtain \( UC \) where \( UV : U \in \left( \left( \left( \text{one-point compactification of} \bigcup T \right) - \{\{\cup T\}\} \right) \cup T \right) \cap \text{Interior}(C, \left( \left( \text{one-point compactification of} \bigcup T \right) - \{\{\cup T\}\} \right) \cup T) \)

unfolding RestrictedTo_def by auto

with top intOP have \( Uop : U \in \left( \left( \left( \text{one-point compactification of} \bigcup T \right) - \{\{\cup T\}\} \right) \cup T \right) \text{ unfolding IsATopology_def by auto} \)

from UV(2) obtain \( VC \) where \( VC \in \left( \left( \left( \text{one-point compactification of} \bigcup T \right) - \{\{\cup T\}\} \right) \cup T \right) \cap \text{Interior}(C, \left( \left( \text{one-point compactification of} \bigcup T \right) - \{\{\cup T\}\} \right) \cup T) \)

unfolding RestrictedTo_def by auto

with top intOP have \( V \in \left( \left( \left( \text{one-point compactification of} \bigcup T \right) - \{\{\cup T\}\} \right) \cup T \right) \)

unfolding RestrictedTo_def by auto

with UV(3-5) \( Uop \) vicinity_betweenDense[of assms(2-4), of UV] union_ordtopology[of assms(2,4)]

have False unfolding T_def by auto

} then have \( \bigcup \left( \left( \left( \text{one-point compactification of} \bigcup T \right) - \{\{\cup T\}\} \right) \cup T \right) \text{ restricted to } \left( \text{Interior}(C, \left( \left( \text{one-point compactification of} \bigcup T \right) - \{\{\cup T\}\} \right) \cup T) \right) \)

by auto

with p1 have \( \bigcup \left( \left( \left( \text{one-point compactification of} \bigcup T \right) - \{\{\cup T\}\} \right) \cup T \right) \text{ restricted to } \left( \text{Interior}(C, \left( \left( \text{one-point compactification of} \bigcup T \right) - \{\{\cup T\}\} \right) \cup T) \right) = \{\cup T\} \)

by auto

with top2 have \( \{\cup T\} \cap \text{Interior}(C, \left( \left( \text{one-point compactification of} \bigcup T \right) - \{\{\cup T\}\} \right) \cup T) \)

unfolding IsATopology_def by auto

then obtain \( W \) where \( UT : UT \in \left( \left( \left( \text{one-point compactification of} \bigcup T \right) - \{\{\cup T\}\} \right) \cup T \right) \cap \text{Interior}(C, \left( \left( \text{one-point compactification of} \bigcup T \right) - \{\{\cup T\}\} \right) \cup T) \)

unfolding RestrictedTo_def by auto

from this(2) have \( \text{Interior}(C, \left( \left( \text{one-point compactification of} \bigcup T \right) - \{\{\cup T\}\} \right) \cup T) \)

- \( \{\{\cup T\}\} \cup T \text{ using } \text{intOP} \)

top unfolding IsATopology_def by auto

with UT(1) have \( \{\cup T\} \in \left( \left( \left( \text{one-point compactification of} \bigcup T \right) - \{\{\cup T\}\} \right) \cup T \right) \)

- \( \{\{\cup T\}\} \cup T \) by auto

then have \( \{\cup T\} \in T \) 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**: $T \times X \times r$
- **Defines**: $T_{\text{def}} : T \equiv (\text{OrdTopology} \times X \times r)$
- **Assumes**: $\text{IsLinOrder}(X, r) \land X \text{ is dense with respect to } r$
- **Shows**: $\neg ((\{\text{one-point compactification of} (\text{CoFinite} (\bigcup T)) \} - \{\bigcup T\} \cup T) \text{ is locally-}T_2)$
- and $(\{\text{one-point compactification of} (\text{CoFinite} (\bigcup T)) \} - \{\bigcup T\} \cup T) \text{ is anti-}\text{IsHConnected}$
- **Using**: $\text{OPComp_cofinite_dense_order_not_loc_T2[OF assms(2-4)]} \land \text{dense_order_infinite[OF assms(2-4)]} \land \text{union_ordtopology[OF assms(2,4)]}$
- **Unfolding**: $T_{\text{def}}$ 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 $T_2 \implies \text{locally-}T_2 \implies \text{anti-hyperconnected}$; all implications proper.

**Theorem (in topology0) T2_imp_loc_T2:**
- **Assumes**: $T \text{ is } T_2$
- **Shows**: $T \text{ is locally-}T_2$
- **Proof**
  1. Fix $x$ assume $x \in \bigcup T$
  2. Fix $b$ assume $\exists b \in T \forall x \in b$
  3. Then have $(T \text{ restricted to } b) \text{ is } T_2$
  4. Moreover from $b$ have $x \in \text{int}(b)$ by auto
  5. Ultimately have $\exists c \in \text{Pow}(b). x \in \text{int}(c) \land (T \text{ restricted to } c) \text{ is } T_2$
  6. By auto
  7. Then have $\forall b \in T. x \in b \implies (\exists c \in \text{Pow}(b). x \in \text{int}(c) \land T_2)$ by auto
  8. Then show thesis unfolding $\text{IsLocallyT2_def} \land \text{IsLocally_def[OF topSpaceAssum]}$ by auto

If there is a closed singleton, then we can consider a topology that makes this point doble.

**Theorem (in topology0) doble_point_top:**
- **Assumes**: $\{m\} \text{ is closed in } T$
- **Shows**: $(T \cup \{(U - \{m\}) \cup (\bigcup T) \cup W \cup (U, W) \in \{V \in T. m \in V\} \times T\}) \text{ is a topology}$
- **Proof**
  1. 

1027
fix M assume M: M ⊆ T ∪ ((U-{m}) ∪ {U} ∪ W. {U,W} ∈ {V ∈ T. m ∈ V} × T)
let MT = {V ∈ M. V ∈ T}
let Mm = {V ∈ M. V /∈ T}
have unm: M = (MT) ∪ (Mm) by auto
have tt: MT ⊆ T using topSpaceAssum unfolding IsATopology_def by auto
{  
assume Mm = 0
then have Mm = 0 by auto
with unm have M = (MT) by auto
with tt have M ∈ T by auto
then have M ∈ T ∪ ((U-{m}) ∪ {U} ∪ 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 ∈ {V ∈ T. m ∈ V} × T by blast
then obtain U W where U: V = (U-{m}) ∪ {U} ∪ W. U ∈ {m ∈ W ∈ T by auto
let U = {V,W} ∈ T. m ∈ V \ (V-{m}) ∪ {U} ∪ W ∈ Mm
let fU = {fst(B). B ∈ U}
let sU = {snd(B). B ∈ U}
have fU ⊆ sU ∩ T by auto
then have P: ∪ fU ∈ T ∪ sU ∈ T using topSpaceAssum unfolding IsATopology_def by auto
moreover
{  
have {V ∈ T. m ∈ V} × T by auto
then have m ∈ fU by auto
ultimately have s: (fU, sU) ∈ {V ∈ T. m ∈ V} × T by auto
moreover have r: ∀ S. ∀ R ∈ {V ∈ T. m ∈ V} → R ∈ T → (S-{m}) ∪ {U} ∪ R ∈ {V ∈ T. m ∈ V} × T
by auto
ultimately have (fU-{m}) ∪ {U} ∪ sU ∈ (U-{m}) ∪ {U} ∪ W. {U,W} ∈ {V ∈ T. m ∈ V} × T by auto
}  
fix v assume v ∈ Mm
then obtain V where v: v ∈ V ∈ Mm by auto
then have V: Vm ∩ T by auto
with M have V ∈ {V ∈ T. m ∈ V} × T by blast
then obtain U W where U: V = (U-{m}) ∪ {U} ∪ W. U ∈ {m ∈ W ∈ T by auto
with v(1) have v ∈ (U-{m}) ∪ {U} ∪ W by auto
then have v ∈ U-{m} ∨ v = ∪ T ∨ v ∈ W by auto
then have (v ∈ U \ v ≠ m) ∨ v = ∪ T ∨ v ∈ W by auto
moreover from U V have {V ∈ T. m ∈ V} by auto
ultimately have v ∈ ((fU)-(m)) ∪ {U} ∪ sU by auto
}
then have Mm ⊆ ((fU)-(m)) ∪ {U} ∪ sU by blast moreover
{  
fix v assume v : v ∈ ((fU)-(m)) ∪ {U} ∪ sU
{  
assume v = ∪ T

1028
then have \( v \in (U-\{m\}) \cup \{\bigcup T\} \cup W \) by auto 
with \( \langle U, W \rangle \in U \) have \( v \in \bigcup M \) by auto 
} 
moreover 
\{ 
assume \( v \not\in \bigcup TV \cup \bigcup U \) 
with \( v \) have \( v \in ((\bigcup fU)-\{m\}) \) by auto 
then have \( (v \in fU \setminus v \neq m) \) by auto 
then obtain \( W \) where \( (v \in W \setminus \bigcup fU \setminus v \neq m) \) by auto 
then have \( v \in (W-\{m\}) \cup \{\bigcup T\} \) \( W \in fU \) by auto 
then obtain \( B \) where \( \text{fst}(B) = W \) \( B \in U \) \( v \in (W-\{m\}) \cup \{\bigcup T\} \) by blast 
then have \( v \in \bigcup M \) by auto 
} 
ultimately have \( v \in \bigcup M \) by auto 
} 
then have \( ((\bigcup fU)-\{m\}) \cup \{\bigcup T\} \cup \bigcup (\bigcup U) \subseteq \bigcup M \) by auto 
ultimately have \( \bigcup M = ((\bigcup fU)-\{m\}) \cup \{\bigcup T\} \cup (\bigcup (\bigcup U)) \) by auto 
then have \( \bigcup M = ((\bigcup fU)-\{m\}) \cup \{\bigcup T\} \cup ((\bigcup (\bigcup U)) \cup (\bigcup (\bigcup U))) \) using \( \text{ummm} \) by auto 
moreover from \( P \) \( tt \) have \( (\bigcup sU) \cup (\bigcup (\bigcup U)) \in T \) using \( \text{topSpaceAssum} \) 
union_open \( \text{of} \) \( \{\bigcup sU, \bigcup (\bigcup U)\} \) by auto 
with \( s \) have \( (\bigcup fU, (\bigcup sU) \cup (\bigcup (\bigcup U))) \in \{V \in T. \ m \in V\} \times T \) by auto 
then have \( (((\bigcup fU)-\{m\}) \cup \{\bigcup T\}) \cup ((\bigcup (\bigcup U)) \cup (\bigcup (\bigcup U))) \in \{U-\{m\}\} \cup \{\bigcup T\} \cup W \). 
\( \langle U, W \rangle \in \{V \in T. \ m \in V\} \times T \) using \( r \) 
by auto 
ultimately have \( \bigcup M \in \{U-\{m\}\} \cup \{\bigcup T\} \cup W \). \( \langle U, W \rangle \in \{V \in T. \ m \in V\} \times T \) by auto 
then have \( \bigcup M \in \{U-\{m\}\} \cup \{\bigcup T\} \cup W \). \( \langle U, W \rangle \in \{V \in T. \ m \in V\} \times T \) by auto 
} 
ultimately 
have \( \bigcup M \in \{U-\{m\}\} \cup \{\bigcup T\} \cup W \). \( \langle U, W \rangle \in \{V \in T. \ m \in V\} \times T \) by auto 
} 
then have \( \forall M \in \text{Pow}(\bigcup (\{U-\{m\}\} \cup \{\bigcup T\} \cup W \). \( \langle U, W \rangle \in \{V \in T. \ m \in V\} \times T \) \) 
by auto 
moreover 
\{ 
fix \( A, B \) assume \( \text{ass} : A \in T \cup \{U-\{m\}\} \cup \{\bigcup T\} \cup W \). \( \langle U, W \rangle \in \{V \in T. \ m \in V\} \times T \) \( B \in T \cup \{U-\{m\}\} \cup \{\bigcup T\} \cup W \). \( \langle U, W \rangle \in \{V \in T. \ m \in V\} \times T \) 
\} 
assume \( A : A \in T \) 
\{ 
assume \( B \in T \) 
with \( A \) have \( A \cap B \in T \) using \( \text{topSpaceAssum} \) unfolding \( \text{IsATopology_def} \) 
by auto 
} 
moreover 
\{ 
assume \( B \notin T \) 
with \( A \) 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 \( U : U \in T m \in U W \in T B = \{U-\{m\}\} \cup \{\bigcup T\} \cup W \) by auto 
\}
moreover
  from A mem_not_refl have \( \bigcup T \notin A \) by auto
ultimately have \( A \cap B = A \cap ((U \setminus \{m\}) \cup W) \) by auto
then have eq: \( A \cap B = (A \cap (U \setminus \{m\})) \cup (A \cap W) \) by auto
have \( \bigcup T \setminus \{m\} \in T \) using assms unfolding IsClosed_def by auto
with U(1) have 0: \( U \cap (\bigcup T \setminus \{m\}) \in T \) using topSpaceAssum unfolding IsATopology_def
by auto
have \( U \cap (\bigcup T \setminus \{m\}) = U \setminus \{m\} \) using U(1) by auto
with 0 have \( U \setminus \{m\} \in T \) by auto
with A have \( (A \cap (U \setminus \{m\})) \in T \) 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 \( (A \cap (U \setminus \{m\})) \cup (A \cap W) \in T \) using
union_open[OF topSpaceAssum, of \( \{A \cap (U \setminus \{m\}), A \cap W\} \)] by auto
with eq have \( A \cap B \in T \) by auto
moreover
{ assume A \( \notin T \)
  with ass(1) have A: \( A \in \{ (U \setminus \{m\}) \cup (\bigcup T) \cup W, \quad (U, W) \in \{ V \in T. \ m \in V \} \times T \} \) by auto
  { assume B: \( B \in T \)
    from A obtain U W where U: \( U \in T \land U \in T \land A = (U \setminus \{m\}) \cup (\bigcup T) \cup W \) by auto
moreover
from B mem_not_refl have \( \bigcup T \notin B \) by auto
ultimately have \( A \cap B = ((U \setminus \{m\}) \cup W) \cap B \) by auto
then have eq: \( A \cap B = ((U \setminus \{m\}) \cap B) \cup (W \cap B) \) by auto
have \( \bigcup T \setminus \{m\} \in T \) using assms unfolding IsClosed_def by auto
with U(1) have 0: \( U \cap (\bigcup T \setminus \{m\}) \in T \) using topSpaceAssum unfolding IsATopology_def
by auto
have \( U \cap (\bigcup T \setminus \{m\}) = U \setminus \{m\} \) using U(1) by auto
with 0 have \( U \setminus \{m\} \in T \) by auto
with B have \( ((U \setminus \{m\}) \cap B) \in T \) using topSpaceAssum unfolding IsATopology_def
by auto
moreover
from B U(3) have \( W \cap B \in T \) using topSpaceAssum unfolding IsATopology_def
by auto
ultimately have \( ((U \setminus \{m\}) \cap B) \cup (W \cap B) \in T \) using
union_open[OF topSpaceAssum, of \( \{((U \setminus \{m\}) \cap B), (W \cap B)\} \)] by auto
with eq have \( A \cap B \in T \) by auto
moreover
{  
  assume B不属于T  
  with assms(2) have B∈{(U-{m})∪{∪T}∪W. ⟨U,W⟩∈{V∈T. m∈V}×T} by auto  
  moreover then obtain U W where U:U∈Tm∈UW∈TB=(U-{m})∪{∪T}∪W by auto  
  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∩(U-{m}))∪((UA-{m})∩W)∪(WA∩W)∪{∪T} by auto  
  have ∪T-{m}∈T using assms unfolding IsClosed_def by auto  
  with U(1) UA(1) have 0:∪∩(∪T-{m})⊂TUA∩(∪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}⊂TUA-{m}∈T 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⊂TWA⊂T using topSpaceAssum unfolding IsATopology_def  
  by auto  
  from TT(2,3) have ((UA-{m})∩W)∪(WA∩W)⊂T using union_open[OF topSpaceAssum,  
  of {(UA-{m})∩W,WA∩W}] by auto  
  with TT(1) have (WA∩(U-{m}))∪(((UA-{m})∩W)∪(WA∩W))⊂T using union_open[OF topSpaceAssum,  
  of {WA∩(U-{m}),((UA-{m})∩W)∪(WA∩W)}] by auto  
  ultimately have A∩B=UA∩U-{m})⊂(U∪T)∪((WA∩(U-{m}))∪(((UA-{m})∩W)∪(WA∩W)))  
  (WA∩(U-{m}))∪(((UA-{m})∩W)∪(WA∩W))⊂T UA∩U∈{V∈T. m∈V} using eq by auto  
  then have ∃W∈T. A∩B=(UA∩U-{m})∪(U∪T)∪W UA∩U∈{V∈T. m∈V} by auto  
  then have A∩B=((U-{m})∪{∪T}∪W. ⟨U,W⟩∈{V∈T. m∈V}×T} by auto  
  }  
  ultimately have A∩B⊂T {U-{m})∪{∪T}∪W. ⟨U,W⟩∈{V∈T. m∈V}×T} by auto  
  ultimately have A∩B⊂T (U-{m})∪{∪T}∪W. ⟨U,W⟩∈{V∈T. m∈V}×T} by auto  
  then have ∀A∈∪{U-{m})∪{∪T}∪W. ⟨U,W⟩∈{V∈T. m∈V}×T). ∀B⊂T (U-{m})∪{∪T}∪W.  
  ⟨U,W⟩∈{V∈T. m∈V}×T}.  
  A∩B⊂T (U-{m})∪{∪T}∪W. ⟨U,W⟩∈{V∈T. m∈V}×T} by blast  

1031
ultimately show thesis unfolding IsATopology_def by auto
def

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 (T\cup(U\{m\})\cup(U\cup(T\cup(W. \langle U,W\rangle\in\{\forall V.T. m\in V\}\times T\}))=\bigcup T \cup\{\bigcup T\}
proof
  \{ 
  fix x assume x\in\bigcup (T\cup(U\{m\})\cup(U\cup(T\cup(W. \langle U,W\rangle\in\{\forall V.T. m\in V\}\times T\}))
  then obtain R where x:x\in R \in T by blast 
  \} 
  moreover 
  \{ 
  assume R\notin T with x(1) have x\in\bigcup T by auto 
  \} 
  ultimately have x\in\bigcup T \cup\{\bigcup T\} by auto 
  \} 
  then show \bigcup (T\cup(U\{m\})\cup(U\cup(T\cup(W. \langle U,W\rangle\in\{\forall V.T. m\in V\}\times T\}))\subseteq\bigcup T \cup\{\bigcup T\}
by auto 
  \} 
  \{ 
  fix x assume x\in\bigcup T \cup\{\bigcup T\}
  then have dis:x\in\bigcup T \cap x=\bigcup T by auto 
  \} 
  \{ 
  assume x\in\bigcup T
  then have x\in\bigcup (T\cup(U\{m\})\cup(U\cup(T\cup(W. \langle U,W\rangle\in\{\forall V.T. m\in V\}\times T\})) by auto 
  \} 
  moreover 
  \{ 
  assume x\notin\bigcup T with dis have x=\bigcup T by auto 
  \} 
  moreover from assms have \bigcup T\{m\}\in\bigcup T unfolding IsClosed_def
by auto 
  \} 
  moreover have 0\in T using empty_open topSpaceAssum by auto
  ultimately have x\in (U\cap(T\{m\})\cup(T)\cup0 \ (T\cap{m})\cup(T)\cup0\in\{U\cap{T\{m\}}\cup(T)\times W. \langle U,W\rangle\in\{\forall V.T. m\in V\}\times T\})
  using union_open[OF topSpaceAssum] by auto
  then have x\in (U\cap(T\{m\})\cup(T)\cup0 \ (T\cap{m})\cup(T)\cup0\in\{U\cap{T\{m\}}\cup(T)\times W. \langle U,W\rangle\in\{\forall V.T. m\in V\}\times T\}) 
  by auto 
  \} 
  \{ 
  \} 

1032
ultimately have \( x \in \bigcup (T \cup ((U \cup \{m\}) \cup (U \cup W)) \setminus \langle U, W \rangle \in \{V \in T, m \in V \} \times T \} \) by auto
}
then show \( \bigcup T \cup \{U\} \subseteq \bigcup (T \cup ((U \cup \{m\}) \cup (U \cup W)) \setminus \langle U, W \rangle \in \{V \in T, m \in V \} \times T \} \) 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 \( \bigcup \{V \cup \{m\}\} \cup \{U \cup T\} \cup \{W\} \cup \{m\} \subseteq \{V \cup \{m\}\} \cup \{U \cup T\} \cup \{W\} \cup \{m\} \) restricted to \( T = T \)
and \( \bigcup \{U \cup \{m\}\} \cup \{V \cup T\} \cup \{m\} \subseteq \{V \cup \{m\}\} \cup \{U \cup T\} \cup \{m\} \) restricted to \( T = T \)
proof-
have \( N \vdash T \notin \bigcup T \) using mem_not_refl by auto
{ fix \( x \) assume \( x \in \bigcup (T \cup ((U \cup \{m\}) \cup (U \cup T)) \setminus \langle U, W \rangle \in \{V \in T, m \in V \} \times T \} \) restricted to \( T = T \)
then obtain \( U \) where \( U : U \in \bigcup (T \cup ((U \cup \{m\}) \cup (U \cup T)) \setminus \langle U, W \rangle \in \{V \in T, m \in V \} \times T \} \) x = \( T \cap U \)
unfolding RestrictedTo_def by blast
{ assume \( U \notin T \)
with \( U(1) \) have \( U \in \{V \cup \{m\}\} \cup \{U \cup T\} \cup \{m\} \in \{V \cup \{m\}\} \cup \{U \cup T\} \cup \{m\} \times T \} \) by auto
then obtain \( V \) and \( W \) where \( V \cap W : U = \{V \cup \{m\}\} \cup \{U \cup T\} \cup \{m\} \in \{V \cup \{m\}\} \cup \{U \cup T\} \cup \{m\} \times T \} \) by auto
with \( N \cup U(2) \) have \( x : x = \{V \cup \{m\}\} \cup W \) by auto
have \( T \setminus \{m\} \notin T \) using assms unfolding IsClosed_def by auto
then have \( V \setminus \{T \setminus \{m\}\} \in T \) using \( V \in \{V \cup \{m\}\} \cup \{U \cup T\} \cup \{m\} \times T \} \) unfolding topSpaceAssum by auto
moreover
have \( V \cup \{m\} = V \cap (U \cup \{m\}) \cup \{U \cup T\} \cup \{W\} \cup \{m\} \cup \{U \cup T\} \cup \{m\} \times T \} \) by auto ultimately
have \( V \cup \{m\} \notin T \) by auto
with \( V \in \{V \cup \{m\}\} \cup \{U \cup T\} \cup \{m\} \times T \} \) using union_open[OF topSpaceAssum, of \( \{V \cup \{m\}\}, W \} \)
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 \( \bigcup (T \cup ((U \cup \{m\}) \cup (U \cup T)) \setminus \langle U, W \rangle \in \{V \in T, m \in V \} \times T \} \) restricted to \( T \subseteq T \)
by auto
moreover
{ fix \( x \) assume \( x : x \in T \)
then have \( x \in \bigcup (T \cup ((U \cup \{m\}) \cup (U \cup T)) \setminus \langle U, W \rangle \in \{V \in T, m \in V \} \times T \} \) by auto
morever
from x have \( \bigcup T \cap x = x \) by auto ultimately
have \( \exists m \in T \{ (U \setminus \{m\}) \cup \{ \bigcup T \} \cup W \}. \{ U, W \} \in \{ \bigcup T \} \times T \} \) by blast
then have \( x \in (T \cup \{ (U \setminus \{m\}) \cup (\bigcup T) \cup W \}. \{ U, W \} \in \{ \bigcup T \} \times T \} \) {restricted
to} \( \bigcup T \) unfolding RestrictedTo_def
by auto

ultimately show \( (T \cup \{ (U \setminus \{m\}) \cup \{ \bigcup T \} \cup W \}. \{ U, W \} \in \{ \bigcup T \} \times T \} \) {restricted
to} \( \bigcup T = T \) by auto
have \( P : \bigcup T \in T \) using topSpaceAssum unfolding IsATopology_def by auto
then show \( \bigcup T \in (T \cup \{ (U \setminus \{m\}) \cup \{ \bigcup T \} \cup W \}. \{ U, W \} \in \{ \bigcup T \} \times T \} \) 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 \{ \{ \text{is closed in } T \} \}
shows \( \neg ((T \cup \{ (U \setminus \{m\}) \cup \{ \bigcup T \} \cup W \}. \{ U, W \} \in \{ \bigcup T \} \times T \} \) {is \( T_2 \})
proof
assume \( \mathfrak{a} : (T \cup \{ (U \setminus \{m\}) \cup \{ \bigcup T \} \cup W \}. \{ U, W \} \in \{ \bigcup T \} \times T \} \) {is \( T_2 \})
then have \( \mathfrak{b} : \bigcup T \in (T \cup \{ (U \setminus \{m\}) \cup \{ \bigcup T \} \cup W \}. \{ U, W \} \in \{ \bigcup T \} \times T \} \) \( \bigcup T = \bigcup T \) by auto
unfolding 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 \( \mathfrak{a} \) have \( \forall x \ y. \ x \in \bigcup T \cup \{ \bigcup T \} \wedge y \in \bigcup T \cup \{ \bigcup T \} \setminus { x \neq y } \rightarrow (\exists m \in (T \cup \{ (U \setminus \{m\}) \cup \{ \bigcup T \} \cup W \}. \{ U, W \} \in \{ \bigcup T \} \times T \}) \ \bigcup T \in \{ \bigcup T \} \times T \}) \). \( x \in \bigcup T \wedge y \in \bigcup T \wedge x \neq y \)
unfolding isT2_def by auto
moreover
from assms(2) have \( m \in \bigcup T \cup \{ \bigcup T \} \) unfolding IsClosed_def by auto
moreover
have \( \bigcup T \in \bigcup T \) by auto ultimately
have \( \exists m \in (T \cup \{ (U \setminus \{m\}) \cup \{ \bigcup T \} \cup W \}. \{ U, W \} \in \{ \bigcup T \} \times T \} \). \( \exists m \in (T \cup \{ (U \setminus \{m\}) \cup \{ \bigcup T \} \cup W \}. \{ U, W \} \in \{ \bigcup T \} \times T \}) \). \( m \in \bigcup T \cup \{ \bigcup T \} \) by auto
then obtain \( U \ W \) where \( (U \in (T \cup \{ (U \setminus \{m\}) \cup \{ \bigcup T \} \cup W \}. \{ U, W \} \in \{ \bigcup T \} \times T \}) \)
\( \forall y \in \bigcup T \cup \{ \bigcup T \} \cup W \). \( y \in \bigcup T \cup \{ \bigcup T \} \) \( m \in \bigcup T \cup \{ \bigcup T \} \) using tot1 by blast
then have \( \bigcup T \notin U \) by auto
with UV(1) have \( P : \bigcup T \in T \) by auto
\{ assume \( U \in T \)
then have \( \forall U \in \bigcup T \) by auto
with UV(4) have \( \bigcup T \in \bigcup T \) using tot1 by auto
then have False using mem_not_refl by auto
\}

1034
with \( U V(2) \) have \( \forall U \in \{ (U, m) \cup \{ U \} \cup W. (U, W) \in \{ U \in T. m \in V \} \times T \} \) by auto 
then obtain \( U W \) where \( V. U = (U, m) \cup \{ U \} \cup W U \cup T m \in U W \in T \) by auto 
from \( V(2, 3) \) \( P \) have \( \text{int} U \cup T m \in U W \) using \( U V(3) \) topSpaceAssum unfolding IsATopology_def by auto 
have \( \{ U \cup \{ (U, m) \cup \{ U \} \cup W \in V \} \cup \{ U \in T. m \in V \} \times T \} \) by auto 
then have \( \{ U \cup \{ (U, m) \cup \{ U \} \cup W \} = 0 \) using \( U V(5) \) by auto 
with int \( (2) \) have \( U \cup \{ m \} \) by auto 
with int \( (1) \) assms \( (1) \) show False by auto 
qed

This topology is locally-\( T_2 \).

Theorem (in topology0) loc_T2_imp_T2_counter_2:
- assumes \( \{ m \} \notin T m \in U T \{ \text{is T}_2 \} \)
- shows \( (U \cup \{ (U, m) \cup U \{ U \} \cup W. (U, W) \in \{ U \in T. m \in V \} \times T \}) \) {is locally-T_2}

proof-
- from assms \( (3) \) have \( T \{ \text{is T}_1 \} \) using T2_is_T1 by auto 
- with assms \( (2) \) have mc: \( \{ m \} \) {is closed in} \( T \) using T1_iff_singleton_closed by auto 
- have \( \forall U \in T \in T \) using mem_not_refl by auto 
- have res: \( (U \cup \{ (U, m) \cup U \{ U \} \cup W. (U, W) \in \{ U \in T. m \in V \} \times T \}) \) {restricted to} \( \forall U T \) 
  and \( P. U \in T \) and \( Q. U \in (U \cup \{ (U, m) \cup U \{ U \} \cup W. (U, W) \in \{ U \in T. m \in V \} \times T \}) \) using openSubspace_double_point mc topSpaceAssum unfolding IsATopology_def by auto 
  \{ 
    fix A assume ass: \( A \in U \) \cup \{ U \} \times T 
    \{ 
      assume A \( \neq U \) 
      with ass have \( A \in U T \) by auto 
      with Q res assms \( (3) \) have \( \forall U \in T (U \cup \{ (U, m) \cup U \{ U \} \cup W. (U, W) \in \{ U \in T. m \in V \} \times T \}) \) \( \forall U \in T \) 
      \( \forall U \in T \) 
      then have \( \exists U \in (U \cup \{ (U, m) \cup U \{ U \} \cup W. (U, W) \in \{ U \in T. m \in V \} \times T \}) \) \( \forall U \in T \) 
      \( A E \{ (U \cup \{ (U, m) \cup U \{ U \} \cup W. (U, W) \in \{ U \in T. m \in V \} \times T \}) \} \{ \text{restricted to} \} \) \( \forall U \in T \) 
      by blast 
    \} 
  \} 
  moreover 
  \{ 
    assume A: \( A = U \) 
    have \( \forall U \in T \) \( \forall U \in T \) \( \forall U \in T \) \( \forall U \in T \) \( \forall U \in T \) 
    using assms \( (2) \) empty_open[of topSpaceAssum] unfolding IsClosed_def by P auto 
    then have \( (U \cup \{ (U, m) \}) \cup (U \cup \{ (U, m) \}) \cup (U \cup \{ (U, m) \}) \cup (U \cup \{ (U, m) \}) \cup (U \cup \{ (U, m) \}) \) \( \forall U \in T \) 
    by auto 
    then have opp: \( (U \cup \{ (U, m) \}) \cup (U \cup \{ (U, m) \}) \cup (U \cup \{ (U, m) \}) \cup (U \cup \{ (U, m) \}) \cup (U \cup \{ (U, m) \}) \) \( \forall U \in T \) 
    by auto 
    \{ 
      fix A A2 assume points: \( A_1 \in (U \cup \{ (U, m) \}) \cup (U \cup \{ (U, m) \}) \cup (U \cup \{ (U, m) \}) \cup (U \cup \{ (U, m) \}) \cup (U \cup \{ (U, m) \}) \) \( \forall U \in T \) 
      from points \( (1, 2) \) have notm: \( A_1 \neq m A_2 \neq m \) using assms \( (2) \) unfolding IsClosed_def using mem_not_refl by auto 
    \} 
  \}

1035
\{ 
  assume or:A1∈∪TA2∈∪T 
  with points(3) assms(3) obtain U V where UV:\∈UV∈TA1∈UA2∈V 
  U\\cap V=0 unfolding isT2_def by blast 
  from UV(1,2) have \((U\{U\cap T-m\})\cup (U\cup T)\cup U.W. (U,W)\in (V∈T.m∈V)\times T\) \{restricted to\} \((U\{U\cap T-m\})\cup (U\cup T)\cup U.W. (U,W)\in (V∈T.m∈V)\times T\) \{restricted to\} \((U\{U\cap T-m\})\cup (U\cup T)\) unfolding RestrictedTo_def by auto moreover 
  then have \(U\{U\cup T-m\}=U\{U\{U\cup T-m\}\cup (U\cup T)\}\cap (U\{U\cup T-m\})=U\{U\{U\cup T-m\}\cup (U\cup T)\}\) using UV(1,2) mem_not_refl[of (TU)] by auto 
  ultimately have opUV:U\{U\cap T-m\}\cup (U\cup T)\cup U.W. (U,W)\in (V∈T.m∈V)\times T\} \{restricted to\} \((U\{U\cap T-m\})\cup (U\cup T)\) by auto 
  moreover have \(U\{U\cup T-m\}\cap (U\{U\cup T-m\})=0\) using UV(5) by auto moreover 
  from UV(3) or(1) notm(1) have A1∈U\{U\cup T-m\} by auto moreover 
  from UV(4) or(2) notm(2) have A2∈U\{U\cup T-m\} by auto ultimately 
  have \(∃V. V∈(U\{U\cap T-m\})\cup (U\cup T)\cup W. (U,W)\in (V∈T.m∈V)\times T\) \{restricted to\} \((U\{U\cap T-m\})\cup (U\cup T)\) ∧ A1∈U\{U\cup T-m\}\cap (U\{U\cup T-m\})\cap W=0\) using exI[where x=U\{U\cap T-m\} and P=λW. W∈(U\{U\cap T-m\})\cup (U\cup T)\cup W. (U,W)\in (V∈T.m∈V)\times T\} \{restricted to\} \((U\{U\cap T-m\})\cup (U\cup T)\) ∧ A1∈U\{U\cup T-m\}\cap (U\{U\cup T-m\})\cap W=0\] using opUV(2) by auto 
  then have \(∃U. U∈(U\{U\cap T-m\})\cup (U\cup T)\cup W. (U,W)\in (V∈T.m∈V)\times T\} \{restricted to\} \((U\{U\cap T-m\})\cup (U\cup T)\) ∧ (∃V. V∈(U\{U\cap T-m\})\cup (U\cup T)\cup W. (U,W)\in (V∈T.m∈V)\times T\} \{restricted to\} \((U\{U\cap T-m\})\cup (U\cup T)\) ∧ A1∈U\{U\cup T-m\}\cap (U\{U\cup T-m\})\cap W=0\] using opUV(1) by auto 
  then have \(∃U. U∈(U\{U\cap T-m\})\cup (U\cup T)\cup W. (U,W)\in (V∈T.m∈V)\times T\} \{restricted to\} \((U\{U\cap T-m\})\cup (U\cup T)\). (∃V. V∈(U\{U\cap T-m\})\cup (U\cup T)\cup W. (U,W)\in (V∈T.m∈V)\times T\} \{restricted to\} \((U\{U\cap T-m\})\cup (U\cup T)\) by blast 
  then have \(∃U. U∈(U\{U\cap T-m\})\cup (U\cup T)\cup W. (U,W)\in (V∈T.m∈V)\times T\} \{restricted to\} \((U\{U\cap T-m\})\cup (U\cup T)\). (∃V. V∈(U\{U\cap T-m\})\cup (U\cup T)\cup W. (U,W)\in (V∈T.m∈V)\times T\} \{restricted to\} \((U\{U\cap T-m\})\cup (U\cup T)\). A1∈U\{U\cup T-m\}\cap (U\{U\cup T-m\})\cap W=0\) by blast 
\}

moreover 
{ 
  assume A1∉∪T 
  then have ig:A1=∪T using points(1) by auto 
  { 
    assume A2∉∪T 
    then have A2=∪T using points(2) by auto 
  } 
}
with points(3) ig have False by auto
}
then have igA2:A2∈∪T by auto moreover
have m∈∪T using assms(2) unfolding IsClosed_def by auto
moreover note notm(2) assms(3) ultimately obtain U V where
UV:U∈TV∈T
m∈UA2∈VV:V=0 unfolding IsT2_def by blast
from UV(1,3) have U∈{U∈T. m∈W} by auto moreover
have 0∈T using empty_open topSpaceAssum by auto ultimately
have (U-{m})∪(∪T)∈(U-{m})∪(∪T)∪W. (U,W)∈(V∈T. m∈V)×T by auto
then have Uop:(U-{m})∪(∪T)∈(T U.(U-{m})∪(∪T)∪W. (U,W)∈(V∈T. m∈V)×T)

by auto
from UV(2) have Vop:V∈(T U.(U-{m})∪(∪T)∪W. (U,W)∈(V∈T. m∈V)×T)}{restricted
then have VV:V∈(T U.(U-{m})∪(∪T)∪W. (U,W)∈(V∈T. m∈V)×T)}{restricted
to}((∪T-{m})∪(∪T)) unfolding RestrictedTo_def
using Vop by blast moreover
from sub(2) have (U-{m})∪(∪T))=(((∪T-{m})∪(∪T))∩((∪T-{m})∪(∪T))
by auto
then have UU:((U-{m})∪(∪T))∈(T U.(U-{m})∪(∪T)∪W. (U,W)∈(V∈T. m∈V)×T)}{restricted
to}((∪T-{m})∪(∪T)) unfolding RestrictedTo_def
using Uop by blast moreover
from UV(2) have ((U-{m})∪(∪T))∩W=(U-{m})∩W using mem_not_refl
by auto
then have ((U-{m})∪(∪T))∩W=0 using UV(5) by auto
with UV(4) VV ig igA2 have ∃V∈(T U.(U-{m})∪(∪T)∪W. (U,W)∈(V∈T. m∈V)×T)}{restricted
to}((∪T-{m})∪(∪T))∧(∃V∈(T (((U-{m})∪(∪T)∪W. (U,W)∈(V∈T. m∈V)×T)}){restricted
to}((∪T-{m})∪(∪T))∧(∃V∈(T (((U-{m})∪(∪T)∪W. (U,W)∈(V∈T. m∈V)×T)}){restricted
A1∈UA2∈V\A∩W=0) using exI[where x=(((U-{m})∪(∪T)) and
P=λU. U∈(T U.(U-{m})∪(∪T)∪W. (U,W)∈(V∈T. m∈V)×T)}{restricted to}((∪T-{m})∪(∪T))∧(∃V∈(T (((U-{m})∪(∪T)∪W. (U,W)∈(V∈T. m∈V)×T)}){restricted
to}((∪T-{m})∪(∪T)). (∃V∈(T (((U-{m})∪(∪T)∪W. (U,W)∈(V∈T. m∈V)×T)}){restricted
A1∈UA2∈V\A∩W=0) by auto
then have ∃V∈(T (((U-{m})∪(∪T)∪W. (U,W)∈(V∈T. m∈V)×T)}){restricted
to}((∪T-{m})∪(∪T)). (∃V∈(T (((U-{m})∪(∪T)∪W. (U,W)∈(V∈T. m∈V)×T)}){restricted
to}((∪T-{m})∪(∪T)).
A1∈UA2∈V\A∩W=0) by blast
}
moreover
{ assume A2∉∪T
then have ig:A2∉∪T using points(2) by auto
}
assume A1€∪_T
then have A1=∪_T using points(1) by auto
with points(3) ig have False by auto

} then have igA2:A1€∪_T by auto moreover
have m€∪_T using assms(2) unfolding IsClosed_def by auto
moreover note notm(1) assms(3) ultimately obtain U V where

UV:U€TV€T

m€UA1€VV∴V=0 unfolding IsT2_def by blast
from UV(1,3) have U€{W€T. m€W} by auto moreover
have 0€T using empty_open topSpaceAssum by auto ultimately
have (U{-m})∪(∪_T)e((U{-m})∪(∪_T)∩U. (U,W)€(V€T. m€V)×T) by auto
then have Uop:(U{-m})∪(∪_T)e(T ∪((U{-m})∪(∪_T)∩U. (U,W)€(V€T. m€V)×T))
by auto
from UV(2) have Vop:V€(T ∪((U{-m})∪(∪_T)∩U. (U,W)€(V€T. m€V)×T))
by auto
from UV(1-3,5) have sub:V€((∪_T{-m})∪(∪_T) ∪((U{-m})∪(∪_T)∩U. (U,W)€(V€T. m€V)×T))
by auto
from sub(1) have V=(((∪_T{-m})∪(∪_T))∩V by auto
then have UV:V€(T ∪((U{-m})∪(∪_T)∩U. (U,W)€(V€T. m€V)×T)) restricted
to{((∪_T{-m})∪(∪_T)) unfolding RestrictedTo_def
using Vop by blast moreover
from sub(2) have ((U{-m})∪(∪_T))=((∪_T{-m})∪(∪_T))∩((U{-m})∪(∪_T))
by auto
then have UV:((U{-m})∪(∪_T))e(T ∪((U{-m})∪(∪_T)∩U. (U,W)€(V€T. m€V)×T)) restricted
to{((∪_T{-m})∪(∪_T)) unfolding RestrictedTo_def
using Uop by blast moreover
from UV(2) have VN=V∩((U{-m})∪(∪_T))=VN using mem_not_refl
by auto
then have VN=V∩((U{-m})∪(∪_T))=0 using UV(5) by auto
with UV UV(4) ig igA2 have ∃U€T ∪((U{-m})∪(∪_T)∩U. (U,W)€(V€T. m€V)×T)) restricted
to{((∪_T{-m})∪(∪_T)).
A1€UA2€UV∀V=0 by auto
with VV igA2 have ∃U. U€T ∪((U{-m})∪(∪_T)∩U. (U,W)€(V€T. m€V)×T)) restricted
to{((∪_T{-m})∪(∪_T)) ∧ (∃V€T ∪((U{-m})∪(∪_T)∩U. (U,W)€(V€T. m€V)×T)) restricted
to{((∪_T{-m})∪(∪_T))}.
A1€UA2€UV∀V=0 using exI[where x=V and P=U]. U€T ∪((U{-m})∪(∪_T)∩U. (U,W)€(V€T. m€V)×T)) restricted
to{((∪_T{-m})∪(∪_T)) ∧ (∃V€T ∪((U{-m})∪(∪_T)∩U. (U,W)€(V€T. m€V)×T)) restricted
to{((∪_T{-m})∪(∪_T))}.
A1€UA2€UV∀V=0) by auto
then have ∃U€T ∪((U{-m})∪(∪_T)∩U. (U,W)€(V€T. m€V)×T)) restricted
to{((∪_T{-m})∪(∪_T))}. (∃V€T ∪((U{-m})∪(∪_T)∩U. (U,W)€(V€T. m€V)×T)) restricted
to{((∪_T{-m})∪(∪_T))}.
A1€UA2€UV∀V=0) by blast
}
ultimately have ∃U€T ∪((U{-m})∪(∪_T)∩U. (U,W)€(V€T. m€V)×T)) restricted
to{((∪_T{-m})∪(∪_T))}. (∃V€T ∪((U{-m})∪(∪_T)∩U. (U,W)€(V€T. m€V)×T)) restricted
to{((∪_T{-m})∪(∪_T))}.

1038
\begin{verbatim}
A1∈U∧A2∈V∧U∩V=0 by blast

}\}
then have ∀A1∈(∪T-{m})∪(∪T). ∀A2∈(∪T-{m})∪(∪T). A1≠A2 →
(∃U∈(T ∪{(U-{m})∪(∪T})∪W . (U,W)∈{V∈T . m∈V}×T)).
(∃V∈(T ∪{(U-{m})∪(∪T})∪W. (U,W)∈{V∈T . m∈V}×T)).
A1∪A2∈(U∧V=0) by auto moreover
have \(∪((T ∪{(U-{m})∪(∪T})∪W . (U,W)∈{V∈T . m∈V}×T))\) restricted to\((∪T-{m})∪(∪T))\)
unfolding RestrictedTo_def by auto
then have \(∪((T ∪{(U-{m})∪(∪T})∪W . (U,W)∈{V∈T . m∈V}×T))\) restricted to\((∪T-{m})∪(∪T))\)=\(∪(T ∪{(U-{m})∪(∪T})∪W. (U,W)∈{V∈T . m∈V}×T))\)∩\((∪T-{m})∪(∪T))\)
using union_doublepoint_top mc by auto
ultimately have ∀A1∈(∪(T ∪{(U-{m})∪(∪T})∪W . (U,W)∈{V∈T . m∈V}×T))\) restricted to\((∪T-{m})∪(∪T))\). A1≠A2 → (∃U∈(T ∪{(U-{m})∪(∪T})∪W . (U,W)∈{V∈T . m∈V}×T)).
A1∪A2∈(U∧V=0) by auto
then have \(∪((T ∪{(U-{m})∪(∪T})∪W. (U,W)∈{V∈T . m∈V}×T))\) restricted to\((∪T-{m})∪(∪T))\) is \(T_2\) unfolding isT2_def by auto
by force
with opp A have \(∃Z∈(TU∪{(U-{m})∪(∪T})∪W . (U,W)∈{V∈T . m∈V}×T))\).
A∈Z∧((TU∪{(U-{m})∪(∪T})∪W. (U,W)∈{V∈T . m∈V}×T)) restricted to\(Z\) is \(T_2\)
by blast
}\}
ultimately
have \(∃Z∈(TU∪{(U-{m})∪(∪T})∪W . (U,W)∈{V∈T . m∈V}×T))\).
A∈Z∧((TU∪{(U-{m})∪(∪T})∪W. (U,W)∈{V∈T . m∈V}×T)) restricted to\(Z\) is \(T_2\)
by blast
\}
then have ∀A∈(TU∪{(U-{m})∪(∪T})∪W . (U,W)∈{V∈T . m∈V}×T)). \(∃Z∈T ∪\{U - \{m\} ∪ \{∪T\} ∪ W . (U,W)∈\{V∈T . m∈V\}×T\}).
A ∈ Z ∧ ((T ∪ {U - \{m\} ∪ \{∪T\} ∪ W . (U,W)∈\{V∈T . m∈V\}×T\}) restricted to\(Z\) is \(T_2\)
using union_doublepoint_top mc by auto
with topology0.loc_T2 show (T ∪ {U - \{m\} ∪ \{∪T\} ∪ W . (U,W)∈\{V∈T . m∈V\}×T\}) is locally-\(T_2\)
unfolding topology0_def using double_point_top mc by auto
qed
\end{verbatim}

There can be considered many more local properties, which; as happens with locally-\(T_2\); can distinguish between spaces other properties cannot.
end
73 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.

73.1 More anti-properties

In this section we study more anti-properties.

73.2 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⊆A ∀M∈Pow(Pow(A)). B⊆∪M → (∃N∈FinPow(M). B⊆∪N)
unfolding IsCompact_def by auto
from B(1) have ∪{x}. x∈B∈Pow(Pow(A)) B⊆∪{x}. x∈B by auto
with B(2) have ∃N∈FinPow({{x}. x∈B}). B⊆∪N by auto
then obtain N where N∈FinPow({{x}. x∈B}) B⊆∪N by auto
then have Finite(N) N⊆{x}. x∈B∈Pow(B)=Pow(A){restricted to}B unfolding RestrictedTo_def by auto
then have B⊆∪N Finite(∪N) using Finite_Union[of N] by auto
then show Finite(B) using subset_Finite by auto
qed

theorem pow_anti_compact:
shows Pow(A){is anti-compact}
proof-
{ fix B assume as: B⊆∪Pow(A) (∪(Pow(A){restricted to}B)){is compact in}(Pow(A){restricted to}B)
  then have sub:B⊆A by auto
  then have Pow(B)=Pow(A){restricted to}B unfolding RestrictedTo_def by blast
  with as(2) have (∪Pow(B)){is compact in}Pow(B) by auto
  then have B{is compact in}Pow(B) by auto
  then have Finite(B) using pow_compact_imp_finite by auto
  then have B{is in the spectrum of}(∪T. (∪T){is compact in}T) using compact_spectrum by auto
} then show thesis unfolding IsAntiComp_def antiProperty_def by auto

1040
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.

### 73.3 Structural results

We first differentiate the spectrum of the lindelöf property depending on some axiom of choice.

**lemma lindeloef_spec1:**

assumes \( \{ \text{the axiom of} \ \text{nat} \ \text{choice holds for subsets}(\Pow(\text{nat})) \)

shows \( (A \ \text{is in the spectrum of} \ \{(\bigcup T)\text{ lindelöf in} T\}) \leftrightarrow (A \lessdot \text{nat}) \)


**lemma lindeloef_spec2:**

assumes \( \neg \{ \text{the axiom of} \ \text{nat} \ \text{choice holds for subsets}(\Pow(\text{nat})) \)

shows \( (A \ \text{is in the spectrum of} \ \{(\bigcup T)\text{ lindelöf in} T\}) \leftrightarrow \text{Finite}(A) \)

**proof**

assume \( \text{Finite}(A) \)

then have \( A:A \text{ is in the spectrum of} \ \{(\bigcup T)\text{ compact in} T\} \)

using `compact_spectrum` by auto

have \( s: \text{nat} \lessdot \text{csucc}(\text{nat}) \) using `le_imp_lesspoll[OF Card_csucc[OF Ord_nat]]`

`lt_csucc[OF Ord_nat] le_iff` by auto

{ fix \( T \) assume \( T\text{ is a topology} \ \{(\bigcup T)\text{ compact in} T\} \)

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 \( (\bigcup T)\text{ is lindelöf in} T \) unfolding `IsLindeloef_def` by auto

} then have \( \forall T. T\text{ is a topology} \ ( (\bigcup T)\text{ is compact in} T) \rightarrow ((\bigcup T)\text{ is lindelöf in} T) \) by auto

with \( A \) show \( A \ \text{is in the spectrum of} \ \{(\bigcup T)\text{ lindelöf in} T\} \)

using `P_imp_Q_spec_inv`

where \( Q:\lambda T. ((\bigcup T)\text{ is compact in} T) \text{ and } P:\lambda T. ((\bigcup T)\text{ is lindelöf in} T) \) by auto

next

assume \( A:A \ \text{is in the spectrum of} \ \{(\bigcup T)\text{ lindelöf in} T\} \)

then have \( \forall T. T\text{ is a topology} \wedge \bigcup T \lessdot A \rightarrow ((\bigcup T)\text{ is compact of cardinal} \ \text{csucc}(\text{nat})(\text{in} T) \) using `Spec_def`
unfolding IsLindeloef_def by auto
then have \(A\) {is compact of cardinal} \(\text{csucc(nat)}\) \{in\} \(\text{Pow}(A)\) using Pow_is_top[of \(A\)] by auto
then have \(\forall M \in \text{Pow}(\text{Pow}(A)) \cdot A \subseteq M \longrightarrow (\exists N \in \text{Pow}(M) \cdot A \subseteq N \land N < \text{csucc(nat)})\) unfolding IsCompactOfCard_def by auto
moreover have \(\{\{x\} \cdot x \in A\} \in \text{Pow}(\text{Pow}(A))\) by auto
moreover have \(A = \bigcup \{\{x\} \cdot x \in A\}\) using FinPow_def by auto
ultimately have \(\exists N \subseteq \{\{x\} \cdot x \in A\} \cdot N < \text{csucc(nat)} \land A \subseteq N\) using FinPow_def by auto
then obtain \(N\) where \(N \subseteq \{\{x\} \cdot x \in A\}\) \(A \subseteq N\) \(N < \text{csucc(nat)}\) using apply_equality by auto
then have \(A \leq N\) using lepoll_def by auto
with \(N < \text{csucc(nat)}\) have \(A < \text{csucc(nat)}\) using lesspoll_trans1 by auto
then have \(A < \text{nat} \lor A \approx \text{nat}\) 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 \(\text{Pow}(\text{nat})\{\text{is a topology}\}\) using Pow_is_top by auto
moreover have \(\bigcup \text{Pow}(\text{nat}) = \text{nat}\) by auto
then have \(\bigcup \text{Pow}(\text{nat}) \approx \text{nat}\) using eqpoll_refl by auto
ultimately have \(\text{nat} \{\text{is compact of cardinal}\} \text{csucc(nat)}\{\text{in}\}\text{Pow(nat)}\) using Spec_def unfolding IsLindeloef_def by auto
then have \(\text{False}\) using Q_disc_comp_csuccQ_eq_Q_choice_csuccQ[OF InfCard_nat] assms by auto
\}
ultimately have \(A < \text{nat}\) by auto
then show \(\text{Finite}(A)\) using lesspoll_nat_is_Finite by auto

1042
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.

**Theorem (in topology0)** no_choice_imp_anti_lindeloef_is_anti_comp:
  assumes ¬{(the axiom of) nat {choice holds for subsets}(Pow(nat))}
  shows T{is anti-compact}
**Proof** -
  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
  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
qed

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-compact}
**Proof** -
  have t:∪{(one-point compactification of)Pow(nat)}={nat}∪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[OF Ord_nat]]
  by auto
  ultimately have {nat}≺csucc(nat) using lesspoll_trans by blast
  with s have {nat}∪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
  
1043
\textbf{Theorem acc_pow_nat_equiv1:} shows \((\text{the axiom of} \ \text{nat \ choice \ holds \ for \ subsets}(\text{Pow}(\text{nat}))) \iff\) \((\text{is anti-lindelöf})\) using \(\text{op_comp_pow_nat_no_anti_comp} \) \(\text{no_choice_imp_anti_lindeloef_is_anti_comp}\).
In the file Topology_ZF_properties.thy, it is proven that \( \mathbb{N} \) is lindeloef if and only if the axiom of countable choice holds for subsets of \( \text{Pow}(\mathbb{N}) \). Now we check that, in ZF, this space is always anti-lindeloef.

**Theorem nat Anti-Lindelöf:**

shows \( \text{Pow}(\mathbb{N}) \) is anti-lindelof

**Proof:**

1. Fix \( A \) assume \( A \subseteq \mathbb{N} \) \( \mathbb{N} \) restricted to \( A \) is lindelof in \( \text{Pow}(\mathbb{N}) \) restricted to \( A \)
   - From \( A(1) \) have \( A \subseteq \mathbb{N} \) by auto
   - Then have \( \text{Pow}(\mathbb{N}) \) restricted to \( A \)=\( \text{Pow}(\mathbb{A}) \) unfolding RestrictedTo_def by blast

2. With \( A(2) \) have \( \text{lin} : A \) is lindelof in \( \text{Pow}(\mathbb{A}) \) using subset_imp_lepoll by auto

3. Fix \( T \) assume \( T \) is a topology \( \mathbb{N} \) \( T \)\( \mathbb{N} \) unfolding eqpoll_sym by auto
   - Then obtain \( f \) where \( f : f \in \text{bij}(A, \mathbb{N}) \) unfolding eqpoll_def by auto
   - Then have \( f \) is surj \( A, \mathbb{N} \) unfolding bij_def by auto
   - Moreover then have \( \text{IsContinuous}(\text{Pow}(\mathbb{A}), T, f) \) unfolding IsContinuous_def
     - Surj_def using \( \text{func1_1_L3} \) by blast
   - Moreover have \( \text{two_top_spaces0}(\text{Pow}(\mathbb{A}), T, f) \) unfolding two_top_spaces0_def
     - Using \( f \) \( T(1) \) \( \text{Pow}(\mathbb{A}) \) unfolding bij_def inj_def by auto
   - Ultimately have \( \mathbb{N} \) is lindelof in \( T \) using \( \text{two_top_spaces0.cont_image_com} \)
     - Lin unfolding \( \text{IsLindelof}(\mathbb{A}) \) by auto

4. Then have \( A \) is in the spectrum of \( \lambda T. ((\bigcup T) \) is lindelof in \( T) \) unfolding Spec_def by auto

5. Then show thesis unfolding \( \text{IsAntiLin}(\mathbb{A}) \) \( \text{antiProperty}(\mathbb{A}) \) 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 Lindelöf.
- Only the compact subspaces of \( \mathbb{N} \) are Lindelöf.
Now, we could wonder if the class of compact spaces and the class of lindelöf spaces being equal is consistent in ZF. Let’s find a topological space which is lindelöf and no compact without assuming any axiom of choice or any negation of one. This will prove that the class of lindelöf spaces and the class of compact spaces cannot be equal in any model of ZF.

**Theorem lord_nat:**

shows \((\text{LOrdTopology } \text{n} \text{at } \text{Le})=\{\text{LeftRayX} (\text{n} \text{at}, \text{Le}, \text{n}). \ n \in \text{nat}\} \cup \{\text{nat}\} \cup \{0\}\)

**Proof:**

- fix \(U\) assume \(U: U \subseteq \{\text{LeftRayX} (\text{n} \text{at}, \text{Le}, \text{n}). \ n \in \text{nat}\} \cup \{\text{nat}\} \cup \{0\}\) \(U \neq \emptyset\)
  - assume \(n \in U\) with \(U\) have \(\bigcup U = \text{nat}\) unfolding \text{LeftRayX}_\text{def} by auto
    then have \(\bigcup U \in \{\text{LeftRayX} (\text{n} \text{at}, \text{Le}, \text{n}). \ n \in \text{nat}\} \cup \{\text{nat}\} \cup \{0\}\) by auto
  
- moreover
  - assume \(n \notin U\) with \(U\) have \(UU: U \subseteq \{\text{LeftRayX} (\text{n} \text{at}, \text{Le}, \text{n}). \ n \in \text{nat}\} \cup \{0\}\) by auto
    - assume \(A: \exists i. \ i \in \text{nat} \land \bigcup U \subseteq \text{LeftRayX} (\text{n} \text{at}, \text{Le}, i)\)
      let \(M = \mu i. \ i \in \text{nat} \land \bigcup U \subseteq \text{LeftRayX} (\text{n} \text{at}, \text{Le}, i)\)
      from \(A\) have \(M: M \in \text{nat} \land \bigcup U \subseteq \text{LeftRayX} (\text{n} \text{at}, \text{Le}, M)\) using \text{LeastI}[OF _ nat_into_\text{Ord}, where \(P = \lambda i. \ i \in \text{nat} \land \bigcup U \subseteq \text{LeftRayX} (\text{n} \text{at}, \text{Le}, i)\)] by auto
    - fix \(y\) assume \(V: y \in \text{LeftRayX} (\text{n} \text{at}, \text{Le}, M)\)
      then have \(y: y \in \text{nat}\) unfolding \text{LeftRayX}_\text{def} by auto
    - assume \(\forall V \in U. \ y \notin V\)
      then have \(\forall m \in \{n \in \text{nat}. \ \text{LeftRayX} (\text{n} \text{at}, \text{Le}, n) \in U\}. \ y \notin \text{LeftRayX} (\text{n} \text{at}, \text{Le}, m)\) using \(UU\) by auto
    - then have \(\forall m \in \{n \in \text{nat}. \ \text{LeftRayX} (\text{n} \text{at}, \text{Le}, n) \in U\}. \ (y, m) \in \text{Le} \lor y = m\) unfolding \text{LeftRayX}_\text{def} using \(y\)
      by auto
    - then have \(RR: \forall m \in \{n \in \text{nat}. \ \text{LeftRayX} (\text{n} \text{at}, \text{Le}, n) \in U\}. \ (m, y) \in \text{Le}\) using \text{Le\_directs\_nat}(1) \(y\) unfolding \text{IsLinOrder\_def} \text{IsTotal\_def} by blast
    - fix \(rr\) assume \(RR \in U\)
      then obtain \(V\) where \(V: V \in U \land rr \in V\) by auto
      - with \(UU\) obtain \(m\) where \(m: V = \text{LeftRayX} (\text{n} \text{at}, \text{Le}, m) m \in \text{nat}\) by auto
        - with \(V(1)\) \(RR\) have \(a: (m, y) \in \text{Le}\) by auto
          from \(V(2)\) \(m(1)\) have \(b: (rr, m) \in \text{Le} \land rr \in \text{nat} \land (\neg m)\) unfolding \text{LeftRayX}_\text{def}
        by auto
          from \(a\) \(b(1)\) have \((rr, y) \in \text{Le}\) using \text{Le\_directs\_nat}(1) unfolding \text{IsLinOrder\_def}
            trans\_def by blast

moreover
\[
\{ 
\begin{align*}
\text{assume } \rr &= y \\
\text{with } a \ b \text{ have } False \text{ using } \text{Le\_directs\_nat}(1) \text{ unfolding } \text{IsLinOrder\_def} \ \text{antisym\_def} \text{ by } \text{blast}
\end{align*}
\]
ultimately have \( \rr \in \text{LeftRayX}(\text{nat}, \text{Le}, y) \) unfolding \( \text{LeftRayX\_def} \) using \( b(2) \) by auto
\[
\begin{align*}
\text{then have } \bigcup U &\subseteq \text{LeftRayX}(\text{nat}, \text{Le}, \text{y}) \text{ unfolding } \text{IsLinOrder\_def} \ \text{antisym\_def} \text{ by } \text{blast} \\
\end{align*}
\]
moreover
\[
\{ 
\begin{align*}
\text{assume } \neg (\exists i. \ i \in \text{nat} \ \land \ \bigcup U \subseteq \text{LeftRayX}(\text{nat}, \text{Le}, i)) \\
\text{then have } A: \forall i. i \in \text{nat} \rightarrow \neg (\bigcup U \subseteq \text{LeftRayX}(\text{nat}, \text{Le}, i)) \text{ by auto}
\end{align*}
\]
\[
\begin{align*}
\text{fix } i \text{ assume } i: i \in \text{nat} \\
\text{with } A \text{ have } AA: \neg (\bigcup U \subseteq \text{LeftRayX}(\text{nat}, \text{Le}, i)) \text{ by auto}
\end{align*}
\]
\[
\begin{align*}
\text{assume } i \not\in \bigcup U \\
\text{then have } \forall V \in U. i \not\in V \text{ by auto} \\
\text{then have } \forall m \in \{n \in \text{nat}. \ \text{LeftRayX}(\text{nat}, \text{Le}, n) \in U\}. i \not\in \text{LeftRayX}(\text{nat}, \text{Le}, m) \text{ by auto}
\end{align*}
\]
unfolding \( \text{LeftRayX\_def} \) by auto
\[
\begin{align*}
\text{with } i \text{ have } \forall m \in \{n \in \text{nat}. \ \text{LeftRayX}(\text{nat}, \text{Le}, n) \in U\}. (i, m) \not\in \text{Le} \ \text{Vi} = m
\end{align*}
\]
\[
\begin{align*}
\text{unfolding } \text{Le\_def} \text{ by auto}
\end{align*}
\]
\[
\begin{align*}
\text{with } i \text{ have } \forall m \in \{n \in \text{nat}. \ \text{LeftRayX}(\text{nat}, \text{Le}, n) \in U\}. \neg (i \leq m) \ \text{Vi} = m
\end{align*}
\]
\[
\begin{align*}
\text{then have } \forall m \in \{n \in \text{nat}. \ \text{LeftRayX}(\text{nat}, \text{Le}, n) \in U\}. m < i \ \text{Vi} = m \text{ using } \text{not\_le\_iff\_lt}[\text{OF } \text{nat\_into\_Ord}[\text{OF } i] \\
\text{nat\_into\_Ord]} \text{ by auto}
\end{align*}
\]
\[
\begin{align*}
\text{then have } M: \forall m \in \{n \in \text{nat}. \ \text{LeftRayX}(\text{nat}, \text{Le}, n) \in U\}. m \leq i \ \text{using } \text{le\_iff} \text{nat\_into\_Ord}[\text{OF } i] \text{ by auto}
\end{align*}
\]
\[
\begin{align*}
\text{fix } s \text{ assume } s \in \bigcup U \\
\text{then obtain } n \text{ where } n: n \in \text{nat} \ s \in \text{LeftRayX}(\text{nat}, \text{Le}, n) \ \text{LeftRayX}(\text{nat}, \text{Le}, n) \in U
\end{align*}
\]
\[
\begin{align*}
\text{using } \text{UU} \text{ by auto}
\end{align*}
\]
\[
\begin{align*}
\text{with } M \text{ have } n: n \leq i \text{ by auto}
\end{align*}
\]
\[
\begin{align*}
\text{from } n(2) \text{ have } sn: s \leq n \ s \neq n \text{ unfolding } \text{LeftRayX\_def} \text{ by auto}
\end{align*}
\]
\[
\begin{align*}
\text{then have } s \leq i \ s \neq i \text{ using } \text{le\_trans}[\text{OF } sn(1) \ ni] \ \text{le\_anti\_sym}[\text{OF } sn(1)] \ ni \text{ by auto}
\end{align*}
\]
then have $s \in \text{LeftRayX}(\text{nats}, \text{Le}, i)$ using $i \le \text{in} \text{nat}$ unfolding \text{LeftRayX_def} by auto \\
} \\
with \text{AA} have False by auto \\
} \\
then have $i \in \bigcup \text{U}$ by auto \\
} \\
then have $\text{nats} \subseteq \bigcup \text{U}$ by auto \\
then have $\bigcup \text{U} = \text{nats}$ using \text{UU unfolding \text{LeftRayX_def}} by auto \\
then have $\bigcup \text{U} \in \{\text{LeftRayX}(\text{nats}, \text{Le}, n). \ n \in \text{nats}\} \cup \{\text{nats}\} \cup \{0\}$ by auto \\
} \\
ultimately have $\bigcup \text{U} \in \{\text{LeftRayX}(\text{nats}, \text{Le}, n). \ n \in \text{nats}\} \cup \{\text{nats}\} \cup \{0\}$ by auto \\
} \\
ultimately have $\bigcup \text{U} \in \{\text{LeftRayX}(\text{nats}, \text{Le}, n). \ n \in \text{nats}\} \cup \{\text{nats}\} \cup \{0\}$ by auto \\
} \\
moreover \\
\{ \\
\fix $\text{U}$ assume $\text{U} = 0$ \\
then have $\bigcup \text{U} \in \{\text{LeftRayX}(\text{nats}, \text{Le}, n). \ n \in \text{nats}\} \cup \{\text{nats}\} \cup \{0\}$ by auto \\
} \\
ultimately have $\forall \text{U}. \ \text{U} \subseteq \{\text{LeftRayX}(\text{nats}, \text{Le}, n). \ n \in \text{nats}\} \cup \{\text{nats}\} \cup \{0\}$ by auto \\
then have $\{\text{LeftRayX}(\text{nats}, \text{Le}, n). \ n \in \text{nats}\} \cup \{\text{nats}\} \cup \{0\} = \{\bigcup \text{U}. \ \text{U} \in \text{Pow}\{\{\text{LeftRayX}(\text{nats}, \text{Le}, n). \ n \in \text{nats}\} \cup \{\text{nats}\}\}\}$ by blast \\
then show thesis using \text{LOrdtopology_ROrdtopology_are_topologies}(2)[OF \text{Le_directs_nats}(1)] unfolding \text{IsAbaseFor_def} by auto \\
qed \\

lemma \text{countable_lord_nats}: \\
shows $\{\text{LeftRayX}(\text{nats}, \text{Le}, n). \ n \in \text{nats}\} \cup \{\text{nats}\} \cup \{0\} \prec \text{csucc(nats)}$ \\
proof - \\
\{ \\
\fix $e$ \\
have $\{e\} \approx 1$ using \text{singleton_eqpoll_1} by auto \\
then have $\{e\} \prec \text{nats}$ using \text{n_lesspoll_nats} eq_lesspoll_trans by auto \\
moreover \\
have $s: \text{nats} \prec \text{csucc(nats)}$ using \text{lt_Card_imp_lesspoll}[OF \text{Card_csucc}] \text{lt_csucc}[OF \text{Ord_nats}] by auto \\
ultimately have $\{e\} \prec \text{csucc(nats)}$ using \text{lesspoll_trans} by blast \\
} \\
then have $\{\text{nats}\} \cup \{0\} \prec \text{csucc(nats)}$ using \text{less_less_imp_un_less}[OF _ _ \text{InfCard_csucc}[OF \text{InfCard_nat}], of \{\text{nats}\} \{0\}] by auto \\
moreover \\
\let FF = $\{n, \text{LeftRayX}(\text{nats}, \text{Le}, n)\}. \ n \in \text{nats}\} \\
\have \text{ff: FF} : \text{nat} \to \{\text{LeftRayX}(\text{nats}, \text{Le}, n). \ n \in \text{nats}\}$ unfolding \text{Pi_def} domain_def function_def by auto \\

1048
then have \( su : FF \in \text{surj}(\text{nat}, \{\text{LeftRayX}(\text{nat}, \text{Le}, n). n \in \text{nat}\}) \) unfolding \text{surj_def}
using apply_equality[OF _ ff] by auto
then have \( \{\text{LeftRayX}(\text{nat}, \text{Le}, n). n \in \text{nat}\} \leq \text{nat} \) using \text{surj_fun_inv_2[OF su lepoll_refl[of \text{nat}]]} \text{Ord_nat}
by auto
ultimately have \( \{\text{LeftRayX}(\text{nat}, \text{Le}, n). n \in \text{nat}\} \preceq \text{nat} \)
using \text{Card_less_csucc_eq_le[OF \text{Card_nat}]} by auto
ultimately have \( \{\text{LeftRayX}(\text{nat}, \text{Le}, n). n \in \text{nat}\} \cup (\{\text{nat}\} \cup \{0\}) \preceq \text{csucc(nat)} \)
ultimately show thesis by auto
qed

corollary lindelof_lord_nat:
shows \( \text{nat}\{\text{is lindeloef in}(\LOrdTopology \text{nat} \text{Le})} \)
unfolding \text{IsLindeloef_def} using \text{countable_lord_nat} \text{lord_nat} \text{card_top_comp[OF \text{Card_csucc[OF \text{Ord_nat}]}}
union_lordtopology_rordtopology(1)[OF Le_directs_nat(1)] by auto

theorem not_comp_lord_nat:
shows \( \neg\ (\text{nat}\{\text{is compact in}(\LOrdTopology \text{nat} \text{Le})} \)
proof
assume \( \text{nat}\{\text{is compact in}(\LOrdTopology \text{nat} \text{Le})} \)
with \text{lord_nat} have \( \text{nat}\{\text{is compact in}(\{\text{LeftRayX}(\text{nat}, \text{Le}, n). n \in \text{nat}\} \cup \{\text{nat}\} \cup \{0\}) \)
by auto
then have \( \forall M \in \text{Pow}(\{\text{LeftRayX}(\text{nat}, \text{Le}, n). n \in \text{nat}\} \cup \{\text{nat}\} \cup \{0\})\). \text{nat} \subseteq \bigcup M
→ (\exists N \in \text{FinPow}(M). \text{nat} \subseteq \bigcup N)
unfolding \text{IsCompact_def} by auto
moreover
{ fix n assume n:n\in\text{nat}
then have n\in\text{succ(n)} by auto
then have \( n, \text{succ}(n) \in \text{Le} \) \( n \neq \text{succ}(n) \) using \text{n nat_succ_iff} by auto
then have n\in\text{LeftRayX}(\text{nat}, \text{Le}, \text{succ(n)}) unfolding \text{LeftRayX_def} using n by auto
then have n\in\bigcup(\{\text{LeftRayX}(\text{nat}, \text{Le}, n). n\in\text{nat}\}) using \text{n nat_succ_iff} by auto
}
ultimately have \( \exists N \in \text{FinPow}(\{\text{LeftRayX}(\text{nat}, \text{Le}, n). n \in \text{nat}\}). \text{nat} \subseteq \bigcup N \)
by blast
then obtain N where N\in\text{FinPow}(\{\text{LeftRayX}(\text{nat}, \text{Le}, n). n \in \text{nat}\}) \text{nat} \subseteq \bigcup N
by auto
then have N:N\subseteq\{\text{LeftRayX}(\text{nat}, \text{Le}, n). n \in \text{nat}\} \text{ Finite(N) nat} \subseteq \bigcup N unfolding \text{FinPow_def by auto}
let \( F = \{n, \text{LeftRayX}(\text{nat}, \text{Le}, n)\}. n \in \{m \in \text{nat}. \text{LeftRayX}(\text{nat}, \text{Le}, m) \in N\}\)
have ff:F:{m\in\text{nat}. \text{LeftRayX}(\text{nat}, \text{Le}, m) \in N} \rightarrow N unfolding Pi_def function_def by auto

1049
then have $F \in \text{surj}(\{m \in \text{nat. LeftRayX(nat,Le,m)} \in N\}, N)$ unfolding surj_def
using $N(1)$ apply_equality[
  OF _ ff] by blast moreover

\{
  fix $x$ $y$ assume xyF:$x \in \{m \in \text{nat. LeftRayX(nat,Le,m)} \in N\}$ $y \in \{m \in \text{nat. LeftRayX(nat,Le,m)} \in N\}$
Fx=Fy
then have $Fx = \text{LeftRayX(nat,Le,x)}$ $Fy = \text{LeftRayX(nat,Le,y)}$
  using apply_equality[OF _ ff] by auto
\}

with $xyF(3)$ have $\text{lxy:LeftRayX(nat,Le,x)} = \text{LeftRayX(nat,Le,y)}$
  by auto

\{
  fix $r$ assume $r < x$
  then have $r \leq x$ $r \neq x$ using leI
  by auto
  with $xyF(1)$ have $r \in \text{LeftRayX(nat,Le,x)}$
  unfolding LeftRayX_def using le_in_nat
  by auto
  then have $r \in \text{LeftRayX(nat,Le,y)}$
  using lxy
  by auto
  then have $r \leq y$ unfolding le_iff
  by auto
\}

then have $\forall r. r < x \rightarrow r < y$ by auto
then have $r : \neg(y < x)$ by auto

\{
  fix $r$ assume $r < y$
  then have $r \leq y$ $r \neq y$ using leI
  by auto
  with $xyF(2)$ have $r \in \text{LeftRayX(nat,Le,y)}$
  unfolding LeftRayX_def using le_in_nat
  by auto
  then have $r \in \text{LeftRayX(nat,Le,x)}$
  using lxy
  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]

\begin{align*}
  xyF(1,2) & \quad \text{le_anti_sym by auto} \\
  & \quad \text{then have } F \in \text{inj}(\{m \in \text{nat. LeftRayX(nat,Le,m)} \in N\}, N)
  \quad \text{unfolding inj_def using ff by auto} \\
  & \quad \quad \text{ultimately have } F \in \text{bij}(\{m \in \text{nat. LeftRayX(nat,Le,m)} \in N\}, N)
  \quad \text{unfolding bij_def by auto} \\
  & \quad \quad \quad \text{then have } \{m \in \text{nat. LeftRayX(nat,Le,m)} \in N\} \approx N
  \quad \text{unfolding eqpoll_def by auto} \\
  & \quad \quad \quad \quad \text{with } N(2)
  \quad \text{have fin:Finite}(\{m \in \text{nat. LeftRayX(nat,Le,m)} \in N\})
  \quad \text{using lepoll_Finite eqpoll_imp_lepoll} \\
  & \quad \quad \quad \quad \quad \text{by auto} \\
  & \quad \quad \quad \quad \quad \text{from } N(3)
  \quad \text{have } N \neq 0 \text{ by auto} \\
  & \quad \quad \quad \quad \quad \text{then have nE:}(m \in \text{nat. LeftRayX(nat,Le,m)} \in N) \neq 0
  \quad \text{using } N(1) \text{ by auto} \\
  & \quad \quad \quad \quad \quad \quad \text{let } M = \text{Maximum}(\{m \in \text{nat. LeftRayX(nat,Le,m)} \in N\}) \\
  & \quad \quad \quad \quad \quad \text{have } M : M \in \text{nat. LeftRayX(nat,Le,m)} \in N \\
  & \quad \quad \quad \quad \quad \quad \forall r \in \{m \in \text{nat. LeftRayX(nat,Le,m)} \in N\}.
  \quad [r, M] \in \text{Le using } \text{fin linord_max_props}(1,3)
  \quad \text{[OF Le_directs_nat(1) _ nE] unfolding FinPow_def by auto} \\
\end{align*}

1050
Fix $V \in N \ x \in V$
then obtain $m$ where $m = \text{LeftRayX}(\text{nat}, \text{Le}, m) \ L e f t R a y X(nat, Le, m) \in N$ $m \in \text{nat}$
using $N(1)$ by auto
with $V(2)$ have $xx : \langle r, m \rangle \in \text{Le} \ r \neq m$ unfolding $\text{LeftRayX}_{\text{def}}$ by auto
from $m(2, 3)$ have $m \in \{ m \in \text{nat}. \ Left Ray X(nat, Le, m) \ \in N \}$ by auto
then have $\langle r, m \rangle \in \text{Le}$ using $M(3)$ by auto
with $xx(1)$ have $\langle r, M \rangle \in \text{Le}$ using $\text{le}_\text{trans}$ unfolding $\text{Le}_{\text{def}}$ by auto
moreover
{
  assume $r = M$
  with $xx \ \text{mM}$ have False using $\text{le}_\text{anti_sym}$ by auto
}
ultimately have $r = \text{LeftRayX}(\text{nat}, \text{Le}, M)$ unfolding $\text{LeftRayX}_{\text{def}}$ by auto

then have $\cup N \subseteq \text{LeftRayX}(\text{nat}, \text{Le}, M)$ by auto
with $M(2)$ have $\cup N = \text{LeftRayX}(\text{nat}, \text{Le}, M)$ by auto
with $N(3)$ have $\text{nat} \subseteq \text{LeftRayX}(\text{nat}, \text{Le}, M)$ by auto
moreover from $M(1)$ have $\text{succ}(M) \in \text{nat}$ using $\text{nat}_{\text{succI}}$ by auto
ultimately have $\text{succ}(M) \in \text{LeftRayX}(\text{nat}, \text{Le}, M)$ by auto
then have $\langle \text{succ}(M), M \rangle \in \text{Le}$ unfolding $\text{LeftRayX}_{\text{def}}$ by blast
then show False by auto
qed

73.4 More Separation properties

In this section we study more separation properties.

73.5 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} ≡ $\forall A \in \text{Pow}(\cup T). \ A\{\text{is compact in}T} \rightarrow A\{\text{is closed in}T}$

Another type of space is an US-space; those where sequences have at most one limit.

definition
IsUS (_ (is US)) where
T{is US} ≡ $\forall N \times y. \ (N : \text{nat} \rightarrow \cup T) \wedge \text{NetConvTop}(\langle N, Le \rangle, x, T) \wedge \text{NetConvTop}(\langle N, Le \rangle, y, T)$
$\rightarrow y = x$

73.6 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 $T_2$
shows $T$ is KC
proof\{-
\begin{align*}
  &\text{fix } A \quad \text{assume } A \text{ is compact in } T \\
  &\quad \text{then have } A \text{ is closed in } T \text{ using in}_T\text{compact_is_cl assms by auto}
\end{align*}
\}
\text{then show thesis unfolding IsKC_def by auto}
qed

From the spectrum of compactness, it follows that any KC-space is $T_1$.

\begin{lemma}[in topology0] KC_imp_T1:
\begin{proof}\{-
\begin{align*}
  &\text{fix } x \quad \text{assume } A: x \in \bigcup T \\
  &\quad \text{have } \text{Finite}\{x\} \text{ by auto} \\
  &\quad \text{then have } \{x\} \text{ is in the spectrum of } \lambda T. \{\bigcup T\} \text{ is compact in } T \text{ using compact_spectrum by auto moreover} \\
  &\quad \text{have } (T\text{restricted to}\{x\}) \text{ is a topology} \text{ using Top1_L4 by auto} \\
  &\quad \text{moreover have } \bigcup (T\text{restricted to}\{x\}) = \{x\} \text{ using } A \text{ unfolding RestrictedTo_def by auto} \\
  &\quad \text{ultimately have } \text{com:}\{x\} \text{ is compact in } (T\text{restricted to}\{x\}) \text{ unfolding Spec_def by auto} \\
  &\quad \text{then have } \{x\} \text{ is compact in } T \text{ using compact_subspace_imp_compact } A \text{ by auto} \\
  &\quad \text{then have } \{x\} \text{ is closed in } T \text{ using assms unfolding IsKC_def using } A \text{ by auto}
\end{align*}
\}
\text{then show thesis using } T_1\text{iff_singleton_closed by auto}
qed
\end{proof}\end{lemma}

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.

\begin{theorem}[in topology0] KC_imp_US:
\begin{proof}\{-
\begin{align*}
  &\text{fix } N x y \quad \text{assume } A: N: \text{nat} \to \bigcup T \quad (N, Le) \to_N x \quad (N, Le) \to_N y \quad x \neq y \\
  &\quad \text{have } \text{dir:Le directs nat using } Le\text{directs_nat by auto moreover} \\
  &\quad \text{from } A(1) \quad \text{have } \text{dom:domain}(N) = \text{nat} \text{ using } func1_1_L1 \text{ by auto} \\
  &\quad \text{moreover note } A(1) \quad \text{ultimately have } \text{Net:}\{N, Le\} \text{ is a net on } \bigcup T \text{ unfolding IsNet_def}
\end{align*}
\}
\end{proof}\end{theorem}
by auto
from A(3) have y: y ∈ ∪ T unfolding NetConverges_def [OF Net] by auto
from A(2) have x: x ∈ ∪ T unfolding NetConverges_def [OF Net] by auto
from A(2) have o1: ∀ U ∈ Pow (∪ T). x ∈ int(U) ⟷ (∃ r ∈ nat. ∀ s ∈ nat. ⟨r, s⟩ ∈ Le ⟷ Ns ∈ U) unfolding NetConverges_def [OF Net]
    using dom by auto
{ assume B : ∃ n ∈ nat. ∀ m ∈ nat. ⟨n, m⟩ ∈ Le ⟷ Nm = y have {y} is closed in T using y T1_iff singleton_closed KC_imp_T1 assms by auto
    then have o2: ∪ T - {y} ∈ T unfolding IsClosed_def by auto
    then have int(∪ T - {y}) = ∪ T - {y} using Top_2_L3 by auto
with A(4) x have o3: x ∈ int(∪ T - {y}) by auto
from o2 have ∪ T - {y} ∈ Pow (∪ T) by auto
with o1 o3 obtain r where r : r ∈ nat ∀ s ∈ nat. ⟨r, s⟩ ∈ Le ⟷ Ns ∈ ∪ T - {y} by auto
from B obtain n where n : n ∈ nat. ∀ m ∈ nat. ⟨n, m⟩ ∈ Le ⟷ Nm = y by auto
from dir r(1) n(1) obtain z where ⟨r, z⟩ ∈ Le ⟨n, z⟩ ∈ Lez ∈ nat unfolding IsDirectedSet_def by auto
with r(2) n(2) have Nz ∈ ∪ T - {y} Nz = y by auto
then have False by auto
}
then have reg: ∀ n ∈ nat. ∃ m ∈ nat. Nm ≠ y ∧ ⟨n, m⟩ ∈ Le by auto
let NN = {⟨n, N(μ i. Ni ≠ y ∧ ⟨n, i⟩ ∈ Le)⟩. n ∈ nat}
{ fix x z assume A1: ⟨x, z⟩ ∈ 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 ≠ y ∧ ⟨n, m⟩ ∈ Le m ∈ nat by auto
    then have LI : N(μ i. Ni ≠ y ∧ ⟨n, i⟩ ∈ Le) ≠ y (n, μ i. Ni ≠ y ∧ ⟨n, i⟩ ∈ Le) ∈ Le
using LeastI[of λm. Nm ≠ y ∧ ⟨n, m⟩ ∈ Le m]
    nat_into_Ord[of m] by auto
    then have ⟨μ i. Ni ≠ y ∧ ⟨n, i⟩ ∈ Le⟩ ∈ nat by auto
    then have N(μ i. Ni ≠ y ∧ ⟨n, i⟩ ∈ Le) ∈ ∪ T using apply_type [OF A(1)]
by auto
    with as have ⟨n, N(μ i. Ni ≠ y ∧ ⟨n, i⟩ ∈ Le)⟩ ∈ nat × ∪ T by auto
} then have NN ∈ Pow(nat × ∪ T) by auto
ultimately have NFun : NN : nat → ∪ T unfolding Pi_def function_def domain_def by auto
{
fix n assume as: n ∈ nat
with reg obtain m where N ≠ y ∧ ⟨n,m⟩ ∈ Le m ∈ nat by auto
then have LI:N(μ i. Ni ≠ y ∧ ⟨n,i⟩ ∈ Le) ≠ y ⟨n,m⟩ i. Ni ≠ y ∧ ⟨n,i⟩ ∈ Le) ∈ Le
using LeastI[of λm. N ≠ y ∧ ⟨n,m⟩ ∈ Le m]
  nat_into_Ord[of m] by auto
then have NNN ≠ y using apply_equality[OF _ NFun] by auto
}
then have noy: ∀ n ∈ nat. NNN ≠ y by auto
have dom2: domain(NN) = nat by auto
then have net2: ⟨NN,Le⟩ ∈ Un unfolding IsNet_def using NFun
  by auto
{
  fix U assume U ∈ Pow(∪ T) x ∈ int(U)
  then have (∃ r ∈ nat. ∀ s ∈ nat. ⟨r,s⟩ ∈ Le → Ns ∈ U) using o1 by auto
  then obtain r where r_def: r ∈ nat ∀ s ∈ nat. ⟨r,s⟩ ∈ Le → Ns ∈ U by auto
  
  fix s assume AA: ⟨r,s⟩ ∈ Le
  with reg obtain m where N ≠ y ⟨s,m⟩ ∈ Le by auto
  then have ⟨s,μ⟩ i. Ni ≠ y ∧ ⟨s,i⟩ ∈ Le using LeastI[of λm. N ≠ y ∧ ⟨s,m⟩ ∈ Le m]
  nat_into_Ord by auto
  with AA have ⟨r,μ⟩ i. Ni ≠ y ∧ ⟨s,i⟩ ∈ Le using le_trans by auto
  with r_def(2) have N ⟨μ⟩ i. Ni ≠ y ∧ ⟨s,i⟩ ∈ U by blast
  then have NNN ∈ U using apply_equality[OF _ NFun] AA by auto
}
then have ∀ s ∈ nat. ⟨r,s⟩ ∈ Le → NNN ∈ U by auto
with r_def(1) have ∃ r ∈ nat. ∀ s ∈ nat. ⟨r,s⟩ ∈ Le → NNN ∈ U by auto
}
then have conv2: ⟨NN,Le⟩ →N x unfolding NetConverges_def[OF net2] using x dom2 by auto
let A = {x ∈ NNN at
{
  fix M assume Acov: A ∈ ∪ M M ⊆ T
  then have x ∈ ∪ M by auto
  then obtain U where U = int(U) using Top_2_L3 by auto
  with Acov(2) have U = int(U) using Top_2_L3 by auto
  then have U = int(U) using Top_2_L3 by auto
  with conv2 obtain r where rr: r ∈ nat ∀ s ∈ nat. ⟨r,s⟩ ∈ Le → NNN ∈ U
  unfolding NetConverges_def[OF net2] using dom2 UT by auto
  have NresFun: restrict(NN, {n ∈ nat. ⟨n,r⟩ ∈ Le}) : {n ∈ nat. ⟨n,r⟩ ∈ Le} → ∪ T
  using restrict_fun
  [OF NFun, of {n ∈ nat. ⟨n,r⟩ ∈ Le}] by auto
  then have restrict(NN, {n ∈ nat. ⟨n,r⟩ ∈ Le}) ∈ surj({n ∈ nat. ⟨n,r⟩ ∈ Le}, range(restrict(NN, {n ∈ nat. ⟨n,r⟩ ∈ Le})))
  using 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 \texttt{surj\_fun\_inv\_2} by auto

moreover
have \{n \in \textit{nat}. \langle n, 0 \rangle \in \textit{Le}\} = \{0\} by auto
then have Finite(\{n \in \textit{nat}. \langle n, 0 \rangle \in \textit{Le}\}) by auto
moreover
\{
  \fix j \assume as: j \in \textit{nat} Finite(\{n \in \textit{nat}. \langle n, j \rangle \in \textit{Le}\})
  \{
    \fix t \assume t \in \{n \in \textit{nat}. \langle n, \textit{succ}(j) \rangle \in \textit{Le}\}
    then have t \in \textit{nat} \langle t, \textit{succ}(j) \rangle \in \textit{Le} by auto
    then have t \leq \textit{succ}(j) by auto
    then have t \subseteq \textit{succ}(j) using \text{le\_imp\_subset} by auto
    then have j \in t \subseteq j by auto
    then have j \subseteq \textit{succ}(j) using \text{succ\_explained} by auto
    then have j \subseteq j \cup \{j\} using \text{succ\_explained} by auto
    then have j \in t \cup \{j\} using \text{le\_imp\_subset} by auto
    then have j \subseteq \textit{succ}(j) \cup \{j\} using \text{succ\_explained} by auto
    then have j \subseteq \textit{succ}(j) \cup \{j\} using \text{succ\_explained} by auto
    then have j \subseteq j \cup \{j\} by auto
    then have j \subseteq j \cup \{j\} using \text{le\_imp\_subset} by auto
    then have j \subseteq j \cup \{j\}\} by auto

unfolding \text{Ord\_def} \ Transset\_def by auto
then have succ(j) \subseteq j \cup \{j\} using \text{succ\_explained} by auto
with \langle t \subseteq j \rangle have t = succ(j) \cup j \leq j by auto
with \langle t \in \textit{nat} \rangle \langle j \in \textit{nat} \rangle have t \in \{n \in \textit{nat}. \langle n, j \rangle \in \textit{Le}\} \cup \{\textit{succ}(j)\}
by auto
\}
then have \{n \in \textit{nat}. \langle n, \textit{succ}(j) \rangle \in \textit{Le}\} \subseteq \{n \in \textit{nat}. \langle n, j \rangle \in \textit{Le}\} \cup \{\textit{succ}(j)\}
by auto

moreover have Finite(\{n \in \textit{nat}. \langle n, j \rangle \in \textit{Le}\} \cup \{\textit{succ}(j)\}) using \text{as}(2)
Finite\_cons
by auto
ultimately have Finite(\{n \in \textit{nat}. \langle n, \textit{succ}(j) \rangle \in \textit{Le}\}) using \text{subset\_Finite}
by auto
\}
then have \forall j \in \textit{nat}. Finite(\{n \in \textit{nat}. \langle n, j \rangle \in \textit{Le}\}) \rightarrow Finite(\{n \in \textit{nat}. \langle n, \textit{succ}(j) \rangle \in \textit{Le}\})
by auto
ultimately have Finite(range(restrict(\textit{NN}, \{n \in \textit{nat} . \langle n, r \rangle \in \textit{Le}\})))
using \text{lepoll\_Finite}[\text{of range}(\text{restrict}(\textit{NN}, \{n \in \textit{nat} . \langle n, r \rangle \in \textit{Le}\}))]
by auto
{\{n \in \textit{nat} . \langle n, r \rangle \in \textit{Le}\}} ind\_on\_nat[OF \langle r \in \textit{nat}\rangle, \text{where P}=\lambda t.\ Finite(\{n \in \textit{nat}. \langle n, t \rangle \in \textit{Le}\})] by auto
then have Finite(\{\text{restrict}(\textit{NN}, \{n \in \textit{nat} . \langle n, r \rangle \in \textit{Le}\})\} \{n \in \textit{nat} . \langle n, r \rangle \in \textit{Le}\}) using \text{range\_image\_domain}[OF \text{NresFun}]
by auto
then have Finite(\text{NN}(\{n \in \textit{nat} . \langle n, r \rangle \in \textit{Le}\})) using \text{restrict\_image}
by auto
then have (\text{NN}(\{n \in \textit{nat} . \langle n, r \rangle \in \textit{Le}\})\{is in the spectrum of\} (\lambda T.\ \langle \bigcup T \rangle\{is compact in\} T) using \text{compact\_spectrum} by auto
moreover have \{\bigcup (\text{restricted to} \text{NN}(\{n \in \textit{nat} . \langle n, r \rangle \in \textit{Le}\})) \{n \in \textit{nat} . \langle n, r \rangle \in \textit{Le}\}\ = \bigcup T \cap \text{NN}(\{n \in \textit{nat} . \langle n, r \rangle \in \textit{Le}\})
unfolding \text{RestrictedTo\_def} by auto
moreover
have $\bigcup T \cap \NN\{ n \in \text{nat} . \langle n, r \rangle \in \text{Le} \}$
using func1_1_L6(2)[OF NFun] by blast
moreover have $(T \cap \NN\{ n \in \text{nat} . \langle n, r \rangle \in \text{Le} \})$ is a topology
using Top_1_L4 by auto
ultimately have $(\NN\{ n \in \text{nat} . \langle n, r \rangle \in \text{Le} \})$ is a topology
by auto
moreover note Acov(2) ultimately
obtain $\mathfrak{M}$ where $\mathfrak{M} : \mathfrak{M} \in \text{FinPow}(M) \ (\NN\{ n \in \text{nat} . \langle n, r \rangle \in \text{Le} \}) \subseteq \bigcup \mathfrak{M}$
unfolding IsCompact_def by force
then have $(\NN\{ n \in \text{nat} . \langle n, r \rangle \in \text{Le} \})$ is compact in $(T \cap \NN\{ n \in \text{nat} . \langle n, r \rangle \in \text{Le} \})$
unfolding Spec_def by force
then have $(\NN\{ n \in \text{nat} . \langle n, r \rangle \in \text{Le} \})$ is compact in $T$
using compact_subspace_imp_compact by auto
moreover from Acov(1) have $(\NN\{ n \in \text{nat} . \langle n, r \rangle \in \text{Le} \}) \subseteq \bigcup M$
by auto
moreover note Acov(2) ultimately
obtain $\mathfrak{M}$ where $\mathfrak{M} : \mathfrak{M} \in \text{FinPow}(M) \ (\NN\{ n \in \text{nat} . \langle n, r \rangle \in \text{Le} \}) \subseteq \bigcup \mathfrak{M}$
unfolding IsCompact_def by blast
from $\mathfrak{M}(1)$ have $\mathfrak{M} \cup \{ U \} \in \text{FinPow}(M)$ using U(2) unfolding FinPow_def
by auto
moreover
{ fix $s$ assume $s : s \in A \ s \notin U$
with U(1) have $s \in \NN\{ n \in \text{nat} . \langle n, r \rangle \in \text{Le} \}$
by auto
then have $s \in \NN\{ n \in \text{nat} . \langle n, r \rangle \in \text{Le} \}$
using func_imagedef[OF NFun] by auto
then obtain $n$ where $n : n \in \text{nat} \ s = \NN\{ n \in \text{nat} . \langle n, r \rangle \in \text{Le} \}$
by auto
{ assume $\langle r, n \rangle \in \text{Le}$
with rr have $\NN\{ n \in \text{nat} . \langle n, r \rangle \in \text{Le} \}$
by auto
with $n(2)$ $s(2)$ have False by auto
} then have $\langle r, n \rangle \notin \text{Le}$
by auto
with rr(1) $n(1)$ have $\lnot (r \leq n)$
by auto
then have $n \leq r$
using Ord_linear_le[where thesis=$\langle n, r \rangle \in \text{Le}$]
nat_into_Ord[OF rr(1)]
nat_into_Ord[OF n(1)]
by auto
with rr(1) $n(1)$ have $\langle n, r \rangle \in \text{Le}$
by auto
with $n(2)$ have $s \in \NN\{ n \in \text{nat} . \langle n, r \rangle \in \text{Le} \}$
by auto
moreover have $\{ n \in \text{nat} . \langle n, r \rangle \in \text{Le} \} \subseteq \text{nat}$
by auto
ultimately have $s \in \NN\{ n \in \text{nat} . \langle n, r \rangle \in \text{Le} \}$
using func_imagedef[OF NFun]
by auto
with $\mathfrak{M}(2)$ have $s \in \bigcup \mathfrak{M}$
by auto
} then have $A \subseteq \bigcup \mathfrak{M} \cup U$
by auto
then have $A \subseteq \bigcup (\mathfrak{M} \cup \{ U \})$
by auto
ultimately have $\exists \mathfrak{M} \in \text{FinPow}(M) \ A \subseteq \bigcup \mathfrak{M}$
by auto
} then have $\forall M \in \text{Pow}(T) \ A \subseteq M \rightarrow (\exists \mathfrak{M} \in \text{FinPow}(M) \ A \subseteq \bigcup \mathfrak{M})$
by auto
moreover have $A \subseteq T$
using func1_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

1056
then have $\bigcup T-A \in T$ unfolding IsClosed_def by auto
then have $\bigcup T-A = \text{int}(\bigcup T-A)$ using Top_2_L3 by auto moreover
{
  assume $y \in A$
  with $A(4)$ have $y \in \text{NNnat}$ by auto
  then have $y \in \{\text{NNn}. n \in \text{nat}\}$ using func_imagedef[OF NFun] by auto
  with noy have False by auto
}
with $y$ have $y \in \bigcup T-A$ by force ultimately
have $y \in \bigcup T-A \cup \text{Pow}(\bigcup T)$ by auto moreover
have $(\forall U \in \text{Pow}(\bigcup T). y \in \text{int}(U) \longrightarrow (\exists t \in \text{nat}. \forall m \in \text{nat}. \langle t, m \rangle \in \text{Le} \longrightarrow N m \in U))$
  using $A(3)$ dom unfolding NetConverges_def[OF Net] by auto
ultimately have $\exists t \in \text{nat}. \forall m \in \text{nat}. \langle t, m \rangle \in \text{Le} \longrightarrow N m \in \bigcup T-A$ by blast
then obtain $r$ where $r \in \text{nat}$ $\forall s \in \text{nat}. \langle r, s \rangle \in \text{Le} \longrightarrow N s \in \bigcup T-A$ by auto
{
  fix $s$ assume $\text{AA} : \langle r, s \rangle \in \text{Le}$
  with $\text{reg}$ obtain $m$ where $N m \neq y \langle s, m \rangle \in \text{Le}$ by auto
  then have $\langle s, \mu. N \mu \neq y \land \langle s, i \rangle \in \text{Le} \rangle \in \text{Le}$ using LeastI[of $\lambda m. N m \neq y$
  $\land \langle s, m \rangle \in \text{Le}$ m] nat_into_Ord by auto
  with $\text{AA}$ have $\langle r, \mu. N \mu \neq y \land \langle s, i \rangle \in \text{Le} \rangle \in \text{Le}$ using le_trans by auto
  with $r \text{def}(2)$ have $N(\mu. N \mu \neq y \land \langle s, i \rangle \in \text{Le}) \in \bigcup T-A$ by force
  then have $NNs \in \bigcup T-A \cup \text{Pow}(\bigcup T)$ by auto
  moreover have $NNs \in \text{NNnat}$ using $\text{AA}$ by auto
  then have $NNs \in \text{A}$ by auto
  ultimately have False by auto
}
moreover have $r \subseteq \text{succ}(r)$ using succ_explained by auto
then have $r \subseteq \text{succ}(r)$ using subset_imp_le nat_into_Ord $\langle r \in \text{nat} \rangle$ nat_succI by auto
then have $\langle r, \text{succ}(r) \rangle \in \text{Le}$ using $\langle r \in \text{nat} \rangle$ nat_succI by auto
ultimately have False by auto
}
then have $\forall N x y. (N: \text{nat} \rightarrow \bigcup T) \land (\langle N, \text{Le} \rangle \rightarrow N x \{\text{in} \} T) \land (\langle N, \text{Le} \rangle \rightarrow N y \{\text{in} \} T)$
$\rightarrow N x = y$ 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∈∪T
    then have {x}⊆∪T by auto
    
    fix y assume y:y≠x y∈cl({x})
    then have x:∀U∈T. y∈U → x∈U using cl_inter_neigh[OF {x}⊆∪T]
    by auto
    
    let N=ConstantFunction(nat,x)
    have fun:N:nat→∪T using x func1_3_L1 by auto
    then have dom:domain(N)=nat using func1_1_L1 by auto
    with fun have Net:(N,Le) is a net on)∪T using Le_directs_nat unfolding IsNet_def
    by auto
    
    fix U assume U∈Pow(∪T) x∈int(U)
    then have x∈U using Top_2_L1 by auto
    then have ∀n∈nat. Nn∈U using func1_3_L2 by auto
    then have ∀n∈nat. (0,n)∈Le → Nn∈U by auto
    then have ∃r∈nat. ∀n∈nat. (r,n)∈Le → Nn∈U by auto
    
    then have (N,Le) →ₙ x unfolding NetConverges_def[OF Net] using x dom by auto moreover
    
    fix U assume U∈Pow(∪T) y∈int(U)
    then have x∈int(U) using r Top_2_L2 by auto
    then have x∈U using Top_2_L1 by auto
    then have ∀n∈nat. Nn∈U using func1_3_L2 by auto
    then have ∀n∈nat. (0,n)∈Le → Nn∈U by auto
    then have ∃r∈nat. ∀n∈nat. (r,n)∈Le → Nn∈U by auto
    
    then have (N,Le) →ₙ y unfolding NetConverges_def[OF Net] using y(2) dom
    Top_3_L11(1)[OF {x}⊆∪T] by auto
    ultimately have x=y using assms unfolding IsUS_def using fun by auto
    with y(1) have False by auto
    
    then have cl({x})⊆{x} by auto
    then have cl({x})={x} using cl_contains_set[OF {x}⊆∪T] by auto
    then have {x} is closed in)∪T using Top_3_L8 x by auto
    
    then show thesis using T1_iff_singleton_closed by auto
  }

73.7 Counter-examples

We need to find counter-examples that prove that this properties are new ones.

1058
We know that $T_2 \Rightarrow \text{loc}.T_2 \Rightarrow \text{anti-hyperconnected} \Rightarrow T_1$ and $T_2 \Rightarrow KC \Rightarrow US \Rightarrow T_1$. The question is: What is the relation between KC or US and, \text{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)** \text{locT2_not_US1**}:

assumes \{m}{\notin}{T} \{m\}{is closed in}{T}. \langle N,Le \rangle \rightarrow _N m \land m{\notin}Nnat

shows \exists N{\in}nat{\cup}\{T\}: \langle N,Le \rangle \rightarrow _N m m{\notin}Nnat by auto

proof

from \text{assms}(3) obtain N where N:N{\in}nat{\cup}\{T\} \langle N,Le \rangle \rightarrow _N m m{\notin}Nnat by auto

have \bigcup \{T\} \subseteq \bigcup \{T\}\cup\{U\}\cup\{U\}. \langle U,W \rangle \in\{V{\in}T. m{\in}V\times T\}\) using \text{assms}(2)

\text{unioin_doublepoint_top} by auto

with \text{N}(1) have \text{fun:nat}{\rightarrow}\bigcup \{T\}\cup\{U\}\cup\{U\}. \langle U,W \rangle \in\{V{\in}T. m{\in}V\times T\}\)

using \text{func1_1_L18} by auto

then have \text{domain}(N){\in}nat using \text{func1_1_L18} by auto

with \text{fun have Net:}(N,Le){\in}\{\text{is a net on}\}\bigcup \{T\}\cup\{U\}\cup\{U\}. \langle U,W \rangle \in\{V{\in}T. m{\in}V\times T\}\)

unfolding \text{IsNet_def} using \text{Le_directs_nat} by auto

from \text{N}(1) \text{ dom have Net2:(N,Le){\in}\{\text{is a net on}\}\bigcup\text{unfolding IsNet_def using Le_directs_nat by auto}

from \text{N}(2) have \text{R:}\forall U{\in}\text{Pow}(\bigcup\text{T}). \text{m}{\in}\text{int}(U) \rightarrow (\exists r{\in}\text{nat.} \ \forall s{\in}\text{nat.} \ \langle r,s\rangle{\in}\text{Le} \rightarrow N{\in}U)

unfolding \text{NetConverges_def[OF Net2]} using dom by auto

\text{fix U}\ \text{assume U:U}{\in}\text{Pow}(\bigcup \{T\}\cup\{U\}\cup\{U\}. \langle U,W \rangle \in\{V{\in}T. m{\in}V\times T\}\)\)

m{\in}\text{Interior}(U,T,\cup\{U\}\cup\{U\}\cup\{U\}. \langle U,W \rangle \in\{V{\in}T. m{\in}V\times T\}\)

let I=\text{Interior}(U,T,\cup\{U\}\cup\{U\}\cup\{U\}. \langle U,W \rangle \in\{V{\in}T. m{\in}V\times T\}\)

have I{\subseteq}\{U\}\cup\{U\}\cup\{U\}. \langle U,W \rangle \in\{V{\in}T. m{\in}V\times T\}\)

using topology0.Top_2_L2

\text{assms}(2) \text{ doble_point_top unfolding topology0_def by blast}

then have \langle \bigcup\text{T}\cap I{\subseteq}\{U\}\cup\{U\}\cup\{U\}. \langle U,W \rangle \in\{V{\in}T. m{\in}V\times T\}\}\text{restricted to}\bigcup\text{T}\cap I{\subseteq}\text{unfolding RestrictedIn_def by blast}

then have \langle \bigcup\text{T}\cap I{\subseteq}\text{using open_subspace_double_point(1) assms(2)}

by auto moreover

then have \text{int}(\bigcup\text{T}\cap I)=\bigcup\text{T}\cap I using Top_2_L3 by auto

with U(2) \text{ assms(2) have m}{\in}\text{int}(\bigcup\text{T}\cap I) unfolding IsClosed_def by auto

moreover note R ultimately have \exists r{\in}\text{nat.} \ \forall s{\in}\text{nat.} \ \langle r,s\rangle{\in}\text{Le} \rightarrow N{\in}(\bigcup\text{T}\cap I)

by blast

then have \exists r{\in}\text{nat.} \ \forall s{\in}\text{nat.} \ \langle r,s\rangle{\in}\text{Le} \rightarrow N{\in}\text{I} by blast

then have \exists r{\in}\text{nat.} \ \forall s{\in}\text{nat.} \ \langle r,s\rangle{\in}\text{Le} \rightarrow N{\in}\text{U} using topology0.Top_2_L1[of \text{T,}\cup\{U\}\cup\{U\}\cup\{U\}. \langle U,W \rangle \in\{V{\in}T. m{\in}V\times T\}\}

\text{doble_point_top assms(2)}

unfolding topology0_def by auto

\}
then have \( \forall U \in \text{Pow}(\bigcup (U - \{m\}) \cup (U \cup \{V\} \cup \{W\} \cup \{W\})) \). \( U \in \text{Pow}(\{V \in T. m \in V\} \times T))) \). \( m \in \text{Interior}(U, U \cup \{V\} \cup \{W\} \cup \{W\} \cup \{W\}) \) by auto
moreover have \( \text{tt}: \text{topology0}(U \cup \{V\} \cup \{W\} \cup \{W\} \cup \{W\}) \) using \( \text{doble_point_top}[\text{OF assms(2)}] \) unfolding \( \text{topology0.NetConverges_def}[\text{OF tt Net}] \)
proof (auto)
unfolding \( \bigcup \) by auto
moreover have \( \text{tt}: \text{topology0}(U \cup \{V\} \cup \{W\} \cup \{W\} \cup \{W\}) \) using \( \text{doble_point_top}[\text{OF assms(2)}] \) using \( \text{topology0.NetConverges_def}[\text{OF tt Net}] \)
proof (auto)
unfolding \( \bigcup \) by auto

In particular, we also know that a locally-\( T_2 \) space doesn’t need to be KC;
since KC⇒US. Also we know that anti-hyperconnected spaces don’t need to be KC or US, since locally-\(T_2\)⇒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) anticompress_KC_equiv_T1:**

assumes \(T\{is anti-compact}\)

shows \(T\{is KC\} \iff T\{is T_1\}\)

**proof**

assume \(T\{is KC\}\)
then show \(T\{is T_1\}\) using KC_imp_T1 by auto

next
assume AS: \(T\{is T_1\}\)

\{
fix A assume A:A\{is compact in\}T A\in\Pow(\bigcup T)
then have A\{is compact in\}(T\{restricted to\}A) A\in\Pow(\bigcup T) using compact_imp_compact_subspace

Compact_is_card_nat by auto

moreover then have \(\bigcup(T\{restricted to\}A) = A\) unfolding RestrictedTo_def
by auto

ultimately have \((\bigcup(T\{restricted to\}A))\{is compact in\}(\bigcup(T\{restricted to\}A))\ A\in\Pow(\bigcup T)\) by auto

with assms have Finite(A) unfolding IsAntiComp_def antiProperty_def

using compact_spectrum by auto

then obtain n where n\in\nat A\approx n unfolding Finite_def by auto

then have A\prec\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 \(\bigcup T-(\bigcup T-A) < \nat\) by auto
then have \(\bigcup T-A \in \CoFinite \bigcup T\) unfolding Cofinite_def CoCardinal_def
by auto

then have \(\bigcup T-A \in T\) using AS T1_cocardinal_coarser by auto

with A(2) have A\{is closed in\}T unfolding IsClosed_def by auto

\}
then show \(T\{is KC\}\) 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 co-countable topology on \(\text{csucc}(\nat)\) is such a topology.

The co-countable topology on \(\nat^+\) is hyperconnected.
lemma cocountable_in_csucc_nat_HConn:
  shows (CoCountable csucc(nat)){is hyperconnected}
proof-
  { fix U V assume as:U∈(CoCountable csucc(nat))V∈(CoCountable csucc(nat)) U∩V=0
  then have csucc(nat)-U<csucc(nat)\U=0 csucc(nat)-V<csucc(nat)\V=0
    unfolding Cocountable_def CoCardinal_def by auto
  then have (csucc(nat)-U)∪(csucc(nat)-V)<csucc(nat)\U=0\V=0 using
    less_less_imp_un_less[OF _ _ InfCard_csucc[OF InfCard_nat]] by auto
  moreover
  { assume (csucc(nat)-U)∪(csucc(nat)-V)<csucc(nat) moreover
    have (csucc(nat)-U)∪(csucc(nat)-V)=csucc(nat)-U∩V by auto
      with as(3) have (csucc(nat)-U)∪(csucc(nat)-V)=csucc(nat) by auto
      ultimately have csucc(nat)<csucc(nat) by auto
      then have False by auto
  }
  ultimately have U=0\V=0 by auto
  }
then show (CoCountable csucc(nat)){is hyperconnected} unfolding IsHConnected_def
by auto
qed

The cocountable topology on $N^+$ is not anti-hyperconnected.

corollary cocountable_in_csucc_nat_notAntiHConn:
  shows ¬((CoCountable csucc(nat)){is anti-}IsHConnected)
proof
  assume as:(CoCountable csucc(nat)){is anti-}IsHConnected
  have (CoCountable csucc(nat)){is hyperconnected} using cocountable_in_csucc_nat_HConn
  by auto moreover
  have csucc(nat)≠0 using Ord_0_lt_csucc[OF Ord_nat] by auto
  then have uni:∪(CoCountable csucc(nat))=csucc(nat) using union_cocardinal
  unfolding Cocountable_def by auto
  have ∀A∈(CoCountable csucc(nat)). A⊆∪(CoCountable csucc(nat)) by fast
    with uni have ∀A∈(CoCountable csucc(nat)). A⊆csucc(nat) by auto
    then have ∀A∈(CoCountable csucc(nat)). csucc(nat)\A=A by auto
    ultimately have ((CoCountable csucc(nat)){restricted to}csucc(nat)){is
      hyperconnected}
    unfolding RestrictedTo_def by auto
    with as have (csucc(nat)){is in the spectrum of}IsHConnected unfolding
      antiProperty_def
      using uni by auto
    then have csucc(nat)≤1 using HConn_spectrum by auto
    then have csucc(nat)<nat using n_lesspoll_nat lesspoll_trans1 by auto
    then show False using lt_csucc[OF Ord_nat] lt_Card_imp_lesspoll[OF
      Card_csucc[OF Ord_nat]] lesspoll_trans by auto
  qed

1062
The cocountable topology on $\mathbb{N}^+$ is not $T_2$.

**Theorem** cocountable_in_csucc_nat_noT2:

- shows $\neg (\text{CoCountable } \text{csucc}(\mathbb{N}))\{\text{is } T_2\}$
- proof
  - assume $(\text{CoCountable } \text{csucc}(\mathbb{N}))\{\text{is } T_2\}$
  - then have antiHC:$(\text{CoCountable } \text{csucc}(\mathbb{N}))\{\text{is anti-}\text{IsHConnected}$
    - using topology0.T2_imp_anti_HConn[OF topology0_CoCardinal[OF InfCard_csucc[OF InfCard_nat]]]
  - unfolding Cocountable_def by auto
  - then show False using cocountable_in_csucc_nat_notAntiHConn by auto
- qed

The cocountable topology on $\mathbb{N}^+$ is $T_1$.

**Theorem** cocountable_in_csucc_nat_T1:

- shows $(\text{CoCountable } \text{csucc}(\mathbb{N}))\{\text{is } T_1\}$
- using cocardinal_is_T1[OF InfCard_csucc[OF InfCard_nat]] unfolding Cocountable_def by auto

The cocountable topology on $\mathbb{N}^+$ is anti-compact.

**Theorem** cocountable_in_csucc_nat_antiCompact:

- shows $(\text{CoCountable } \text{csucc}(\mathbb{N}))\{\text{is anti-compact}\}$
- proof
  - have noE:csucc(\mathbb{N})\neq0 using Ord_0_lt_csucc[OF Ord_nat] by auto
  - fix $A$ assume as:A\subseteq\bigcup(\text{CoCountable } \text{csucc}(\mathbb{N}))\{\text{restricted to}\}A$\{is compact in\}(\text{CoCountable } \text{csucc}(\mathbb{N}))\{\text{restricted to}\}A
  - from as(1) have ass:A\subseteq\text{csucc}(\mathbb{N})$ using union_cocardinal[OF noE] unfolding Cocountable_def by auto
  - have $(\text{CoCountable } \text{csucc}(\mathbb{N}))\{\text{restricted to}\}A$=CoCountable (A\cap\text{csucc}(\mathbb{N}))
  - using subspace_cocardinal
    - unfolding Cocountable_def by auto moreover
  - from ass have A\cap\text{csucc}(\mathbb{N})=A by auto
  - ultimately have $(\text{CoCountable } \text{csucc}(\mathbb{N}))\{\text{restricted to}\}A$=CoCountable A by auto
  - with as(2) have comp:$(\bigcup(\text{CoCountable } A))\{\text{is compact in\}(\text{CoCountable } A)$
  - by auto
    - assume as2:A\prec\text{csucc}(\mathbb{N})$ moreover
      - fix $t$ assume t:t\in A
        - have A-{t}\subseteq A by auto
          - then have A-{t}\subseteq A using subset_imp_lepoll by auto
            - with as2 have A-{t}\prec\text{csucc}(\mathbb{N})$ using lesspoll_trans1 by auto
    - moreover note noE
      - ultimately have $(A-{t})$\{is closed in\}(CoCountable A) using closed_sets_cocardinal[OF \text{csucc}(\mathbb{N})]
        - A-{t}A$ unfolding Cocountable_def by auto
          - then have A-{A-(t)}\subseteq(\text{CoCountable } A)$ unfolding IsClosed_def using union_cocardinal[OF noE, of A]
unfolding Cocountable_def by auto moreover
from t have A-(A-{t})={t} by auto ultimately
have \{t\}∈(CoCountable A) by auto
}\thence have r:∀ t∈A. \{t\}∈(CoCountable A) by auto

fix U assume U∈Pow(A)
\{ fix t assume t∈U 
with U r have t∈\{t\}⊆U \{t\}∈(CoCountable A) by auto 
then have ∃ V∈(CoCountable A). t∈V ∧ V⊆U by auto 
\} then have U∈(CoCountable A) using topology0.open_neigh_open[OF topology0_CoCardinal]
OF InfCard_csucc[OF InfCard_nat]] unfolding Cocountable_def by auto
\} then have Pow(A)⊆(CoCountable A) by auto moreover
\{ fix B assume B∈(CoCountable A)
then have B∈Pow(∪(CoCountable A)) by auto
then have B∈Pow(A) using union_cocardinal[OF noE] unfolding Cocountable_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 ∪(CoCountable A)=∪Pow(A) by auto
then have tot:∪(CoCountable A)=A by auto
from comp have (∪(CoCountable A){restricted to}){is compact in}(∪(CoCountable A)) using compact_imp_compact_subspace
Compact_is_card_nat tot unfolding RestrictedTo_def by auto
ultimately have A{is in the spectrum of}(λ T. (∪ T){is compact in}T) using comp tot unfolding IsAntiComp_def antiProperty_def by auto
moreover
\{ assume as1:-A≺csucc(nat)
from ass have A⊆csucc(nat) using subset_imp_lepoll by auto
with as1 have A≈csucc(nat) using lepoll_iff_leqpoll by auto
then have csucc(nat)≈A using eqpoll_sym by auto
then have nat≺A using ltpoll_eq_trans lt_csucc[OF Ord_nat]
lt_Card_imp_lepoll[OF Card_csucc[OF Ord_nat]] by auto
then have nat⊆A using lepoll_iff_leqpoll by auto
then obtain f where f∈inj(nat,A) unfolding lepoll_def by auto
moreover
\then have fun:f:nat→A unfolding inj_def by auto
\then have f∈surj(nat,range(f)) using fun_is_surj by auto

1064
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 $\text{as2} : \text{range}(f) < \text{csucc}(\text{nat})$ using lt_card_imp_lesspoll[OF Card_csucc[OF Ord_nat]]
lt_csucc[OF Ord_nat] eq_lesspoll_trans by auto
then have $\text{range}(f) \{\text{is closed in}\} (\text{CoCountable} \ A)$ using closed_sets_cocardinal[of csucc(nat)]
range(f)A unfolding Cocountable_def using func1_1_L5B[OF fun]
noE by auto
then have $(A \cap \text{range}(f)) \{\text{is compact in}\} (\text{CoCountable} \ A)$ using compact_closed
union_cocardinal[OF noE, of A]
compact_is_card_nat unfolding Cocountable_def by auto
moreover have $\text{int} : A \cap \text{range}(f) = \text{range}(f) \cap A = \text{range}(f)$ using
func1_1_L5B[OF fun] by auto
ultimately have $\text{range}(f) \{\text{is compact in}\} (\text{CoCountable} \ A) \{\text{restricted to}\} \text{range}(f))$
using compact_imp_compact_subspace
compact_is_card_nat by auto
moreover have $((\text{CoCountable} \ A) \{\text{restricted to}\} \text{range}(f)) = \text{CoCountable} \ (\text{range}(f) \cap A)$
using subspace_cocardinal unfolding Cocountable_def by auto
with int(2) have $((\text{CoCountable} \ A) \{\text{restricted to}\} \text{range}(f)) = \text{CoCountable} \ \text{range}(f)$ by auto
ultimately have $\text{comp2} : \text{range}(f) \{\text{is compact in}\} (\text{CoCountable} \ \text{range}(f))$
by auto
{
fix $t$ assume $t : t \in \text{range}(f)$
have $\text{range}(f) - \{t\} \subseteq \text{range}(f)$ by auto
then have $\text{range}(f) - \{t\} \subseteq \text{range}(f)$ using subset_imp_lepoll by auto
with as2 have $\text{range}(f) - \{t\} < \text{csucc}(\text{nat})$ using lesspoll_trans1 by auto
moreover note noE
ultimately have $(\text{range}(f) - \{t\}) \{\text{is closed in}\} (\text{CoCountable} \ \text{range}(f))$
using closed_sets_cocardinal[of csucc(nat)]
range(f)-{t}range(f) unfolding Cocountable_def by auto
then have $\text{range}(f) - (\text{range}(f) - \{t\}) = \{t\}$ using IsClosed_def unfolding union_cocardinal[OF noE, of range(f)]
unfolding Cocountable_def by auto
moreover from $t$ have $\text{range}(f) - (\text{range}(f) - \{t\}) = \{t\}$ by auto
ultimately have $\{t\} \in (\text{CoCountable} \ \text{range}(f))$ by auto
}
then have $r : \forall t \in \text{range}(f) \cdot \{t\} \in (\text{CoCountable} \ \text{range}(f))$ by auto
{
fix $U$ assume $U : U \in \text{Pow}(\text{range}(f))$
{
fix $t$ assume $t \in U$
with $U$ have $t \in \{t\} \subseteq U \{t\} \in (\text{CoCountable} \ \text{range}(f))$ by auto
then have $\exists V \in (\text{CoCountable} \ \text{range}(f)). t \in V \land V \subseteq U$ by auto
}
then have \( U \subseteq (\text{CoCountable range}(f)) \) using topology0.open_neigh_open[OF topology0_CoCardinal[OF InfCard_csucc[OF InfCard_nat]]] unfolding Cocountable_def by auto

then have \( \text{Pow}(\text{range}(f)) \subseteq (\text{CoCountable range}(f)) \) by auto moreover 

\( \text{fix } B \text{ assume } B \subseteq (\text{CoCountable range}(f)) \)

then have \( B \subseteq \text{Pow}(\bigcup (\text{CoCountable range}(f))) \) by auto

then have \( B \subseteq \text{Pow}(\text{range}(f)) \) using union_cocardinal[OF noE] unfolding Cocountable_def by auto

ultimately have \( p : \text{Pow}(\text{range}(f)) = (\text{CoCountable range}(f)) \) by blast

then have \( (\text{CoCountable range}(f)) \{\text{is anti-compact}\} \) using pow_anti_compact[of range(f)] by auto

moreover from \( p \) have \( \bigcup (\text{CoCountable range}(f)) = \bigcup \text{Pow}(\text{range}(f)) \) by auto

then have \( \text{Finite}(\text{range}(f)) \) using compact_spectrum by auto

then have \( \text{Finite}(\text{nat}) \) using e_eqpoll_imp_Finite_iff by auto

ultimately have \( \text{False} \) using nat_not_Finite by auto

\}

ultimately have \( A \{\text{is in the spectrum of}\} (\lambda T. (\bigcup T) \{\text{is compact in}\} T) \)

by auto

\}

then have \( \forall A \subseteq \text{Pow}(\bigcup (\text{CoCountable csucc(nat))) \bigcup (\text{CoCountable csucc(nat)) \{\text{restricted to}\} A)) \{\text{is compact in}\} \text{(CoCountable csucc(nat)) \{restricted to}\} A) \)

\( \rightarrow (A \{\text{is in the spectrum of}\} (\lambda T. (\bigcup T) \{\text{is compact in}\} T)) \)

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.

\begin{enumerate}
\item theorem cofinite_not_US:
\end{enumerate}

\item shows \( \neg ((\text{CoFinite nat}) \{\text{is US}\}) \)

\item proof
\begin{enumerate}
\item assume \( A : (\text{CoFinite nat}) \{\text{is US}\} \)
\item let \( N = \text{id}(\text{nat}) \)
\end{enumerate}

1066
have \( f : \mathbb{N} \to \mathbb{N} \) using \( \text{id}_\mathbb{N} \) by auto
then have \( \text{fun} : \mathbb{N} \to \bigcup (\text{CoCardinal}(\mathbb{N}, \mathbb{N})) \) using \( \text{union}_\text{cocardinal} \) unfolding \( \text{Cofinite}\_\text{def} \) by auto
then have \( \text{dom} : \text{domain}(	ext{fun}) = \mathbb{N} \) using \( \text{func1}_1\_\text{L1} \) by auto
with \( \text{fun} \) have \( \text{NET} : \langle \mathbb{N}, \text{Le} \rangle \) is a net on \( \bigcup (\text{CoCardinal}(\mathbb{N}, \mathbb{N})) \) unfolding \( \text{IsNet}\_\text{def} \)
using \( \text{Le}_\text{directs}_\mathbb{N} \) by auto
have \( \text{tot} : \bigcup (\text{CoCardinal}(\mathbb{N}, \mathbb{N})) = \mathbb{N} \) using \( \text{union}_\text{cocardinal} \) by auto
\{ fix \( U \) n assume \( U : U \in \text{Pow}(\bigcup (\text{CoFinite} \mathbb{N})) \) n \( \in \) \( \text{Interior}(U, (\text{CoFinite} \mathbb{N})) \)
have \( \text{Interior}(U, (\text{CoFinite} \mathbb{N})) \in (\text{CoFinite} \mathbb{N}) \) using \( \text{topology0}_\text{Top}_2\_\text{L2} \) unfolding \( \text{Cofinite}\_\text{def} \) by auto
then have \( \text{nat-interior}(U, (\text{CoFinite} \mathbb{N})) \subset \mathbb{N} \) using \( \text{U(2)} \) unfolding \( \text{Cofinite}\_\text{def} \)
\( \text{CoCardinal}\_\text{def} \) by auto
then have \( \text{Finite}(\text{nat-interior}(U, (\text{CoFinite} \mathbb{N}))) \) using \( \text{lesspoll}_\mathbb{N}\_\text{is}_\text{Finite} \) by auto
moreover have \( \text{nat-U} \subseteq \text{nat-interior}(U, (\text{CoFinite} \mathbb{N})) \) using \( \text{topology0}_\text{Top}_2\_\text{L1} \) unfolding \( \text{Cofinite}\_\text{def} \) by auto
ultimately have \( \text{fin} : \text{Finite}(\text{nat-U}) \) using \( \text{subset}_\text{Finite} \) by auto
moreover have \( \text{lin} : \text{IsLinOrder}(\mathbb{N}, \text{Le}) \) using \( \text{Le}_\text{directs}_\mathbb{N}(1) \) by auto
then have \( \text{IsLinOrder}(\text{nat-U}, \text{Le}) \) using \( \text{ord_linear_subset}[\mathbb{N} \text{ Le} \text{nat-U}] \) by auto
ultimately have \( r : \text{nat-U} = 0 \lor (\forall r \in \text{nat-U}. \langle r, \text{Maximum}(\text{Le}, \text{nat-U}) \rangle \in \text{Le}) \) using \( \text{linord}\_\text{max}_\text{props}(3) [\text{of nat}\_\text{U}\_\text{lenat-U}] \) unfolding \( \text{FinPow}\_\text{def} \) by auto
\{ assume \( \text{reg} : \forall s \in \mathbb{N}. \exists r \in \mathbb{N}. \langle s, r \rangle \in \text{Le} \land \text{Nr} \notin U \) with \( r \) have \( s : (\forall r \in \text{nat-U}. \langle r, \text{Maximum}(\text{Le}, \text{nat-U}) \rangle \in \text{Le}) \) nat-U\( \neq 0 \) using \( \text{apply_type}[\text{OF } f] \) by auto
have \( \text{Maximum}(\text{Le}, \text{nat-U}) \subset \text{nat} \) using \( \text{linord}\_\text{max}_\text{props}(2) [\text{OF } \text{lin } s(2)] \) unfolding \( \text{FinPow}\_\text{def} \) by auto
then have \( \text{succ}(\text{Maximum}(\text{Le}, \text{nat-U})) \subset \text{nat} \) using \( \text{nat}\_\text{succI} \) by auto
with \( r \) have \( \exists r \in \text{nat}. (\text{succ}(\text{Maximum}(\text{Le}, \text{nat-U})), r) \in \text{Le} \land \text{Nr} \notin U \) by auto
then obtain \( r \) where \( r \_\text{def} : \text{r} \in \text{nat} \) \( (\text{succ}(\text{Maximum}(\text{Le}, \text{nat-U})), r) \in \text{Le} \land \text{Nr} \notin U \) by auto
from \( r \_\text{def}(1, 3) \) have \( \text{Nr} \in \text{nat-U} \) using \( \text{apply_type}[\text{OF } f] \) by auto
with \( s(1) \) have \( (\text{Nr}, \text{Maximum}(\text{Le}, \text{nat-U})) \in \text{Le} \) by auto
then have \( r : (\text{Maximum}(\text{Le}, \text{nat-U})) \in \text{Le} \) using \( \text{id}_\text{conv} r \_\text{def}(1) \) by auto
then have \( r : \text{succ}(\text{Maximum}(\text{Le}, \text{nat-U})) \) by auto
with \( r \_\text{def}(2) \) have \( r < r \) using \( \text{lt}_\text{trans2} \) by auto
then have \( \text{False} \) by auto
\}
then have \( \exists s \in \mathbb{N}. \forall r \in \mathbb{N}. (s, r) \in \text{Le} 
\Rightarrow \text{Nr} \in U \) by auto
\} then have \( \forall n \in \mathbb{N}. \forall U \in \text{Pow}(\bigcup (\text{CoFinite} \mathbb{N})). n \in \text{Interior}(U, (\text{CoFinite} \mathbb{N})) \Rightarrow (\exists s \in \mathbb{N}. \forall r \in \mathbb{N}. (s, r) \in \text{Le} \Rightarrow \text{Nr} \in U) \) by auto
1067
with tot have \( \forall n \in \bigcup (\text{CoCardinal}(\text{nat},\text{nat})). \forall U \in \text{Pow}(\bigcup (\text{CoCardinal}(\text{nat},\text{nat}))). \)
\( n \in \text{Interior}(U, \text{CoCardinal}(\text{nat},\text{nat})) \rightarrow (\exists s \in \text{nat}. \forall r \in \text{nat}. (s,r) \in \text{Le} \rightarrow Nr \in U) \)
unfolding Cofinite_def by auto
then have \( \forall n \in \bigcup (\text{CoCardinal}(\text{nat},\text{nat})). (\langle N,\text{Le} \rangle \rightarrow N n \in \{\text{in}\}(\text{CoCardinal}(\text{nat},\text{nat}))) \)
unfolding topology0.NetConverges_def[of topology0_CoCardinal[of InfCard_nat]
NET]
using dom by auto
with tot have \( \forall n \in \text{nat}. (\langle N,\text{Le} \rangle \rightarrow N n \in \{\text{in}\}(\text{CoFinite nat})) \)
unfolding Cofinite_def by auto
then have \( (\langle N,\text{Le} \rangle \rightarrow N 0 \in \{\text{in}\}(\text{CoFinite nat})) \land (\langle N,\text{Le} \rangle \rightarrow N 1 \in \{\text{in}\}(\text{CoFinite nat})) \land 0 \neq 1 \) 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 \( \text{csucc}(\text{nat}) \).

theorem extension_pow_top:
\( \text{shows} \ (\text{Pow}(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable} \text{ csucc(nat)})-{0})\{\text{is a topology}\}

proof-
have noE:csucc(nat)\( \neq 0 \) using Ord_0_lt_csucc[of Ord_nat] by auto
\{ fix M assume M:M\( \subseteq \)co{\text{Pow}(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable} \text{ csucc(nat)})-{0})\}
let MP=\( \{U \in M. U \in \text{Pow}(\text{csucc}(\text{nat}))\}
let MN=\( \{U \in M. U \notin \text{Pow}(\text{csucc}(\text{nat}))\}
have unM:\( \bigcup M = (\bigcup MP) \cup (\bigcup MN) \) by auto
have csucc(nat)\( \notin \text{csucc(nat)} \) using mem_not_refl by auto
with M have MN=MN=\( \{U \in M. U = \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable} \text{ csucc(nat)})-{0})\}
by auto
have unMP:\( \bigcup MP \in \text{Pow}(\text{csucc}(\text{nat})) \) by auto
then have MN=0\( \rightarrow \bigcup M = (\text{Pow}(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable} \text{ csucc(nat)})-{0})\)
using unM by auto moreover
\{ assume MN\( \neq 0 \)
with MN have \( \{U \in M. U = \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable} \text{ csucc(nat)})-{0})\} \neq 0 \)
by auto
then obtain U where U:M \( \subseteq \{\text{csucc(nat)}\} \cup S. S \in (\text{CoCountable} \text{ csucc(nat)})-{0}) \)
by blast
then obtain S where S:U=\( \{\text{csucc(nat)}\} \cup S \in (\text{CoCountable} \text{ csucc(nat)})-{0} \)
by auto
with U MN have csucc(nat)\( \subseteq U \in MN \) by auto
then have a1:csucc(nat)\( \subseteq \bigcup MN \) by auto
let SC=\( \{S \in (\text{CoCountable} \text{ csucc(nat)}) \cup \text{csucc(nat)}\} \cup S \in M \}
have unSC:\( \bigcup SC \in (\text{CoCountable} \text{ csucc(nat)}) \) using CoCar_is_topology[of
InfCard_csucc[DF InfCard_nat]
  unfolding IsATopology_def unfolding Ccountable_def by blast
  { fix s assume s∈(csucc(nat))∪∪SC
    then have s=csucc(nat)∨s∈∪∪SC by auto
    then have s∈∪∪MN∨∃S∈SC. s∈S) using a1 by auto
    then have s∈∪∪MN∨(∃S∈SC. {csucc(nat)}∪∪S∈M
    ∧ s∈S) by auto
    with MN have s∈∪∪MN∨∃S∈(CoCountable csucc(nat)). {csucc(nat)}∪∪S∈MN
    ∧ s∈S) by auto
    then have s∈∪∪MN by blast
  }
  then have (csucc(nat))∪∪SC∪∪MN by blast
  moreover
  { fix s assume s∈∪∪MN
    then obtain U where U:s∈U U∈M U∈Pow(csucc(nat)) by auto
    with M have U∈{(csucc(nat))∪∪S. S∈(CoCountable csucc(nat))) by auto
    then obtain S where S:U={csucc(nat)}∪∪S S∈(CoCountable csucc(nat))
    by auto
    with U(1) have s=csucc(nat)∨s∈S by auto
    with S U(2) have s=csucc(nat)∨s∈∪∪SC by auto
  }
  then have ∪∪MN⊆(csucc(nat))∪∪SC by blast
  ultimately have unMN:∪∪MN=(csucc(nat))∪∪SC by auto
  from unSC have bl:csucc(nat)−∪∪SC<csucc(nat)∪∪SC=0 unfolding Ccountable_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 U(2) have x∈{csucc(nat)} by auto
      then have x=csucc(nat) by auto
    }
    then have S⊆{csucc(nat)} by auto
    with S U(1) have S={csucc(nat)} by auto
    with S U(1) have csucc(nat)−{csucc(nat)}<csucc(nat) unfolding Ccountable_def
  CoCardinal_def
  by auto
moreover
then have $\text{csucc(nat)}-\{\text{csucc(nat)}\} = \text{csucc(nat)}$ using \text{mem_not_refl[of csucc(nat)]} by force

ultimately have False by auto

} then have $0 \notin SC$ by auto moreover
  from $S \cup \{1\}$ have $S \subseteq SC$ by auto
  ultimately have $S \subseteq \bigcup SC \neq 0$ by auto
  then have noe:$\bigcup SC \neq 0$ by auto

moreover have $\text{csucc(nat)}- (\bigcup SC \cup MP) \subseteq \text{csucc(nat)}- \bigcup SC$ using \text{subset_imp_lepoll} by auto

ultimately have $\text{csucc(nat)}- (\bigcup SC \cup MP) \prec \text{csucc(nat)}$ using \text{lesspoll_trans1} by auto

moreover have $\bigcup SC \subseteq \bigcup (\text{CoCountable csucc(nat)})$ using \text{unSC} by auto

then have $\bigcup SC \subseteq \text{csucc(nat)}$ using \text{union_cocardinal[OF noE]} unfolding \text{Cocountable_def} by auto

ultimately have $(\bigcup SC \cup MP) \in (\text{CoCountable csucc(nat)})$ using \text{unMP} unfolding \text{Cocountable_def CoCardinal_def} by auto

then have $\{\text{csucc(nat)}\} \cup (\bigcup SC \cup MP) \in (\text{Pow(csucc(nat))} \cup \{\text{csucc(nat)}\} \cup S. S \in (\text{CoCountable csucc(nat)})-\{0\}$)

ultimately have $\bigcup M \in (\text{Pow(csucc(nat))} \cup \{\text{csucc(nat)} \cup S. S \in (\text{CoCountable csucc(nat)})-\{0\}$)

moreover

\{ fix $U$ $V$ assume $UV : U \in (\text{Pow(csucc(nat))} \cup \{\text{csucc(nat)}\} \cup S. S \in (\text{CoCountable csucc(nat)})-\{0\}) V \in (\text{Pow(csucc(nat))} \cup \{\text{csucc(nat)}\} \cup S. S \in (\text{CoCountable csucc(nat)})-\{0\})$

  \{ assume $\text{csucc(nat)} \notin U \cup \text{csucc(nat)} \notin V$

    with $UV$ have $U \in \text{Pow(csucc(nat))} \forall V \in \text{Pow(csucc(nat))}$ by auto

    then have $U \cap V \in \text{Pow(csucc(nat))}$ by auto

    then have $U \cap V \in (\text{Pow(csucc(nat))} \cup \{\text{csucc(nat)}\} \cup S. S \in (\text{CoCountable csucc(nat)})-\{0\}$)

  } moreover

  \{ assume $\text{csucc(nat)} \in U \setminus \text{csucc(nat)} \in V$

    then obtain $SU$ $SV$ where $S : U = \{\text{csucc(nat)}\} \cup SU$ $V = \{\text{csucc(nat)}\} \cup SV$
SU ∈ (CoCountable csucc(nat))-{0}  
SV ∈ (CoCountable csucc(nat))-{0}  using UV mem_not_refl by auto
from S(1,2) have U∩V={csucc(nat)}∪(SU∩SV) by auto moreover
from S(3,4) have SU<SV∈(CoCountable csucc(nat)) using CoCar_is_topology[OF InfCard_csucc[OF InfCard_nat]] unfolding IsATopology_def
unfolding Cocountable_def by blast moreover
from S(3,4) have SU∩SV∈(CoCountable csucc(nat)) unfolding IsATopology_def
unfolding IsHConnected_def
ultimately have U∩V∈{{csucc(nat)}∪S. S∈(CoCountable csucc(nat))}-{0}} by auto
ultimately have U∩V∈(Pow(csucc(nat))∪{{csucc(nat)}∪S. S∈(CoCountable csucc(nat))}-{0}})
ultimately show thesis unfolding IsATopology_def by auto
qed

This topology is defined over \( N^+ \cup \{N^+\} \) or \( \text{c succ}(n at) \cup \text{c succ}(n at) \).

lemma extension_pow_union:
  shows \( \bigcup (\text{Pow}(\text{c succ}(n at)) \cup \{\bigcup \{\text{c succ}(n at)\} ∪ S. S∈(\text{Co Countable csucc(nat))}-{0}}) = \text{c succ}(n at) ∪ \{\bigcup \{\text{c succ}(n at)\} ∪ S. S∈(\text{Co Countable csucc(nat))}-{0}}) \)
proof
  have noE:csucc(nat)≠0 using Ord_0_lt_csucc[OF Ord_nat] by auto
  have \( \bigcup (\text{Pow}(\text{c succ}(n at)) \cup \{\bigcup \{\text{c succ}(n at)\} ∪ S. S∈(\text{Co Countable csucc(nat))}-{0}}) = \bigcup (\text{Pow}(\text{c succ}(n at)) \cup (\bigcup \{\text{c succ}(n at)\} ∪ S. S∈(\text{Co Countable csucc(nat))}-{0}})) \)
  by blast
  also have ...=csucc(nat) ∪ (\bigcup \{\text{c succ}(n at)\} ∪ S. S∈(\text{Co Countable csucc(nat))}-{0}})
  by auto
  ultimately have A:0:1∪(\text{Pow}(\text{c succ}(n at)) \cup \{\bigcup \{\text{c succ}(n at)\} ∪ S. S∈(\text{Co Countable csucc(nat))}-{0}})
  by auto
  have \( \bigcup (\text{Co Countable csucc(nat)}) \in (\text{Co Countable csucc(nat)}) \) using CoCar_is_topology[OF InfCard_csucc[OF InfCard_nat]]
  unfolding IsATopology_def Cocountable_def by auto
  then have csucc(nat)∈(CoCountable csucc(nat)) using union_cocardinal[OF noE] unfolding Cocountable_def
  by auto
  with noE have csucc(nat)∈(CoCountable csucc(nat))-{0} by auto
  then have \( \{\text{c succ}(n at)\} ∪ \text{c succ}(n at) \subseteq \bigcup \{\text{c succ}(n at)\} ∪ S. S∈(\text{Co Countable csucc(nat))}-{0}} \)
  by auto
  then have \( \{\text{c succ}(n at)\} ∪ \text{c succ}(n at) \subseteq \bigcup (\text{Pow}(\text{c succ}(n at)) \cup \{\bigcup \{\text{c succ}(n at)\} ∪ S. S∈(\text{Co Countable csucc(nat))}-{0}}) \)
  by blast
  with A show csucc(nat)∪csucc(nat)⊆∪(Pow(csucc(nat)) ∪ \{csucc(nat)∪S. S∈(CoCountable csucc(nat))-{0}})

1071
by auto
{
  fix x assume x:x∈(∪{{csucc(nat)}∪S. S∈(CoCountable csucc(nat))−{0}})
x≠csucc(nat)
  then obtain U where U:U∈{{csucc(nat)}∪S. S∈(CoCountable csucc(nat))−{0}}
x∈U by blast
  then obtain S where S:U={csucc(nat)}∪S S∈(CoCountable csucc(nat))−{0}
  by auto
    with U(2) x(2) have x∈S by auto
  then have x∈csucc(nat) using union_cocardinal[OF noE]
    unfolding Cocountable_def by auto
}
then have (∪{{csucc(nat)}∪S. S∈(CoCountable csucc(nat))−{0}})⊆csucc(nat)
∪{csucc(nat)} by blast
  with A show ∪(Pow(csucc(nat)) ∪ {{csucc(nat)}∪S. S∈(CoCountable csucc(nat))−{0}})⊆csucc(nat)
    by blast
qed

This topology has a discrete open subspace.

lemma extension_pow_subspace:
  shows (Pow(csucc(nat)) ∪ {{csucc(nat)}∪S. S∈(CoCountable csucc(nat))−{0}}){restricted
to}csucc(nat)=Pow(csucc(nat))
  and csucc(nat)∈(Pow(csucc(nat)) ∪ {{csucc(nat)}∪S. S∈(CoCountable
  csucc(nat))−{0}})
proof
  show csucc(nat)∈(Pow(csucc(nat)) ∪ {{csucc(nat)}∪S. S∈(CoCountable
  csucc(nat))−{0}}) by auto
  {
    fix x assume x∈(Pow(csucc(nat)) ∪ {{csucc(nat)}∪S. S∈(CoCountable
    csucc(nat))−{0}}){restricted to}csucc(nat)
    then obtain R where x=csucc(nat)∩R R∈(Pow(csucc(nat)) ∪ {{csucc(nat)}∪S.
    S∈(CoCountable csucc(nat))−{0}}) 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
  {
    fix x assume x:x∈Pow(csucc(nat))
    then have x=csucc(nat)∩x by auto
      with x have x∈(Pow(csucc(nat)) ∪ {{csucc(nat)}∪S. S∈(CoCountable
      csucc(nat))−{0}}){restricted to}csucc(nat)
        unfolding RestrictedTo_def by auto
      }
    then show Pow(csucc(nat))⊆(Pow(csucc(nat)) ∪ {{csucc(nat)}∪S. S∈(CoCountable
    csucc(nat))−{0}}){restricted to}csucc(nat) by auto
qed

This topology is Hausdorff.
theorem extension_pow_T2:
shows (Pow(succ(nat)) ∪ {succ(nat)} ∪ S. S ∈ (CoCountable succ(nat)) - {0})) is T_2
proof
have noE:succ(nat) ≠ 0 using Ord_0_lt_csucc[OF Ord_nat] by auto
{ fix A B assume A ∈ (Pow(succ(nat)) ∪ {succ(nat)} ∪ S. S ∈ (CoCountable succ(nat)) - {0})) B ∈ (Pow(succ(nat)) ∪ {succ(nat)} ∪ S. S ∈ (CoCountable succ(nat)) - {0})) A ≠ B
then have AB:A ∈ succ(nat) ∪ {succ(nat)} B ∈ succ(nat) ∪ {succ(nat)} A ≠ B using extension_pow_union by auto
{ assume A ≠ succ(nat) B ≠ succ(nat)
then have A ∈ succ(nat) B ∈ succ(nat) using AB by auto
then have sub:(A) ∈ (Pow(succ(nat)) ∪ {succ(nat)} ∪ S. S ∈ (CoCountable succ(nat)) - {0})) {B} ∈ (Pow(succ(nat)) ∪ {succ(nat)} ∪ S. S ∈ (CoCountable succ(nat)) - {0})) {restricted to}succ(nat)
by auto
then obtain RA RB where \{A\} = succ(nat) ∩ RA \{B\} = succ(nat) ∩ RB RA ∈ (Pow(succ(nat)) ∪ \{succ(nat)\} ∪ S. S ∈ (CoCountable succ(nat)) - {0})) RB ∈ (Pow(succ(nat)) ∪ \{succ(nat)\} ∪ S. S ∈ (CoCountable succ(nat)) - {0}))
unfolding RestrictedTo_def by auto
then have \{A\} ∈ (Pow(succ(nat)) ∪ \{succ(nat)\} ∪ S. S ∈ (CoCountable succ(nat)) - {0})) \{B\} ∈ (Pow(succ(nat)) ∪ \{succ(nat)\} ∪ S. S ∈ (CoCountable succ(nat)) - {0}))
using extension_pow_subspace(1) extension_pow_top unfolding IsATopology_def by auto
moreover from AB(3) have \{A\} ∩ \{B\} = 0 by auto ultimately have \exists U ∈ (Pow(succ(nat)) ∪ \{succ(nat)\} ∪ S. S ∈ (CoCountable succ(nat)) - {0})). \exists V ∈ (Pow(succ(nat)) ∪ \{succ(nat)\} ∪ S. S ∈ (CoCountable succ(nat)) - {0})). \{A\} ∩ \{B\} = 0 by auto
} moreover { assume A = succ(nat) ∨ B = succ(nat)
with AB(3) have disj:(A = succ(nat) ∧ B ≠ succ(nat)) ∨ (B = succ(nat) ∧ A ≠ succ(nat)) by auto
{ assume ass:A = succ(nat) ∧ B ≠ succ(nat)
then have p:B ∈ succ(nat) using AB(2) by auto
have \{B\} ≠ 1 using singleton_eqpoll_1 by auto
then have \{B\} ≠ nat using eq_lesspoll_trans n_lesspoll_nat by auto
then have \{B\} ⊆ nat using lesspoll_imp_lepoll by auto
then have \{B\} ⊆ succ(nat) using Card_less_succ_eq_le[OF Card_nat] by auto
} by auto

1073
with p have {B} {is closed in} (CoCountable csucc(nat)) unfolding Cocountable_def
  using closed_sets_cocardinal [OF noE] by auto
then have csucc(nat)-{B} ∈ (CoCountable csucc(nat)) unfolding IsClosed_def
  Cocountable_def using union_cocardinal [OF noE] by auto more-
over
  
  assume csucc(nat)-{B}=0
  with p have csucc(nat)={B} by auto
  then have csucc(nat)≈1 using singleton_eqpoll_1 by auto
lesspoll_imp_lepoll by auto
  then have csucc(nat)≺csucc(nat) using Card_less_csucc_eq_le [OF Card_nat] by auto
ultimately have {csucc(nat)} ∪ (csucc(nat)-{B}) ∈ {{csucc(nat)}∪S. S ∈ (CoCountable
  csucc(nat))-{0}} by auto
  then have U1:{csucc(nat)} ∪ (csucc(nat)-{B}) ∈ (Pow(csucc(nat)) ∪
  {{csucc(nat)}∪S. S ∈ (CoCountable csucc(nat))-{0}}) by auto
  have {B}∈Pow(csucc(nat)) using p by auto
  then have {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 R where R∈Pow(csucc(nat)) ∪ {{csucc(nat)}∪S. S ∈ (CoCountable
  csucc(nat))-{0}} {B}=csucc(nat) ∩ R
  unfolding RestrictedTo_def by auto
  then have U2:{B}∈Pow(csucc(nat)) ∪ {{csucc(nat)}∪S. S ∈ (CoCountable
  csucc(nat))-{0}} using extension_pow_subspace(2)
  extension_pow_top unfolding IsATopology_def by auto
  have ((csucc(nat)) ∪ (csucc(nat)-{B}))∩{B}=0 using p mem_not_refl [OF
csucc(nat)] by auto
  with U1 U2 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}}.
  A∈U∧B∈V∧U∩V=0 using ass(1) 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

1074
with \( p \) have \( \{ A \} \) is closed in \( \text{CoCountable} \) \( csucc(nat) \) unfolding \text{Cocountable_def} using \( \text{closed_sets_cocardinal}[OF noE] \) by auto
then have \( csucc(nat) - \{ A \} \in \text{CoCountable} \) \( csucc(nat) \) unfolding \text{IsClosed_def} \text{Cocountable_def} using \( \text{union_cocardinal}[OF noE] \) by auto more-
over
{ assume \( csucc(nat) - \{ A \} = 0 \)
with \( p \) have \( csucc(nat) = \{ A \} \) by auto
then have \( csucc(nat) \approx 1 \) using \( \text{singleton_eqpoll_1} \) by auto
ultimately have \( \{ csucc(nat) \} \cup (csucc(nat) - \{ A \}) \in \{ \{ csucc(nat) \} \cup S. S \in \text{CoCountable} \} - \{ 0 \} \) by auto
ultimately have \( \{ A \} \in \text{Pow}(csucc(nat)) \) unfolding \text{RestrictedTo_def} by auto
ultimately have \( \{ A \} \leq \text{csucc}(nat) \) unfolding \text{IsATopology_def} by auto
ultimately have \( \{ A \} \cap (csucc(nat) - \{ A \}) = 0 \) using \( \text{mem_not_refl[of csucc(nat)]} \) by auto
ultimately have \( \{ A \} \in \text{csucc}(nat) \) unfolding \text{Bex_def} by auto
ultimately have \( \{ A \} \in \text{csucc}(nat) \) unfolding \text{Bex_def} by auto
ultimately have \( \{ A \} \in \text{csucc}(nat) \) unfolding \text{Bex_def} by auto
ultimately have \( \{ A \} \in \text{csucc}(nat) \) unfolding \text{Bex_def} by auto
ultimately have \( \exists U \in \text{Pow}(\text{csucc(nat)}) \cup \{\text{csucc(nat)}\} \cup S. S \in (\text{CoCountable csucc(nat)}) - \{0\} \). 
\( \exists V \in \text{Pow}(\text{csucc(nat)}) \cup \{\text{csucc(nat)}\} \cup S. S \in (\text{CoCountable csucc(nat)}) - \{0\} \).

\( A \in U \land B \in V \land U \cap V = 0 \) by auto

then show thesis unfolding \text{isT2_def} by auto 

qed

The topology we built is not discrete; i.e., not every set is open.

\text{theorem extension_pow_notDiscrete:} 
\text{shows} \{\text{csucc(nat)}\} \notin (\text{Pow}(\text{csucc(nat)}) \cup \{\text{csucc(nat)}\} \cup S. S \in (\text{CoCountable csucc(nat)}) - \{0\})

\text{proof}
\begin{itemize}
\item assume \{\text{csucc(nat)}\} \notin (\text{Pow}(\text{csucc(nat)}) \cup \{\text{csucc(nat)}\} \cup S. S \in (\text{CoCountable csucc(nat)}) - \{0\})
\item then have \{\text{csucc(nat)}\} \notin (\{\text{csucc(nat)}\} \cup S. S \in (\text{CoCountable csucc(nat)}) - \{0\})
\end{itemize}

using \text{mem_not_refl} by auto
\begin{itemize}
\item then obtain \( S \) where \( S:S \in (\text{CoCountable csucc(nat)}) - \{0\} \) \{\text{csucc(nat)}\} \{\text{csucc(nat)}\} \cup S
\end{itemize}
by auto
\begin{itemize}
\item fix \( x \) assume \( x \in S \)
\item then have \( x \in \{\text{csucc(nat)}\} \cup S \) by auto
\item with \text{(2)} have \( x \in \{\text{csucc(nat)}\} \) by auto
\item then have \( x = \text{csucc(nat)} \) by auto
\end{itemize}
then have \( S \subseteq \{\text{csucc(nat)}\} \) by auto
\begin{itemize}
\item with \text{(1)} have \( S = \{\text{csucc(nat)}\} \) by auto
\item with \text{(1)} have \( \text{csucc(nat)} - \{\text{csucc(nat)}\} = \text{csucc(nat)} \) unfolding \text{Cocountable_def}
\item CoCardinal_def
\end{itemize}
by auto moreover
\begin{itemize}
\item then have \( \text{csucc(nat)} - \{\text{csucc(nat)}\} = \text{csucc(nat)} \) using \text{mem_not_refl[of csucc(nat)]} by force
\item ultimately show False by auto
\end{itemize}

qed

The topology we built is anti-compact.

\text{theorem extension_pow_antiCompact:} 
\text{shows} (\text{Pow}(\text{csucc(nat)}) \cup \{\text{csucc(nat)}\} \cup S. S \in (\text{CoCountable csucc(nat)}) - \{0\}) \{\text{is anti-compact}\}

\text{proof}
\begin{itemize}
\item have \( \text{noE:csucc(nat)} \neq 0 \) using \text{Ord_0_lt_csucc[OF Ord_nat]} by auto
\item \begin{itemize}
\item fix \( K \) assume \( K:K \subseteq \bigcup (\text{Pow}(\text{csucc(nat)}) \cup \{\text{csucc(nat)}\} \cup S. S \in (\text{CoCountable csucc(nat)}) - \{0\})
\end{itemize}
\end{itemize}
\[
\bigcup \left( (\text{Pow}(\text{csucc}(\text{nat}))) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable } \text{csucc}(\text{nat})) \{\text{restricted to } K}\right) \text{ is compact in } \bigcup \left( (\text{Pow}(\text{csucc}(\text{nat}))) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable } \text{csucc}(\text{nat})) \{\text{restricted to } K}\right)
\]

from \(K(1)\) have \(\text{sub}:K \subseteq \text{csucc}(\text{nat}) \cup \{\text{csucc}(\text{nat})\}\) using \text{extension}_\text{pow}_\text{union}

by auto

have \(\bigcup \left( (\text{Pow}(\text{csucc}(\text{nat}))) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable } \text{csucc}(\text{nat})) \{\text{restricted to } K}\right) = (\text{csucc}(\text{nat}) \cup \{\text{csucc}(\text{nat})\} \cap K
\)

using \text{extension}_\text{pow}_\text{union} \text{unfolding} \text{RestrictedTo_def} by auto moreover

from \(\text{sub}\) have \((\text{csucc}(\text{nat}) \cup \{\text{csucc}(\text{nat})\}) \cap K = K\) by auto

ultimately have \(\bigcup \left( (\text{Pow}(\text{csucc}(\text{nat}))) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable } \text{csucc}(\text{nat})) \{\text{restricted to } K}\right) = K\) by auto

with \(K(2)\) have \(K \text{is compact in } (\bigcup \left( (\text{Pow}(\text{csucc}(\text{nat}))) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable } \text{csucc}(\text{nat})) \{\text{restricted to } K}\right)\) by auto

then have \(\text{comp}:K \text{is compact in } (\bigcup \left( (\text{Pow}(\text{csucc}(\text{nat}))) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable } \text{csucc}(\text{nat})) \{\text{restricted to } K}\right)
\)

using \text{compact}_\text{subspace}_\text{imp}_\text{compact} by auto

\[
\{ \begin{align*}
&\text{assume ss:}:K \subseteq \text{csucc}(\text{nat}) \\
&\text{then have } K \text{is compact in } (\bigcup \left( (\text{Pow}(\text{csucc}(\text{nat}))) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable } \text{csucc}(\text{nat})) \{\text{restricted to } K}\right)) = K \text{ by auto} \\
&\text{using compact}_\text{imp}_\text{compact}_\text{subspace} \text{ comp Compact_is_card_nat by auto}
\end{align*}
\]

then have \(K \text{is compact in } \text{Pow}(\text{csucc}(\text{nat}))\) using \text{extension}_\text{pow}_\text{subspace}(1)

by auto

then have \(K \text{is compact in } (\bigcup \left( (\text{Pow}(\text{csucc}(\text{nat}))) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable } \text{csucc}(\text{nat})) \{\text{restricted to } K}\right))\) using \text{compact}_\text{imp}_\text{compact}_\text{subspace}

Compact_is_card_nat by auto

then have \(K \text{is compact in } (\bigcup \left( (\text{Pow}(\text{csucc}(\text{nat}))) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable } \text{csucc}(\text{nat})) \{\text{restricted to } K}\right))\) by auto

ultimately have \(\bigcup \left( (\text{Pow}(\text{csucc}(\text{nat}))) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable } \text{csucc}(\text{nat})) \{\text{restricted to } K}\right)\) is compact in \(\left( \bigcup \left( (\text{Pow}(\text{csucc}(\text{nat}))) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable } \text{csucc}(\text{nat})) \{\text{restricted to } K}\right)\right)\) using \text{pow}_\text{anti}_\text{compact}

\text{unfolding} \text{IsAntiComp_def} \text{ antiProperty_def} using ss by auto

}

moreover

\[
\{ \begin{align*}
&\text{assume } \neg (K \subseteq \text{csucc}(\text{nat})) \\
&\text{with sub have } \text{csucc}(\text{nat}) \subseteq K \text{ by auto} \\
&\text{with sub have ss: } K \subseteq \text{csucc}(\text{nat}) \subseteq \text{csucc}(\text{nat}) \text{ by auto} \\
&\{ \begin{align*}
&\text{assume prec: } K \subseteq \text{csucc}(\text{nat}) \\
&\text{then have } (K \subseteq \text{csucc}(\text{nat})) \{\text{is closed in } (\text{CoCountable } \text{csucc}(\text{nat}))\} \text{ by auto} \\
&\text{using closed_sets_cocardinal[OF noE] ss}\text{ unfolding Cocardinal_def by auto}
\end{align*}
\end{align*}
\]

then have \(\text{csucc}(\text{nat}) \subseteq (K \subseteq \text{csucc}(\text{nat})) \subseteq (\text{CoCountable } \text{csucc}(\text{nat}))\) unfolding \text{IsClosed_def} using \text{union}_\text{cocardinal}[OF noE] by auto more-
over

{  
  assume csucc(nat)-(K-{csucc(nat)})=0
  with ss have csucc(nat)=(K-{csucc(nat)}) by auto
  with prec have False by auto
}
ultimately have {csucc(nat)} \cup (csucc(nat)-(K-{csucc(nat)}))∈{(csucc(nat))∪S.
  S∈(CoCountable csucc(nat))-\{0\})
  by auto
moreover have {csucc(nat)} \cup (csucc(nat)-(K-{csucc(nat)}))=({csucc(nat)}
U csucc(nat))-K-{csucc(nat)} by blast
ultimately have ({csucc(nat)} \cup csucc(nat))-K-{csucc(nat)})∈{(csucc(nat))∪S.
  S∈(CoCountable csucc(nat))-\{0\}) by auto
then have ({csucc(nat)} \cup csucc(nat))-K-{csucc(nat)})∈(Pow(csucc(nat))
U \{\{csucc(nat)\}∪S. S∈(CoCountable csucc(nat))-\{0\}\})
  by auto moreover
have csucc(nat) \cup \{csucc(nat)\}=\{csucc(nat)\} \cup csucc(nat) by auto
ultimately have (csucc(nat) \cup csucc(nat))-K-{csucc(nat)})∈(Pow(csucc(nat))
U \{\{csucc(nat)\}∪S. S∈(CoCountable csucc(nat))-\{0\}\})
  by auto
then have (∪(Pow(csucc(nat)) \cup \{\{\{csucc(nat)\}∪S. S∈(CoCountable
  csucc(nat))-\{0\}\})-K-{csucc(nat)})∈(Pow(csucc(nat)) \cup \{\{csucc(nat)\}∪S.
  S∈(CoCountable csucc(nat))-\{0\}\})
  by auto
using extension_pow_union by auto
then have (K-{csucc(nat)})\{is closed in}(Pow(csucc(nat)) \cup \{\{\{csucc(nat)\}∪S.
  S∈(CoCountable csucc(nat))-\{0\}\})
  by auto unfolding IsClosed_def using ss by auto
with comp have (K\cap(K-{\{csucc(nat)\}))\{is compact in}(Pow(csucc(nat))
U \{\{csucc(nat)\}∪S. S∈(CoCountable csucc(nat))-\{0\}\}) using compact_closed
Compact_is_card_nat by auto
moreover have K\cap(K-{\{csucc(nat)\})=K-{\{csucc(nat)\}) by auto
ultimately have (K-{\{csucc(nat)\})\{is compact in}(Pow(csucc(nat))
U \{\{csucc(nat)\}∪S. S∈(CoCountable csucc(nat))-\{0\}\}) by auto
with ss have (K-{\{csucc(nat)\})\{is compact in}(Pow(csucc(nat))
U \{\{csucc(nat)\}∪S. S∈(CoCountable csucc(nat))-\{0\}\})(restricted to)\{csucc(nat)
  by auto
using compact_imp_compact_subspace comp Compact_is_card_nat
by auto
then have (K-{\{csucc(nat)\})\{is compact in}(Pow(csucc(nat)) using extension_pow_subspace(1) by auto
then have (K-{\{csucc(nat)\})\{is compact in}(Pow(csucc(nat))(restricted
to}(K-{\{csucc(nat)\}) using compact_imp_compact_subspace
Compact_is_card_nat by auto moreover
have ∪(Pow(csucc(nat))(restricted to}(K-{\{csucc(nat)\}))=K-{\{csucc(nat)\})
using ss unfolding RestrictedTo_def by auto
ultimately have (∪(Pow(csucc(nat))(restricted to}(K-{\{csucc(nat)\}))\{is
  compact in}(Pow(csucc(nat))(restricted to}(K-{\{csucc(nat)\})) by auto
then have (K-{\{csucc(nat)\})\{is in the spectrum of}(\{\{\}\})\{is
  compact in\{\\}) using pow_anti_compact
unfolding IsAntiComp_def antiProperty_def using ss by auto

1078
then have Finite(K-{csucc(nat)}) using compact_spectrum by auto
  moreover have Finite({csucc(nat)}) by auto ultimately
  have Finite(K) using Diff_Finite[of {csucc(nat)} K] by auto
  then have K{is in the spectrum of} (\forall T. (S(T){is compact in} T)
  using compact_spectrum by auto
  moreover have Finite(K-{csucc(nat)}) by auto
  using Diff_Finite[of K-{csucc(nat)}]
  by auto
  ultimately have Finite(K) using Diff_Finite[of K-{csucc(nat)}]
  by auto

then have K{is in the spectrum of} (\forall T. (S(T){is compact in} T)
  using compact_spectrum by auto
  moreove assume \neg(K-{csucc(nat)}≺csucc(nat))
  with ss have K-{csucc(nat)}≤csucc(nat) using lepoll_iff_leqpoll
  subset_imp_lepoll[of K-{csucc(nat)}]
  then have csucc(nat)≤K-{csucc(nat)} using eqpoll_sym by auto
  then have nat≺K-{csucc(nat)} using lesspoll_eq_trans lt_csucc[OF Ord_nat]
  lt.Card_imp_leqpoll[OF Card_csucc[OF Ord_nat]] by auto
  then have nat≺K-{csucc(nat)} using lepoll_iff_leqpoll by auto
  then obtain f where f∈inj(nat,K-{csucc(nat)}) unfolding lepoll_def
  surj_def by auto
  then have fun: f:nat→K-{csucc(nat)} unfolding inj_def by auto
  then have e:range(f)≈nat using eqpoll_sym by auto
  ultimately have f∈bij(nat,range(f)) unfolding bij_def inj_def
  surj_def by auto
  then have nat≈range(f) unfolding eqpoll_def by auto
  then have e:range(f)≈nat using eqpoll_sym by auto
  then have as2:range(f)≺csucc(nat) using lt.Card_imp_leqpoll[OF Card_csucc[OF Ord_nat]]
  lt_csucc[OF Ord_nat] lt.Crd_imp_lesspoll_trans by auto
  then have range(f){is closed in}(CoCountable csucc(nat)) using
  closed_sets_cocardinal[of csucc(nat)]
  range(f)csucc(nat)] unfolding Cocountable_def using func1_1_L5B[OF fun] ss noE by auto
  then have csucc(nat)-(range(f))∈(CoCountable csucc(nat)) unfolding IsClosed_def
  Cocountable_def using union_cocardinal[OF noE] by auto
  moreover 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 \( \{\text{csucc}(\text{nat})\} \cup \text{csucc}(\text{nat}) \setminus \text{range}(f) \in (\powerset(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup \text{range}(f) \in (\powerset(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable } \text{csucc}(\text{nat})) \setminus \{0\}) \) by auto

moreover have \( \text{csucc}(\text{nat}) \cup \{\text{csucc}(\text{nat})\} = \{\text{csucc}(\text{nat})\} \cup \text{csucc}(\text{nat}) \) by auto

ultimately have \( (\text{csucc}(\text{nat}) \cup \{\text{csucc}(\text{nat})\}) \setminus \text{range}(f) \in (\powerset(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable } \text{csucc}(\text{nat})) \setminus \{0\}) \) by auto

then have \( \bigcup (\powerset(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable } \text{csucc}(\text{nat})) \setminus \{0\}) \setminus \text{range}(f) \in (\powerset(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable } \text{csucc}(\text{nat})) \setminus \{0\}) \) using \text{extension_pow_union} by auto

moreover have \( \text{range}(f) \subseteq \bigcup (\powerset(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable } \text{csucc}(\text{nat})) \setminus \{0\}) \) using \text{ss func1_1_L5B[OF fun]} by auto

ultimately have \( (\text{range}(f)) \setminus \text{is closed in}(\powerset(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable } \text{csucc}(\text{nat})) \setminus \{0\}) \) unfolding \text{IsClosed_def} by blast

moreover have \( \text{range}(f) \setminus \text{is compact in}(\powerset(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable } \text{csucc}(\text{nat})) \setminus \{0\}) \) using \text{compact_closed} by auto

ultimately have \( (\text{range}(f)) \setminus \text{is compact in}(\powerset(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable } \text{csucc}(\text{nat})) \setminus \{0\}) \) by auto

then have \( (\bigcup (\powerset(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable } \text{csucc}(\text{nat})) \setminus \{0\}) \setminus \text{range}(f)) \setminus \text{range}(f) \in (\powerset(\text{csucc}(\text{nat})) \cup \{\text{csucc}(\text{nat})\} \cup S. S \in (\text{CoCountable } \text{csucc}(\text{nat})) \setminus \{0\}) \) using \text{extension_pow_subspace(1)} by auto

ultimately have \( (\text{range}(f)) \in \text{the spectrum of}(\lambda T. (\bigcup T) \setminus \text{is compact in} T) \) using \text{pow_anti_compact[of csucc(nat)]} unfolding \text{IsAntiComp_def} \text{antiProperty_def} using ss \text{func1_1_L5B[OF fun]} by auto

then have \( \text{Finite}(\text{range}(f)) \) using \text{compact_spectrum} by auto moreover

then have \( \text{Finite}(\text{nat}) \) using \text{eqpoll_imp_Finite_iff} by auto ultimately have \( \text{False} \) using \text{nat_not_Finite} by auto

ultimately have \( K \) \{is in the spectrum of\}( \lambda T. (\bigcup T) \in \text{is compact in} T) \) by auto

1080
ultimately have \( K \) is in the spectrum of (\( \lambda T \) (\( \bigcup T \)) is compact in \( T \)) by auto

then show thesis unfolding IsAntiComp_def antiProperty_def by auto

qed

If a topological space is KC, then its one-point compactification is US.

theorem (in topology0) KC_imp_OP_comp_is_US:
  assumes \( T \) is KC
  shows \( \{ \text{one-point compactification of} T \} \) is US
proof-
  { fix \( N \), \( x \), \( y \) assume \( A : N : \text{nat} \rightarrow \bigcup \{ \text{one-point compactification of} T \} \) \( N, \text{Le} \rightarrow N \) \( x \in (\text{one-point compactification of} T) \) \( N, \text{Le} \rightarrow N \) \( y \in (\text{one-point compactification of} T) \) \( x \neq y \)
    have dir: Le directs nat using \( \text{Le directs nat(2)} \).
    from \( A(1) \) have dom: domain(\( N \)) = nat using \( \text{func1_1_L1} \) by auto
    with dir \( A(1) \) have NET: \( \{ N, \text{Le} \} \) is a net on \( \bigcup \{ \text{one-point compactification of} T \} \) unfolding \( \text{IsNet_def} \) by auto
    have xy: \( x \in \bigcup \{ \text{one-point compactification of} T \} \)
      unfolding topology0.NetConverges_def [OF \( _\text{NET} \), of \( x \)]
    unfolding topology0_def using \( \text{op_comp_is_top dom op_compact_total} \) by auto
    from \( A(2) \) have comp: \( \forall U \in \text{Pow}(\bigcup \{ \text{one-point compactification of} T \}). \)
      \( x \in \text{Interior}(U, \{ \text{one-point compactification of} T \}) \) \( \forall t \in \text{nat}. \forall m \in \text{nat}. \langle t, m \rangle \in \text{Le} \rightarrow N \hspace{1em} m \in U \) using topology0.NetConverges_def [OF \( _\text{NET} \), of \( x \)]
    unfolding topology0_def using \( \text{op_comp_is_top dom op_compact_total} \) by auto
    from \( A(3) \) have op2: \( \forall U \in \text{Pow}(\bigcup \{ \text{one-point compactification of} T \}). \)
      \( y \in \text{Interior}(U, \{ \text{one-point compactification of} T \}) \) \( \forall t \in \text{nat}. \forall m \in \text{nat}. \langle t, m \rangle \in \text{Le} \rightarrow N \hspace{1em} m \in U \) using topology0.NetConverges_def [OF \( _\text{NET} \), of \( y \)]
    unfolding topology0_def using \( \text{op_comp_is_top dom op_compact_total} \) by auto
  }
  assume p: \( x \in \bigcup T \) \( y \in \bigcup T \)
  { assume B: \( \exists n \in \text{nat.} \forall m \in \text{nat}. (n, m) \in \text{Le} \rightarrow N \hspace{1em} m = \bigcup T \)
    have \( \bigcup T \in (\{ \text{one-point compactification of} T \}) \) using \( \text{open_subspace} \) 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 \( \text{op_comp_is_top} \) by auto
    then have \( x \in \text{Interior}(\bigcup T, \{ \text{one-point compactification of} T \}) \) using \( p(1) \) by auto
    moreover

1081
have $\bigcup T \subseteq \text{Pow}(\bigcup \{\text{one-point compactification of } T\})$ using open_subspace(1)
by auto
ultimately have $\exists t \in \text{domain}(\text{ fst}(\{N, \text{Le}\}))$. $\forall m \in \text{domain}(\text{fst}(\{N, \text{Le}\}))$.
$(t, m) \in \text{snd}(\{N, \text{Le}\}) \rightarrow \text{fst}(\{N, \text{Le}\})$ $m \in \bigcup T$ using A(2)
using topology0.NetConverges_def[OF _ NET] op_comp_is_top unfolding topology0_def by blast
then have $\exists t \in \text{nat}$. $\forall m \in \text{nat}$. $(t, m) \in \text{Le} \rightarrow N$ $m \in \bigcup T$ using dom by auto
then obtain $t$ where $t : t : \text{nat} \ \forall m \in \text{nat}$. $(t, m) \in \text{Le} \rightarrow N$ $m \in \bigcup T$
by auto
from $B$ obtain $n$ where $n : n \in \text{nat} \ \forall m \in \text{nat}$. $(n, m) \in \text{Le} \rightarrow N = \bigcup T$ by auto
from $t(1)$ $n(1)$ dir obtain $z$ where $z : z \in \text{nat} \ \langle n, z \rangle \in \text{Le} \ \langle t, z \rangle \in \text{Le}$ 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 $\text{reg} : \forall n \in \text{nat}$. $\exists m \in \text{nat}$. $N m \notin \bigcup T \land (n, m) \in \text{Le}$ by auto
let $\text{NN} = \{(n, N(\mu \ i. \ Ni \neq \bigcup T \land (n, i) \in \text{Le})) \cdot n \in \text{nat}\}$
{
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 $s : n \in \text{nat}$
with $\text{reg}$ obtain $m$ where $N m \neq \bigcup T \land (n, m) \in \text{Le}$ $m \in \text{nat}$ by auto
then have $LI : N(\mu \ i. \ Ni \neq \bigcup T \land (n, i) \in \text{Le}) \neq \bigcup T \land (n, i) \in \text{Le}$ using LeastI[of $\lambda m$. $N m \neq \bigcup T \land (n, m) \in \text{Le}$ $m$]
nat_into_Ord[of $m$] by auto
then have $(\mu \ i. \ Ni \neq \bigcup T \land (n, i) \in \text{Le}) \in \text{nat}$ by auto
then have $N(\mu \ i. \ Ni \neq \bigcup T \land (n, i) \in \text{Le}) \in \bigcup \{\text{one-point compactification of } T\}$ using apply_type[OF A(1)] op_comp_total by auto
with as have $(n, N(\mu \ i. \ Ni \neq \bigcup T \land (n, i) \in \text{Le})) \in \text{nat} \times \bigcup \{\text{one-point compactification of } T\}$ by auto
}
then have $\text{NN} \in \text{Pow}(\text{nat} \times \bigcup \{\text{one-point compactification of } T\})$ by auto
ultimately have $\text{NFun} : \text{NN} : \text{nat} \rightarrow \bigcup \{\text{one-point compactification of } T\}$ unfolding Pi_def function_def domain_def by auto
{
fix $n$ assume $s : n \in \text{nat}$

with reg obtain \( m \) where \( \text{Nm} \neq \bigcup T \land \langle n, m \rangle \in \text{Le} \) \( m \in \text{nat} \) by auto

then have \( \text{LI} : \forall \mu. \text{Nm} \neq \bigcup T \land \langle n, \mu \rangle \in \text{Le} \) using \( \text{LeastI} \) of \( \lambda m. \text{Nm} \neq \bigcup T \land \langle n, m \rangle \in \text{Le} \) \( m \)

\( \text{nat_into_Ord} \) of \( m \) by auto

then have \( \text{NNn} \neq \bigcup T \) using \( \text{apply_equality} \) of \( \text{OF NFun} \) by auto

then have \( \forall n \in \text{nat}. \text{NNn} \neq \bigcup T \) by auto

then have \( \forall n \in \text{nat}. \text{NNn} \in \bigcup T \) using \( \text{apply_type} \) of \( \text{NFun} \) \( \text{op_compact_total} \) by auto

then have \( \text{R} : \text{NN} : \text{nat} \to \bigcup T \) using \( \text{func1_1_L1A} \) of \( \text{NFun} \) by auto

have \( \text{dom2} : \text{domain}(\text{NN}) = \text{nat} \) by auto

then have \( \text{net2} : \langle \text{NN}, \text{Le} \rangle \) is a net on \( \bigcup T \) unfolding \( \text{IsNet_def} \) using \( \text{R} \) dir by auto

\{ fix \( U \) assume \( U : U \subseteq \bigcup T \) \( x \in \text{int}(U) \)

have \( \text{intT} : \text{int}(U) \subseteq \bigcup T \) using \( \text{Top_2_L2} \) by auto

then have \( \text{int}(U) \subseteq \{ \text{one-point compactification of} \ T \} \) unfolding \( \text{OPCompactification_def} \) by auto

\{ fix \( s \) assume \( \text{AA} : \langle r, s \rangle \in \text{Le} \)

with reg obtain \( m \) where \( \text{Nm} \neq \bigcup T \land \langle s, m \rangle \in \text{Le} \) by auto

then have \( \langle s, \mu \rangle \in \text{Le} \) using \( \text{LeastI} \) of \( \lambda m. \text{Nm} \neq \bigcup T \land \langle n, m \rangle \in \text{Le} \) \( m \)

\( \text{nat_into_Ord} \) by auto

with \( \text{AA} \) have \( \langle r, \mu \rangle \in \text{Le} \) using \( \text{le_trans} \) by auto

with \( \text{r_def}(2) \) have \( N(\mu \land \langle s, \mu \rangle \in \text{Le} \) \( \in U \) by blast

then have \( \text{NNs} \subseteq U \) using \( \text{apply_equality} \) of \( \text{OF NFun} \) \( \text{AA} \) by auto

\}

then have \( \forall s \in \text{nat}. \langle r, s \rangle \in \text{Le} \to \text{NNs} \subseteq U \) by auto

with \( \text{r_def}(1) \) have \( \exists r \in \text{nat}. \forall s \in \text{nat}. \langle r, s \rangle \in \text{Le} \to \text{NNs} \subseteq U \) by auto

\}

then have \( \forall U \in \text{Pow}(\bigcup T). x \in \text{int}(U) \)

\to \( \langle r \in \text{nat}. \forall s \in \text{nat}. \langle r, s \rangle \in \text{Le} \to \text{NN} \ s \in U \) by auto

then have \( \text{conx} : \langle \text{NN}, \text{Le} \rangle \to N x(\text{in}) \bigcup T \) using \( \text{NetConverges_def} \) of \( \text{OF net2} \)

p(1) \( \text{op_comp_is_top} \)

unfolding \( \text{topology0_def} \) using xy(1) \( \text{dom2} \) by auto

\{ fix \( U \) assume \( U : U \subseteq \bigcup T \) \( y \in \text{int}(U) \)
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) by auto
with intT 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 ∀s∈nat. ⟨r,s⟩∈Le → Ns∈U using
Top_2_L1 by auto
{
  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 using LeastI[of λm.
Nm≠∪T ∧ ⟨s,m⟩∈Le m]
  nat_into_Ord by auto
  with AA have ⟨r,µ i. Ni≠∪T ∧ ⟨s,i⟩∈Le using le_trans by auto
  with r_def(2) have N(µ i. Ni≠∪T ∧ ⟨s,i⟩∈Le)∈U by blast
  then have NNs∈U using apply Equality[OF _ NFUn] AA by auto
}
then have ∀s∈nat. ⟨r,s⟩∈Le → NNs∈U by auto
with r_def(1) have ⟨r,s⟩∈Le → NNs∈U by auto

then have ∀U∈Pow(∪T). y ∈ int(U)
   → (∃r∈nat. ∀s∈nat. ⟨r, s⟩ ∈ Le ↔ NN s ∈ U) by auto
then have cony:(NN,Le)→N y{in}T using NetConverges_def[OF net2]
p(2) op_comp_is_top
unfolding topology0_def using xy(2) dom2 by auto
with conx assms have x=y using KC_imp_US unfolding IsUS_def using
R by auto
with A(4) have False by auto
}
moreover
{
  assume AAA:x∈∪TVy∈∪T
  with pp have x∈∪TVy∈∪T by auto
  
  assume x:x∈∪T
  with A(4) have y∈∪T using pp(2) by auto
  
  assume B:∃n∈nat. ∀m∈nat. ⟨n,m⟩∈Le → Nm=∪T
  have ∪T∈({one-point compactification of}T) using open_subspace
  by auto
  then have ∪T=Interior(∪T,{one-point compactification of}T)
  using topology0.Top_2_L3

1084
unfolding topology0_def using op_comp_is_top by auto
then have \( y \in \text{Interior}(\bigcup T, \{\text{one-point compactification of}\} T) \)
using y by auto moreover
have \( \bigcup T \in \text{Pow}(\{\text{one-point compactification of}\} T) \)
using open_subspace(1) by auto
ultimately have \( \exists t \in \text{domain}(\text{fst}(\langle N, \text{Le} \rangle)). \forall m \in \text{domain}(\text{fst}(\langle N, \text{Le} \rangle)). \langle t, m \rangle \in \text{snd}(\langle N, \text{Le} \rangle) \rightarrow \text{fst}(\langle N, \text{Le} \rangle) m \in \bigcup T \) using A(3)
unfolding topology0.NetConverges_def[OF _ NET] op_comp_is_top
then have \( \exists t \in \text{domain}(\text{fst}(\langle N, \text{Le} \rangle)). \forall m \in \text{domain}(\text{fst}(\langle N, \text{Le} \rangle)). \langle t, m \rangle \in \text{snd}(\langle N, \text{Le} \rangle) \rightarrow \text{fst}(\langle N, \text{Le} \rangle) m \in \bigcup T \) usingAuto
ultimately have \( \exists t \in \text{nat}. \forall m \in \text{nat}. \langle t, m \rangle \in \text{Le} \rightarrow N m \in \bigcup T \) by auto
then obtain \( t \) where \( t : t \in \text{nat} \land \forall m \in \text{nat}. \langle t, m \rangle \in \text{Le} \rightarrow N m \in \bigcup T \) by auto
from \( B \) obtain \( n \) where \( n : n \in \text{nat} \land \forall m \in \text{nat}. \langle n, m \rangle \in \text{Le} \rightarrow N m = \bigcup T \) by auto
from \( t(1) n(1) \) dir obtain \( z \) where \( z : z \in \text{nat} \land \langle n, z \rangle \in \text{Le} \land \langle t, z \rangle \in \text{Le} \) unfolding IsDirectedSet_def by auto
from \( t(2) z(1,3) \) have \( N z \in \bigcup T \) by auto
moreover from \( n(2) z(1,2) \) have \( N z = \bigcup T \) by auto
ultimately have \( \text{False} \) using mem_not_refl by auto
then have \( \text{reg} : \forall n \in \text{nat}. \exists m : m \in \text{nat} \land N m \neq \bigcup T \land \langle n, m \rangle \in \text{Le} \) by auto
let \( NN = \{ \langle n, N(\mu i. Ni \neq \bigcup T \land \langle n, i \rangle \in \text{Le}) \rangle : n \in \text{nat} \} \)
fix \( x \) \( z \) assume \( A1 : \langle x, z \rangle \in NN \)
fix \( y' \) assume \( A2 : \langle x, y' \rangle \in NN \)
with \( A1 \) have \( z = y' \) by auto
then have \( \forall y'. \langle x, y' \rangle \in NN \rightarrow z = y' \) by auto
then have \( \forall x z. \langle x, z \rangle \in NN \rightarrow (\forall y'. \langle x, y' \rangle \in NN \rightarrow z = y') \) by auto
moreover
fix \( n \) assume as : \( n \in \text{nat} \)
with \( \text{reg} \) obtain \( m \) where \( N m \neq \bigcup T \land \langle n, m \rangle \in \text{Le} \) m \in \text{nat} \) by auto
then have LI : \( \forall (\mu i. Ni \neq \bigcup T \land \langle n, i \rangle \in \text{Le}) \neq \bigcup T \) \( \langle n, i \rangle \in \text{Le} \) using LeastI[of \( \lambda m. N m \neq \bigcup T \land \langle n, m \rangle \in \text{Le} \) m]
then have \( \langle n, i \rangle \in \text{Le} \) using nat_into_Ord[of \( m \)] by auto
then have \( \langle n, i \rangle \in \text{Le} \) \( n \in \text{nat} \) by auto
then have \( \langle n, \mu i. Ni \neq \bigcup T \land \langle n, i \rangle \in \text{Le} \rangle \in \text{nat} \times \bigcup \{\text{one-point compactification of}\} T \) using apply_type[OF A(1)] op_compact_total by auto
then have \( \langle n, \mu i. Ni \neq \bigcup T \land \langle n, i \rangle \in \text{Le} \rangle \in \text{nat} \times \bigcup \{\text{one-point compactification of}\} T \) by auto
then have \( NN \in \text{Pow}(\text{nat} \times \bigcup \{\text{one-point compactification of}\} T) \) by auto
ultimately have \(\text{NFun} : \text{nat} \rightarrow \bigcup \{\text{one-point compactification of} T\}\)

unfolding \(\Pi_{\text{def}}\) \(\text{function_def}\) \(\text{domain_def}\) by auto

\{ 
  fix \(n\) assume as:\(n \in \text{nat}\)
  with \(\text{reg}\) obtain \(m\) where \(\text{Nm} \neq \bigcup T \land (n,m) \in \text{Le} m \in \text{nat}\) by auto 
  then have \(\text{LI} : (\mu i. \text{Ni} \neq \bigcup T \land (n,i) \in \text{Le}) \neq \bigcup T (n,m) \in \text{Le} m\)
  nat_into_Ord[of \(m\)] by auto 
  then have \(\text{NNn} \neq \bigcup T\) using apply_equality[OF \(\text{NFun}\)] by auto 
\} 

then have \(\forall n \in \text{nat}. \text{NNn} \neq \bigcup T\) by auto 
then have \(\forall n \in \text{nat}. \text{NNn} \in \bigcup T\) using apply_type[OF \(\text{NFun}\)] op_compact_total by auto 
then have \(\forall n \in \text{nat}. \text{NNn} \rightarrow \bigcup T\) using func1_1_L1A[OF \(\text{NFun}\)] by auto 

have \(\text{dom2} : \text{domain(NN)} = \text{nat}\) by auto 
then have \(\text{net2} : \langle \text{NN},\text{Le} \rangle\) {is a net on} \(\bigcup T\) unfolding IsNet_def using \(\text{R dir}\) by auto 

\{ 
  fix \(U\) assume \(\text{U(2)}\) have \(\text{intT} : \text{int}(U) \subseteq \bigcup T\) (\(\forall y \in \text{int}(U)\) unfolding \(\text{op2 op_compact_total}\) by auto 
  then obtain \(r\) where \(\text{r_def}(2)\) have \(N(\mu i. \text{Ni} \neq \bigcup T \land (s,i) \in \text{Le}) \in \text{U}\) by blast 
  with \(\text{r_def}(1)\) have \(\exists r \in \text{nat}. \forall s \in \text{nat}. \langle r,s \rangle \in \text{Le} \rightarrow \text{Ns} \in \text{int}(U)\) by auto 
  then have \(\forall s \in \text{nat}. \langle r,s \rangle \in \text{Le} \rightarrow \text{NNs} \in \text{U}\) by blast 
\} 

then have \(\forall U \in \text{Pow}(\bigcup T). \forall y \in \text{int}(U)\) unfolding NetConverges_def[OF \(\text{net2}\)] by auto 

1086
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 V: V ∈ T by auto
 then have V = int(V) using Top_2_L3 by auto
 with cony obtain r where r: r ∈ nat ∀ s ∈ nat. ⟨r, s⟩ ∈ Le → NNs ∈ 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 restrict(NN, {n ∈ nat. ⟨n, r⟩ ∈ Le}) ∈ surj({n ∈ nat. ⟨n, r⟩ ∈ Le}, range(restrict(NN, {n ∈ nat. ⟨n, r⟩ ∈ Le})))
 using fun_is_surj by auto
 moreover 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} ⊆ {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 have {n ∈ nat. ⟨n, j⟩ ∈ Le} ∪ {succ(j)} using subset_imp_lepoll by auto
 unfolding Ord_def
 Transset_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. ⟨n, j⟩ ∈ Le⟩ have t ∈ {n ∈ nat. ⟨n, j⟩ ∈ Le} by auto
 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
}

1087
as(2) Finite_cons

by auto

ultimately have Finite({n ∈ nat. (n, succ(j)) ∈ Le}) using subset_Finite

by auto

then have ∀ j ∈ nat. Finite({n ∈ nat. (n, j) ∈ Le}) → Finite({n ∈ nat. (n, succ(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 ] where P=λ t. Finite({n ∈ nat. ⟨n, t⟩ ∈ Le})

by auto

then have Finite((restrict(NN, {n ∈ nat. ⟨n, r⟩ ∈ Le})){n ∈ nat. ⟨n, r⟩ ∈ Le}) using range_image_domain[OF NresFun]

by auto

then have Finite((NN{n ∈ nat. ⟨n, r⟩ ∈ Le}){is in the spectrum of}(λ T. (∪ 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{restricted to}NN{n ∈ nat. ⟨n, r⟩ ∈ Le}))=NN{n ∈ nat. ⟨n, r⟩ ∈ Le}

using func1_1_L6(2)[OF R] by blast

moreover have (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:∀ M∈FinPow(M) (NN{n ∈ nat. ⟨n, r⟩ ∈ Le})⊂∪ N

unfolding IsCompact_def by blast

from N(1) have N ⊂{V}∈FinPow(M) using V(2) unfolding FinPow_def

by auto

moreover

{ fix s assume s:s∈A seg V
  with V(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

1088
with \( rr \) have \( \forall n \in V \) by auto
with \( n(2) \) \( s(2) \) have False by auto
}

then have \( \langle r, n \rangle \notin \text{Le} \) by auto
with \( rr(1) \) \( n(1) \) have \( (r \leq n) \) by auto
then have \( n \leq r \) using \text{Ord_linear_le[where thesis=\( \langle n, r \rangle \in \text{Le} \)]}
\text{nat_into_Ord[OF rr(1)]}
\text{nat_into_Ord[OF n(1)]} by auto

with \( rr(1) \) \( n(1) \) have \( \langle n, r \rangle \in \text{Le} \) by auto

then have \( \langle n, r \rangle \in \text{Le} \) by auto

with \( n(2) \) have \( \neg (r \leq n) \) by auto
then have \( n \leq r \) using \text{Ord_linear_le[where thesis=\( \langle n, r \rangle \in \text{Le} \)]}
\text{nat_into_Ord[OF rr(1)]}
\text{nat_into_Ord[OF n(1)]} by auto

with \( rr(1) \) \( n(1) \) have \( \langle n, r \rangle \in \text{Le} \) by auto

with \( n \in \{ n \in \text{nat}. \ t \in \{ n \in \text{nat}. \ (n, r) \in \text{Le} \} \} \) by auto
moreover have \( \{ n \in \text{nat}. \ (n, r) \in \text{Le} \} \subseteq \text{nat} \) by auto
ultimately have \( s \in \text{NN} \{ n \in \text{nat}. \ (n, r) \in \text{Le} \} \) using \text{func_imagedef[OF NFun]}

by auto
with \( N(2) \) have \( s \in \bigcup N \) by auto
}

then have \( \forall M \in \text{Pow}(T). \ A \subseteq \bigcup M \) by auto
then have \( \forall M \in \text{Pow}(T). \ A \subseteq \bigcup (M \cup \{V\}) \) by auto
ultimately have \( \exists N \in \text{FinPow}(M). \ A \subseteq \bigcup N \) by auto

have ss:A \subseteq \bigcup (T) using \text{func1_1_L6(2)[OF R]} y by blast ultimately have \( A \) \( \text{is compact in}(T) \) unfolding \text{IsCompact_def} by auto moreover
with assms have \( A \) \( \text{is closed in}(T) \) unfolding \text{IsKC_def IsCompact_def} by auto ultimately have \( A \subseteq \bigcup (T) \) using \text{ss} by auto
then have \( \{ \bigcup T \} \cup (\bigcup T-A) \subseteq (\{ \text{one-point compactification of} \ T \}) \) unfolding \text{OPCompactification_def}
by auto
then have \( \{ \bigcup T \} \cup (\bigcup T-A) = \text{Interior}(\{ \bigcup T \} \cup (\bigcup T-A), \{ \text{one-point compactification of} \ T \}) \) using \text{topology0.Top_2_L3 op_comp_is_top unfolding topology0_def} by auto
moreover have \( \forall U \in \text{Pow}(\bigcup (\{ \text{one-point compactification of} \ T \})) \) by auto

assume \( x \in A \)
with \( A(4) \) have \( x \in \text{NNnat} \) by auto
then have \( x \in \{ n \in \text{NNnat}. \ n \in \text{nat} \} \) using \text{func_imagedef[OF NFun] by auto}
then obtain \( n \) where \( n \in \text{natNNn=x} \) by auto
with noy \( x \) have False by auto
}

with \( y \) have \( x \in \{ \bigcup T \} \cup (\bigcup T-A) \) using \( x \) by force ultimately have \( x \in \text{Interior}(\{ \bigcup T \} \cup (\bigcup T-A), \{ \text{one-point compactification of} \ T \}) \) \( (\bigcup T) \cup (\bigcup T-A) \in \text{Pow}((\{ \text{one-point compactification of} \ T \})) \) using \text{op_compact_total} by auto
moreover have \( \forall U \in \text{Pow}(\bigcup (\{ \text{one-point compactification of} \ T \})) \) \( . \ x \in \text{Interior}(U, \{ \text{one-point compactification of} \ T \}) \) \( \rightarrow (\exists t \in \text{nat}. \ \forall m \in \text{nat}. \ (t, m) \in \text{Le} \rightarrow N \ m \in U) \)
using A(2) dom topology0.NetConverges_def[OF _ NET] op_comp_is_top
unfolding topology0_def by auto
ultimately have \( \exists t \in \text{nat.} \ \forall m \in \text{nat.} \ (t, m) \in L_e \rightarrow N \ m \in (U T) \cup (U T-A) \)
by blast
then obtain r where r_def:r \in \text{nat.} \ \forall s \in \text{nat.} \. \ (r, s) \in L_e \rightarrow N s \in (U T) \cup (U T-A) 
by auto
{ fix s assume AA:(r,s)\in L_e
with reg obtain m where Nm\neq U T \ (s,m)\in L_e by auto
then have \( (s,\mu) \ i. \ Ni\neq U T \land \ (s,i) \in L_e \) using LeastI[of \( \lambda m. \)
Nm\neq U T \land (s,m)\in L_e \]
\text{nat_into_Ord} by auto
with AA have \( (r,\mu) \ i. \ Ni\neq U T \land \ (s,i) \in L_e \) using le_trans by auto
with r_def(2) have N(\( \mu \ i. \ Ni\neq U T \land \ (s,i) \in L_e \))\in (U T) \cup (U T-A) 
by auto
then have NNs\in (U T) \cup (U T-A) using apply_equality[OF _ NFun]
AA by auto
with noy have NNs\in (U T-A) using AA by auto
moreover have NNs\in \{N\text{nt.} \ t\in \text{nat.}\} using AA by auto
then have NNs\in NNnat using func_imagedef[OF NFun] by auto
then have NNs\in A by auto
ultimately have False by auto
}
moreover have \( r \subseteq \text{succ}(r) \) using succ_explained by auto
then have \( r \subseteq \text{succ}(r) \) using subset_imp_le nat_into_Ord \( <r \in \text{nat} \) nat_succl by auto
then have \( (r,\text{succ}(r)) \in L_e \) using \( <r \in \text{nat} \) nat_succl by auto
ultimately have False by auto
}
then have \( x \neq U T \) by auto
with xy(1) AAA have yf\in U T x\in U T using op_comp_total by auto
with xy(2) have y:y=U T and x:x\in U T using op_comp_total by auto
{ assume B:\( \exists n \in \text{nat.} \ \forall m \in \text{nat.} \. \ (n,m) \in L_e \rightarrow Nm=U T \)
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 \( \exists t \in \text{domain(fst}(N, \text{Le})) \). \( \forall m \in \text{domain(fst}(N, \text{Le})) \). \( (t, m) \in \text{snd}(N, \text{Le}) \) \rightarrow \( \text{fst}(N, \text{Le}) \) \ m \in \bigcup T \) using A(2)
using topology0.NetConverges_def[OF _ NET] op_comp_is_top_unfolding topology0_def by blast
then have \( \exists t \in \text{nat.} \ \forall m \in \text{nat.} \. \ (t, m) \in L_e \rightarrow N \ m \in \bigcup T \) using dom
by auto
  then obtain t where t:t∈nat ∀m∈nat. (t, m) ∈ Le → N m ∈ \bigcup T
by auto
  from B obtain n where n:n∈nat ∀m∈nat. (n,m)∈Le → Nm=\bigcup T by auto
from t(1) n(1) dir obtain z where z:z∈nat ⟨n,z⟩∈Le (t,z)∈Le unfolding IsDirectedSet_def
  by auto
  from t(2) z(1,3) have Nz∈\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:∀n∈nat. \exists m∈nat. Nm≠\bigcup T ∧ ⟨n,m⟩∈Le by auto
let NN={(n,N(μ i. Ni≠\bigcup T ∧ ⟨n,i⟩∈Le)). n∈nat}
{  fix x z assume A1:(x, z) ∈ 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≠\bigcup T ∧ ⟨n,m⟩∈Le m∈nat by auto
  then have LI:N(μ i. Ni≠\bigcup T ∧ ⟨n,i⟩∈Le)≠\bigcup T (n,μ i. Ni≠\bigcup T ∧ ⟨n,i⟩∈Le)∈Le using LeastI[of λm. Nm≠\bigcup T ∧ ⟨n,m⟩∈Le]
       nat_into_Ord[of m] by auto
  then have (μ i. Ni≠\bigcup T ∧ ⟨n,i⟩∈Le)∈nat by auto
  then have N(μ i. Ni≠\bigcup T ∧ ⟨n,i⟩∈Le)∈\bigcup \{one-point compactification of T\} using apply_type[OF A(1)] op_compact_total by auto
       with an have (n,N(μ i. Ni≠\bigcup T ∧ ⟨n,i⟩∈Le))∈nat×\bigcup \{one-point compactification of T\} by auto
} then have NN∈Pow(nat×\bigcup \{one-point compactification of T\}) by auto
ultimately have NFun:NN:nat→\bigcup \{one-point compactification of T\} unfolding Pi_def function_def domain_def by auto
{  fix n assume as:n∈nat
  with reg obtain m where Nm≠\bigcup T ∧ ⟨n,m⟩∈Le m∈nat by auto
  then have LI:N(μ i. Ni≠\bigcup T ∧ ⟨n,i⟩∈Le)≠\bigcup T (n,μ i. Ni≠\bigcup T ∧ ⟨n,i⟩∈Le)∈Le using LeastI[of λm. Nm≠\bigcup T ∧ ⟨n,m⟩∈Le]
       nat_into_Ord[of m] by auto
  then have NNN≠\bigcup T using apply_equality[OF _ NFun] by auto
} then have nøy:∀n∈nat. NNN≠\bigcup T by auto

1091
then have \( \forall n \in \text{nat.} \, \text{NNn} \in \bigcup T \) using apply_type[of NFun] op_compact_total by auto
then have \( R : \text{NN} : \text{nat} \rightarrow \bigcup T \) using func1_1_L1A[of NFun] by auto
have \( \text{dom2 : domain(\text{NN}) = \text{nat} by auto} \)
then have \( \text{net2 : } \langle \text{NN, Le} \rangle \{\text{is a net on}\} \bigcup T \) unfolding IsNet_def using R dir by auto
\{
fix \( U \) assume \( U : U \subseteq \bigcup T \, x \in \text{int}(U) \)
have \( \text{intT : int}(U) \subseteq T \) using Top_2_L2 by auto
then have \( \text{int}(U) \in \{\text{one-point compactification of } T\} \) unfolding OPCompactification_def by auto
\{
fix \( U \) assume \( U : U \subseteq \bigcup T \, x \in \text{int}(U) \)
then have \( \langle \text{r, s} \rangle \in \text{le} \rightarrow \text{NNs} \in U \) using comp op_compact_total by auto
then obtain \( r \) where \( r \in \text{nat} \) \( \forall s \in \text{nat.} \, \langle r, s \rangle \in \text{le} \rightarrow \text{NNs} \in \bigcup \text{U} \) using Top_2_L1 by auto
\{
fix \( s \) assume \( \text{AA : } \langle r, s \rangle \in \text{le} \)
with reg obtain \( m \) where \( \text{Nm} \neq \bigcup T \wedge \langle s, m \rangle \in \text{le} \) using LeastI[of \( \lambda \text{m.} \text{Nm} \neq \bigcup T \wedge \langle s, m \rangle \in \text{le} \)] nat_into_Ord by auto
with \( \text{AA} \) have \( \langle r, \mu \rangle \in \text{le} \rightarrow \text{NNs} \in \bigcup \text{U} \) using le_trans by auto
with \( r \in \text{nat} \) \( \forall s \in \text{nat.} \, \langle r, s \rangle \in \text{le} \rightarrow \text{NNs} \in \bigcup \text{U} \) using Top_2_L1 by auto
\{
fix \( M \) assume \( \text{Acov : } A \subseteq \bigcup M \subseteq T \)
then have \( x \in \bigcup M \) by auto
then obtain \( V \) where \( V : x \in V \, V \in M \) by auto
with \( \text{Acov}(2) \) have \( \text{V = int}(V) \) using Top_2_L3 by auto
with \( V \in \text{int}(V) \) by auto
with cony VT obtain \( r \) where \( r \in \text{nat} \) \( \forall s \in \text{nat.} \, \langle r, s \rangle \in \text{le} \rightarrow \text{NNs} \in V \) using NetConverges_def[of net2] x op_compact_is_top unfolding topology0_def using xy(2) dom2 by auto
let \( A = \{x\} \cap \text{NNnat} \)
\{
fix \( M \) assume \( \text{Acov : } A \subseteq \bigcup M \subseteq T \)
then have \( x \in \bigcup M \) by auto
then have \( V : x \in V \, V \in M \) by auto
with \( \text{Acov}(2) \) have \( \text{V = int}(V) \) using Top_2_L3 by auto
with \( V \in \text{int}(V) \) by auto
with cony VT obtain \( r \) where \( r \in \text{nat} \) \( \forall s \in \text{nat.} \, \langle r, s \rangle \in \text{le} \rightarrow \text{NNs} \in V \) using NetConverges_def[of net2] x op_compact_is_top unfolding topology0_def using xy(2) dom2 by auto
let \( A = \{x\} \cap \text{NNnat} \)
\{
fix \( M \) assume \( \text{Acov : } A \subseteq \bigcup M \subseteq T \)
then have \( x \in \bigcup M \) by auto
then have \( V : x \in V \, V \in M \) by auto
with \( \text{Acov}(2) \) have \( \text{V = int}(V) \) using Top_2_L3 by auto
with \( V \in \text{int}(V) \) by auto
with cony VT obtain \( r \) where \( r \in \text{nat} \) \( \forall s \in \text{nat.} \, \langle r, s \rangle \in \text{le} \rightarrow \text{NNs} \in V \) using NetConverges_def[of net2] x op_compact_is_top unfolding topology0_def using xy(2) dom2 by auto
let \( A = \{x\} \cap \text{NNnat} \)
\{
fix \( M \) assume \( \text{Acov : } A \subseteq \bigcup M \subseteq T \)
then have \( x \in \bigcup M \) by auto
then have \( V : x \in V \, V \in M \) by auto
with \( \text{Acov}(2) \) have \( \text{V = int}(V) \) using Top_2_L3 by auto
with \( V \in \text{int}(V) \) by auto
with cony VT obtain \( r \) where \( r \in \text{nat} \) \( \forall s \in \text{nat.} \, \langle r, s \rangle \in \text{le} \rightarrow \text{NNs} \in V \) using NetConverges_def[of net2] x op_compact_is_top unfolding topology0_def using xy(2) dom2 by auto
let \( A = \{x\} \cap \text{NNnat} \)
unfolding NetConverges_def[of net2, of x] using dom2 y by auto
have NresFun: restrict(NN, {n ∈ nat. ⟨n, r⟩ ∈ Le}) : {n ∈ nat. ⟨n, r⟩ ∈ Le} → \bigcup 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})))
using 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
{ fix j assume as:j∈nat Finite({n∈nat. ⟨n, j⟩ ∈ Le})
  { fix t assume t∈{n∈nat. ⟨n, succ(j)⟩ ∈ Le}
    then have t∈nat {t, succ(j)} ∈ Le by auto
    then have t≤succ(j) by auto
    then have t≤succ(j) using le_imp_subset by auto
    then have t≤j ∪ {j} using succ_explained by auto
    then have j∈t∨t≤j by auto
    then have j∈t∨t≤j using subset_imp_le < t∈nat > < j∈nat > nat_into_Ord by auto
  then have j ∪ {j} ⊆ t∨t≤j using < t∈nat > < j∈nat > nat_into_Ord
  unfolding Ord_def
  Transset_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}) using subset_Finite
by auto
}
then have ∀ j∈nat. Finite({n∈nat. ⟨n, j⟩ ∈ Le}) → Finite({n∈nat. ⟨n, succ(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})), where P=λt.
Finite({n\in\text{nat}.\ (n,t)\in\text{Le}})\text{ by auto}
then have Finite((\text{restrict} (\text{NN}, \{n \in \text{nat} .\ (n, r) \in \text{Le} \})\{n \in \text{nat}\} .\ (n, r) \in \text{Le}\})\text{ using range_image_domain[of NresFun]}
by auto
then have Finite(\text{NN}(n \in \text{nat} .\ (n, r) \in \text{Le}))\text{ using restrict_image by auto}
then have (\text{NN}(n \in \text{nat} .\ (n, r) \in \text{Le}))\{n \in \text{nat} .\ (n, r) \in \text{Le}\}\text{ using range_image_domain[of NresFun] by auto}
then have \text{NN}(n \in \text{nat} .\ (n, r) \in \text{Le})\text{ using restrict_image by auto}
then have \text{(NN\{n \in \text{nat}.\ (n, r) \in \text{Le}\})\{is in the spectrum of\}(\lambda T.}(T\{\text{restricted to}\}\text{NN}(n \in \text{nat} .\ (n, r) \in \text{Le}))=U T\cap \text{NN}(n \in \text{nat} .\ (n, r) \in \text{Le})\text{ using compact_spectrum by auto}
moreover have \text{\bigcup (T\{\text{restricted to}\}\text{NN}(n \in \text{nat}.\ (n, r) \in \text{Le}))=\bigcup T}\cap \text{NN}(n \in \text{nat} .\ (n, r) \in \text{Le})\text{ using func1_1_L6[of R] by blast}
moreover have \text{(T\{\text{restricted to}\}\text{NN}(n \in \text{nat}.\ (n, r) \in \text{Le}))}\text{ is a topology} using Top_1_L4 unfolding topology0_def by auto
ultimately have \text{(NN\{n \in \text{nat}.\ (n, r) \in \text{Le}\})\{is compact in\}(T\{\text{restricted to}\}\text{NN}(n \in \text{nat}.\ (n, r) \in \text{Le}))\text{ using compact_subspace_imp_compact by auto}
given A: A\subseteq U M by auto
moreover note \text{Acov(1) have (NN\{n \in \text{nat}.\ (n, r) \in \text{Le}\})\subseteq M}\text{ using FinPow_def by auto}
moreover have \text{(NN\{n \in \text{nat}.\ (n, r) \in \text{Le}\})\subseteq M}\text{ using FinPow_def by auto}
moreover have \text{\forall s: s\in A s\notin V}\text{ using def[of V]\ by auto}
fix s assume s: s\in A s\notin V with V(1) have s\notin NNnat by auto
then have s\in (\text{NNn. n\in\text{nat}})\text{ using func_imagedef[of NFun] by auto}
then obtain n where n: n\in\text{nat} s=\text{NNn} by auto
{ assume \langle r,n\rangle\in\text{Le}
with rr have \text{NNn}\in V by auto
with n(2) s(2) have False by auto
}
then have \langle r,n\rangle\notin\text{Le} by auto
with rr(1) n(1) have \neg(r\leq n) by auto
then have n\leq r using Ord_linear_le[where thesis=(n,r)\in\text{Le}] nat_into_Ord[of rr(1)]

moreover have \text{\forall n\in\text{nat.} \ (n,r)\in\text{Le})\subseteq \text{nat}} by auto
ultimately have s\in (\text{NN\{n\in\text{nat.} \ (n,r)\in\text{Le}\}})\text{ using func_imagedef[of NFun]}

1094
over blast \[\rightarrow\] OPCompactification_def by auto
unfolding auto
\{ using ∧ ⟨ s, m \} ultimately have \( A \{is compact in\} (T) \) unfolding IsCompact_def by auto moreover
with assms have \( A \{is closed in\} (T) \) unfolding IsKC_def IsCompact_def by auto ultimately
have \( A \in \text{Pow}(\bigcup T). \) \( B \{is compact in\} (T) \land B \{is closed in\} (T) \) using ss by auto
then have \( \bigcup \{ (\bigcup T) \} \bigcup (\bigcup (T-A)) \in \{\text{one-point compactification of} T\} \) unfolding OPCompactification_def by auto
then have \( \bigcup \{ (\bigcup T) \} \bigcup (\bigcup (T-A)) = \text{Interior}(\{ (\bigcup T) \} \bigcup (\bigcup (T-A)) \} \) using topology0.Top_2_L3 op_comp_is_top unfolding topology0_def by auto moreover
\{ assume \( y \in A \)
with \( A(4) \) have \( y \in \text{NNnat} \) by auto
then have \( y \in \{ \text{NNnat. } n \in \text{nat}\} \) using func_imagedef[OF NFun] by auto
then obtain n where \( n \in \text{natNNn} y \) by auto
with noy y have False by auto
\}
with y have \( y \in \{ (\bigcup T) \} \bigcup (\bigcup (T-A)) \) by force ultimately
have \( y \in \text{Interior}(\{ (\bigcup T) \} \bigcup (\bigcup (T-A)) \} \) using \( \bigcup \{ (\bigcup T) \} \bigcup (\bigcup (T-A)) \) by auto moreover
have \( \forall \in \text{Pow}(\{ (\bigcup T) \} \bigcup (\bigcup (T-A)) \} \) unfolding topology0.Top_2_L3 op_comp_is_top
by blast
then obtain r where \( r \_def : r \in \text{nat. } \forall x \in \text{nat. } \langle r, x \rangle \in L \rightarrow N x \in \{ (\bigcup T) \} \bigcup (\bigcup (T-A)) \) by auto
\{ fix s assume \( AA : \{ s, r \} \in L \)
with reg obtain m where \( N m \neq \bigcup T \) \( \langle s, m \rangle \in L \) by auto
then have \( \langle s, m \rangle \in L \) \( \forall i. N i \neq \bigcup T \) \( \langle s, i \rangle \in L \) using LeastI[of \( \lambda m. N m \neq \bigcup T \land (s, m) \in L \) nat_into_Ord by auto
with \( AA \) have \( \langle r, \mu i. N i \neq \bigcup T \land (s, i) \rangle \in L \) by le_trans by auto
\}

1095
with $r_{\text{def}}(2)$ have $N(\mu \ i. \ Ni \neq \bigcup T \land (s,i) \in \text{Le}) \in \bigcup T \cup \bigcup (T-A)$ by auto
then have $NNs \in \bigcup T \cup \bigcup (T-A)$ using apply_equality[OF _ NFun] AA by auto
then have $NNs \in (\bigcup T) \cup (\bigcup (T-A))$ by auto
moreover have $NNs \subseteq (\bigcup T) \cup (\bigcup (T-A))$ by auto
moreover have $NNs \in (\bigcup T) \cup (\bigcup (T-A))$ using apply_equality[OF _ NFun] AA
ultimately have 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 <r>nat succI by auto
ultimately have False by auto
ultimately have False by auto
then have $\forall N \ x \ y. \ N:nat \rightarrow \bigcup \{(\text{one-point compactification of} T) \land \langle N,\text{Le}\rangle \rightarrow N \ x\{\text{in}\}(\{(\text{one-point compactification of} T)) \land \langle (N,\text{Le}) \rightarrow N \ y\{\text{in}\}(\{(\text{one-point compactification of} T)) \rightarrow x=y \text{ by auto}
then show thesis unfolding IsUS_def by auto
qed

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 $T \{\text{is anti-compact}\} \ A \subseteq \bigcup \{(\text{one-point compactification of} T) \cup T \in A$
shows $A \{\text{is compact in} \ (\text{one-point compactification of} T)$
proof
{
fix $M$ assume $M:M \subseteq \{(\text{one-point compactification of} T) \ A \subseteq M$
with assms(3) obtain $U$ where $U: \bigcup T \in U \ U \in M$ by auto
with $M(1)$ obtain $K$ where $K:K \{\text{is compact in} \ T \ K \{\text{is closed in} \ T \ U=\bigcup T \cup (T-K)$
unfolding OPCompactification_def using mem_not_refl[of $\bigcup T$] by auto
from $K(1)$ have $K \{\text{is compact in} \ T \{\text{restricted to} \ K$ using compact_imp_compact_subspace
Compact_is_card_nat
by auto
moreover have $\bigcup (T \{\text{restricted to} \ K) \subseteq \bigcup T \cup K$ unfolding RestrictedTo_def by auto
with $K(1)$ have $\bigcup (T \{\text{restricted to} \ K) = K$ unfolding IsCompact_def by auto ultimately
have $\bigcup (T \{\text{restricted to} \ K) \{\text{is compact in} \ T \{\text{restricted to} \ K$ by auto
with assms(1) have $K \{\text{is in the spectrum of} \ \{\lambda T. \ (\bigcup T) \{\text{is compact in} \ T$ unfolding IsAntiComp_def
antiProperty_def using $K(1)$ unfolding IsCompact_def by auto
then have fin$K$:Finite($K$) using compact_spectrum by auto
from assms(2) have $A-U \subseteq (\bigcup T \cup (\bigcup T)) - U$ using op_compact_total by

1096
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}(λ_T. (∪_T){is compact in}T) using compact_spectrum by auto moreover
have ∪(((one-point compactification of)T){restricted to}(A-U))=A-U unfolding RestrictedTo_def using assms(2) K(3) op_comp_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}(λ_T. (∪_T){is compact in}T) unfolding Spec_def by auto
then have (A-U){is compact in}({one-point compactification of}T){restricted to}(A-U)
  unfolding Spec_def by auto
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
qed

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:
shows ¬(((one-point compactification of)Pow(csucc(nat)) ∪ {{csucc(nat)}} ∪ S. S∈(CoCountable csucc(nat))-{0}}){is KC})
proof
have noE:csucc(nat) ≠ 0 using Ord_0.lt_csucc[OF Ord_nat] by auto
let T=(Pow(csucc(nat)) ∪ {{csucc(nat)}} ∪ S. S∈(CoCountable csucc(nat))-{0}})
assume ass:(one-point compactification of)T{is KC}
from extension_pow_notDiscrete have {csucc(nat)} ∉ (Pow(csucc(nat)) ∪ {{csucc(nat)}}) S ∈ (CoCountable csucc(nat)) - {0}})
  by auto
{ assume csucc(nat)=csucc(nat)∪{csucc(nat)} moreover
  have csucc(nat)∈csucc(nat)∪{csucc(nat)} by auto
  ultimately have csucc(nat)∈csucc(nat) by auto
  then have False using mem_not_refl by auto
} then have dist:csucc(nat) ≠ csucc(nat)∪{csucc(nat)} by blast
{ assume {csucc(nat)}∈{(one-point compactification of)Pow(csucc(nat))}
\[ \bigcup \{\text{csucc(nat)}\} \cup S \in (\text{CoCountable \ csucc(nat)) - \{0\})\]
then have \{\text{csucc(nat)}\} \in \{\bigcup T \cup (\bigcup T - K) \cdot K \in \text{Pow}(\bigcup T) \cdot B(\text{is compact in}) T \land B(\text{is closed in}) T\}\]

unfolding OPCompactification_def using extension_pow_notDiscrete
by auto
then obtain K where \{\text{csucc(nat)}\} = \{\bigcup T \cup (\bigcup T - K) \} by auto moreover
have \bigcup T \in \{\bigcup T \cup (\bigcup T - K) \} by auto
ultimately have \bigcup T \in \{\text{csucc(nat)}\} by auto
with dist have False using extension_pow_union by auto

then have \{\text{csucc(nat)}\} \notin \{\text{one-point compactification of}} T\} by auto moreover
have \bigcup \{\text{one-point compactification of}} T\} - \{\bigcup \{\text{one-point compactification of}} T\} - \{\text{csucc(nat)}\} = \{\text{csucc(nat)}\} using extension_pow_union

\text{topology0.op_compact_total unfolding topology0_def using extension_pow_top}
by auto ultimately

have di: \{\bigcup \{\text{one-point compactification of}} T\} - \{\bigcup \{\text{one-point compactification of}} T\} - \{\text{csucc(nat)}\} \notin \{\text{one-point compactification of}} T\} by auto

assume \{\bigcup \{\text{one-point compactification of}} T\} - \{\text{csucc(nat)}\}\} \text{is closed in}\{\{\text{one-point compactification of}} T\}
then have \bigcup \{\text{one-point compactification of}} T\} - \{\bigcup \{\text{one-point compactification of}} T\} - \{\text{csucc(nat)}\}\} \in \{\text{one-point compactification of}} T\}\} \text{IsClosed_def by auto}

with di have False by auto

then have n: \neg(\{\bigcup \{\text{one-point compactification of}} T\} - \{\text{csucc(nat)}\}\} \text{is closed in}\{\{\text{one-point compactification of}} T\}\} by auto moreover
from dist have \bigcup T \in \{\bigcup \{\text{one-point compactification of}} T\} - \{\text{csucc(nat)}\}\} 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\} - \{\text{csucc(nat)}\}\} \text{is compact in}\{\{\text{one-point compactification of}} T\} using topology0.anti_comp_imp_OP_inf_comp[of T
\{\bigcup \{\text{one-point compactification of}} T\} - \{\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\} - \{\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, \( U \neq K \).

### 73.8 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.
73.9 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**

\[
\text{IsPerf } \equiv \forall x \in \bigcup T. \{x\} \notin T
\]

An anti-perfect space is called scattered.

**definition**

\[
\text{IsScatt } \equiv \text{Is anti-IsPerf}
\]

A topological space with two disjoint dense subspaces is called resolvable.

**definition**

\[
\text{IsRes } \equiv \exists U \in \text{Pow}(\bigcup T). \exists V \in \text{Pow}(\bigcup T). \text{Closure}(U,T)=\bigcup T \land \text{Closure}(V,T)=\bigcup T \land U \cap V = 0
\]

A topological space where every dense subset is open is called submaximal.

**definition**

\[
\text{IsSubMax } \equiv \forall U \in \text{Pow}(\bigcup T). \text{Closure}(U,T)=\bigcup T \rightarrow U \in T
\]

A subset of a topological space is nowhere-dense if the interior of its closure is empty.

**definition**

\[
\text{IsNowhereDense } \equiv \forall A \in \text{Pow}(\bigcup T). \text{Interior(Closure(A,T),T)} = 0
\]

A topological space is then a Luzin space if every nowhere-dense subset is countable.

**definition**

\[
\text{IsLuzin } \equiv \forall A \in \text{Pow}(\bigcup T). (A \text{ is nowhere dense in } T) \rightarrow A \leq \text{nat}
\]

An also useful property is local-connexion.

**definition**

\[
\text{IsLocConn } \equiv \forall A \in \text{Pow}(\bigcup T). (A \text{ is locally-connected}) \rightarrow (A \text{ is connected})
\]

An SI-space is an anti-resolvable perfect space.

**definition**

\[
\text{IsAntiRes } \equiv \text{Is anti-IsRes}
\]

1099
definition
IsSI \(_{\text{(is Strongly Irresolvable)}}\) where
\(T\text{is Strongly Irresolvable} \equiv (T\text{is anti-resolvable}) \land (T\text{is perfect})\)

73.10 First examples

Firstly, we need to compute the spectrum of the being perfect.

lemma spectrum_perfect:
  shows \((A\{\text{is in the spectrum of}Is\text{Perf}\}) \iff A=0\)
proof
  assume \(A\{\text{is in the spectrum of}Is\text{Perf}\)
  then have \(\exists T\{\text{is a topology}\} \cup T\approx A\)
    with \(T\{\text{is perfect}\}\) unfolding Spec_def using Pow_is_top by auto
  then have \(\forall b\in A. \ {b} \notin \text{Pow}(A)\) unfolding IsPerf_def by auto
  then show \(A=0\) by auto
next
  assume \(A:A=0\)
  \{
    fix \(T\) assume \(T\{\text{is a topology}\} \cup T\approx A\)
    with \(T\{\text{is a topology}\}\) \(\cup T\approx 0\) by auto
    then have \(\cup T=0\) using eqpoll_0_is_0 by auto
    then have \(T\{\text{is perfect}\}\) unfolding IsPerf_def by auto
  \}
  then show \(A\{\text{is in the spectrum of}Is\text{Perf}\}\) unfolding Spec_def by auto
qed

The discrete space is clearly scattered:

lemma pow_is_scattered:
  shows \(\exists A\{\text{is scattered}\}\)
proof-
  \{
    fix \(B\) assume \(B:B \subseteq \bigcup \text{Pow}(A)\) \(\{\text{Pow(A) restricted to}B\}\{\text{is perfect}\}\)
    from \(B\{\text{is in the spectrum of}Is\text{Perf}\}\) unfolding RestrictedTo_def
    by blast
    with \(B\{\text{is a topology}\}\) \(\bigcup T\approx 0\) by auto
    then have \(\forall b\in B. \ {b} \notin \text{Pow}(B)\) unfolding IsPerf_def by auto
    then have \(B=0\) by auto
  \}
  then show \(\exists A\{\text{is perfect}\}\) 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 \(\exists x\ y. \ x\in X \land y\in X \land x\neq y\)
  shows \(\{0,X\}\{\text{is perfect}\}\)
proof-
The trivial topology is resolvable, if it is defined over a set with more than one point.

lemma trivial_is_resolvable:
  assumes ∃ x y. x∈X ∧ y∈X ∧ x≠y
sows {0,X} is resolvable}
proof-
  from assms obtain x y where xy:x∈X y∈X x≠y by auto
  {
    fix A assume A:A is closed in}{0,X} A⊆X
    then have X−A∈{0,X} unfolding IsClosed_def by auto
    then have X−A=0∨X−A=X by auto
    with A(2) have A=X∨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
    }
    ultimately have A=0∨A=X by auto
    then have cl:∀ A∈Pow(X). A is closed in}{0,X} → A=0∨A=X by auto
    from xy(3) have {x}∩{y}=∅ by auto moreover
    {
      have {X} is a partition of X using indiscrete_partition xy(1) by auto
      then have top:topology0 PTopology X {X} using topology0_ptopology
      by auto
      have X≠0 using xy(1) by auto
      then have (PTopology X {X})={0,X} using indiscrete_ptopology[of X]
      by auto
      with top have top0:topology0({0,X}) by auto
      then have x∈Closure({x},{0,X}) using topology0.cl_contains_set xy(1)
      by auto moreover
      have Closure({x},{0,X}) is closed in}{0,X} using topology0.cl_is_closed
      top0 xy(1) by auto
      moreover note cl
      moreover have Closure({x},{0,X})⊆X using topology0.Top_3_L11(1)
      top0 xy(1) by auto
      ultimately have Closure({x},{0,X})=X by auto
    }
  }
have \{X\} is a partition of \{X\} using indiscrete_partition xy(1) by auto
then have top: topology0 \{PTopology X \{X\}\} using topology0_ptopology by auto
  have \{X\} \neq \emptyset using xy(1) by auto
then have \{PTopology X \{X\}\} = \{\emptyset, \{X\}\} using indiscrete_ptopology[of \{X\}] by auto
  with top have top0: topology0 \{\{0, X\}\} by auto
then have \{\{\{y\}, \{0, X\}\}\} using topology0.cl_contains_set xy(2) by auto
moreover have \{\{\{y\}, \{0, X\}\}\} \subseteq \{\emptyset, \{X\}\} using topology0.cl_is_closed
  ultimately have \{\{\{y\}, \{0, X\}\}\} = \{\emptyset, \{X\}\} by auto
ultimately show 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 \(A\{\text{is in the spectrum of}IsLuzin\} \iff A \leq \text{nat}\)
proof
  assume A:A \{\text{is in the spectrum of}IsLuzin\}
  \{ assume A=0 then have A \leq \text{nat} using empty_lepollI by auto \}
moreover
  \{ assume A\neq0 then obtain x where x:x \in A by auto
    \{ fix M assume M \subseteq \{\emptyset, \{x\}, A\}
      then have \bigcup M \subseteq \{\emptyset, \{x\}, A\} using x by blast \}
  moreover
    \{ fix U V assume U \subseteq \{\emptyset, \{x\}, A\} V \subseteq \{\emptyset, \{x\}, A\}
      then have U \cap V \subseteq \{\emptyset, \{x\}, A\} by auto \}
  ultimately have top: \{\{0, \{x\}, A\}\} is a topology unfolding IsATopology_def by auto
moreover have tot: \bigcup \{\{0, \{x\}, A\}\} = A using x by auto
moreover note A ultimately have luz: \{\{0, \{x\}, A\}\} is luzin unfolding Spec_def by auto

1102
moreover have \( \{x\} \subseteq \{0,\{x\}\} \) by auto
then have \( (\bigcup \{0,\{x\}\}) \setminus \{x\} \) is closed in \( \{0,\{x\}\} \) using topology0.Top_3_L9 unfolding topology0_def using top by blast
then have \( (A \setminus \{x\}) \) is closed in \( \{0,\{x\}\} \) using tot by auto
then have \( \text{Closure}(A \setminus \{x\}, \{0,\{x\}\}) = A \setminus \{x\} \) using top top topology0.Top_3_L8[of \( \{0,\{x\}\} \)] unfolding topology0_def by auto
then have \( (\bigcup \{0,\{x\}\}) \setminus \{x\} \subseteq A \setminus \{x\} \) using topology0.Top_2_L1 unfolding topology0_def by auto
then have \( \text{Interior}(\text{Closure}(A \setminus \{x\}, \{0,\{x\}\}), \{0,\{x\}\}) = \text{Interior}(A \setminus \{x\}, \{0,\{x\}\}) \) by auto
then have \( \text{Interior}(\text{Closure}(A \setminus \{x\}, \{0,\{x\}\}), \{0,\{x\}\}) \subseteq A \setminus \{x\} \) using topology0.Top_2_L2 unfolding topology0_def using top by auto
from \( x \) have \( \neg (A \subseteq A \setminus \{x\}) \) by auto
then have \( \text{Interior}(\text{Closure}(A \setminus \{x\}, \{0,\{x\}\}), \{0,\{x\}\}) = 0 \) by auto
then have \( \text{Closure}(A \setminus \{x\}, \{0,\{x\}\}) \) is nowhere dense in \( \{0,\{x\}\} \) unfolding IsNowhereDense_def by auto
with \( \neg (A \subseteq A \setminus \{x\}) \) have \( \text{Interior}(\text{Closure}(A \setminus \{x\}, \{0,\{x\}\}), \{0,\{x\}\}) = 0 \) by auto
then have \( \text{Closure}(A \setminus \{x\}, \{0,\{x\}\}) \subseteq A \setminus \{x\} \) using topology0.Top_2_L2 unfolding topology0_def using top by auto
then have \( (A \setminus \{x\}) \subseteq \{0,\{x\}\} \) using topology0.Top_2_L2 unfolding topology0_def using top by auto
from \( x \) have \( \neg (A \subseteq A \setminus \{x\}) \) by auto
then have \( \text{Interior}(\text{Closure}(A \setminus \{x\}, \{0,\{x\}\}), \{0,\{x\}\}) = 0 \) by auto
then have \( \text{Closure}(A \setminus \{x\}, \{0,\{x\}\}) \) is nowhere dense in \( \{0,\{x\}\} \) unfolding IsNowhereDense_def by auto
with \( luz \) have \( A \setminus \{x\} \leq \text{nat} \) unfolding IsLuzin_def using tot by auto
then have \( U1: A \setminus \{x\} \subseteq \text{csucc}(\text{nat}) \) using Card_less_csucc_eq_le[of Card_nat] by auto
then have \( (A \setminus \{x\}) \cup \{x\} \subseteq \text{csucc}(\text{nat}) \) using less_less_imp_un_less[of _ _ _ InfCard_csucc[of InfCard_nat]] by auto
ultimately show \( A \leq \text{nat} \) by auto
next
assume \( A: A \leq \text{nat} \)
fix \( T \) assume \( T: \text{T is a topology} \) \( \bigcup T = A \)
fix \( B \) assume \( B \subseteq \bigcup T \) \( B \) is nowhere dense in \( T \)
then have \( B \subseteq \bigcup T \) using subset_imp_lepoll by auto
with \( T(2) \) have \( B \leq A \) using lepoll_imp_lepoll by auto
with \( A \) have \( B \leq \text{nat} \) using lepoll_trans by blast
then have \( \forall B \in \text{Pow}(\bigcup T). \ (B \text{ is nowhere dense in } T) \rightarrow B \leq \text{nat} \) by auto
then have \( T: \text{T is luzin} \) unfolding IsLuzin_def by auto

1103
then show $A$ is in the spectrum of $\text{IsLuzin}$ unfolding $\text{Spec}_\text{def}$ by auto

**73.11 Structural results**

Every resolvable space is also perfect.

**Theorem (in topology0) resolvable_imp_perfect:**

assumes $T$ is resolvable

shows $T$ is perfect

**Proof**

- Assume $\neg (T$ is perfect)
  
  then obtain $x$ where $x : x \in \bigcup T \{x\} \in T$ unfolding $\text{IsPerf}_\text{def}$ by auto
  
  then have $\text{cl} : (\bigcup T - \{x\}) \{x\}$ is closed in $T$ using $\text{Top}_3_{L9}$ by auto
  
  from asms obtain $U V$ where $U \subseteq \bigcup T V \subseteq \bigcup T$ $\text{cl}(U) = \bigcup T \text{ cl}(V) = \bigcup T U \cap V = 0$
  
  unfolding $\text{IsRes}_\text{def}$ by auto
  
  - fix $W$ assume $x \notin W W \subseteq \bigcup T$
    
    then have $W \subseteq \bigcup T - \{x\}$ by auto
    
    then have $\text{cl}(W) \subseteq \bigcup T - \{x\}$ using $\text{cl} \text{ Top}_3_{L13}$ by auto
    
    with $x(1)$ have $\neg (\bigcup T \subseteq \text{cl}(W))$ by auto
    
    then have $\neg (\text{cl}(W) = \bigcup T)$ by auto
  
  with $UV$ have False by auto

then show thesis by auto

**Qed**

The spectrum of being resolvable follows:

**Corollary spectrum_resolvable:**

shows $(A$ is in the spectrum of $\text{IsRes}) \iff A = 0$

**Proof**

- Assume $A : A$ is in the spectrum of $\text{IsRes}$
  
  have $\forall T. T$ is a topology} $\to \text{IsRes}(T) \to \text{IsPerf}(T)$ using $\text{topology0}\_\text{resolvable}_\text{imp}_\text{perfect}$
  
  unfolding $\text{topology0}_\text{def}$ by auto
  
  with $A$ have $A$ is in the spectrum of $\text{IsPerf}$ using $\text{P}_\text{imp}_Q$ spec inv[of $\text{IsRes IsPerf}$] by auto
  
  then show $A = 0$ using $\text{spectrum}_\text{perfect}$ by auto

next

- Assume $A = 0$
  
  fix $T$ assume $T : T$ is a topology} $\bigcup T \approx A$
  
  with $T(2)$ $A$ have $\bigcup T \approx 0$ by auto
  
  then have $\bigcup T = 0$ using $\text{eqpoll\_0\_is\_0}$ by auto
  
  then have $\text{Closure}(0, T) = \bigcup T$ using $\text{topology0}\_\text{Top}_3_{L2} T(1)$
  
  $\text{topology0}\_\text{Top}_3_{L8}$ unfolding $\text{topology0}_\text{def}$ by auto
  
  then have $T$ is resolvable} unfolding $\text{IsRes}_\text{def}$ by auto

1104
then show A is in the spectrum of IsRes unfolding Spec_def by auto
qed

The cofinite space over N is a $T_1$, perfect and luzin space.

**Theorem:** cofinite_nat_perfect:

**Shows:** (CoFinite nat) is perfect

**Proof:**

- Fix x assume $x \in \bigcup (\text{CoFinite nat})$ \{x\} $\in (\text{CoFinite nat})$
  then have $x : x \in \text{nat}$ using union_cocardinal unfolding Cofinite_def
  by auto
    with $x(2)$ have $\text{nat} - \{x\} \prec \text{nat}$ unfolding Cofinite_def CoCardinal_def
  by auto
    moreover have $\text{Finite}(\{x\})$ by auto
      then have $\{x\} \prec \text{nat}$ unfolding Finite_def using n_lesspoll_nat eq_lesspoll_trans
      by auto
      ultimately have $(\text{nat} - \{x\}) \cup \{x\} \prec \text{nat}$ using xn by auto
      ultimately have False by auto
  
then show thesis unfolding IsPerf_def by auto
qed

**Theorem:** cofinite_nat_luzin:

**Shows:** (CoFinite nat) is luzin

**Proof:**

- have $\text{nat} \{\text{is in the spectrum of}\} \text{IsLuzin}$ using spectrum_Luzin by auto
  moreover have $\bigcup (\text{CoFinite nat}) = \text{nat}$ using union_cocardinal unfolding Cofinite_def
  by auto
  moreover have (CoFinite nat) is a topology unfolding Cofinite_def
  using CoCar_is_topology[OF InfCard_nat] by auto
  ultimately show thesis unfolding Spec_def by auto
qed

The cocountable topology on $N^+$ or $\text{csucc(nat)}$ 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)$\neq 0$ using lt_csucc[OF Ord_nat] by auto
  
then show thesis unfolding IsPerf_def by auto
qed

1105
with \( x(2) \) have \( \text{csucc}(\text{nat}) \prec \text{csucc}(\text{nat}) \) unfolding Ccountable_def

CoCardinal_def by auto

moreover have \( \text{Finite}(\{x\}) \) by auto

then have \( \{x\} \prec \text{nat} \) unfolding Finite_def using n_lesspoll_nat eq_lesspoll_trans by auto

ultimately have \( (\text{csucc}(\text{nat}) \setminus \{x\}) \cup \{x\} \prec \text{csucc}(\text{nat}) \) using less_less_imp_un_less[of _ _ InfCard_csucc[OF InfCard_nat]] by auto

then have \( \{x\} \upprec \text{nat} \) unfolding Finite_def using n_lesspoll_nat eq_lesspoll_trans by auto

ultimately have \( \text{False} \) by auto

qed

theorem cocountable_csucc_nat_luzin:
  shows \( \text{(CoCountable} \text{csucc}(\text{nat}))\{\text{is luzin}\} \)

proof
  have noE:\( \text{csucc}(\text{nat}) \neq 0 \) using lt_csucc[OF Ord_nat] by auto

  \{ fix \( B \) assume \( B : B \in (\bigcup (\text{CoCountable csucc(nat)))) \) \( B \{\text{is nowhere dense in}(\text{CoCountable csucc(nat))} \) \( \neg (B \subseteq \text{nat}) \)

  from \( B(1) \) have \( B \subseteq \text{csucc(nat)} \) using union_cocardinal noE unfolding Cocountable_def by auto

  moreover from \( B(3) \) have \( \neg (B \prec \text{csucc(nat)}) \) using Card_less_csucc_eq_le[OF Card_nat] by auto

  ultimately have \( \text{Closure}(B, \text{CoCountable csucc(nat)}) = \text{csucc(nat)} \) using closure_set_cocardinal noE unfolding Cocountable_def by auto

  then have \( \text{Interior}(\text{Closure}(B, \text{CoCountable csucc(nat))), \text{CoCountable csucc(nat)}) = \text{csucc(nat)} \) unfolding IsNowhereDense_def by auto moreover

  have \( \text{csucc(nat)} \prec \text{csucc(nat)} = 0 \) by auto

  then have \( \text{csucc(nat)} \prec \text{csucc(nat)} = \text{csucc(nat)} \) using empty_lepollI Card_less_csucc_eq_le[OF Card_nat] by auto

  ultimately have \( \text{False} \) using noE by auto

  then have \( \forall B \in \bigcup (\text{CoCountable csucc(nat)}) \). (\( B \{\text{is nowhere dense in}(\text{CoCountable csucc(nat))} \) \( \neg (B \subseteq \text{nat}) \) by auto

  then show thesis unfolding IsLuzin_def by auto

qed

The existence of \( T_2 \), uncountable, perfect and luzin spaces is unprovable in \( ZFC \). It is related to the \( CH \) and Martin’s axiom.

end
74 Uniform spaces

theory UniformSpace_ZF imports Topology_ZF_4a
begin

This theory defines uniform spaces and proves their basic properties.

74.1 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
IsUniformity (\_ {is a uniformity on} \_ 90) where
\( \Phi \) {is a uniformity on} $X \equiv (\Phi \) {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)\)

A Member of a uniformity on $X$ is a reflexive relation on $X$.

lemma unif_props: assumes $\Phi$ {is a uniformity on} $X$ $A \in \Phi$
shows $A \subseteq X \times X$ and $\text{id}(X) \subseteq A$
using assms IsUniformity_def IsFilter_def by auto

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} $X$ $U \in \Phi$
shows $\exists W \subseteq \Phi. \ W \circ W \subseteq U \land W = \text{converse}(W)$
proof -
from assms obtain $V$ where $V \in \Phi$ and $V \circ V \subseteq U$
unfolding IsUniformity_def by auto
let $W = V \cap \text{converse}(V)$
from assms(1) have $W \in \Phi$ and $W = \text{converse}(W)$
unfolding IsUniformity_def IsFilter_def by auto
moreover from $V \circ V \subseteq U$ have $W \circ W \subseteq U$ by auto
ultimately show thesis by blast
qed
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 $X$
- $V \in \Phi$ and $x \in X$

**Shows**

- $V\{x\} \neq 0$ and $x \in V\{x\}$

**Proof** -

- from assms(1,2) have $id(X) \subseteq V$ using IsUniformity_def IsFilter_def

  by auto

with $<x \in X>$ show $x \in V\{x\}$ and $V\{x\} \neq 0$ by auto

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**

- $\Phi$ is a uniformity on $X$
- $W \in \Phi$

**Shows**

- $W \subseteq X \times X$ and $\text{domain}(W) = X$

**Proof** -

- from assms show $W \subseteq X \times X$ unfolding IsUniformity_def IsFilter_def

  by blast

  show $\text{domain}(W) = X$

  proof

  from assms show $\text{domain}(W) \subseteq X$ unfolding IsUniformity_def IsFilter_def

  by auto

  from assms show $X \subseteq \text{domain}(W)$ unfolding IsUniformity_def by blast

  qed

  qed

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**

- $M \equiv \{(x,\{V\{x\}.V \in \Phi\}).x \in X\}$

**Shows**

- $M : X \to \text{Pow(Pow(X))}$ and $\forall x \in X. M(x) = \{V\{x\}.V \in \Phi\}$

**Proof** -

- from assms have $\forall x \in X. \{V\{x\}.V \in \Phi\} \in \text{Pow(Pow(X))}$

  using IsUniformity_def IsFilter_def image_subset by auto

  with assms show $M : X \to \text{Pow(Pow(X))}$ using ZF_fun_from_total by simp

  with assms show $\forall x \in X. M(x) = \{V\{x\}.V \in \Phi\}$ using ZF_fun_from_tot_val

  by simp
In the next lemma we show that the collection defined in lemma \texttt{neigh\_filt\_fun} is a filter on $X$. The proof is kind of long, but it just checks that all filter conditions hold.

**lemma** \texttt{filter\_from\_uniformity}:  
assumes $\Phi$ \{is a uniformity on\} $X$ and $x \in X$  
defines $M \equiv \{x, \{V(x).V \in \Phi\}\}.x \in X$  
shows $M(x)$ \{is a filter on\} $X$

**proof** -  
from \texttt{assms} have PhiFilter: $\Phi$ \{is a filter on\} $(X \times X)$ and  
$M : X \rightarrow \text{Pow(Pow}(X))$ and $M(x) = \{V(x).V \in \Phi\}$  
using \texttt{IsUniformity\_def neigh\_filt\_fun} by auto  
have $0 \notin M(x)$  
**proof** -  
from \texttt{assms} have 0 : $x \in X$ have $0 \notin \{V(x).V \in \Phi\}$ using \texttt{neigh\_not\_empty} by blast  
with $M(x) = \{V(x).V \in \Phi\}$ show $0 \notin M(x)$ by simp  
qed  
moreover have $X \in M(x)$  
**proof** -  
note $M(x) = \{V(x).V \in \Phi\}$  
moreover from \texttt{assms} have $X \times X \in \Phi$ unfolding \texttt{IsUniformity\_def IsFilter\_def} by blast  

hence $(X \times X)\{x\} \in \{V(x).V \in \Phi\}$ by auto  
moreover from $<x \in X>$ have $(X \times X)\{x\} = X$ by auto  
ultimately show $X \in M(x)$ by simp  
qed  
moreover from $<M : X \rightarrow \text{Pow(Pow}(X))>$ $<x \in X>$ have $M(x) \subseteq \text{Pow}(X)$ using \texttt{apply\_funtype} by blast  
moreover have LargerIn: $\forall B \in M(x). \forall C \in \text{Pow}(X). B \subseteq C \longrightarrow C \in M(x)$  
**proof** -  
{  
fix B assume B : $C \in M(x)$  
fix C assume C : $\in \text{Pow}(X)$ and $B \subseteq C$  
from $<M(x) = \{V(x).V \in \Phi\}>$ $<B \in M(x)>$ obtain $U$ where  
$U \in \Phi$ and $B = U(x)$ by auto  
let $V = U \cup C \times C$  
from \texttt{assms} have $U \in \Phi$ \{is a filter on\} $(X \times X)$ and $U \subseteq V$  
using \texttt{IsUniformity\_def IsFilter\_def} by auto  
with $<U \in \Phi>$ PhiFilter have $V \in \Phi$ using \texttt{IsFilter\_def} by simp  
moreover from \texttt{assms} $<U \in \Phi>$ $<x \in X>$ $<B = U(x) >$ $<B \subseteq C>$ have $C = V(x)$  
using \texttt{neigh\_not\_empty image\_greater\_rel} by simp  
ultimately have $C \in \{V(x).V \in \Phi\}$ by auto  
with $<M(x) = \{V(x).V \in \Phi\}>$ $<x \in X>$ $<B = U(x) >$ $<B \subseteq C>$ have $C \in M(x)$ by simp  
}  
thus thesis by blast  
qed  
moreover have $\forall A \in M(x). \forall B \in M(x). A \cap B \in M(x)$

1109
proof -
{ fix A B assume A ∈ M(x) and B ∈ M(x)
  with <M(x) = {V{x}.V∈Φ}> obtain V_A V_B where
  A = V_A(x) B = V_B(x) and V_A ∈ Φ V_B ∈ Φ
  by auto
  let C = V_A(x) ∩ V_B{x}
  from assms <V_A ∈ Φ> <V_B ∈ Φ> have V_A∩V_B ∈ Φ using IsUniformity_def
  IsFilter_def
  by simp
  with <M(x) = {V{x}.V∈Φ}> have (V_A∩V_B){x} ∈ M(x) by auto
  moreover from PhiFilter <V_A ∈ Φ> <V_B ∈ Φ> have C ∈ Pow(X) unfolding IsFilter_def
  by auto
  moreover have (V_A∩V_B){x} ⊆ C using image_Int_subset_left by simp
  moreover note LargerIn
  ultimately have C ∈ M(x) by simp
  with <A = V_A(x)> <B = V_B{x}> have A∩B ∈ M(x) by blast
} thus thesis by simp
qed
ultimately show thesis unfolding IsFilter_def by simp
qed

The function defined in the premises of lemma neigh_filt_fun (or filter_from_uniformity)
is a neighborhood system. The proof uses the existence of the "half-the-size" neighborhood condition (∀V∈Φ. V O V ⊆ U) of the uniformity definition, but not the converse(U) ∈ Φ part.

theorem neigh_from_uniformity:
  assumes Φ {is a uniformity on} X
  shows {⟨x,{V{x}.V∈Φ}⟩.x∈X} {is a neighborhood system on} X
proof -
  let M = {⟨x,{V{x}.V∈Φ}⟩.x∈X}
  from assms have M:X→Pow(Pow(X)) and Mval: ∀x∈X. M(x) = {V{x}.V∈Φ}
  using IsUniformity_def neigh_filt_fun by auto
  moreover from assms have ∀x∈X. (M(x) {is a filter on} X) using filter_from_uniformity
  by simp
  moreover
  { fix x assume x∈X
    have ∀N∈M(x). x∈N ∧ (∃U∈M(x).∀y∈U.(N∈M(y)))
    proof -
      { fix N assume N∈M(x)
        have x∈N and ∃U∈M(x).∀y∈U.(N∈M(y))
        proof -
          from <M:X→Pow(Pow(X))> Mval <x∈X> <N∈M(x)>
          obtain U where U∈Φ and N = U{x} by auto
          with assms <x∈X> show x∈N using neigh_not_empty by simp
          from assms <U∈Φ> obtain V where V∈Φ and V O V ⊆ U
          unfolding IsUniformity_def by auto
          let W = V{x}
          from <V∈Φ> Mval <x∈X> have W ∈ M(x) by auto
          1110
        }
moreover have \( \forall y \in W \; N \in M(y) \)

proof -
{ fix \( y \) assume \( y \in W \)
with \( \langle M : X \to \text{Pow(\text{Pow}(X))} \rangle \langle x \in X \rangle \langle W \in M(x) \rangle \) have \( y \in X \)
using apply_funtype by blast
with assms have \( M(y) \) \{is a filter on\} \( X \) using filter_from_uniformity
by simp
moreover from assms \( x \in X \; V \in \Phi \) have \( V \{y\} \in M(y) \)
using neigh_filt_fun by auto
moreover from \( \langle M : X \to \text{Pow(\text{Pow}(X))} \rangle \langle x \in X \rangle \langle N \in M(x) \rangle \) have \( N \in \text{Pow}(X) \)
using apply_funtype by blast
moreover from \( \langle V \cup V \subseteq U \rangle \langle y \in W \rangle \) have \( V \{y\} \subseteq (V \cup V) \{x\} \) and \( (V \cup V) \{x\} \subseteq U \{x\} \)
by auto
with \( \langle N = U \{x\} \rangle \) have \( V \{y\} \subseteq N \) by blast
ultimately have \( N \in M(y) \) unfolding IsFilter_def by simp
} thus thesis by simp
qed
ultimately show \( \exists U \in M(x). \forall y \in U. (N \in M(y)) \) by auto
qed
}
{ thus thesis by simp
qed
}
ultimately show thesis unfolding IsNeighSystem_def by simp
qed

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 \( \text{UniformTopology}(\Phi) \).

The definition may be a bit cryptic but it just combines the construction of a neighborhood system from uniformity as in the assumptions of lemma \( \text{filter_from_uniformity} \) and the construction of topology from a neighborhood system from theorem \( \text{topology_from_neighs} \). 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
\( \text{UniformTopology}(\Phi,X) \equiv \{ U \in \text{Pow}(X). \forall x \in U. U \in \langle \langle t, \{V \in \Phi\} \rangle. t \in X \rangle(x) \} \)

The collection of sets constructed in the \( \text{UniformTopology} \) definition is indeed a topology on \( X \).

definition
\( \text{UniformTopology}(\Phi,X) \equiv \{ U \in \text{Pow}(X). \forall x \in U. U \in \langle \langle t, \{V \in \Phi\} \rangle. t \in X \rangle(x) \} \)

Theorem \( \text{uniform_top_is_top} \):
assumes \( \Phi \) \{is a uniformity on\} \( X \)
shows
\( \text{UniformTopology}(\Phi,X) \) \{is a topology\} and \( \bigcup \text{UniformTopology}(\Phi,X) = X \)
using assms neigh_from_uniformity UniformTopology_def topology_from_neighs by auto
75 More on uniform spaces

theory UniformSpaceZF_1 imports func_ZF_1 UniformSpaceZF TopologyZF_2
begin

This theory defines the maps to study in uniform spaces and proves their basic properties.

75.1 Uniformly continuous functions

Just as 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→Y ==> Φ {is a uniformity on}X ==> Γ {is a uniformity on}Y ==> 
  f {is uniformly continuous between} Φ {and} Γ Ξ ∀V∈Γ. (ProdFunction(f,f)-V)∈Φ

Any uniformly continuous function is continuous when considering the topologies on the uniformities.

lemma uniformly_cont_is_cont:
  assumes f:X→Y Φ {is a uniformity on}X Γ {is a uniformity on}Y f {is uniformly continuous between} Φ {and} Γ
  shows IsContinuous(UniformTopology(Φ,X),UniformTopology(Γ,Y),f)
proof -
  { fix U assume op: U ∈ UniformTopology(Γ,Y)
    have f-(U) ∈ UniformTopology(Φ,X)
      proof -
        from assms(1) have f-(U) ⊆ X using func1_1_L3 by simp
        moreover
        { fix x xa assume as:(x,xa) ∈ f xa ∈ U
          with assms(1) have x:x∈X unfolding Pi_def by auto
          from as(2) op have U:U ∈ {{t,⟨V(t),V∈Γ⟩}.t∈Γ}(xa) unfolding UniformTopology_def
          by auto
        from as(1) assms(1) have xa:xa ∈ Y unfolding Pi_def by auto
        have {{t,⟨V(t),V∈Γ⟩}.t∈Y} ∈ Pi(Y,%t. {{V(t),V∈Γ}}) unfolding Pi_def function_def
          by auto
          with U xa have U ∈{V(xa),V∈Γ} using apply_equality by auto
          then obtain V where V:U = V(xa) ∀V∈Γ by auto
          with assms have ent:(ProdFunction(f,f)-(V))∈Φ using IsUniformlyCont_def
        by simp
      by auto
    }
have ∀t. t ∈ (ProdFunction(f,f)-V){x} <-> (x,t) ∈ ProdFunction(f,f)-(V)

using image_def by auto
with assms(1) x have ∀t. t: (ProdFunction(f,f)-V){x} <-> (t∈X ∧ (fx,ft) ∈ V)
  using prodFunVimage by auto
with assms(1) as(1) have ∀t. t ∈ (ProdFunction(f,f)-V){x} <-> (t∈X ∧ f(t) ∈ U)
  by auto
with assms(1) U have ∀t. t ∈ (ProdFunction(f,f)-V){x} <-> t ∈ f-U
  using func1_1_L15 by simp
hence f-U = (ProdFunction(f,f)-V){x} by blast

moreover have {{t, {V{t}.V ∈ Φ} . t ∈ X} . t ∈ X} ∈ Pi(X,%t. {V{t}.V ∈ Φ})

ultimately have f-(U) ∈ UniformTopology(Φ, X) unfolding UniformTopology_def
by blast
then show thesis unfolding IsContinuous_def by simp
qed

76 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 of the notion of uniform covers.

76.1 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 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. The members of such family are
called a "uniform cover", but one has to remember that this notion has
meaning only in the contexts a the whole family of uniform covers. Looking
at a specific cover in isolation we can not say whether it is a uniform cover
or not.

The set of all covers of $X$ is called $\text{Covers}(X)$.

**definition**

\[ \text{Covers}(X) \equiv \{ P \in \text{Pow}(\text{Pow}(X)). \cup P = X \} \]

A cover of a nonempty set must have a nonempty member.

**lemma** $\text{cover_nonempty}$: assumes $X \neq 0$ $P \in \text{Covers}(X)$
shows $\exists U \in P. U \neq 0$
using assms unfolding $\text{Covers_def}$ by blast

A "star" of $R$ with respect to $\mathcal{R}$ is the union of all $S \in \mathcal{R}$ that intersect $R$.

**definition**

\[ \text{Star}(U,P) \equiv \bigcup \{ V \in P. V \cap U \neq 0 \} \]

An element of $\mathcal{R}$ is a subset of its star with respect to $\mathcal{R}$.

**lemma** $\text{element_subset_star}$: assumes $U \in P$ shows $U \subseteq \text{Star}(U,P)$
using assms unfolding $\text{Star_def}$ by auto

An alternative formula for star of a singleton.

**lemma** $\text{star_singleton}$: shows $(\bigcup \{ V \times V. V \in P \}) \{ x \} = \text{Star}(\{ x \}, P)$
unfolding $\text{Star_def}$ by blast

Star of a larger set is larger.

**lemma** $\text{star_mono}$: assumes $U \subseteq V$ shows $\text{Star}(U,P) \subseteq \text{Star}(V,P)$
using assms unfolding $\text{Star_def}$ by blast

In particular, star of a set is larger than star of any singleton in that set.

**corollary** $\text{star_single_mono}$: assumes $x \in U$ shows $\text{Star}(\{ x \}, P) \subseteq \text{Star}(U,P)$
using assms $\text{star_mono}$ by auto

A cover $\mathcal{R}$ (of $X$) is said to be a "barycentric refinement" of a cover $\mathcal{C}$ iff for
every $x \in X$ the star of $\{ x \}$ in $\mathcal{R}$ is contained in some $C \in \mathcal{C}$.

**definition**

\[ \text{IsBarycentricRefinement} (\_ <^B \_ 90) \]
where $P <^B Q \equiv \forall x \in \bigcup P. \exists U \in Q. \text{Star}(\{ x \}, P) \subseteq U$

A cover is a barycentric refinement of the collection of stars of the singletons
$\{ x \}$ as $x$ ranges over $X$.

**lemma** $\text{singl_star_bary}$: assumes $P \in \text{Covers}(X)$ shows $P <^B \{ \text{Star}(\{ x \}, P). x \in X \}$
using assms unfolding $\text{Covers_def}$ $\text{IsBarycentricRefinement_def}$ 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**

```plaintext
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 $P \in$ 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 $Q \in$ Covers($X$) and $Q <∗ P$ shows $Q <B P$

proof -

```plaintext
from assms(1) have $\bigcup Q = X$ unfolding Covers_def by simp
{ fix $x$ assume $x \in X$
  with $\langle \bigcup Q = X \rangle$ obtain $R$ where $R \in Q$ and $x \in R$ by auto
  with assms(2) obtain $U$ where $U \in P$ and $\operatorname{Star}(R,Q) \subseteq U$
    unfolding IsStarRefinement_def by auto
  from $\langle x \in R \rangle$ $\langle \operatorname{Star}(R,Q) \subseteq U \rangle$ have $\operatorname{Star}(\{x\},Q) \subseteq U$
    using star_single_mono by blast
  with $\langle U \in P \rangle$ have $\exists V \in P . \operatorname{Star}(\{x\},Q) \subseteq V$ by auto
} with $\langle \bigcup Q = X \rangle$ show thesis unfolding IsBarycentricRefinement_def by simp
qed
```

Barycentric refinement of a barycentric refinement is a star refinement.

**lemma** bary_bary_star:

assumes $P \in$ Covers($X$) $Q \in$ Covers($X$) $R \in$ Covers($X$) $P <B Q$ $Q <B R$ $X \neq 0$

shows $P <∗ R$

proof -

```plaintext
{ fix $U$ assume $U \in P$
  { assume $U = 0$
    then have $\operatorname{Star}(U,P) = 0$ unfolding Star_def by simp
    from assms(6,3) obtain $V$ where $V \in R$ using cover_nonempty by auto
    with $\langle \operatorname{Star}(U,P) = 0 \rangle$ have $\exists V \in R . \operatorname{Star}(U,P) \subseteq V$ 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 $\operatorname{Star}(\{x_0\},Q) \subseteq V$
      unfolding Covers_def IsBarycentricRefinement_def by auto
    have $\operatorname{Star}(U,P) \subseteq V$
      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
```
The notion of a filter defined in Topology_ZF_4 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 $D$.

This departs slightly from the definition in Wikipedia that requires that $\Theta$ contains the trivial cover $\{X\}$. As we show in lemma unicov_contains_trivial below we don’t loose anything by weakening the definition this way.

definition
AreUniformCovers (_ {are uniform covers of} _ 90)
where
\[ \Theta \ {\text{are uniform covers of}} \ X \equiv \ \Theta \subseteq \text{Covers}(X) \land \Theta \neq \emptyset \land \]
\[ (\forall R \in \Theta. \forall C \in \text{Covers}(X). (\ (R <^* C) \rightarrow C \in \Theta)) \land \]
\[ (\forall C \in \Theta. \forall D \in \Theta. \exists R \in \Theta. (\ R <^* C) \land (\ R <^* D)) \]
A family of uniform covers contain the trivial cover $\{X\}$.

lemma unicov_contains_triv: assumes $\Theta \ {\text{are uniform covers of}} \ X$
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 $\Theta \ {\text{are uniform covers of}} \ X$
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

1116
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

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 \( X \neq 0 \)
shows \( Q \in \Theta \)
proof -
from assms(1,2) obtain \( R \) where \( R \in \Theta \) and \( R \prec^B P \)
using unicov_has_bar_ref by blast
from assms(1,2,3) \( \prec^B \) have \( P \in \text{Covers}(X) \) \( Q \in \text{Covers}(X) \) \( R \in \text{Covers}(X) \)
unfolding AreUniformCovers_def by auto
with assms(1,3,4,5) \( \prec^B \) show thesis using bary_bary_star unfolding AreUniformCovers_def by auto

A technical lemma to simplify proof of the uniformity_from_unicov theorem.

**lemma star_ref_mem:**
assumes \( U \in P \prec^* Q \) and \( \bigcup \{W \times W. W \in Q\} \subseteq A \)
says \( U \times U \subseteq A \)
proof -
from assms(1,2) obtain \( W \) where \( W \in Q \) and \( \bigcup \{S \in P. S \cap U \neq 0\} \subseteq W \)
unfolding IsStarRefinement_def Star_def by auto
with assms(1,3) show \( U \times U \subseteq A \) by blast

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.
lemma rel_square_starr: shows 
\( (\bigcup \{ U \times U. \ U \in \mathcal{P} \}) \circ (\bigcup \{ U \times U. \ U \in \mathcal{P} \}) = \bigcup \{ U \times \text{Star}(U, \mathcal{P}). \ U \in \mathcal{P} \} \)

unfolding Star_def by blast

An identity similar to rel_square_starr but with Star on the left side of the Cartesian product:

lemma rel_square_starl: shows 
\( (\bigcup \{ U \times U. \ U \in \mathcal{P} \}) \circ (\bigcup \{ U \times U. \ U \in \mathcal{P} \}) = \bigcup \{ \text{Star}(U, \mathcal{P}) \times U. \ U \in \mathcal{P} \} \)

unfolding Star_def by blast

A somewhat technical identity about the square of a symmetric relation:

lemma rel_sq_image: assumes \( W = \text{converse}(W) \) domain(W) \( \subseteq X \) shows \( \text{Star}\{x\},\{\text{W}\{t\}. \ t \in X\} = (W \circ W)\{x\} \)

proof

have I: \( \text{Star}\{x\},\{\text{W}\{t\}. \ t \in X\} = \bigcup \{ S \in \{\text{W}\{t\}. \ t \in X\}. \ x \in S \} \)

unfolding Star_def by auto

{ fix y assume y \in \text{Star}\{x\},\{\text{W}\{t\}. \ t \in X\} 
  with I obtain S where y \in S x \in S S \in \{\text{W}\{t\}. \ t \in X\} by auto 
  from \( \langle S \in \{\text{W}\{t\}. \ t \in X\} \rangle \) obtain t where t \in X and S = \text{W}\{t\} by auto 
  with \( \langle x \in S \rangle \langle y \in S \rangle \) have \( \langle t, x \rangle \in \text{W} \) and \( \langle t, y \rangle \in \text{W} \) using rel_compdef by auto 
} then show \( \text{Star}\{x\},\{\text{W}\{t\}. \ t \in X\} \subseteq (W \circ W)\{x\} \) by blast

{ fix y assume y \in (W \circ W)\{x\} 
  then obtain t where \( \langle x, t \rangle \in \text{W} \) and \( \langle t, y \rangle \in \text{W} \) using rel_compdef by auto 
  from assms(2) \( \langle t, y \rangle \in \text{W} \) have t \in X by auto 
  from \( \langle x, t \rangle \in \text{W} \) have \( \langle t, x \rangle \in \text{converse}(\text{W}) \) by auto 
  with assms(1) \( \langle t, y \rangle \in \text{W} \) \( \langle t \in X \rangle \) have \( y \in \text{Star}\{x\},\{\text{W}\{t\}. \ t \in X\} \) by auto 
} then show \( (W \circ W)\{x\} \subseteq \text{Star}\{x\},\{\text{W}\{t\}. \ t \in X\} \) 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 \( \text{UniformityFromUniCov}(X, \Theta) \equiv \text{Supersets}(X \times X, \{\bigcup \{ U \times U. \ U \in \mathcal{P}\}. \ P \in \Theta\}) \)

For any member \( P \) of a cover \( \Theta \) the set \( \bigcup \{ U \times U : U \in P \} \) is a member of UniformityFromUniCov(X, \Theta).
lemma basic_unif: assumes $\Theta \subseteq \text{Covers}(X)$ $P \in \Theta$
shows $\bigcup \{U \times U. \ U \in P\} \in \text{UniformityFromUniCov}(X, \Theta)$
using assms unfolding UniformityFromUniCov_def Supersets_def Covers_def
by blast

If $\Theta$ is a family of uniform covers of $X$ then $\text{UniformityFromUniCov}(X, \Theta)$ is a uniformity on $X$

theorem uniformity_from_unicov:
assumes $\Theta$ {are uniform covers of} $X$ $X \neq 0$
shows $\text{UniformityFromUniCov}(X, \Theta)$ {is a uniformity on} $X$
proof -
let $\Phi = \text{UniformityFromUniCov}(X, \Theta)$
have $\Phi$ {is a filter on} $(X \times X)$
proof -
{ assume $0 \notin \Phi$
then obtain $P$ where $P \in \Theta$ and $0 = \bigcup \{U \times U. \ U \in P\}$
unfolding UniformityFromUniCov_def Supersets_def by auto
hence $\bigcup \{U \times U. \ U \in P\} = 0$ by auto
with assms $P \in \Theta$ have False unfolding AreUniformCovers_def Covers_def
by auto
} thus thesis by auto
qed moreover have $X \times X \in \Phi$
proof -
from assms have $X \times X \in \{\bigcup \{U \times U. \ U \in P\}. \ P \in \Theta\}$
using unicov_contains_triv unfolding AreUniformCovers_def
by auto
then show thesis unfolding Supersets_def UniformityFromUniCov_def
by blast
qed
moreover have $\Phi \subseteq \text{Pou}(X \times X)$
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 $P_A \ P_B$ where
$P_A \in \Theta \ P_B \in \Theta$ and $I: \bigcup \{U \times U. \ U \in P_A\} \subseteq A \ \bigcup \{U \times U. \ U \in P_B\} \subseteq B$
unfolding UniformityFromUniCov_def Supersets_def by auto
from assms(1) $P_A \in \Theta \ P_B \in \Theta$ obtain $P$
where $P \in \Theta$ and $P \prec P_A$ and $P \prec P_B$
unfolding AreUniformCovers_def by blast
have $\bigcup \{U \times U. \ U \in P\} \subseteq A \cap B$
proof -
{ fix $U$ assume $U \in P$

1119
with \( \langle P' \rangle < A \) \( \langle P' \rangle > B \) I have \( U \times U \subseteq A \) and \( U \times U \subseteq B \)

using \texttt{star_ref_mem} by \texttt{auto}

\} thus thesis by \texttt{blast}

\begin{document}

\section*{Theorem 1}

\begin{proof}

\begin{enumerate}

\item \( A \cap B \subseteq X \times X \)

\end{enumerate}

\end{proof}

\section*{Theorem 2}

\begin{proof}

\begin{enumerate}

\item \( B \subseteq C \) \( \Rightarrow \) \( C \subseteq \Phi \)

\end{enumerate}

\end{proof}

ultimately show thesis unfolding \texttt{IsFilter_def} by \texttt{simp}

\begin{proof}

\begin{enumerate}

\item \( A \in \Phi \)

\item \( \exists B \in \Phi \) \( B \subseteq A \)

\end{enumerate}

\end{proof}

moreover have \( \forall A \in \Phi \). \texttt{id}(X) \subseteq A \land (\exists B \in \Phi \cdot B \bullet B \subseteq A) \land \texttt{converse}(A)

\begin{proof}

\begin{enumerate}

\item \( A \in \Phi \)

\end{enumerate}

\end{proof}

moreover have \( \forall B \in \Phi \cdot C \subseteq \texttt{Pow}(X \times X) \rightarrow C \subseteq \Phi \)

\begin{proof}

\begin{enumerate}

\item \( B \subseteq C \)

\end{enumerate}

\end{proof}

\section*{Theorem 3}

\begin{proof}

\begin{enumerate}

\item \( B \subseteq C \)

\end{enumerate}

\end{proof}

moreover have \( \forall B \in \Phi \cdot \exists C \in \texttt{Pow}(X \times X) \cdot B \subseteq C \)

\begin{proof}

\begin{enumerate}

\item \( B \subseteq C \)

\end{enumerate}

\end{proof}

ultimately show thesis unfolding \texttt{IsFilter_def} by \texttt{simp}

\end{document}
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 \( \langle Q \triangleleft P \rangle \) obtain \( V \in P \) and \( \text{Star}(U,Q) \subseteq V \times V \) unfolding \text{IsStarRefinement_def} by blast
  unfolding \text{IsStarRefinement_def} by blast
thus \( \exists V \in P. U \times \text{Star}(U,Q) \subseteq V \times V \) using \text{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 \( \langle V \times V. V \in P \rangle \subseteq A \rangle \) 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 \( \langle A \in \Phi. \langle P \in \Theta. \langle \bigcup \{ U \times U. U \in P \} \subseteq A \rangle \rangle \langle A \rangle \) have \( \text{converse}(A) \in \Phi \)
  unfolding \text{AreUniformCovers_def UniformityFromUniCov_def Supersets_def}
  by auto
ultimately show \( \text{id}(X) \subseteq A \land (\exists B \in \Phi. B \circ B \subseteq A) \land \text{converse}(A) \in \Phi \)
  by simp
qed
ultimately show \( \Phi \) \{is a uniformity on\} \( X \) unfolding \text{IsUniformity_def}
  by simp
qed

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} \( X \neq 0 \)

\( \text{shows} \) \( \text{UniCovFromUniformity}(X,\Phi) = (\bigcup U \in \Phi. \bigcap x \in X. \{ P \in \text{Covers}(X). \exists A \in P. U({x}) \subseteq A \}) \)
proof -
have \( \{ P \in \text{Covers}(X) : \exists U \in \Phi. \forall x \in X. \exists A \in P. U(\{x\}) \subseteq A \} \) = 
\((\bigcup U \in \Phi. \bigcap x \in X. \{ P \in \text{Covers}(X) : \exists A \in P. U(\{x\}) \subseteq A \})\)

proof
\{ fix P assume \( P \in \text{Covers}(X) \) and \( \exists U \in \Phi. \forall x \in X. \exists A \in P. U(\{x\}) \subseteq A \) 
then obtain \( U \) where \( U \in \Phi \) and \( \forall x \in X. \exists A \in P. U(\{x\}) \subseteq A \) by auto 
with 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 \( \{ U \in \Phi \} \) have \( P \in \bigcup U \in \Phi. \bigcap x \in X. \{ P \in \text{Covers}(X) : \exists A \in P. U(\{x\}) \subseteq A \} \)
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 
\( (\bigcup U \in \Phi. \bigcap x \in X. \{ P \in \text{Covers}(X) : \exists A \in P. U(\{x\}) \subseteq A \}) \)
using subset_iff by simp
\(
\{ fix P assume \( P \in (\bigcup U \in \Phi. \bigcap x \in X. \{ P \in \text{Covers}(X) : \exists A \in P. U(\{x\}) \subseteq A \}) \) 
then obtain \( U \) where \( U \in \Phi \) and \( P \in \{ \bigcap x \in X. \{ P \in \text{Covers}(X) : \exists A \in P. U(\{x\}) \subseteq A \} \} \)
\}

by auto
with assms have \( P \in \text{Covers}(X) \) and \( \forall x \in X. \exists A \in P. U(\{x\}) \subseteq A \) by auto
with \( \{ U \in \Phi \} \) have \( P \in \{ \bigcap U \in \Phi. \{ P \in \text{Covers}(X) : \exists A \in P. U(\{x\}) \subseteq A \} \} \)
by auto
\}

then show \( (\bigcup U \in \Phi. \bigcap x \in X. \{ P \in \text{Covers}(X) : \exists A \in P. U(\{x\}) \subseteq A \}) \subseteq 
\{ P \in \text{Covers}(X) : \exists U \in \Phi. \forall x \in X. \exists A \in P. U(\{x\}) \subseteq A \} \) by auto
qed
then show thesis unfolding UniCovFromUniformity_def by simp
qed

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 \( P \in \text{Covers}(X) \)
proof -
from assms have \( W \subseteq X \times X \) and \( P \in \text{Pow}(\text{Pow}(X)) \)
using unif_props(1) by auto
moreover have \( \bigcup P = X \)
proof
from \( \{ W \subseteq X \times X \} \) show \( \bigcup P \subseteq X \) by auto
from assms show \( X \subseteq \bigcup P \) using neigh_not_empty(2) by auto
qed
ultimately show thesis unfolding Covers_def by simp
qed
moreover from assms(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 \text{common_bar_refinement}:

assumes
\( \Phi \; \{\text{is a uniformity on}\} \; X \)
\( \Theta = \text{UniCovFromUniformity}(X, \Phi) \)
\( C \in \Theta \; \; D \in \Theta \)

shows \( \exists R \in \Theta . (R \prec^B C) \land (R \prec^B D) \)

proof -

from assms(2,3) obtain \( U \) where \( U \in \Phi \) and \( I: \forall x \in X. \exists C \in C. U\{x\} \subseteq C \)

unfolding \text{UniCovFromUniformity}\_\text{def} by auto

from assms(2,4) obtain \( V \) where \( V \in \Phi \) and \( II: \forall x \in X. \exists D \in D. V\{x\} \subseteq D \)

unfolding \text{UniCovFromUniformity}\_\text{def} by auto

from assms(1) \( \prec U \prec V \in \Phi \)

unfolding \text{IsUniformity}\_\text{def} \text{IsFilter}\_\text{def} by auto

with assms(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 assms(1) \( \prec W \in \Phi \)

unfolding \text{IsUniformity}\_\text{def} \text{IsFilter}\_\text{def} by auto

let \( P = \{ W\{t\}. t \in X \} \)

have \( P \in \Theta \)

\( P \prec^B C \) \( P \prec^B D \)

proof -

from assms(1,2) \( \prec W \in \Phi \)

show \( P \in \Theta \) using \text{cover_image} by simp

with assms(2) have \( \bigcup P = X \)

unfolding \text{UniCovFromUniformity}\_\text{def} \text{Covers}\_\text{def} by simp

fix \( x \) assume \( x \in X \)

from \( \prec W \circ \text{converse}(W) \prec \text{domain}(W) \subseteq X \)

have \( \text{Star}({}\{x\},P) \subseteq U\{x\} \) and \( \text{Star}({}\{x\},P) \subseteq V\{x\} \)

using \text{rel}\_\text{sq}\_\text{image} by auto

from \( \prec x \in X \)

obtain \( C \) where \( C \in C \) and \( U\{x\} \subseteq C \)

by auto

with \( \text{Star}({}\{x\},P) \subseteq U\{x\} \prec C \in C \) have \( \exists C \in C. \text{Star}({}\{x\},P) \subseteq C \)

by auto

moreover

from \( \prec x \in X \)

obtain \( D \) where \( D \in D \) and \( V\{x\} \subseteq D \)

by auto

with \( \text{Star}({}\{x\},P) \subseteq V\{x\} \prec D \in D \) 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 \( \bigcup P = X \) show \( P \prec^B C \) and \( P \prec^B D \)
If \( \Phi \) is a (diagonal) uniformity on \( X \), then every element of \( \text{UniCovFromUniformity}(X, \Phi) \) has a barycentric refinement there.

**corollary** \text{bar_refinement_ex}:  
~~~  
assumes \( \Phi \) {is a uniformity on} \( X \)  
\[ \Theta = \text{UniCovFromUniformity}(X, \Phi) \]  
\[ C \in \Theta \]  
shows \( \exists R \in \Theta. (R <^\Theta C) \)  
using 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** \text{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 assms(1) have \( \Theta \subseteq \text{Covers}(X) \) unfolding \text{UniCovFromUniformity_def}  
moreover have \( \forall R \in \Theta. \forall C \in \text{Covers}(X). ((R <^* C) \longrightarrow C \in \Theta) \)  
proof -  
~~~  
{ fix \( R, C \) assume \( R \in \Theta \land C \in \text{Covers}(X) \land R <^* C \) having \( C \in \Theta \)  
proof -  
~~~  
from \( <R \in \Theta> \) obtain \( U \) where \( U \subseteq \Phi \) and \( I: \forall x \in X. \exists R \in R. U(\{x\}) \subseteq R \)  
by auto  
moreover have \( \forall R \in \Theta \land C \in \text{Covers}(X) \land R <^* C \)  
~~~  
with \( U \subseteq \Phi \) \, \text{element_subset_star} \quad \text{by blast}  
~~~  
with \( C \subseteq C \) \, \text{by auto}  
\} hence \( \forall x \in X. \exists C \subseteq C \land U(\{x\}) \subseteq C \) by auto  
with \( <U \in \Phi> \) \, \text{Covers} \quad \text{by auto}  
\} thus thesis unfolding \text{UniCovFromUniformity_def} by auto  
~~~
qed

moreover have \( \forall C \in \Theta. \forall D \in \Theta. \exists R \in \Theta. (R <^* C) \land (R <^* D) \)

proof -

\{ fix \( C \) \( D \) assume \( C \in \Theta \) \( D \in \Theta \)
  with assms(1) obtain \( P \) where \( P \in \Theta \) and \( P <^B C \) \( P <^B D \)
  using common_bar_refinement by blast
  from assms(1) \( <P \in \Theta> \) obtain \( R \) where \( R \in \Theta \) and \( R <^B P \)
  using bar_refinement_ex by blast
  from \( <R \in \Theta> <P \in \Theta> <C \in \Theta> <D \in \Theta> \) have
  \( P \in \text{Covers}(X) \) \( R \in \text{Covers}(X) \) \( C \in \text{Covers}(X) \) \( D \in \text{Covers}(X) \)
  unfolding UniCovFromUniformity_def by auto
  with assms(2) \( <R \in \Theta> <P \in \Theta> <C \in \Theta> <D \in \Theta> \) have \( R <^* C \) and \( R <^* D \)
  using bary_bary_star by auto
  with \( <R \in \Theta> \) have \( \exists R \in \Theta. (R <^* C) \land (R <^* D) \) by auto
\} thus thesis by simp
qed

ultimately show thesis unfolding AreUniformCovers_def by simp
qed

The UniCovFromUniformity operation is the inverse of UniformityFromUniCov.

theorem unicov_from_unif_inv: assumes \( \Theta \) \{are uniform covers of\} \( X \)
\( X \neq 0 \) shows UniCovFromUniformity(\( X \), UniformityFromUniCov(\( X \), \( \Theta \))) = \( \Theta \)

proof
  let \( \Phi = \text{UniformityFromUniCov}(X, \Theta) \)
  let \( L = \text{UniCovFromUniformity}(X, \Phi) \)
  from assms have I: \( \Phi \) \{is a uniformity on\} \( X \)
    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 \( P \in L \)
    with I obtain \( Q \) where \( Q \in L \) and \( Q <^B P \)
    using bar_refinement_ex by blast
    from \( <Q \in L> \) obtain \( U \) where \( U \in \Phi \) and \( U(x) \subseteq A \)
      unfolding UniCovFromUniformity_def by auto
    from \( <U \in \Phi> \) have \( U \in \text{Supersets}(X \times X, \{ \bigcup \{ U \times U. U \in P \}. P \in \Theta \}) \)
      unfolding UniformityFromUniCov_def by simp
    then obtain \( B \) where \( B \subseteq X \times X \) \( B \subseteq U \) and \( \exists C \in \bigcup \{ U \times U. U \in P \}. P \in \Theta \). \( C \subseteq B \)
      unfolding Supersets_def by auto
    then obtain \( C \) where \( C \in \bigcup \{ U \times U. U \in P \}. P \in \Theta \) and \( C \subseteq B \)
    by auto
    then obtain \( R \) where \( R \in \Theta \) and \( C = \bigcup \{ V \times V. V \in R \} \) by auto
    with \( <C \subseteq B> <B \subseteq U> \) have \( \bigcup \{ V \times V. V \in R \} \subseteq U \) by auto
    from assms(1) II \( <P \in L> <Q \in L> <R \in \Theta> \) have
    IV: \( P \in \text{Covers}(X) \) \( Q \in \text{Covers}(X) \) \( R \in \text{Covers}(X) \)
      unfolding AreUniformCovers_def by auto
    have \( R <^B Q \)
    proof -
      \{ fix \( x \) assume \( x \in X \)

1125
with III obtain $A$ where $A \in \mathbb{Q}$ and $U\{x\} \subseteq A$ by auto
with $\text{refl}(\bigcup \{V \times V. V \in \mathbb{R}\}) \subseteq U$ have $\bigcup \{V \times V. V \in \mathbb{R}\}\{x\} \subseteq A$
by auto
with $\{A \in \mathbb{Q}. \text{Star}(\{x\}, R) \subseteq A\}$ using star_singleton by auto
} then have $\forall x \in X. \exists A \in \mathbb{Q}. \text{Star}(\{x\}, R) \subseteq A$ by simp
moreover from $\text{refl}(\bigcup \{V \times V. V \in \mathbb{R}\}) \subseteq U$
unfolding Covers_def by simp
ultimately show thesis unfolding IsBarycentricRefinement_def by simp
qed

with assms(2) $Q \in \mathbb{B}$ have $R \subseteq \mathbb{B}$
by bary_bary_star unfolding Covers_def by auto
with assms(1) $P \in \bigcup \text{Covers}(X)$ have $P \in \mathbb{B}$
unfolding AreUniformCovers_def by simp
thus $L \subseteq \mathbb{B}$ by auto

fix $P$ assume $P \in \mathbb{B}$
with assms(1) have $P \in \bigcup \text{Covers}(X)$
unfolding AreUniformCovers_def by auto
from assms(1) $P \in \mathbb{B}$ obtain $Q$ where $Q \in \mathbb{B}$ and $Q \subseteq \mathbb{B}$
using unicov_has_bar_ref by blast
let $A = \bigcup \{V \times V. V \in \mathbb{Q}\}$
have $A \in \mathbb{B}$ by auto
proof -
from assms(1) $Q \in \mathbb{B}$ have $A \subseteq X \times X$ and $A \in \{\bigcup \{V \times V. V \in \mathbb{Q}\}. Q \in \mathbb{B}\}$
unfolding AreUniformCovers_def Covers_def by auto
then show thesis
using superset_gen unfolding UniformityFromUniCov_def by auto
qed

with I II $B \in \mathbb{B}$ have $R \subseteq \mathbb{B}$
unfolding half_size_symm by blast
let $R = \{B\{x\}, x \in X\}$
from I II $B \in \mathbb{B}$ have $R \subseteq \bigcup R = X$
unfolding cover_image unfolding UniCovFromUniformity_def Covers_def by auto
have $R \subseteq \mathbb{B}$ by auto
proof -
fix $x$ assume $x \in X$
from assms(1) $Q \in \mathbb{B}$ have $\bigcup Q = X$
unfolding AreUniformCovers_def Covers_def by auto
with $Q \subseteq \mathbb{B}$ $x \in X$ obtain $C$ where $C \subseteq \mathbb{B}$ and $\text{Star}(\{x\}, Q) \subseteq C$
unfolding IsBarycentricRefinement_def by auto
from $B = \text{converse}(B)$ I $B \in \mathbb{B}$ have $\text{Star}(\{x\}, R) = (B \circ B)\{x\}$
using uni_domain rel_sq_image by auto
moreover from $B \subseteq A$ have $B \circ B \subseteq A\{x\}$ by blast
moreover have $A\{x\} = \text{Star}(\{x\}, Q)$ using star_singleton by simp
ultimately have $\text{Star}(\{x\}, R) \subseteq \text{Star}(\{x\}, Q)$ by auto
with $\text{Star}(\{x\}, Q) \subseteq C$ $C \subseteq \mathbb{B}$ have $\exists C \subseteq \mathbb{B}$. $\text{Star}(\{x\}, R) \subseteq C$ by auto
with \( \bigcup R = X \) show thesis unfolding IsBarycentricRefinement_def by auto

qed

with assms(2) II \( \langle P \in \text{Covers}(X) \rangle \langle R \in L \rangle \langle R \prec^B P \rangle \) have \( P \in L \)
using unicovert_bary_cov by simp

\} thus \( \Theta \subseteq L \) by auto

qed

The UniformityFromUniCov operation is the inverse of UniCovFromUniformity.

theorem unif_from_unicov_inv: assumes \( \Phi \{\text{is a uniformity on} \} X \neq 0 \)
shows \( \text{UniformityFromUniCov}(X, \text{UniCovFromUniformity}(X, \Phi)) = \Phi \)
proof
let \( \Theta = \text{UniCovFromUniformity}(X, \Phi) \)
let \( L = \text{UniformityFromUniCov}(X, \Theta) \)
from assms have I: \( \Theta \{\text{are uniform covers of} \} X \)
using unicovert_from_uniformity by simp
with assms have II: \( L \{\text{is a uniformity on} \} X \)
using unif_from_unicov by simp

\{ fix \( A \) assume \( A \in \Phi \)
with assms(1) obtain \( B \in \Phi \) \( B \circ B \subseteq A \) and \( B = \text{converse}(B) \)
using half_size_symm by blast
from assms(1) \( \langle A \in \Phi \rangle \) have \( A \subseteq X \times X \)
using uni_domain(1) by simp
let \( P = \{ B(x). \ x \in X \} \)
from assms(1) \( \langle B \in \Phi \rangle \) have \( P \in \Theta \)
using cover_image by simp
let \( C = \bigcup \{ U \times U. \ U \in P \} \)
from I \( \langle P \in \Theta \rangle \) have \( C \subseteq L \)
unfolding AreUniformCovers_def using basic_unif by blast
from assms(1) \( \langle B \in \Phi \rangle \langle B = \text{converse}(B) \rangle \langle B \circ B \subseteq A \rangle \) have \( C \subseteq A \)
using uni_domain(2) symm_sq_prod_image by simp
with II \( \langle A \subseteq X \times X \rangle \langle C \subseteq L \rangle \) 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 \( \langle A \subseteq X \times X \rangle \) using unif_props(1) by simp
from \( \langle A \in L \rangle \) obtain \( P \) where \( P \in \Theta \) and \( \bigcup \{ U \times U. \ U \in P \} \subseteq A \)
unfolding \( \text{UniformityFromUniCov}(X, \text{UniCovFromUniformity}(X, \Phi)) = \Phi \)
Supersets_def by blast
from \( \langle P \in \Theta \rangle \) obtain \( B \) where \( B \in \Phi \) and \( \langle \forall x \in X. \ \exists V \in P. B(x) \subseteq V \rangle \)
unfolding \( \text{UniformityFromUniCov}(X, \text{UniCovFromUniformity}(X, \Phi)) = \Phi \)
by auto
have \( B \subseteq A \)
proof -
from assms(1) \( \langle B \in \Phi \rangle \) have \( B \subseteq \bigcup \{ B(x) \times B(x). \ x \in X \} \)
using unif_props 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
\}
This theory is about the first subject of algebraic topology: topological groups.

77.1 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**

\[
IsATopologicalGroup(T,f) \equiv (T \text{ is a topology}) \land \text{IsAgroup}(\bigcup T,f) \land \\
\text{IsContinuous(ProductTopology}(T,T),T,f) \land \\
\text{IsContinuous}(T,T,\text{GroupInv}(\bigcup T,f))
\]

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 \( \langle A, B \rangle \mapsto \{ x + y | x \in A, y \in B \} \). 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 neighborhoods 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]: \( \tau \equiv \text{ProductTopology}(T,T) \)

  fixes \( f \)
  assumes Ggroup: IsAgroup(G,f)
  assumes fcon: IsContinuous(\( \tau,T,f \))
  assumes inv_cont: IsContinuous(T,T,GroupInv(G,f))

  fixes grop (infixl + 90)
  defines grop_def [simp]: \( x+y \equiv f(x,y) \)

  fixes grinv (- _ 89)
  defines grinv_def [simp]: \( (-x) \equiv \text{GroupInv}(G,f)(x) \)

  fixes grsub (infixl - 90)
  defines grsub_def [simp]: \( x-y \equiv x+(-y) \)

  fixes setinv (- _ 72)
  defines setinv_def [simp]: \( -A \equiv \text{GroupInv}(G,f)(A) \)

  fixes ltrans (infix + 73)
  defines ltrans_def [simp]: \( x + A \equiv \text{LeftTranslation}(G,f,x)(A) \)

  fixes rtrans (infix + 73)
  defines rtrans_def [simp]: \( A + x \equiv \text{RightTranslation}(G,f,x)(A) \)

  fixes setadd (infixl + 71)
  defines setadd_def [simp]: \( A + B \equiv (f \{\text{lifted to subsets of} \} \ G)(A,B) \)

  fixes gzero (0)
  defines gzero_def [simp]: \( 0 \equiv \text{TheNeutralElement}(G,f) \)
fixes zerohoods \( (N_0) \)
defines zerohoods_def [simp]: \( N_0 \equiv \{ A \in \text{Pow}(G). \ 0 \in \text{int}(A) \} \)

fixes listsum \( (\sum_{-70}) \)
defines listsum_def [simp]: \( \sum_k \equiv \text{Fold1}(f,k) \)

The first lemma states that we indeed talk about topological group in the context of topgroup locale.

**lemma (in topgroup) topGroup**: shows \( \text{IsAtopologicalGroup}(T,f) \)

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 \( \text{IsAtopologicalGroup}(T,f) \)

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 \( g \in G \) shows \( (-g) \in G \)

using assms inverse_in_group by simp

Sum of two group elements is in the group.

**lemma (in topgroup) group_op_closed_add**: assumes \( x_1 \in G \ x_2 \in G \)

shows \( x_1 + x_2 \in G \)

using assms group_op_closed by simp

Zero is in the group.

**lemma (in topgroup) zero_in_tgroup**: shows \( 0 \in G \)
Another lemma about canceling with two group elements written in additive notation

**lemma (in topgroup) inv_cancel_two_add:**
assumes \( x_1 \in G \ x_2 \in G \)
shows
\[ x_1 + (\neg x_2) + x_2 = x_1 \]
\[ x_1 + x_2 + (\neg x_2) = x_1 \]
\[ (\neg x_1) + (x_1 + x_2) = x_2 \]
\[ x_1 + ((\neg x_1) + x_2) = x_2 \]
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 \( x_1 \in G \ x_2 \in G \ x_3 \in G \)
shows
\[ (x_1 + (\neg x_2)) + (x_2 + (\neg x_3)) = x_1 + (\neg x_3) \]
\[ ((\neg x_1) + x_2) + ((\neg x_2) + x_3) = (\neg x_1) + x_3 \]
\[ (\neg x_1) + (x_1 + x_2) + (\neg x_3) = x_2 \]
\[ (x_1 + x_2) + (\neg (x_3 + x_2)) = x_1 + (\neg x_3) \]
\[ (\neg x_1) + (x_1 + x_2 + x_3) + (\neg x_3) = x_2 \]
proof -
from assms have \( x_1 + (\neg x_3) = (x_1 + (\neg x_2)) + (x_2 + (\neg x_3)) \)
using group0_2_L14A(1) by blast
thus \( (x_1 + (\neg x_2)) + (x_2 + (\neg x_3)) = x_1 + (\neg x_3) \) by simp
from assms have \( (\neg x_1) + x_3 = ((\neg x_1) + x_2) + ((\neg x_2) + x_3) \)
using group0_2_L14A(2) by blast
thus \( ((\neg x_1) + x_2) + ((\neg x_2) + x_3) = (\neg x_1) + x_3 \) by simp
from assms show \( (\neg x_1) + (x_1 + x_2) + (\neg x_3) = x_2 \)
using cancel_middle(1) by simp
from assms show \( (x_1 + x_2) + (\neg (x_3 + x_2)) = x_1 + (\neg x_3) \)
using cancel_middle(2) by simp
from assms show \( (\neg x_1) + (x_1 + x_2 + x_3) + (\neg x_3) = x_2 \)
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 $x_1 \in G$ $x_2 \in G$ $x_3 = x_1 \cdot x_2$
shows $x_3 \cdot (-x_2) = x_1$ and $(-x_1) \cdot x_3 = x_2$
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 $x_1 \in G$ $x_2 \in G$ $x_3 \in G$ $x_1 \cdot (-x_2) = (-x_3)$
shows $x_3 = x_2 \cdot (-x_1)$
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 $x_1 \in G$ $x_2 \in G$ $x_3 \in G$ $(-x_1) \cdot x_2 = (-x_3)$
shows $x_3 = (-x_2) \cdot x_1$
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

**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 $G \times G$).

**lemma** (in topgroup) prod_top_on_G:
shows $\tau$ {is a topology} 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.

**lemma** (in topgroup) topgroup_f_binop: 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 $T$ {restricted to} $H$ which is defined to be $\{ V \cap H : V \in T \}$ in ZF1 theory.
lemma (in topgroup) top_subgroup: assumes A1: IsAsubgroup(H,f)
  shows IsAtopologicalGroup(T {restricted to} H,restrict(f,H×H))
proof -
  let τ₀ = T {restricted to} H
  let f_H = restrict(f,H×H)
  have ∪τ₀ = G ∩ H using union_restrict by simp
  also from A1 have ... = H
  using group0_3_L2 by blast
  finally have ∪τ₀ = H by simp
  have τ₀ is a topology} using Top_1.L4 by simp
  moreover from A1 <∪τ₀ = H> have IsAgroup(∪τ₀,f_H)
  using IsAsubgroup_def by simp
  moreover have IsContinuous(ProductTopology(τ₀,τ₀),τ₀,f_H)
proof -
  have two_top_spaces0(τ, T,f)
  using topSpaceAssum prod_top_on_G topgroup_f_binop prod_top_on_G
  two_top_spaces0_def by simp
  moreover from A1 have H ⊆ G using group0_3_L2 by simp
  then have H×H ⊆ ∪τ using prod_top_on_G by auto
  moreover have IsContinuous(τ,T,f) using fcon by simp
  ultimately have IsContinuous(τ {restricted to} H×H, T {restricted to} f_H(H×H),f_H)
  using two_top_spaces0.restr_restr_image_cont
  by simp
  moreover have ProductTopology(τ₀,τ₀) = τ {restricted to} H×H using topSpaceAssum
  prod_top_restr_comm
  by simp
  moreover from A1 have f_H(H×H) = H using image_subgr_op
  by simp
  ultimately show thesis by simp
qed
moreover have IsContinuous(τ₀,τ₀,GroupInv(∪τ₀,f_H))
proof -
  let g = restrict(GroupInv(G,f),H)
  have GroupInv(G,f) : G -> 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 ⊆ ∪T using group0_3_L2 by simp
  ultimately have IsContinuous(τ₀,T {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 GroupInv(H,f_H) = g using group0_3_T1 by simp
  with <∪τ₀ = H> have g = GroupInv(∪τ₀,f_H) by simp
ultimately show thesis by simp
qed
ultimately show thesis unfolding IsAtopologicalGroup_def by simp
qed

77.2 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 $A \subseteq G \land B \subseteq G$ 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 $A \subseteq G$ 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 $V \subseteq G \land x \in G$
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

1134
A corollary from interval_add

corollary (in topgroup) elements_in_set_sum: assumes A ⊆ G B ⊆ G
t ∈ A+B shows ∃ s ∈ A. ∃ q ∈ B. t = s + q
using assms interval_add(4) by auto

A corollary from lrtrans_image

corollary (in topgroup) elements_in_ltrans:
  assumes B ⊆ G g ∈ G t ∈ g+B
  shows ∃ q ∈ B. t = g + q
  using assms lrtrans_image(1) by simp

Another corollary of lrtrans_image

corollary (in topgroup) elements_in_rtrans:
  assumes B ⊆ G g ∈ G t ∈ B+g shows ∃ q ∈ 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 ⊆ G B ⊆ G s ∈ A q ∈ B
  shows t ∈ A+B
  using assms interval_add by auto

Another corollary of lrtrans_image

corollary (in topgroup) elements_in_ltrans_inv:
  assumes B ⊆ G g ∈ G q ∈ B
  shows t ∈ g+B
  using assms lrtrans_image(1) by auto

Another corollary of rtrans_image_add

lemma (in topgroup) elements_in_rtrans_inv:
  assumes B ⊆ G g ∈ G q ∈ B
  shows t ∈ B+g
  using assms lrtrans_image(2) by auto

Right and left translations are continuous.

lemma (in topgroup) trans_cont: assumes g ∈ G shows
  IsContinuous(T, T, RightTranslation(G, f, g)) and
  IsContinuous(T, T, LeftTranslation(G, f, g))
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.

lemma (in topgroup) open_tr_open: assumes g ∈ G and V ∈ T
  shows g+V ∈ T and V+g ∈ T
  using assms neg_in_tgroup trans_cont IsContinuous_def trans_image_vimage
  by auto

1135
Right and left translations are homeomorphisms.

**Lemma** (in topgroup) `tr_homeo`:
- Assumes `g ∈ G` shows `IsAhomeomorphism(T,T,RightTranslation(G,f,g))` and `IsAhomeomorphism(T,T,LeftTranslation(G,f,g))`
  - Using `assms` `trans_bij` `trans_cont` `open_tr_open` `bij_cont_open_homeo`
  - By `auto`

Left translations preserve interior.

**Lemma** (in topgroup) `ltrans_interior`:
- Assumes `A1: g ∈ G` and `A2: A ⊆ G` shows `g + int(A) = int(g+A)`
  - Proof -
    - From `assms` have `A ⊆ ∪T` and `IsAhomeomorphism(T,T,LeftTranslation(G,f,g))`
      - Using `tr_homeo`
        - By `auto`
      - Then show thesis using `int_top_invariant` by `simp`
    - Qed

Right translations preserve interior.

**Lemma** (in topgroup) `rtrans_interior`:
- 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

Translating by an inverse and then by an element cancels out.

**Lemma** (in topgroup) `trans_inverse_elem`:
- Assumes `g ∈ G` and `A ⊆ 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 ∈ T` shows `(-V) ∈ 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 ⊆ G` shows `int(-A) = -(int(A))`
  - Using `assms` `inv_homeo` `int_top_invariant` by `simp`

1136
77.3 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) nneigh_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 \( g \in G \) \( H \in N_0 \)
  shows \( g \in \text{int}(g + H) \)
proof -
  from A2 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 \( g \in \text{int}(H) + g \) 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 \( g \in \text{int}(H+g) \) \( \Rightarrow \) \text{assms(2)} show \( g \in \text{int}(H+g) + \text{int}(H) \)
using \text{elem_in_int_sad by simp}

qed

Negative of a neighborhood of zero is a neighborhood of zero.

\textbf{lemma (in topgroup) neg_neigh_neigh: assumes } H \in N_0 \text{ shows } (-H) \in N_0
\textbf{proof -}
from \text{assms} have \( \text{int}(H) \subseteq G \) and \( 0 \in \text{int}(H) \) using \text{Top_2_L1} by auto
with \text{assms} have \( 0 \in \text{int}(-H) \) using \text{neut_inv_neut inv_int inv_int} by simp

moreover
have \( \text{GroupInv}(G,f):G \rightarrow G \) using \text{Ggroup group0_2_T2} by simp
then have \( (-H) \subseteq G \) using \text{func1_1_L6} by simp

ultimately show \( \text{thesis} \) by simp

qed

Left translating an open set by a negative of a point that belongs to it makes it a neighborhood of zero.

\textbf{lemma (in topgroup) open_trans_neigh: assumes } A1: U \in T \text{ and } g \in U \text{ shows } (-g)+U \in N_0
\textbf{proof -}
let \( H = (-g)+U \)
from \text{assms} have \( g \in G \) by auto
then have \( (-g) \in G \) using \text{neg_in_tgroup} by simp
with \text{A1} have \( H \in T \) using \text{open_tr_open} by simp

hence \( H \subseteq G \) by auto
moreover have \( 0 \in \text{int}(H) \)

\textbf{proof -}
from \text{assms} have \( U \subseteq G \) and \( g \in U \) by auto
with \( <H \in T> \) show \( 0 \in \text{int}(H) \) using \text{elem_trans_neut Top_2_L3} by auto

ultimately show \( \text{thesis} \) by simp

qed

Right translating an open set by a negative of a point that belongs to it makes it a neighborhood of zero.

\textbf{lemma (in topgroup) open_trans_neigh_2: assumes } A1: U \in T \text{ and } g \in U \text{ shows } U+(-g) \in N_0
\textbf{proof -}
let \( H = U+(-g) \)
from \text{assms} have \( g \in G \) by auto
then have \( (-g) \in G \) using \text{neg_in_tgroup} by simp
with \text{A1} have \( H \in T \) using \text{open_tr_open} by simp

hence \( H \subseteq G \) by auto
moreover have \( 0 \in \text{int}(H) \)

\textbf{proof -}
from \text{assms} have \( U \subseteq G \) and \( g \in U \) by auto

1138
with $\langle H := T \rangle$ 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 $W \in \mathcal{N}_0$ and $x \in G$
shows $x + (W + (-x)) \in \mathcal{N}_0$ and $(x + W) + (-x) \in \mathcal{N}_0$
proof -
  from assms(2) have $x + (W + (-x)) \subseteq G$ using lrtrans_in_group_add(1) by simp
  moreover have $0 \in \text{int}(x + (W + (-x)))$
  proof -
    from assms(2) have $\text{int}(W + (-x)) \subseteq G$
    using neg_in_tgroup lrtrans_in_group_add(2) Top_2_L1 by blast
    with assms(2) have $(x + \text{int}((W + (-x)))) = \{x + y. y \in \text{int}(W + (-x))\}$
    using lrtrans_image(1) by simp
    moreover from assms have $(-x) \in \text{int}(W + (-x))$
    using neg_in_tgroup elem_in_int_ltrans(1) by simp
    ultimately have $x + (-x) \in x + \text{int}(W + (-x))$ by auto
    with assms show thesis using group0_2_L6 neg_in_tgroup lrtrans_in_group_add(2)
ltrans_interior
    by simp
  qed

ultimately show $x + (W + (-x)) \in \mathcal{N}_0$ by simp
from assms(2) have $(x + W)^+(-x) \subseteq G$ using lrtrans_in_group_add(2) neg_in_tgroup

\[
\begin{align*}
  \text{by simp} \\
  \text{moreover have } 0 \in \text{int}((x + W)^+(-x))
\end{align*}
\]
proof -
  from assms(2) have $\text{int}(x + W) \subseteq G$ using lrtrans_in_group_add(1) Top_2_L1
by blast
  with assms(2) have $\text{int}(x + W) + (-x) = \{y + (-x). y \in \text{int}(x + W)\}$
  using neg_in_tgroup lrtrans_image(2) by simp
  moreover from assms have $x \in \text{int}(x + W)$ using elem_in_int_ltrans(1)
  by simp
  ultimately have $x + (-x) \in \text{int}(x + W) + (-x)$ by auto
  with assms(2) have $0 \in \text{int}(x + W) + (-x)$ using group0_2_L6 by simp
  with assms show thesis using group0_2_L6 neg_in_tgroup lrtrans_in_group_add(1)
rtrans_interior
  by auto
  qed
ultimately show $(x + W)^+(-x) \in \mathcal{N}_0$ 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 $A \subseteq ((-x)+B)x \in G \subseteq G$
shows $x+A \subseteq B$

proof-
  from asms(1) have $x+A \subseteq (x+((-x)+B))$ by auto
  with asms(2,3) show $x+A \subseteq B$
  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 $U \in \mathcal{N}_0$
  shows $\exists V \in \mathcal{N}_0. (V \subseteq U \land (-V)=V)$

proof
let $V = U \cap (-U)$
  have $U \subseteq G$ using asms unfolding zerohoods_def by auto
  then have $V \subseteq G$ by auto
  have invg: GroupInv(G, f) $\in G \to G$ using group0_2_T2 Ggroup by auto
  have invb: GroupInv(G, f) $\in bij(G,G)$ using group_inv_bij(2) by auto
  have $(-V)=\text{GroupInv}(G,f)-V$ unfolding setninv_def using inv_image_vimage
    by auto
  also have $\ldots=(\text{GroupInv}(G,f)-U) \cap (\text{GroupInv}(G,f)-(-U))$ using invim_inter_inter_invim
    by auto
  also have $\ldots=(-U) \cap (\text{GroupInv}(G,f)-(\text{GroupInv}(G,f)U))$
    unfolding setninv_def using inv_image_vimage by auto
  also from $U \subseteq G$ have $\ldots=(-U) \cap U$ using 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 asms have $(-U) \in \mathcal{N}_0$ using neg_neigh_neigh by auto
  with asms have $0 \in \text{int}(U) \cap \text{int}(-U)$ unfolding zerohoods_def by auto
  moreover have $\text{int}(U) \cap \text{int}(-U) = \text{int}(V)$ using int_inter_int by simp
  ultimately have $0 \in \text{int}(V)$ by (rule set_mem_eq)
  with $V \subseteq G$ show $V \in \mathcal{N}_0$ using zerohoods_def by auto
qed

We can say even more than in 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 $U$.

theorem (in topgroup) exists_procls_zerohood:
  assumes $U \in \mathcal{N}_0$
  shows $\exists V \in \mathcal{N}_0. (V \subseteq U \land (V+V) \subseteq U \land (-V)=V)$

proof-
  have $\text{int}(U) \in \mathcal{T}$ using Top_2_L2 by auto
  then have $f^{-1}(\text{int}(U)) \in \mathcal{T}$ using fcon IsContinuous_def by auto

1140
moreover have fn: f ⟨0, 0⟩ = 0 using group0_2_L2 by auto
moreover
have 0 ∈ int(U) using assms unfolding zerohoods_def by auto
then have f - {0} ⊆ f -(int(U)) using func1_1_L8 vimage_def by auto
then have GroupInv(G,f) ⊆ f -(int(U)) using group0_2_T3 by auto
then have ⟨0, 0⟩ ∈ f -(int(U)) using fn: zero_in_tgroup unfolding GroupInv_def by auto
ultimately obtain W V where
  wop: W ∈ T and vop: V ∈ T and cartsub: W × V ⊆ f -(int(U)) and zerhood: ⟨0, 0⟩ ∈ W × V

  using prod_top_point_neighb topSpaceAssum
  unfolding prodtop_def by force
then have 0 ∈ W and 0 ∈ V by auto
then have 0 ∈ W ∩ V by auto
have sub: W ∩ V ⊆ G using wop vop G_def by auto
have assoc: f ⊢ G × G → G using group_oper_fun by auto
{
  fix t, s
  assume t ∈ W ∩ V and s ∈ W ∩ V
  then have t ∈ W and s ∈ V by auto
  then have ⟨t, s⟩ ∈ W × V by auto
  then have ⟨t, s⟩ ∈ f -(int(U)) using cartsub by auto
  then have f ⟨t, s⟩ ∈ int(U) using func1_1_L15 assoc by auto
} hence {f ⟨t, s⟩. ⟨t, s⟩ ∈ (W ∩ V) × (W ∩ V)} ⊆ int(U) by auto
then have (W ∩ V) × (W ∩ V) ⊆ int(U)
  unfolding setadd_def using lift_subsets_explained(4) assoc sub by auto
then have (W ∩ V) + (W ∩ V) ⊆ U using Top_2_L1 by auto
from topSpaceAssum have W ∩ V ∈ T using vop wop unfolding IsATopology_def by auto
then have int(W ∩ V) = W ∩ V using Top_2_L3 by auto
with sub <0 ∈ W ∩ V> have W ∩ V ⊆ N₀ unfolding zerohoods_def by auto
then obtain Q where Q ∈ N₀ and Q ⊆ W ∩ V and {Q = Q} using exists_sym_zerohood by blast
then have Q × Q ⊆ (W ∩ V) × (W ∩ V) by auto
moreover from <Q ⊆ W ∩ V> have W ∩ V ⊆ G and Q ⊆ G using vop wop unfolding
G_def by auto
ultimately have Q + Q ⊆ (W ∩ V) + (W ∩ V) using interval_add(2) func1_1_L8 by auto
with ⟨(W ∩ V) + (W ∩ V) ⊆ U> have Q + Q ⊆ U by auto
from <Q ∈ N₀> have θ ∈ Q unfolding zerohoods_def using Top_2_L1 by auto
with <Q + θ ⊆ U> <Q ⊆ G> have 0 ∈ Q using interval_add(3) by auto
with <Q ⊆ G> have Q ⊆ U unfolding ltrans_def gzero_def using trans_neutral(2)
image_id_same by auto
with <Q ∈ N₀> <Q + Q ⊆ U> ⟨Q⟩ = Q> show thesis by auto
qed

1141
77.4 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.

Lemma (in topgroup) cl_contains_zneigh:
assumes A1: A ⊆ G and A2: H ∈ N₀
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 <x ∈ cl(A)> have x ∈ G by auto
have int(H) ⊆ G using Top_2.L2 by auto
let V = x + (-int(H))
have V = x + (-int(H))
proof -
from A2 <x ∈ G> 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 ∩ V ≠ 0
proof -
from A2 <x ∈ G> <x ∈ cl(A)> have ∀U ∈ T. x ∈ U induces U ∩ A ≠ 0 using cl_inter_neigh by simp
then obtain y where y ∈ A and y ∈ V by auto
with <V = x + (-int(H))> <int(H) ⊆ G> <x ∈ G> have x ∈ y + int(H)
using ltrans_inv_in by simp
with <y ∈ A> have ∀U ∈ T. x ∈ U induces y ∈ 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 ⊆ G shows cl(A) = (∪ H ∈ N₀. A + H)
proof
from assms show cl(A) ⊆ (∪ H ∈ N₀. A + H)
using zneigh_not_empty cl_contains_zneigh by auto
next
{ fix x
assume x ∈ (∪ H ∈ N₀. A + H)
then have x ∈ A + G using zneigh_not_empty by auto
with assms have x ∈ G using interval_add by blast
have ∀ U ∈ T. x ∈ U → U ∩ A ≠ 0
proof -
{ fix U
assume U ∈ T and x ∈ U
let H = -((<x>) + U)
from $\langle U \in T \rangle$ and $\langle x \in U \rangle$ have $(-x)+U \subseteq G$ and $H \in N_0$ using open_trans_neigh neg_neigh_neigh by auto
with $\langle x \in (\bigcap H \in N_0. A+H) \rangle$ have $x \in A+H$ by auto
with assms $H \in N_0$ obtain $y$ where $y \in A$ and $x \in y+H$
using interval_add(3) by auto
have $y \in U$
proof -
  from assms $\langle y \in A \rangle$ have $y \in G$ by auto
  with $\langle (-x)+U \subseteq G \rangle$ and $\langle x \in y+H \rangle$ have $y \in x+(-x)+U$
  using ltrans_inv_in by simp
  with $\langle U \in T \rangle$ $\langle x \in G \rangle$ show $y \in U$
  using neg_in_tgroup trans_comp_image group0_2_L6 trans_neutral
  image_id_same
  by auto
  qed
  with $\langle y \in A \rangle$ have $U \cap A \neq 0$ by auto
} thus thesis by simp
qed

Thus (\bigcap H \in N_0. A+H) \subseteq cl(A) by auto
qed

77.5 Sums of sequences of elements and subsets

In this section we consider properties of the function $G^n \to G, x = (x_0, x_1, ..., x_{n-1}) \mapsto \sum_{i=0}^{n-1} x_i$. We will model the cartesian product $G^n$ by the space of sequences $n \to G$, where $n = \{0, 1, ..., n-1\}$ 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 group.

lemma (in topgroup) sum_list_in_group:
  assumes $n \in \text{nat}$ and $x: \text{succ}(n) \to G$
  shows $(\sum x) \in G$
proof -
  from assms have semigr0(G,f) and $n \in \text{nat} x: \text{succ}(n) \to G$
  using semigr0_valid_in_tgroup by auto
  then have Fold1(f,x) \in G by (rule semigr0.prod_type)
  thus $(\sum x) \in G$ 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 a 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 \( n \in \text{nat} \) and \( x : \text{succ}(\text{succ}(n)) \rightarrow G \)

shows \( (\sum x) = (\sum \text{Init}(x)) + (x(\text{succ}(n))) \)

proof -

from assms have semigr0(G,f) and \( n \in \text{nat} \)

using semigr0_valid_in_tgroup by auto

then have \( \text{Fold1}(f,x) = f(\text{Fold1}(f,\text{Init}(x)),x(\text{succ}(n))) \)

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

shows \( \text{IsContinuous}(\text{SeqProductTopology}(\text{succ}(n),T),T,\{(x,\sum x).x\in\text{succ}(n)\rightarrow G\}) \)

proof -

note \( <n \in \text{nat}> \)

moreover have \( \text{IsContinuous}(\text{SeqProductTopology}(\text{succ}(0),T),T,\{(x,\sum x).x\in\text{succ}(0)\rightarrow G\}) \)

proof -

have \( \{(x,\sum x).x\in\text{succ}(0)\rightarrow G\} = \{(x,x(0)). x\in1\rightarrow G\} \)

using semigr0_valid_in_tgroup semigr0.prod_of_1elem by simp

moreover have \( \text{IsAhomeomorphism}(\text{SeqProductTopology}(1,T),T,\{(x,x(0)). x\in1\rightarrow \bigcup T\}) \)

using topSpaceAssum singleton_prod_top1 by simp

ultimately show thesis using IsAhomeomorphism_def by simp

qed

moreover have \( \forall k \in \text{nat} \).

\( \text{IsContinuous}(\text{SeqProductTopology}(\text{succ}(k),T),T,\{(x,\sum x).x\in\text{succ}(k)\rightarrow G\}) \)

\( \rightarrow \)

\( \text{IsContinuous}(\text{SeqProductTopology}(\text{succ}(\text{succ}(k)),T),T,\{(x,\sum x).x\in\text{succ}(\text{succ}(k))\rightarrow G\}) \)

proof -

{ fix \( k \) assume \( k \in \text{nat} \)

let \( s = \{(x,\sum x).x\in\text{succ}(k)\rightarrow G\} \)

let \( g = \{(p,(\text{fst}(p)),\text{snd}(p))). p \in (\text{succ}(k)\rightarrow G)\times G\} \)

let \( h = \{(x,\langle \text{Init}(x),x(\text{succ}(k))\rangle). x \in \text{succ}(\text{succ}(k))\rightarrow G\} \)

let \( \varphi = \text{SeqProductTopology}(\text{succ}(k),T) \)

let \( \psi = \text{SeqProductTopology}(\text{succ}(\text{succ}(k)),T) \)

assume \( \text{IsContinuous}(\varphi,T,s) \)

from \( <k \in \text{nat}> \) have \( s : (\text{succ}(k)\rightarrow G) \rightarrow G \)

using sum_list_in_group ZF_fun_from_total by simp

have \( h : (\text{succ}(\text{succ}(k))\rightarrow G)\rightarrow (\text{succ}(k)\rightarrow G)\times G \)

proof -

{ fix \( x \) assume \( x \in \text{succ}(\text{succ}(k))\rightarrow G \)

with \( <k \in \text{nat}> \) have \( \text{Init}(x) \in (\text{succ}(k)\rightarrow G) \)

using init_simps by simp

with \( <k \in \text{nat}> \) \( <x : \text{succ}(\text{succ}(k))\rightarrow G> \)

have \( (\text{Init}(x),x(\text{succ}(k))) \in (\text{succ}(k)\rightarrow G)\times G \)

using apply_funtype by blast

} then show thesis using ZF_fun_from_total by simp

1144
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 O g O 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 seq_prod_top_is_top by auto
moreover have g = {⟨p,⟨s(fst(p)),snd(p)⟩⟩. p ∈ ∪ϕ×∪T}
by simp
moreover note <f O g O h :(succ(succ(k))→G)→G
ultimately have IsContinuous(ProductTopology(ϕ,T),ProductTopology(T,T),g)
using cart_prod_cont1 by blast
thus thesis by simp
qed
moreover have IsContinuous(τ,T,f) using fcon by simp
moreover have {⟨x,∑x⟩.x∈succ(succ(k))→G} = f 0 g 0 h
proof -
let d = {⟨x,∑x⟩.x∈succ(succ(k))→G}
from <k ∈ nat> have ∀x∈succ(succ(k))→G. (∑x) ∈ G
  using sum_list_in_group by blast
then have d:(succ(succ(k))→G)→G
  using sum_list_in_group ZF_fun_from_total by simp
moreover have ∀x∈succ(succ(k))→G. d(x) = (f 0 g 0 h)(x)
proof
fix x assume x∈succ(succ(k))→G
then have I: h(x) = ⟨Init(x),x(succ(k))⟩
  using ZF_fun_from_tot_val1 by simp
moreover from <k ∈ nat> <x∈succ(succ(k))→G>
  have Init(x): succ(k)→G
  using init_props by simp
moreover from <k ∈ nat> <x: succ(succ(k))→G>

1145
have II: x(succ(k)) ∈ G
using apply_funtype by blast
ultimately have h(x) ∈ (succ(k)→G)×G by simp
then have g(h(x)) = ⟨s(fst(h(x))),snd(h(x))⟩
using ZF_fun_from_tot_val1 by simp
with I have g(h(x)) = ⟨s(Init(x)),x(succ(k))⟩
by simp
with <Init(x): succ(k)→G> have g(h(x)) = ⟨sum Init(x),x(succ(k))⟩
using ZF_fun_from_tot_val1 by simp
with <k ∈ nat> <x: succ(succ(k))→G>
have f(g(h(x))) = ⟨sum x⟩
using shorter_set_add by simp
with <x ∈ succ(succ(k))→G> have f(g(h(x))) = d(x)
using ZF_fun_from_tot_val1 by simp
moreover from
<h: (succ(succ(k))→G)→(succ(k)→G)×G>
<g:((succ(k)→G)×G)→(G×G)>
<f:G×G→G> <x∈succ(succ(k))→G>
have (f o g o h)(x) = f(g(h(x))) by (rule func1_1_L18)
ultimately show d(x) = (f o g o h)(x) by simp
qed
ultimately show {⟨x, sum x⟩.x∈succ(succ(k))→G} = f o g o h
using func_eq by simp
qed
moreover note <IsContinuous(τ,T,f)>
ultimately have IsContinuous(ψ,T,{⟨x, sum x⟩.x∈succ(succ(k))→G})
using comp_cont3 by simp
} thus thesis by simp
qed
ultimately show thesis by (rule ind_on_nat)
qed
end

78 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.

78.1 Separation properties of topological groups

The topological groups have very specific properties. For instance, G is T₀
iff it is T₃.

theorem (in topgroup) cl_point:
  assumes x∈G
  shows cl{⟨x⟩} = ⟨⋂H∈N₀. x+H⟩
  proof-

have \( c:\text{cl}\{x\} = (\bigcap_{H \in \mathcal{N}_0} \{x\} + H) \) using \text{cl_topgroup} \text{assms} by auto

\[
\text{fix } H \\
\text{assume } H \in \mathcal{N}_0 \\
\text{then have } \{x\} + H = x + H \text{ using interval_add(3) \text{assms}} \\
\text{by auto} \\
\text{with } <H \in \mathcal{N}_0> \text{ have } \{x\} + H \in \{x + H. H \in \mathcal{N}_0\} \text{ by auto}
\]

then have \( \{x\} + H. H \in \mathcal{N}_0 \subseteq \{x + H. H \in \mathcal{N}_0\} \) by auto moreover

\[
\text{fix } H \\
\text{assume } H \in \mathcal{N}_0 \\
\text{then have } \{x\} + H = x + H \text{ using interval_add(3) \text{assms}} \\
\text{by auto} \\
\text{with } <H \in \mathcal{N}_0> \text{ have } x + H \in \{x + H. H \in \mathcal{N}_0\} \text{ by auto}
\]

then have \( \{x + H. H \in \mathcal{N}_0\} \subseteq \{x + H. H \in \mathcal{N}_0\} \) by auto ultimately have \( \{x\} + H \subseteq \{x + H. H \in \mathcal{N}_0\} \) 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 \( \{0\} \text{is closed in} \) \( T \) shows \( T \{\text{is } T_1\} \)

proof-

\[
\text{fix } x \ z \text{ assume } x \in G \text{ and } z \in G \text{ and } \text{dis:} x \neq z \\
\text{then have } \text{clx:} \text{cl}\{x\} = (\bigcap_{H \in \mathcal{N}_0} x + H) \text{ using \text{cl_point} by auto}
\]

\[
\text{fix } y \\
\text{assume } y \in \text{cl}\{x\} \\
\text{with } \text{clx have } y \in (\bigcap_{H \in \mathcal{N}_0} x + H) \text{ by auto} \\
\text{then have } t: \forall H \in \mathcal{N}_0. \ y \in x + H \text{ by auto} \\
\text{from } <y \in \text{cl}\{x\}> \ x \in G \text{ have } y \in G \text{ using } \text{Top_3_L11(1) } G \text{def by auto}
\]

\[
\text{fix } H \\
\text{assume } H \text{Neig:} H \in \mathcal{N}_0 \\
\text{with } t \text{ have } y \in x + H \text{ by auto} \\
\text{then obtain } n \text{ where } y = x + n \text{ and } n \in H \text{ unfolding ltrans_def grop_def LeftTranslation_def by auto}
\]

with \( H \text{Neig have } n \in G \text{ unfolding zerohoods_def by auto} \)

\[
\text{from } <y = x + n> \text{ and } <n \in H> \text{ have } (-x) + y \in H \text{ using } \text{group0.group0_2_L18(2) group0_valid_in_tgroup x G n G unfolding grinv_def grop_def by auto}
\]

1147
have \((-x)+y \in (\bigcap N_0)\) using zneigh_not_empty by auto

\begin{align*}
\text{have } \cl\{0\} &= (\bigcap H \in N_0. \ 0+H) \text{ using cl_point zero_in_tgroup by auto} \\
\end{align*}

moreover 
\begin{align*}
\text{fix } H &\text{ assume } H \in N_0 \\
\text{then have } H \subseteq G \text{ unfolding zerohoods_def by auto} \\
\text{then have } 0+H = H \text{ using image_id_same group0.trans_neutral(2) group0_valid_in_tgroup unfolding gzero_def ltrans_def} \\
\text{by auto} \\
\text{with } \langle H \in N_0 \rangle &\text{ have } 0+H \in N_0 \\
\text{ultimately have } \cl\{0\} &= (\bigcap N_0) \text{ by auto} 
\end{align*}

\begin{align*}
\text{with } \langle H \in N_0 \rangle &\text{ have } 0+H \in N_0 \\
\end{align*}

moreover 
\begin{align*}
\text{fix } H &\text{ assume } H \in N_0 \\
\text{then have } H \subseteq G \text{ unfolding zerohoods_def by auto} \\
\text{then have } 0+H = H \text{ using image_id_same group0.trans_neutral(2) group0_valid_in_tgroup unfolding gzero_def ltrans_def} \\
\text{by auto} \\
\text{with } \langle H \in N_0 \rangle &\text{ have } 0+H \in N_0 \\
\text{ultimately have } \cl\{0\} &= (\bigcap N_0) \text{ by auto} 
\end{align*}

\begin{align*}
\text{then have } (-x)+y \in \cl\{0\} &\text{ using assms Top_3.L8 G_def zero_in_tgroup by auto} \\
\text{then have } (-x)+y = 0 &\text{ by auto} \\
\text{then have } y = -(-x) &\text{ using group0.group0_2_L9(2) group0_valid_in_tgroup unfolding grinv_def by auto} \\
\text{ultimately have } \exists V \in T. z \in V \wedge x \notin V &\text{ by (safe,auto)} \\
\text{then show } T \{\text{is T1}\} &\text{ using ist1_def by auto} \\
\text{qed} \\
\end{align*}

\begin{align*}
\text{theorem } \text{(in topgroup) T0_imp_neu_closed:} \\
\text{assumes } T \{\text{is T0}\} \\
\text{shows } \{0\} \{\text{is closed in}\} T \\
\text{proof-} \\
\text{fix } x &\text{ assume } x \in \cl\{0\} \text{ and } x \neq 0 \\
\text{have } \cl\{0\} &= (\bigcap H \in N_0. \ 0+H) \text{ using cl_point zero_in_tgroup by auto} \\
\text{moreover} \\
\text{fix } H &\text{ assume } H \in N_0 \\
\text{then have } H \subseteq G \text{ unfolding zerohoods_def by auto} \\
\text{then have } 0+H = H \text{ using image_id_same group0.trans_neutral(2) group0_valid_in_tgroup unfolding gzero_def ltrans_def} \\
\text{by auto} \\
\text{with } \langle H \in N_0 \rangle &\text{ have } 0+H \in N_0 \\
\text{ultimately have } \exists V \in T. z \in V \wedge x \notin V &\text{ by (safe,auto)} \\
\text{then show } T \{\text{is T1}\} &\text{ using ist1_def by auto} \\
\text{qed} \\
\end{align*}
then have \( \{0 + H. H \in \mathcal{N}_0 \} = \mathcal{N}_0 \) by blast
ultimately have \( \operatorname{cl} \{0\} = (\bigcap \mathcal{N}_0) \) by auto
from \(<x \neq 0\) and \(<x \in \operatorname{cl} \{0\}\) obtain \( U \) where \( U \in T \) and \((x \notin U \land 0 \in U) \lor (0 \notin U \land x \in U)\)
using assms Top_3_L11(1)

zero_in_tgroup unfolding isT0_def G_def by blast
moreover
{ assumes 0 \( \in U \) with \(<U \in T> \) have \( U \in \mathcal{N}_0 \) by auto
with \(<x \in \operatorname{cl} \{0\}>\) and \(<\operatorname{cl} \{0\} = (\bigcap \mathcal{N}_0)\) have \( x \in U \) by auto
}
ultimately have \( 0 \notin U \) and \( x \in U \) by auto
where \( U \in T \) and \((x / \in U \land 0 \in U) \lor (0 \in U \land x \in U)\)
using assms Top_3_L11(1)

zero_in_tgroup unfolding isT0_def G_def by blast

ultimately have \( \operatorname{cl} \{0\} \subseteq \{0\} \) by auto
ultimately have \( \operatorname{cl} \{0\} = \{0\} \) by auto
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

78.2 Existence of nice neighbourhoods.

lemma (in topgroup) exist_basehoods_closed:
assumes \( U \in \mathcal{N}_0 \)
sows \( \exists V \in \mathcal{N}_0. \ \operatorname{cl}(V) \subseteq U \)
proof-
from assms obtain \( V \) where \( V \in \mathcal{N}_0 \) \( V \subseteq U \) \((V + V) \subseteq U \) \((-V) = V \)
using exists_procls_zerohood by blast
have inv_fun: \( \text{GroupInv}(G, f) \in G \to G \) using group0_2_T2 Ggroup by auto
have f_fun: \( f : G \times G \to G \) using group0.group_oper_fun group0_valid_in_tgroup
by auto
fix \( x \) assume \( x \in \operatorname{cl}(V) \)
with \( V \in \mathcal{N}_0 \) have \( x \in \bigcup T \) \( V \subseteq \bigcup T \) using Top_3_L11(1)
unfolding zerohoods_def G_def by blast+
with \( V \in \mathcal{N}_0 \) have \( x \in \operatorname{int}(x + V) \) using elem_in_int_ltrans G_def by auto
with \( V \subseteq \bigcup T \) \( x \in \operatorname{cl}(V) \) have \( \operatorname{int}(x + V) \cap V \neq 0 \) using cl_inter_neigh Top_2_L2
by blast
then have \( (x + V) \cap V \neq 0 \) using Top_2_L1 by blast
then obtain \( q \) where \( q \in (x + V) \) and \( q \in V \) by blast
with \( V \subseteq \bigcup T \) \( x \in \bigcup T \) obtain \( v \) where \( q = x + v \) \( v \in V \) unfolding ltrans_def
by blast
then have \( V \subseteq \bigcup T \) \( x \in \bigcup T \) using Top_2_L1 by blast
then obtain \( q \) where \( q \in (x + V) \) and \( q \in V \) by blast
with \( V \subseteq \bigcup T \) \( x \in \bigcup T \) have \( q = v = x \) using group0.group0_2_L18(1) group0_valid_in_tgroup
by blast
unfolding G_def
by blast

unfolding gsub_def grinv_def group_def by auto
moreover
from \( <V \subseteq \bigcup T> \) \( <v \in V> \) \( <q \in V> \) have \( v \in \bigcup T \) \( q \in \bigcup T \) by auto
with \( q = x + v \) \( x \in \bigcup T \) have \( q = v = x \) using group0.group0_2_L18(1) group0_valid_in_tgroup
by blast

unfolding G_def
by blast
moreover
from \( <v \in V> \) have \( (-v) \in (-V) \) unfolding setninv_def grinv_def using func_imagedef
by auto

1149
then have \((-v)\in V\) using \((-V)\subseteq V\) by auto
with \(<q\in V>\) have \((q,-v)\in V\times V\) by auto
then have \(f(q,-v)\in V\times V\) using lift_subset_suff f_fun unfolding setadd_def by auto
with \(<V\subseteq U>\) have \(-v\in U\) unfolding grsub_def grop_def by auto
with \(<q=x>\) have \(x\in U\) by auto
\}
then have \(cl(V)\subseteq U\) by auto
with \(<V\in N_0>\) show thesis by auto
qd

78.3 Rest of separation axioms

Theorem (in topgroup) T1_imp_T2:
assumes \(T\{is T_1\}\)
shows \(T\{is T_2\}\)
proof-
\{
  fix \(x \ y\) assume ass:\(x\in\bigcup T \ y\in\bigcup T \ x\neq y\)
  \{
    assume \((-y)+x=0\)
    with ass(1,2) have \(y=x\) using group0.group0_2_L11[where \(a=y\) and \(b=x\)] group0_valid_in_tgroup by auto
    with ass(3) have \(False\) by auto
  \}
  then have \((-y)+x\neq 0\) by auto
  then have \(0\neq (-y)+x\) by auto
  from \(<y \in \bigcup T>\) have \((-y)\in \bigcup T\) using neg_in_tgroup G_def by auto
  with \(<x \in \bigcup T>\) have \((-y)+x\in \bigcup T\) using group0.group_op_closed[where \(a=-y\) and \(b=x\)] group0_valid_in_tgroup unfolding G_def by auto
  with assms \(<0\neq (-y)+x>\) obtain \(U\) where \(U\subseteq T\) and \((-y)+x\notin U\) and \(0\in U\) unfolding isT1_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 \(Q\) where \(Q\subseteq U\) \((Q+Q)\subseteq U\) \((-Q)=Q\) using exists_procls_zerohood by blast
  with \(<-y)+x\notin U>\) have \((-y)+x\notin Q\) by auto
  from \(<Q\subseteq G>\) have \(Q\subseteq G\) unfolding zerohoods_def by auto
  \{
    assume \(x\in y+Q\)
    with \(<Q\subseteq G>\ <y\in \bigcup T>\) obtain \(u\) where \(u\subseteq Q\) and \(x=y+u\) unfolding ltrans_def
going G_def by auto
    with \(<Q\subseteq G>\ <y\in \bigcup T>\ <x\in \bigcup T>\ <Q\subseteq G>\) have \((-y)+x\in u\) using group0.group0_2_L18(2) group0_valid_in_tgroup unfolding G_def
    unfolding grsub_def grinv_def group_def by auto
    with \(<u\subseteq Q>\) have \((-y)+x\in Q\) by auto
  \}
\}
then have False using \((-y)+x\notin Q\) by auto 
} 
then have \(x\notin y+Q\) by auto moreover 
{
assume \(y\in x+Q\)
with \(<Q\subseteq G\times x\subseteq T\) 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 \(<Q\subseteq G\times u\subseteq T\) unfolding G_def by auto 
with \(<y=x+u\times y\subseteq T\times Q\subseteq G\) have \((-x)+y=u\) using group0.group0_2_L18(2) 
group0_valid_in_tgroup unfolding G_def 
unfolding grsub_def grinv_def grop_def by auto 
moreover from \(<u\in Q\) have \((-u)\in (-Q)\) unfolding setninv_def grinv_def 
using func_imagedef[OF group0_2_T2[OF Ggroup]] unfolding G_def 
by auto 
ultimately have \((-y)+x\in Q\) using 
group0.group0_2_L18(2)[OF group0_valid_in_tgroup,of y]
unfolding G_def 
by auto 
moreover note \(<x\in T\times y\in T\) unfolding G_def 
ultimately have \((-y)+(x+u)=v\) using group0.group_inv_of_inv[OF group0_valid_in_tgroup] 
by auto 
moreover note \(<x\in T\times y\in T\times u\in T\) unfolding G_def 
by auto 
moreover note \(<x\in T\times y\in T\times u\in T\times v\in T\) unfolding G_def 
by auto 
moreover note \(<x\in T\times y\in T\times u\in T\times v\in T\times w\in T\) unfolding G_def 
by auto 
moreover note \(<x\in T\times y\in T\times u\in T\times v\in T\times w\in T\times x\in T\) unfolding G_def 
by auto 
then have \((-y)+x\in Q\) by auto
then have \((-u) \in Q\) using \(\langle -Q \rangle = Q\) by \textit{auto}
with \(\langle v, -u \rangle \in Q \times Q\) by \textit{auto}
then have \(f(v, -u) \in Q + Q\) using \textit{lift_subset_suff[OF group0.group_oper_func[OF 
\textit{group0_valid_in_tgroup} \(Q \subseteq G\) \(\langle Q \subseteq G \rangle\)]}
unfolding \textit{setadd_def} by \textit{auto}
with \(\langle v, -u \rangle \in Q \times Q\) have \(\langle v, -u \rangle \in Q \times Q\) by \textit{auto}
then have \(f\langle v, -u \rangle \in Q + Q\)
using \textit{lift_subset_suff[OF group0.group_oper_func[OF 
\textit{group0_valid_in_tgroup} \(\langle Q \subseteq G \rangle\)]]
unfolding \textit{setadd_def} by \textit{auto}
with \(\langle Q + Q \subseteq U \rangle\) have \(v - u \in U\) unfolding \textit{grsub_def} \textit{grop_def} by \textit{auto}
ultimately have \(\langle (-y) + x \in U \rangle\) by \textit{auto}
with \(\langle (-y) + x \notin U \rangle\) have \(\text{False}\) by \textit{auto}

\}\then show \(\text{thesis}\) using \textit{isT2_def} by \textit{auto}
\end{quote}

Here follow some auxiliary lemmas.

\textbf{lemma (in topgroup) trans_closure:}
\begin{itemize}
  \item \textit{assumes} \(x \in G\) \(A \subseteq G\)
  \item \textit{shows} \(\text{cl}(x + A) = x + \text{cl}(A)\)
\end{itemize}
\textbf{proof-}
\begin{itemize}
  \item have \(\bigcup T - \bigcup T - (x + A) = (x + A)\) unfolding \textit{ltrans_def} using \textit{group0.group0_5_L1(2)[OF 
  \textit{group0_valid_in_tgroup assms(1)}]}\)
    unfolding \textit{image_def} \textit{range_def} \textit{domain_def} \textit{converse_def} \textit{Pi_def} by \textit{auto}
    then have \(\text{cl}(x + A) = \bigcup T - \text{int}(\bigcup T - (x + A))\) using \textit{Top_2_L11(2)[of \bigcup T - (x + A)]}
    by \textit{auto}
    moreover have \(x + G = G\) using \textit{surj_image_eq[OF \textit{group0.group_oper_func[OF 
  \textit{group0_valid_in_tgroup assms(1)}]}]}\)
    unfolding \textit{ltrans_def} \textit{G_def} by \textit{auto}
    then have \(\bigcup T - (x + A) = x + \bigcup T - (x + A)\) unfolding \textit{G_def} \textit{substr_eq[OF assms(2)]}
    by \textit{auto}
    unfolding \textit{ltrans_def} \textit{G_def} using \textit{group0.group_oper_func[OF 
  \textit{group0_valid_in_tgroup assms(1)}]}\}
    \textit{bij_def} by \textit{auto}
    then have \(\text{int}(\bigcup T - (x + A)) = \text{int}(\bigcup T - (x + A))\) by \textit{auto}
    then have \(\text{int}(\bigcup T - (x + A)) = x + \text{int}(\bigcup T - (x + A))\) using \textit{Top_2_L2[of \bigcup T - (x + A)]}
    unfolding \textit{G_def} by \textit{force}
    have \(\bigcup T - \text{int}(\bigcup T - (x + A)) = \text{cl}(\bigcup T - (x + A))\)
    by \textit{auto}
    with \(\bigcup T - \text{int}(\bigcup T - (x + A)) = \text{cl}(\bigcup T - (x + A))\) have \(\bigcup T - \text{int}(\bigcup T - (x + A)) = \text{cl}(\bigcup T - (x + A))\)
    by \textit{auto}
\end{itemize}
with \( x \in G \) have \( \text{int}(\bigcup T-(x+A)) = \bigcup T-(x+cl(A)) \) using inj_image_dif[of LeftTranslation(G, f, x)GGcl(A)]

unfolding ltrans_def using group0.group0.trans_bij(2)[OF group0_valid_in_tgroup assms(1)] Top_3_L11(1) assms(2) unfolding bij_def G_def by auto
then have \( \bigcup T-int(\bigcup T-(x+A)) = \bigcup T-(\bigcup T-(x+cl(A))) \) by auto
then have \( \bigcup T-int(\bigcup T-(x+A)) = x+cl(A) \) unfolding ltrans_def using group0.group0_5_L1(2)[OF group0_valid_in_tgroup assms(1)]

unfolding image_def range_def domain_def converse_def Pi_def by auto
with \( cl(x+A) = \bigcup T-int(\bigcup T-(x+A)) \) show thesis by auto
qed

lemma (in topgroup) trans_interior2: assumes \( g \in G \) and \( A \subseteq G \) shows \( \text{int}(A)+g = \text{int}(A+g) \)
proof -
from assms have \( A \subseteq \bigcup T \) 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 \( x \in G \) A \( \subseteq G \) shows \( \text{cl}(A+x) = \text{cl}(A)+x \)
proof -
have \( \bigcup T-(\bigcup T-(A+x)) = (A+x) \) 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 \( \text{cl}(A+x) = \bigcup T-int(\bigcup T-(A+x)) \) using Top_3_L11(2)[of \( \bigcup T-(A+x) \)]
by auto moreover
have \( G+x=G \) using surj_image_eq group0.group0.trans_bij(1)[OF group0_valid_in_tgroup 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 \( \text{int}(\bigcup T-(A+x)) = \text{int}(\bigcup T-(A)+x) \) using trans_interior2[of assms(1),of \( \bigcup T-(A) \)] unfolding G_def by force
have \( \bigcup T-int(\bigcup T-A) = cl(\bigcup T-(\bigcup T-A)) \) using Top_3_L11(2)[of \( \bigcup T-A \)] by force
have \( \bigcup T-(\bigcup T-A) = A \) using assms(2) G_def by auto
with \( \bigcup T-int(\bigcup T-A) = cl(\bigcup T-(\bigcup T-A)) \) have \( \bigcup T-int(\bigcup T-A) = cl(A) \) by auto
have \( \bigcup T-(\bigcup T-(\bigcup T-A)) = \text{int}(\bigcup T-A) \) using Top_2_L2 by auto
with \( \bigcup T-(\bigcup T-A) = cl(A) \) have \( \text{int}(\bigcup T-A) = \text{cl}(A) \) by auto
with \( \text{int}(\bigcup T-A+x) = \text{int}(\bigcup T-A)+x \) have \( \text{int}(\bigcup T-A+x) = (\bigcup T-cl(A))+x \) by auto
with \( G+x=G \) have \( \text{int}(\bigcup T-(A+x)) = \bigcup T-(cl(A)+x) \) using inj_image_dif[of RightTranslation(G, f, x)GGcl(A)] unfolding rtrans_def using group0.group0.trans_bij(1)[OF group0_valid_in_tgroup

1153
assms(1) Top_3_L11(1) assms(2) unfolding bij_def G_def
by auto
then have \( \bigcup \) Int(\( \bigcup \) T-A+x)=\( \bigcup \) T-cl(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 \( \bigcup \) T-int(\( \bigcup \) T-A+x)=cl(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
with <cl(A+x)=\( \bigcup \) T-int(\( \bigcup \) T-A+x)> show thesis by auto
qed

lemma (in topgroup) trans_subset:
assumes A \subseteq (-x)+B x \in GA \subseteq GB \subseteq G
shows x+A \subseteq B
proof-
{ fix t assume t \in x+A
  with <x \in G> \<A \subseteq G> 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 G> \<u \in A> have (-x)+t=u using group0.group0_2_L18(2)[OF group0_valid_in_tgroup,of x u]
group0.group_op_closed[OF group0_valid_in_tgroup,of x u] unfolding grop_def grinv_def by auto
  with <u \in A> <v \in B> have (-x)+t=(-x)+v \forall v \in B unfolding ltrans_def
  have LeftTranslation(G,f,-x) \in inj(G,G) using group0.trans_bij(2)[OF group0_valid_in_tgroup neg_in_tgroup[of x]]
  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 x]]
  <A \subseteq G> <B \subseteq G> by auto
  with eq <A \subseteq G> <B \subseteq G> have A=B by auto
  \}
then have eq1: \forall A \subseteq G. \forall B \subseteq G. f(-x,A)=f(-x,B) \rightarrow A=B by auto
  from <A \subseteq G> <u \in A> have u \in G by auto
  with <v \in B> <B \subseteq G> <t=x+u> have t \in G v \subseteq G using group0.group_op_closed[OF group0_valid_in_tgroup[of u]] unfolding grop_def by auto
  with eq1 \<(-x)+t=(-x)+v> have t=v unfolding grop_def by auto
  with \<v \in B> have t \in B by auto
  \}
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.

### 78.4 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** $\forall U \in T \cap N_0. \exists N \in N_0. N \subseteq U \land P(N, T) \land P(N, T)$
- **shows** $T \{\text{is locally}\} P$

**proof**

1. fix $x \in U$ assume $x \in \bigcup T \subseteq T \in U$
   - then have $(-x) + U \subseteq T \cap N_0$ using open_tr_open(1) open_trans_neigh neg_in_tgroup
   - unfolding $G \text{ def}$ by auto with assms(1)
   - obtain $N$ where $N \subseteq ((-x) + U)$ using $P(N, T)$
   - note $< x \in \bigcup T >$-$< N \subseteq ((-x) + U) >$ moreover
   - from $< U \subseteq T >$ have $U \subseteq T$ by auto moreover
   - from $< N \subseteq N_0 >$ have $N \subseteq G$ unfolding zerohoods_def by auto
   - ultimately have $(x + N) \subseteq U$ using trans_subset unfolding $G \text{ def}$ by auto
   - moreover have $x + A \subseteq G$ unfolding ltrans_def using $group0.group0_5_L1(2)$[OF group0_valid_in_tgroup $x \in G$]
   - unfolding image_def range_def domain_def converse_def $Pi$ def by auto
   - ultimately have $\exists N \in Pow(U). x \in int(N) \land P(N, T)$ by auto

2. then show thesis unfolding $IsLocally_def[\text{OF topSpaceAssum}]$ by auto

**qed**

**lemma** (in topgroup) trans_closed:
- **assumes** $A \{\text{is closed in}\} G \in T x \in G$
- **shows** $(x + A) \{\text{is closed in}\} T$

**proof**

1. from assms(1) have $cl(A) = A$ using $Top\_3\_L8$ unfolding $IsClosed_def$ by auto
   - then have $x + cl(A) = x + A$ by auto
   - then have $cl(x + A) = x + A$ using trans_closure assms unfolding $IsClosed_def$ by auto
   - moreover have $x + A \subseteq G$ unfolding ltrans_def using $group0.group0_5_L1(2)$[OF group0_valid_in_tgroup $x \in G$]
   - unfolding image_def range_def domain_def converse_def $Pi$ def by auto
   - ultimately show thesis using $Top\_3\_L8$ unfolding $G \text{ def}$ by auto

1155
As it is written in the previous section, every topological group is regular.

**Theorem (in topgroup) topgroup_reg:**

- shows \( T \{ \text{is regular} \} \)

**Proof:**

```plaintext
{-
  fix \( U \) assume \( U \in T \cap N_0 \)
  then obtain \( V \) where \( cl(V) \subseteq UV \in N_0 \) using exist_basehoods_closed by blast
  then have \( V \subseteq cl(V) \) using clcontains_set unfolding zerohoods_def G_def by auto
  then have \( int(V) \subseteq cl(V) \) using interior_mono by auto
  with \( \langle V \in N_0 \rangle \) have \( cl(V) \in N_0 \) unfolding zerohoods_def G_def using Top_3_L11(1)

- by auto
  from \( \langle V \in N_0 \rangle \) have \( cl(V) \{ \text{is closed in} \} T \) using cl_is_closed unfolding zerohoods_def G_def by auto
  with \( \langle cl(V) \in N_0 \rangle \langle cl(V) \subseteq U \rangle \) have \( \exists N \in N_0 . N \subseteq U \) unfolding zerohoods_def G_def by auto
  then have \( \forall U \in T \cap N_0 . \exists N \in N_0 . N \subseteq U \) unfolding zerohoods_def G_def by auto
  moreover have \( \forall N \in Pow(G) . ( \forall x \in G . (N \{ \text{is closed in} \} T \longrightarrow (x+N) \{ \text{is closed in} \} T)) \) unfolding trans_closed by auto
  ultimately have \( T \{ \text{is locally-closed} \} \) using local_iff_neutral unfolding IsLocallyClosed_def by auto
  then show \( T \{ \text{is regular} \} \) using regular_locally_closed by auto
- qed
```

The promised corollary follows:

**Corollary (in topgroup) T2_imp_T3:**

- assumes \( T \{ \text{is } T_2 \} \)
- shows \( T \{ \text{is } T_3 \} \) using T2_is_T1 topgroup_reg isT3_def assms by auto

**End**

---

### 79 Topological groups - uniformity

**Theory** TopologicalGroup_Uniformity_ZF imports TopologicalGroup_ZF UniformSpace_ZF_1

**Begin**

Each topological group is a uniform space. This theory is about the uniformities that are naturally defined by a topological group structure.

#### 79.1 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

definition (in topgroup) leftUniformity
where leftUniformity ≡ \{V ∈ \text{Pow}(G × G). \exists U ∈ \mathbb{N}_0. \{⟨s, t⟩ ∈ G × G. (-s)+t ∈ U\} ⊆ V\}

Definition of right uniformity

definition (in topgroup) rightUniformity
where rightUniformity ≡ \{V ∈ \text{Pow}(G × G). \exists U ∈ \mathbb{N}_0. \{⟨s, t⟩ ∈ G × G. s+(-t) ∈ U\} ⊆ V\}

Right and left uniformities are indeed uniformities.

lemma (in topgroup) side_uniformities:
shows leftUniformity {is a uniformity on} G and rightUniformity {is a uniformity on} G
proof-
{}
assumption O ∈ leftUniformity
then obtain U where U:U ∈ \mathbb{N}_0 \{⟨s, t⟩ ∈ G × G. (-s)+t ∈ U\} ⊆ O unfolding leftUniformity_def by auto

have \{(0,0):G × 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 ∈ \text{int}(U)\} using U(1) by auto
then have \{0 ∈ U\} using Top_2_L1 by auto
ultimately have \{(0,0) ∈ \{⟨s, t⟩ ∈ G × G. (-s)+t ∈ U\}\} by auto
with U(2) have \{(0,0) ∈ O\} by blast
hence False by auto
hence O \notin leftUniformity by auto
moreover have leftUniformity ⊆ \text{Pow}(G × G) unfolding leftUniformity_def by auto

moreover
{}
have \{G × G ∈ \text{Pow}(G × G)\} by auto moreover
have \{⟨s, t⟩:G × G. (-s)+t:G\} ⊆ G × G by auto moreover
note zneigh_not_empty
ultimately have \{G × G ∈ leftUniformity\} unfolding leftUniformity_def by auto

moreover
{}
fix A B assume as:A ∈ leftUniformity B ∈ leftUniformity
from as(1) obtain AU where AU:AU ∈ \mathbb{N}_0 \{⟨s, t⟩ ∈ G × G. (-s)+t ∈ AU\} ⊆ A
A ∈ \text{Pow}(G × G)

unfolding leftUniformity_def by auto

1157
from as(2) obtain \( BU \) where \( BU : BU \subseteq N \), \( \{ (s, t) \in G \times G. (-s) + t \in BU \} \subseteq B \)

unfolding 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 \( \text{op} : \text{int}(AU) \cap \text{int}(BU) \in T \)
using Top_2_L2 topSpaceAssum
IsATopology_def by auto
moreover have \( \text{int}(AU) \cap \text{int}(BU) \subseteq AU \cap BU \)
using Top_2_L1 by auto
with \( \text{op} \) have \( \text{int}(AU) \cap \text{int}(BU) \subseteq \text{int}(AU \cap BU) \)
using Top_2_L5 by auto
moreover note AU(1) BU(1)
ultimately have \( AU \cap BU : N \)
unfolding zerrohoods_def by auto
moreover have \( \{ (s, t) \in G \times G. (-s) + t \in \text{int}(AU \cap BU) \} \subseteq \{ (s, t) \in G \times G. (-s) + t \in \text{int}(AU) \} \)
by auto
with \( AU(2) \) \( BU(2) \) have \( \{ (s, t) \in G \times G. (-s) + t \in \text{int}(AU \cap BU) \} \subseteq A \cap B \)
by auto
ultimately have \( A \cap B : N \)
unfolding zerohoods_def by auto
moreover have \( \{ (s, t) \in G \times G. (-s) + t \in \text{int}(AU \cap BU) \} \subseteq \{ (s, t) \in G \times G. (-s) + t \in \text{int}(AU) \} \)
by auto
with \( AU(3) \) \( BU(3) \) by blast
then have \( A \cap B \in \text{leftUniformity} \)
unfolding 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 \( BU : BU \subseteq N \), \( \{ (s, t) \in G \times G. (-s) + t \in BU \} \subseteq B \)
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 \( C \in \{ V \in \text{Pow}(G \times G). \exists U \in N, \{ (s, t) \in G \times G. (-s) + t \in V \} \subseteq U \} \subseteq V \)
by auto
then have \( C \in \text{leftUniformity} \)
unfolding leftUniformity_def by auto
} then have \( \forall B \in \text{leftUniformity}. \forall C \in \text{Pow}(G \times G). B \subseteq C \rightarrow C \in \text{leftUniformity} \)
by auto
ultimately have \( \text{leftFilter} : \text{leftUniformity} \) \( \{ \) is a filter on \( \} \) \( G \times G \)
unfolding IsFilter_def
by auto
\( \{ \)
assume \( 0 \in \text{rightUniformity} \)
then obtain \( U \) where \( U : U \subseteq N \), \( \{ (s, t) \in G \times G. s + (-t) \in U \} \subseteq 0 \)
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.group1_of_one group0.group0_2_L2
zero_in_tgroup

1158
by auto
moreover
have \(0\in\text{int}(U)\) 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 \(0\notin \text{rightUniformity}\) by auto
moreover have \(\text{rightUniformity}\subseteq \text{Pow}(G\times G)\) unfolding \rightUniformity_def\ by auto
moreover
\[
\begin{align*}
\text{have } & G\times G\subseteq \text{Pow}(G\times G) \text{ by auto} \\
\text{moreover have } & \{(s,t):G\times G. (-s)+t:G\} \subseteq G\times G \text{ by auto} \\
\text{moreover note } & \text{zneigh_not_empty} \\
\text{ultimately have } & G\times G \in \text{rightUniformity} \text{ unfolding \rightUniformity_def}\ by auto
\end{align*}
\]
moreover
\[
\begin{align*}
\text{fix } & A B \text{ assume as:\(A\in \text{rightUniformity} B\in \text{rightUniformity}\)} \\
\text{from as(1) obtain } & AU \text{ where } AU:AU \in \mathcal{N}_0 \\{(s,t)\in G\times G. s+(-t)\in AU\} \subseteq A \\
A\in\text{Pow}(G\times G) \text{ unfolding \rightUniformity_def}\ by auto
\end{align*}
\]
moreover
\[
\begin{align*}
\text{from } & AU(1) \text{ BU(1) have } 0\in \text{int}(AU)\cap \text{int}(BU) \text{ by auto} \\
\text{moreover from } & AU \text{ BU have } \text{op: int}(AU)\cap \text{int}(BU)\in T \\
\text{using } & \Top_2_L2 \text{ topSpaceAssum IsATopology_def}\ by auto \\
\text{moreover have } & \text{int}(AU)\cap \text{int}(BU) \subseteq AU\cap BU \text{ using } \Top_2_L1\) by auto \\
\text{with op have } & \text{int}(AU)\cap \text{int}(BU)\subseteq \text{int}(AU\cap BU) \text{ using } \Top_2_L5\) by auto \\
\text{moreover note } & AU(1) \text{ BU(1)} \\
\text{ultimately have } & AU\cap BU: \mathcal{N}_0 \text{ unfolding \zerohoods_def}\ by auto \\
\text{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\} \\
\text{by auto} \\
\text{with } & AU(2) \text{ BU(2) have } \{(s,t)\in G\times G. s+(-t)\in AU\cap BU\} \subseteq A\cap B \text{ by auto} \\
\text{ultimately have } & A\cap B \in \{V\in \text{Pow}(G\times G). \exists \mathcal{N}_0. \{\langle s,t\rangle\in G\times G. s+(-t)\in V\}\subseteq V\} \\
\text{using } & AU(3) \text{ BU(3) by blast} \\
\text{then have } & A\cap B \in \text{rightUniformity} \text{ unfolding \rightUniformity_def}\ by simp \\
\text{hence } & \forall A\in \text{rightUniformity. } \forall B\in \text{rightUniformity. } A\cap B \in \text{rightUniformity} \\
\text{by auto} \\
\text{moreover }
\end{align*}
\]
fix B C assume as:B∈rightUniformity C∈Pow(G × G) B ⊆ C
from as(1) obtain BU where BU:BU∈N₀ \{⟨s,t⟩∈G×G. s+(-t) ∈ BU\} ⊆ B

unfolding rightUniformity_def by blast
from as(3) BU(2) have \{(s,t)∈G×G. s+(-t) ∈ BU\} ⊆ C by auto
then have C ∈ rightUniformity using as(2) BU(1) unfolding rightUniformity_def by auto
}
then have ∀B∈rightUniformity. ∀C∈Pow(G×G). B ⊆ C

ultimately have rightFilter:rightUniformity \{is a filter on\} (G×G)
unfolding IsFilter_def by auto

{ fix U assume as:U∈leftUniformity
from as obtain V where V:V∈N₀ \{⟨s,t⟩∈G×G. (-s)+t ∈ V\} ⊆ U
unfolding leftUniformity_def by auto
then have ∀V⊆G by auto

{ fix x assume as2:x∈id(G)
from as obtain V where V:V∈N₀ \{⟨s,t⟩∈G×G. (-s)+t ∈ V\} ⊆ U
unfolding leftUniformity_def by auto
from V(1) have 0∈int(V) by auto
then have V0:0∈V using Top_2_L1 by auto
from as2 obtain t where t:x=(t,t) t:G by auto
from t(2) have (-t)+t =0 using group0_valid_in_tgroup group0.group0_2_L6 by auto
with V0 t V(2) have x∈U by auto
}
then have id(G) ⊆ U by auto
moreover

{ fix x assume as:⟨x∈(s,t)∈G×G. (-s)+t ∈ -V\}
then obtain s t where as:a∈G t∈G (-s)+t ∈ -V x=(s,t) by force
from as(3) -V⊆G have (-s)+t∈{-q. q∈V} using ginv_image_add
by simp
then obtain q where q: q∈V (-s)+t = -q by auto
with -V⊆G have q∈G by auto
with <s∈G. t∈G> (-s)+t = -q> have q=(-t)+s using simple_equation1_add by blast
with q(1) have (-t)+s ∈ V by auto
with as(1,2) have ⟨t,s⟩ ∈ U using V(2) by auto
then have ⟨s,t⟩ ∈ converse(U) by auto
with as(4) have x ∈ converse(U) by auto
}
then have \{(s,t)∈G×G. (-s)+t ∈ -V\} ⊆ converse(U) by auto
moreover have (-V):N₀ using neg_neigh_neigh V(1) by auto
moreover note as as

1160
ultimately have converse(U) ∈ leftUniformity unfolding leftUniformity_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} O (s,t)∈G×G. (-s)+t ∈ W
        
        then obtain 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 WW:W ∈ G×G by auto
        moreover 
        from x(4,5) have ((-x_1)+x_2,(-x_2)+x_3):W×W by auto
        with WW have f((((-x_1)+x_2,(-x_2)+x_3)):f(W×W)
        using func_imagedef topgroup_f_binop by auto
        ultimately have (-x_1)+ x_3∈W by auto
        with x(2) have (-x_1)+ x_3∈V by auto
        with x(1,3,6) have x:{(s,t)∈G×G. (-s)+t ∈ V} by auto
    }
    then have (s,t)∈G×G. (-s)+t ∈ W} O (s,t)∈G×G. (-s)+t ∈ W} ⊆ U
        using V(2) by auto moreover
        have {(s,t)∈G×G. (-s)+t ∈ W}∈leftUniformity unfolding leftUniformity_def using W(1) by auto
        ultimately have ∃Z∈leftUniformity. Z O Z⊆U by auto
    }
ultimately have id(G)⊆U ∧ (∃Z∈leftUniformity. Z O Z⊆U) ∧ converse(U)∈leftUniformity by blast
}
then have 
∀U∈leftUniformity. id(G)⊆U ∧ (∃Z∈leftUniformity. Z O Z⊆U) ∧ converse(U)∈leftUniformity by auto
with leftFilter show leftUniformity {is a uniformity on} G unfolding IsUniformity_def by auto 
{ 
    fix U assume as:U∈rightUniformity
    from as obtain V where V:V∈N_0 {(s,t)∈G×G. s*(-t) ∈ V} ⊆ U
    unfolding rightUniformity_def by auto 
    { 
        fix x assume as2:x∈id(G)
    }

1161
from as obtain \( V \) where \( V : \forall \in N \_0 \� \{ (s,t) \in G \times G . \ s+(-t) \in V \} \subseteq U \\
unfolding \text{rightUniformity}\_def \text{ by auto} \\
from \( V(1) \) have \( 0 \in \text{int}(V) \text{ by auto} \\
then have \( 0 \cdot 0 \in V \text{ using Top}_2\_L1 \text{ by auto} \\
from as2 obtain \( t \) where \( t : 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} \\
\text{ by auto} \\
with \( V_0 t V(2) \) have \( x \in U \text{ by auto} \}
then have \( \text{id}(G) \subseteq U \text{ by auto} \\
moreover 
\{ 
\{ 
\text{fix } x \text{ assume } ass : x \in \{ (s,t) \in G \times G . \ s+(-t) \in -V \} \\
\text{then obtain } s t \text{ where } as : s \in G t \in G s+(-t) \in -V x = (s,t) \\
\text{by force} \\
from as(3) \( V(1) \) have \( s+(-t) \in \{ -q . q \in V \} \\
\text{using ginv\_image\_add by simp} \\
\text{then obtain } q \text{ where } q : q \in V s+(-t) = -q \text{ by auto} \\
\text{with } \langle V \in N_0 \rangle \text{ have } q \in G \text{ by auto} \\
\text{with } as(1,2) q(1,2) \text{ have } t+(-s) \in V \text{ using simple\_equation0\_add} \\
\text{by blast} \\
\text{with } as(1,2,4) \( V(2) \) have \( x \in \text{converse}(U) \text{ by auto} \}
\} \text{then have } \{ (s,t) \in G \times G . \ s+(-t) \in -V \} \subseteq \text{converse}(U) \text{ by auto} \\
moreover \text{from } \( V(1) \) have \( (-V) \in N_0 \text{ using neg\_neigh\_neigh by auto} \\
ultimately have \( \text{converse}(U) \in \text{rightUniformity} \text{ using as rightUniformity\_def} \\
\text{by auto} \}
\} \text{moreover } 
\{ 
\text{from } \( V(1) \) obtain \( W \) where \( W : W : N_0 \ W + W \subseteq V \text{ using exists\_procls\_zerohood} \\
\text{by blast} \\
\{ 
\text{fix } x \text{ assume } as : x : \{ (s,t) \in G \times G . \ s+(-t) \in W \} \ O \{ (s,t) \in G \times G . \ s+(-t) \in W \} \\
\} \text{then obtain } x_1 x_2 x_3 \text{ where} \\
x : x_1 : G x_2 : G x_3 : G x_1 + (-x_2) \in W x_2 + (-x_3) \in W x = (x_1, x_3) \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 \text{from } W(1) \text{ have } W : W \times W \subseteq G \times G \text{ by auto} \\
moreover 
\text{from } x(4,5) \text{ have } (x_1+(-x_2), x_2+(-x_3)) \in W \times W \text{ by auto} \\
\text{with } W \times W \text{ have } f((x_1+(-x_2), x_2+(-x_3))) \in f(W \times W) \\
\text{using func\_imagedef topgroup\_f\_binop by auto} \\
ultimately have \( (x_1+(-x_2)) + (x_2+(-x_3)) \in W + W \text{ 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 $x_1+(-x_3) \in W+W$ by auto
with $W(2)$ have $x_1+(-x_3) \in V$ by auto then have $x \in \{(s,t)\in G\times G. s+(-t) \in V\}$ using $x(1,3,6)$ by auto

with $V(2)$ have $\{(s,t)\in G\times G. s+(-t) \in W\} \subseteq U$ by auto
moreover from $W(1)$ have $\{(s,t)\in G\times G. s+(-t) \in W\} \in \text{rightUniformity}$ unfolding rightUniformity_def by auto
ultimately have $\exists Z \in \text{rightUniformity}. Z \circ Z \subseteq U$ by auto

ultimately have $\text{id}(G) \subseteq U \land (\exists Z \in \text{rightUniformity}. Z \circ Z \subseteq U) \land \text{converse}(U) \in \text{rightUniformity}$ by blast

then have $\forall U \in \text{rightUniformity}. \text{id}(G) \subseteq U \land (\exists Z \in \text{rightUniformity}. Z \circ Z \subseteq U) \land \text{converse}(U) \in \text{rightUniformity}$ by auto
with rightFilter show rightUniformity $\{\text{is a uniformity on} \ G\}$ 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 $M = \{(t, \{V \{t\} . V \in \text{leftUniformity}\}). t \in G\}$
have fun:M:G:Pow(Pow(G)) using neigh_from_uniformity side_uniformities(1) IsNeighSystem_def by auto
let $N = \{(t, \{V \{t\} . V \in \text{rightUniformity}\}). t \in G\}$
have funN:N:G:Pow(Pow(G)) using neigh_from_uniformity side_uniformities(2) IsNeighSystem_def by auto
\{
fix $U$ assume $\text{op}:U \subseteq T$
hence $U \subseteq G$ by auto
\{
fix $x$ assume $x:x \in U$
with $\text{op}$ have $xg:x \in G$ and $(-x) \in G$ using neg_in_tgroup by auto
then have $(x, \{V\{x\} . V \in \text{leftUniformity}\}) \in \{(t, \{V\{t\} . V \in \text{leftUniformity}\}). t \in G\}$
\}\}
by auto
with fun have app:M(x) = \{V{x}. V \in \text{leftUniformity}\} using ZF_fun_from_tot_val

by auto
have \((-x)+U : N\) using open_trans_neigh op x by auto
then have \(V:\{(s,t)\in G\times G. (-s)+t\in((-x)+U)\} \in \text{leftUniformity}\)
  unfolding leftUniformity_def by auto
with \(x\) have \(N:\forall t\in G. t:\{(s,t)\in G\times G. (-s)+t\in((-x)+U)\} \longleftrightarrow (-x)+t\in((-x)+U)\)
  using image_iff by auto

fix \(t\) assume \(t:t\in G\)
{
  assume as:\((-x)+t\in((-x)+U)\)
  then have \((-x)+t\in\text{LeftTranslation}(G,f,-x)U\) by auto
  then obtain \(q\) where \(q:q\in U \ (q,(-x)+t)\in\text{LeftTranslation}(G,f,-x)\)
    using image_iff by auto
  with op have \(q\in G\) by auto
  from \(q(2)\) have \((-x)+q = (-x)+t\) unfolding LeftTranslation_def
    by auto
  with \((-x)\in G\) \(q\in G\) \(t\in G\) have \(q = t\) using neg_in_tgroup
  cancel_left_add
    by blast
  with \(q(1)\) have \(t\in U\) by auto
}
moreover
{
  assume \(t:t\in U\)
  with \((-x)\in G\) \((-x)\in G\) \(t\in G\) have \(q = t\) using neg_in_tgroup
  cancel_left_add
    by blast
  with \(t\in U\) have \(t\in U\) by auto
}
ultimately have \((-x)+t\in((-x)+U) \longleftrightarrow t\in U\) by blast
}
with \(N\) have \(\forall t\in G. t:\{(s,t)\in G\times G. (-s)+t\in((-x)+U)\} \longleftrightarrow t\in U\)
  by blast
with \(op\) have \(\forall t. t:\{(s,t)\in G\times G. (-s)+t\in((-x)+U)\} \longleftrightarrow t\in U\)
  by auto
hence \(U = \{(s,t)\in G\times G. (-s)+t\in((-x)+U)\}\) by auto
with \(V\) have \(\exists V\in\text{leftUniformity}. U=V\{x\}\) by blast
with app have \(U = \{t, \{V \{t\} . V \in \text{leftUniformity}\}\} . t \in G\}\{x\}
  by auto
moreover from \(x\in G\) funN have app:N(x) = \{V{x}. V \in \text{rightUniformity}\}
  using ZF_fun_from_tot_val by simp
moreover
from x op have openTrans:U+(-x) : \(N\) using open_trans_neigh_2 by auto
then have \(V:\{(s,t)\in G \times G. \ s+(-t)\in (U+(-x))\} \in \text{rightUniformity}\)

unfolding \text{rightUniformity_def} by auto

with \(xg\) have

\[N: \forall t\in G. \ t:\{(s,t)\in G \times G. \ s+(-t)\in (U+(-x))\}-\{x\} \iff t+(-x)\in (U+(-x))\]

using \text{vimage_iff} by auto

moreover

\[
\begin{align*}
\text{fix } t & \text{ assume } t\in G \\
& \text{ assume as: } t+(-x)\in (U+(-x)) \\
& \text{ hence } t+(-x)\in \text{RightTranslation}(G,f,-x)U \text{ by auto} \\
& \text{ then obtain } q \text{ where } q\in U \langle q,t+(-x)\rangle\in \text{RightTranslation}(G,f,-x) \\
& \text{ using } \text{image_iff} \text{ by auto} \\
& \text{ with } op \text{ have } q\in G \text{ by auto} \\
& \text{ from q(2) have } q+(-x) = t+(-x) \text{ unfolding } \text{RightTranslation_def} \\
& \text{ by auto} \\
& \text{ with } \langle q\in G \rangle \langle (-x) \in G \rangle \langle t\in G \rangle \text{ have } q = t \text{ using } \text{cancel_left_add} \\
& \text{ by simp} \\
& \text{ with } q(1) \text{ have } t\in U \text{ by auto} \\
\end{align*}
\]

moreover

\[
\begin{align*}
\text{assume } t\in U \\
& \text{ with } \langle (-x)\in G \rangle \text{ have } t+(-x)\in (U+(-x)) \text{ using } \text{lrtrans_image(2)} \\
& \text{ by auto} \\
\end{align*}
\]

ultimately have \(t+(-x)\in (U+(-x)) \iff t:U \text{ by blast} \)

with \(N\) have \(\forall t\in G. \ t:\{(s,t)\in G \times G. \ s+(-t)\in (U+(-x))\}-\{x\} \iff t:U \)

by blast

with \(op\) have \(\forall t. \ t:\{(s,t)\in G \times G. \ s+(-t)\in (U+(-x))\}-\{x\} \iff t:U \)

by auto

hence \(\langle (s,t)\in G \times G. \ s+(-t)\in (U+(-x))\rangle - \{x\} = U \text{ by auto} \)

then have \(U = \text{converse}(\{(s,t)\in G \times G. \ s+(-t)\in (U+(-x))\})\{x\} \)

unfolding \text{vimage_def} by simp

with \(V\) app have \(U \in \{(t, \{V \ t\} . V \in \text{leftUniformity}) \ . t \in G\}(x)\)

using \text{side_uniformities(2)} \text{IsUniformity_def} by auto

ultimately have

\[
\begin{align*}
U \in \{(t, \{V \ t\} . V \in \text{leftUniformity}) \ . t \in G\}(x) \text{ and} \\
U \in \{(t, \{V \ t\} . V \in \text{rightUniformity}) \ . t \in G\}(x) \\
\end{align*}
\]

by auto

} hence

\[
\begin{align*}
\forall x\in U. \ U \in \{(t, \{V \ t\} . V \in \text{leftUniformity}) \ . t \in G\} \ \text{x and} \\
\forall x\in U. \ U \in \{(t, \{V \ t\} . V \in \text{rightUniformity}) \ . t \in G\} \ \text{x} \\
\end{align*}
\]

by auto

} hence

\[
\begin{align*}
T \subseteq \{U \in \text{Pow}(G). \ \forall x\in U. \ U \in \{(t, \{V \ t\} . V \in \text{leftUniformity}) \ . t \in G\} \\
\end{align*}
\]

1165
\[ t \in G \} \) and \\
\{ U \cap \{ t \} \subseteq U \} \} . \ t \in G \} \} . \\
by \text{ auto} \\
moreover \\
\{ \\
\text{ fix } U \text{ assume as: } U \in \{ t \} \subseteq U \} \} . \ t \in G \} \} \\
by \text{ auto} \\
moreover \\
\{ \\
\text{ fix } x \text{ assume } x : x \subseteq U \\
\text{ with as(1) have } x g : x \subseteq G \text{ by auto} \\
\text{ from x as(2) have } U \subseteq \{ t \} \subseteq U \} \} . \ t \in G \} \} \\
by \text{ auto} \\
\text{ with xg fun have } U \subseteq \{ x \} \subseteq U \} \} \\
by \text{ auto} \\
\text{ then obtain } V \text{ where } V : U = V \} \subseteq V \} \subseteq U \} \} \\
\text{ unfolding leftUniformity def by auto} \\
\text{ from V(2) obtain } W \text{ where } W : W \subseteq \mathcal{N}_0 \} \subseteq \{ (s, t) : G \times G \} . \ (-s) + t : W \subseteq V \text{ by auto} \\
\text{ using image_iff by auto} \\
\text{ hence } B : \{ (s, t) : G \times G \} . \ (-s) + t : W \} \subseteq \{ t \in G \} . \ (-x) + t : W \} \subseteq \text{ by auto} \\
\text{ from } W(1) \text{ have } W G : W \subseteq G \text{ by auto} \\
\{ \\
\text{ fix } t \text{ assume } t : t \in x + W \\
\text{ then have } t \in \text{ LeftTranslation}(G, f, x) \subseteq W \text{ by auto} \\
\text{ then obtain } s \text{ where } s : s \subseteq W \} \text{ using apply_equality} \\
\text{ image_iff by auto} \\
\text{ with } < W \subseteq G \} \text{ have } s \subseteq G \text{ by auto} \\
\text{ from s(2) have } t = x + t \subseteq W \text{ unfolding LeftTranslation_def by auto} \\
\text{ with } < W \subseteq G \} \text{ have } (-x) + t = s \text{ using put_on_the_other_side(2)} \\
by \text{ simp} \\
\text{ with s(1) have } (-x) + t : W \subseteq W \text{ by auto} \\
\text{ with } < W \subseteq G \} \text{ have } t \in \{ s \subseteq G \} . \ (-x) + s : W \} \subseteq W \text{ by auto} \\
\} \\
\text{ then have } x + W \subseteq \{ t \subseteq G \} . \ (-x) + t : W \} \subseteq W \text{ by auto} \\
\text{ with } B \text{ have } x + W \subseteq \{ (s, t) : G \times G \} . \ (-s) + t \in W \} \subseteq \{ x \} \text{ by auto} \\
\text{ with } A \text{ have } x + W \subseteq V \} \subseteq \{ x \} \text{ by blast} \\
\text{ with } V(1) \text{ have } x + W \subseteq U \text{ by auto} \\
\text{ then have } \text{ int}(x + W \} \subseteq \text{ U using Top_2_L1 by blast} \\
\text{ moreover from xg W(1) have } x \in \text{ int}(x + W \} \subseteq \text{ using elem_in_int_ltrans(1)} \\
\text{ by auto} \\
\text{ moreover have } \text{ int}(x + W \} \subseteq \text{ E using Top_2_L2 by auto} \\
\text{ ultimately have } \exists Y \subseteq T . \ x \in Y \wedge Y \subseteq U \text{ by auto} \\
\} \\
\text{ then have } U \subseteq U \text{ using open_neigh_open by auto} \\
\} \\
\text{ hence } \{ U \in \text{ Pow}(G \} . \ \forall x \subseteq U . \ U \subseteq \{ t \} \subseteq U \} . \ \text{ E using leftUniformity} \\
. \ t \in G \} \} \\
\text{ by auto} \\
1166
moreover

\{ 
  \text{fix } U \text{ assume } as:U \in \text{Pow}(G) \forall x \in U. U \in \{ \{ t, \{ V \{ t \} \} \cdot V \in \text{rightUniformity} \} \cdot t \in G \} \}

x

\{ 
  \text{fix } x \text{ assume } x:x \in U 
  \text{with } as(1) \text{ have } xg:x \in G \text{ by auto} 
  \text{from } x \text{ as } as(2) \text{ have } U \in \{ \{ t, \{ V \{ t \} \} \cdot V \in \text{rightUniformity} \} \cdot t \in G \} \cdot x \text{ by auto} 
  \text{with } xg \text{ funN} \text{ have } U \in \{ V \{ x \} \cdot V \in \text{rightUniformity} \} \text{ using } \text{apply_equality} 
  \text{by auto} 
  \text{then obtain } V \text{ where } V:U=V\{ x \} \text{ using } \text{apply_equality} 
  \text{by auto} 
  \text{then obtain } W \text{ where } W:W\subseteq \text{converse}(V) 
  \text{unfolding } \text{rightUniformity_def} \text{ by auto} 
  \text{from } W(2) \text{ have } A:\{ \{ s,t \} : G \times G. s+-(-t):W\} -\{ x \} \subseteq V\{ x \} \text{ by auto} 
  \text{from } xg \text{ have } \forall q \in G. q \in \{ \{ \{ s,t \} : G \times G. s+-(-t):W\} -\{ x \} \} \iff q+-(-x):W 
  \text{using } \text{image_iff} \text{ by auto} 
  \text{hence } B:\{ \{ s,t \} : G \times G. s+-(-t):W\} -\{ x \} = \{ t \in G. t+-(-x):W \} \text{ by auto} 
  \text{from } W(1) \text{ have } W:W\subseteq G \text{ by auto} 
  \{ 
    \text{fix } t \text{ assume } t \in W+x 
    \text{with } W\subseteq G \text{ obtain } s \text{ where } s \in W \text{ and } t=s+x \text{ using } \text{lrtrans_image}(2) 
    \text{by auto} 
    \text{with } W\subseteq G \text{ have } s \in G \text{ by auto} 
    \text{with } W\subseteq G \text{ have } t=s+x \text{ have } t+-(-x) = s \text{ using } \text{put_on_the_other_side} 
    \text{by simp} 
    \text{with } s \in W \text{ have } t \in \{ s \in G. s+-(-x) \in W \} \text{ by auto} 
  \} 
  \text{then have } W+x \subseteq \{ t:G. t+-(-x):W \} \text{ by auto} 
  \text{with } B \text{ have } W+x \subseteq \{ \{ s,t \} : G \times G. s + (-t) \in W \} -\{ x \} \text{ by auto} 
  \text{with } A \text{ have } W+x \subseteq V\{ x \} \text{ by blast} 
  \text{with } V(1) \text{ have } W+x \subseteq U \text{ by auto} 
  \text{then have } \text{int}(W+x) \subseteq U \text{ using } \text{Top_2_L1} \text{ by blast} 
  \text{moreover} 
  \text{from } xg \text{ have } x \in \text{int}(W+x) \text{ using } \text{elem_in_int_rtrans}(1) \text{ by auto} 
  \text{moreover have } \text{int}(W+x) \subseteq T \text{ using } \text{Top_2_L2} \text{ by auto} 
  \text{ultimately have } \exists Y \in T. x \in Y \lor Y \subseteq U \text{ by auto} 
  \} 
\text{then have } U \in T \text{ using } \text{open_neigh_open} \text{ by auto} 
\}
ultimately have
\{U \in \text{Pow}(G). \ \forall x \in U. U \in \{\{t, \{V\} . V \in \text{leftUniformity}\}. t \in G\}(x)\} = T
\{U \in \text{Pow}(G). \ \forall x \in U. U \in \{\{t, \{V\} . V \in \text{rightUniformity}\}. t \in G\}(x)\} = T

by auto

then show UniformTopology(leftUniformity,G) = T and UniformTopology(rightUniformity,G) = T

unfolding UniformTopology_def 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\in 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 assms have L: LeftTranslation(G,f,x):G \rightarrow G and leftUniformity \{is a uniformity on\} G
using group0_5_L1 side_uniformities(1) by auto

moreover have \(\forall V \in \text{leftUniformity}. P-(V) \in \text{leftUniformity}\)

proof -

\{ fix V assume V \in \text{leftUniformity} then obtain U where U \in \mathcal{N}_0 and \{(s,t) \in G \times G . (- s) + t \in U\} \subseteq V \}

unfolding leftUniformity_def by auto

with V \in leftUniformity have
\{ fix z assume z:z \in \{(s,t) \in G \times G . (- s) + t \in U\} then obtain s t where st:z=(s,t) s\in G t\in G by auto
from st(1) z have st2: (- s) + t \in U by auto
from assms st have
P(z) = \langle \text{LeftTranslation}(G, f, x)(s), \text{LeftTranslation}(G, f, x)(t)\rangle
using prodFunctionApp group0_5_L1(2) by blast
with assms st(2,3) have P(z) = \langle x+s,x+t\rangle using group0_5_L2(2)
by auto
moreover
from \langle x\in G, s\in G, t\in G \rangle have (- (x+s)) + (x+t) = (-s)+t
using cancel_middle_add(3) by simp
with st2 have (- (x+s)) + (x+t) \in U by auto
ultimately have P(z) \in \{(s,t) \in G \times G . (- s) + t \in U\}
using assms st(2,3) group_op_closed by auto
with as(3) have P(z) \in V by force
with L z have z \in P-(V) using prodFunction func1_1_L5A vimage_iff

1168
by blast } 
with as(2) have $\exists U \in N_0. \{ (s,t) \in G \times G . (-s) + t \in U \} \subseteq P-(V)$

by blast with $\langle \text{LeftTranslation}(G,f,x):G \rightarrow G \rangle \subseteq G \times G$ have $P-(V) \in \text{leftUniformity}$ 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 $x \in G$
shows
$\text{RightTranslation}(G,f,x) \{ \text{is uniformly continuous between} \} \text{rightUniformity}$
{and} \text{rightUniformity}

proof -
let $P = \text{ProdFunction}(\text{RightTranslation}(G, f, x), \text{RightTranslation}(G, f, x))$
from asms have $R: \text{RightTranslation}(G, f, x):G \rightarrow G$ and \text{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 $U \in N_0$ and \{ $s,t) \in G \times G . s + (-t) \in U \} \subseteq V$
\unfolding rightUniformity_def by auto
with $\langle V \in \text{rightUniformity} \rangle$ have $\{ (s,t) \in G \times G . s + (-t) \in U \} \subseteq V$
\unfolding rightUniformity_def by auto
\{ fix $z$ assume $z:z \in \{ (s,t) \in G \times G . s + (-t) \in U \}$
then obtain $s,t$ where $st:z=(s,t) s \in G t \in G$ by auto
from $st(1)$ $z$ have $st2: s + (-t) \in U$ by auto
from asms $st$ have $P(z) = (\text{RightTranslation}(G, f, x)(s), \text{RightTranslation}(G, f, x)(t))$
\using prodFunctionApp group0_5_L1(1) by blast
with asms $st(2,3)$ have $P(z) = (s+x,t+x)$ using group0_5_L2(1)
by auto
moreover
from $x \in G <s \in G < t \in G$ have $(s+x) + (- (t+x)) = s + (- t)$
\using cancel_middle_add(4) by simp
with $st2$ have $(s+x) + (- (t+x)) \in U$ by auto
ultimately have $P(z) \in \{ (s,t) \in G \times G . s + (- t) \in U \}$
\using asms $st(2,3)$ group_op_closed by auto
with as(3) have $P(z) \in V$ by force

1169
with \( R \ z \) have \( z \in P-(V) \) using prodFunction func1_1_L5A vimage_iff by blast

\}

with as(2) have \( \exists U \in N_0. \{ (s,t) \in G \times G . s + (- t) \in U \} \subseteq P-(V) \)

by blast

with \(<\)RightTranslation(G,f,x):G→G,<V \subseteq G \times G. have P-(V) \in rightUniformity unfolding rightUniformity_def using prodFunction func1_1_L6A by blast

\} thus thesis by simp

qed

ultimately show thesis using IsUniformlyCont_def by auto

qed

The third uniformity important on topological groups is called the uniformity of Roelcke.

definition (in topgroup) roelckeUniformity
where roelckeUniformity ≡ \{ V \in Pow(G×G). \exists U \in 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 (in topgroup) roelcke_uniformity:
shows roelckeUniformity \{ is a uniformity on \} G

proof -
let \( \Phi = roelckeUniformity \)
have \( \forall U \in \Phi. id(G) \subseteq U \land (\exists V \in \Phi. V \cap V \subseteq U) \land \text{converse}(U) \in \Phi \)

proof
fix \( U \) assume \( U \in roelckeUniformity \)
then obtain \( V \) where \( V:V \subseteq G \times G \)

unfolding roelckeUniformity_def by auto

from \( V(2) \) have \( \forall V:V \subseteq G \) by auto

have \( id(G) \subseteq U \)

proof -
from \( V(2) \) have \( 0 \in int(V) \) by auto

then have \( V:0 \in V \) using Top_2_L1 by auto

\{ fix \( x \) assume \( x:x \in G \)

with \( V\in N_0 \) have \( x \in V+x \) using elem_in_int_rtrans(1) Top_2_L1

by blast
with \( V\subseteq G < x \in G < 0 \in V \) have \( x+0 : (V+x)+V \)

using lrtrans_in_group_add(2) interval_add(4) by auto

with \( x < G \) have \( x : (V+x)+V \) using group0_2_L2 by auto

with \( x < G \) have \( \{ (x,x) : (V+s)+V \} \subseteq U \)

with \( V(1) \) have \( (x,x) \in U \) by auto

\} thus \( id(G) \subseteq U \) by auto

qed

moreover have \( \text{converse}(U) \in \Phi \)

proof -
\{ fix 1 assume \( 1 \in \{ (s,t) \in G \times G. t \in ((-V)+s)+(-V) \} \)

1170
then obtain \( s, t \) where \( st : s \in G \quad t \in ((-V)+s)+(-V) \) 
by force
from \( \langle V \subseteq G \rangle \) have \( smG : (-V) \subseteq G \) using \( \text{ginv_image_add(1)} \) by simp
with \( s \in G \) have \( VxG : (-V)+s \subseteq G \) using \( \text{lrtrans_in_group_add(2)} \)
by simp
from \( \langle V \subseteq G \rangle \) \( \langle t \in G \rangle \) have \( VsG : V+t \subseteq G \)
using \( \text{lrtrans_in_group_add(2)} \)
by simp
from \( st(3) \) \( V \times G \)
\( smG \) obtain \( x, y \) where \( xy : t = x+y \) \( x \in (-V)+s \) \( y \in (-V) \)
using \( \text{elements_in_set_sum} \) by blast
from \( xy(2) \) \( smG \) \( st(1) \)
obtain \( z \) where \( z : x = z+s \) \( z \in (-V) \)
using \( \text{elements_in_rtrans} \) by blast
with \( y \in (-V) \) \( (-V) \subseteq G \) \( s \in G \) \( t = x+y \)
have \( ts : (-z)+t+(-y) = s \) using \( \text{cancel_middle_add(5)} \) by blast

\{ fix \( u \) assume \( u \in (-V) \)
with \( \langle V \subseteq G \rangle \) have \( (-u) \in V \) using \( \text{ginv_image_el_add} \) by simp \}
hence \( R : \forall u \in (-V). (-u) \in V \) by simp
with \( z(2) \) \( xy(3) \)
have \( zy : (-z) \in V \) \( (-y) \in V \) by auto
from \( zy(1) \) \( V \times G \)
\( st(2) \) have \( (-z)+t : V+t \) using \( \text{lrtrans_image(2)} \)
by auto
moreover from \( V(2) \) have \( (-V) : N_0 \) using \( \text{neg_neigh_neigh} \) by auto
ultimately have \( \exists V \in N_0. \{ (s,t) \in G \times G. t \in (V+s)+V \} \subseteq \text{converse(U)} \) by auto
ultimately show \( \text{converse(U)} \in \text{roelckeUniformity} \) unfolding \( \text{roelckeUniformity_def} \) by auto
moreover have \( \exists Z \in \Phi. \ Z \circ Z \subseteq U \)
proof -
from \( V(2) \) obtain \( W \) where \( W : W \in N_0 \) \( W+W \subseteq V \) using \( \text{exists_procls_zerohood} \)
by blast
then have \( WG : W \subseteq G \) by auto
moreover
\{ fix \( k \) assume \( \text{as:} k : \{ (s,t) \in G \times G. t \in (W+s)+W \} \circ \{ (s,t) \in G \times G. t \in (W+s)+W \} \)
then obtain \( x_1, x_2, x_3 \) where \( x : x_1 \in G \) \( x_2 \in G \) \( x_3 \in G \) \( x_1 \in (W+x_1)+W \) \( x_2 \in (W+x_2)+W \) \( k = \langle x_1, x_3 \rangle \)
unfolding \( \text{comp_def} \) by auto
from \( \langle x_1 \in G \rangle \) have \( VsG : W+x_1 \subseteq G \) and \( Vx1G : V+x_1 \subseteq G \)
}\}
using \text{lrtrans\_in\_group\_add}(2) by auto
from x(4) VsG WG obtain x y where xy:x_2 = x+y x \in W+x_1 y \in W
using \text{elements\_in\_set\_sum} by blast
from xy(2) WG x(1) obtain z where z:x = z+x_1 z \in W using \text{elements\_in\_rtrans}
by blast
from z(2) xy(3) WG have yzG:y \in G by auto
from x(2) have VsG:W+x_2 \subseteq G using \text{lrtrans\_in\_group\_add} by simp
from x(5) VsG WG obtain x' y' where xy2:x_3 = x'+y' x' \in W+x_2 y' \in W
using \text{elements\_in\_set\_sum} by blast
from xy2(2) WG x(2) obtain z' where z2:x' = z'+x_2 z' \in W using \text{elements\_in\_rtrans}
by blast
from z2(2) xy2(3) WG have yzG2:y' \in G by auto
from x(2) have VsG:W+x_1 \subseteq G using \text{lrtrans\_in\_group\_add} by simp
from x(5) VsG WG obtain z'+z \in W+W y+y' \in W+W
using \text{interval\_add}(4) by auto
with W(2) have yzV:z'+z \in V y+y' \in V by auto
from yzV(1) VG x(1) have (z'+z)+x_1 \in V+x_1 using \text{lrtrans\_image}(2) by auto
with yzV(2) Vx1G VG have ((z'+z)+x_1)+(y+y') \in (V+x_1)+V
using \text{interval\_add}(4) by auto
moreover have 0 = 0+0+0 using group0_2_L2 zero_in_tgroup by auto
moreover have \(0,0):G \times G\) using zero_in_tgroup by auto
ultimately show \(\exists Z \in \text{roelcke\_Uniformity}. Z \circ Z \subseteq U\) by simp
qed
ultimately show \(\exists \phi \in \Phi. \phi \circ \phi \subseteq U\) by \text{roof}
have \text{roelcke\_Uniformity} \{\text{is a filter on} (G \times G)
proof -

\{ assume 0 \in \text{roelcke\_Uniformity}
then obtain \(\omega\) where U:W+V \subseteq \(\{(s,t)\in G \times G. t \in (W+s)+W\}\) by auto
ultimately show \(\exists Z \in \text{roelcke\_Uniformity}. Z \circ Z \subseteq U\) by simp
have \text{roelcke\_Uniformity} \{\text{is a filter on} (G \times G)
from U(1) have 0 ∈ int(W) by auto
then have 0 ∈ W using Top_2_L1 by auto
with ⟨W ∈ N_0⟩ have 0 + 0 + 0 ∈ (W + 0) + W
using group_0_2_L2 group_op_closed trans_neutral_image interval_add_zero
by auto
ultimately have ⟨0, 0⟩ ∈ {⟨s, t⟩ ∈ G × G. t ∈ (W + s) + W} by auto
with U(2) have False by blast

moreover { 
fix x xa assume as: x ∈ roelckeUniformity xa ∈ x
have roelckeUniformity ⊆ Pow(G × G) unfolding roelckeUniformity_def
by auto
with as have xa ∈ G × G by auto
}
moreover { 
have G × G ∈ Pow(G × G) by auto
moreover have {⟨s, t⟩ ∈ G × G. t ∈ (G + s) + G} ⊆ G × G by auto
moreover note zneigh_not_empty
ultimately have G × G ∈ roelckeUniformity unfolding roelckeUniformity_def
by auto
}
moreover { 
fix A B assume as: A ∈ roelckeUniformity B ∈ roelckeUniformity
from as(1) obtain AU where
AU: AU ∈ N_0 {⟨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 ∈ N_0 {⟨s, t⟩ ∈ G × G. t ∈ (BU + s) + BU} ⊆ B ∈ Pow(G × G)
unfolding roelckeUniformity_def by auto
from AU(1) BU(1) have 0 ∈ int(AU) ∩ int(BU) 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_L1 by auto
with op have int(AU) ∩ int(BU) ⊆ int(AU ∪ BU) using Top_2_L5
by auto
moreover note AU(1) BU(1)
ultimately have interNeigh : AU ∩ BU ∈ N_0 unfolding zerohoods_def by auto
moreover { 
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 \(<AU \cap BU \in N_0> <s\in G> have AU \cap BU \subseteq G\) and \((AU \cap BU)+s \subseteq G\)
using lrtrans_in_group_add(2) by auto

with z(4) obtain \(x y\) where \(t : t = x+y\ x \in (AU \cap BU)+s\ y \in AU \cap BU\)
using elements_in_set_sum by blast

from \(t(2) z(2)\) interNeigh obtain q where \(x : x = q+s\ q \in AU \cap BU\)
using lrtrans_image(2)
by auto
with \(AU(1) BU(1) z(2)\)

have \(x \in AU+s\ x \in BU+s\)
using lrtrans_image(2)
by auto
with \(<y \in AU \cap BU> <AU \in N_0> <BU \in N_0> <s \in G> <t = x+y> have\)
\(t \in (AU+s)+AU\ and t \in (BU+s)+BU\)
using lrtrans_in_group_add(2) elements_in_set_sum_inv by auto

with z(1,2,3) have
\(z \in \{(s,t)\in G\times G. t \in (AU+s)+AU\} and z \in \{(s,t)\in G\times G. t \in (BU+s)+BU\}\)
by auto

then have
\(\{(s,t)\in G\times G. t \in ((AU \cap BU)+s)+(AU \cap BU)\} \subseteq\)
\(\{(s,t)\in G\times G. t \in (AU+s)+AU\} \cap \{(s,t)\in G\times G. t \in (BU+s)+BU\}\)
by auto

with \(AU(2) BU(2)\) have \(\{(s,t)\in G\times G. t \in ((AU \cap BU)+s)+(AU \cap BU)\} \subseteq A \cap B\)
by blast
ultimately have \(A \cap B \in roelckeUniformity\) using \(AU(3) BU(3)\)
unfolding roelckeUniformity_def
by blast

moreover
\{ fix B C assume as:B \in roelckeUniformity C \subseteq (G \times G) B \subseteq C \}
from as(1) obtain BU where \(BU:BU \in N_0 \\{(s,t)\in G\times G. t \in (BU+s)+BU\} \subseteq B\)
unfolding roelckeUniformity_def by blast
from as(3) \(BU(2)\) have \(\{(s,t)\in G\times G. t \in (BU+s)+BU\} \subseteq C\) by auto
then have \(C \in roelckeUniformity\) using as(2) \(BU(1)\)
unfolding roelckeUniformity_def
by auto

ultimately show thesis unfolding IsFilter_def by auto
qed
ultimately show thesis using IsUniformity_def by auto
qed

The topology given by the roelcke uniformity is the original topology

lemma (in topgroup) top_generated_roelcke_uniformity:
  shows UniformTopology(roelckeUniformity,G) = T
proof -

1174
\[
\text{let } M = \{ \langle t, \{ V \{ t \} . V \in \text{roelckeUniformity} \rangle \mid t \in G \} \n\]

have fun: M: G \to \text{Pow(Pow(G))} using IsNeighSystem_def neigh_from_uniformity
roelcke_uniformity
by auto

{ fix U assume as: U \in \{ U \in \text{Pow(G)}. \forall x \in U. U \in Mx \} }
{ fix x assume x: x \in U with as have xg: x \in G by auto from x as have U \in \{ \langle t, \{ V \{ t \} . V \in \text{roelckeUniformity} \rangle \mid t \in G \}(x) \} by auto with fun <x \in G> have U \in \{ V \{ x \} . V \in \text{roelckeUniformity} \} using ZF_fun_from_tot_val by simp then obtain V where V: U = V{x} V \in \text{roelckeUniformity} by auto from V(2) obtain W where W: W \in N_0 \{ \langle s, t \rangle \in G \times G. t \in (W+s)+W \} \subseteq V unfolding roelckeUniformity_def by auto from W(1) have WG: W \subseteq G by auto from W(2) have A: \{ \langle s, t \rangle \in G \times G. t \in (W+s)+W \} \subseteq V \{ x \} by auto have \{ \langle s, t \rangle \in G \times G. t \in (W+s)+W \} \{ x \} = (W+x)+W proof - let A = \{ \langle s, t \rangle \in G \times G. t \in (W+s)+W \} from <W \subseteq G> \langle x \in G \rangle have I: (W+x)+W \subseteq G using lrtrans_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 <W \subseteq G> <x \in G> I have (W+x)+W \subseteq \{ t \in G. \langle x, t \rangle \in A \} 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 (int(W)+x) \times (int(W)) \subseteq (W+x) \times W using Top_2.L1 by auto then have f((int(W)+x) \times (int(W))) \subseteq f((W+x) \times W) using image_mono by auto moreover from xg WG have (\langle int(W)+x, int(W) \rangle) \in \text{Pow(G)} \times \text{Pow(G)} and ((W+x), W) \in \text{Pow(G)} \times \text{Pow(G)} using Top_2.L2 lrtrans_in_group_add(2) by auto then have (int(W)+x)+(int(W)) = f((int(W)+x) \times (int(W))) and (W+x)+W = f((W+x) \times W) using interval_add(2) by auto ultimately have (int(W)+x)+(int(W)) \subseteq (W+x)+W by auto with xg WG have int(W+x)+(int(W)) \subseteq (W+x)+W using rtrans_interior by auto moreover

have \( \int(W+x) + (\int(W)) = (\bigcup t \in \int(W+x). t + (\int(W))) \)
using interval_add(3) Top_2_L2 by auto
moreover have \( \forall t \in \int(W+x). t + (\int(W)) = \int(t+W) \)
proof =
{ fix t assume \( t \in \int(W+x) \)
  from \( <x \in G> \) have \( (W+x) \subseteq G \) using lrtrans_in_group_add(2)
  by simp
  with \( <t \in \int(W+x)> \) have \( t \in G \) using Top_2_L2 by auto
  with \( <W \subseteq G> \) have \( t + \int(W) = \int(t+W) \) using ltrans_interior
}
thus thesis by simp
qed
ultimately have \( \int(W+x) + (\int(W)) = (\bigcup t \in \int(W+x). \int(t+W)) \)
by auto
with topSpaceAssum have \( \int(W+x) + (\int(W)) \in T \) using Top_2_L2
proof =
}
moreover from \( <x \in G> <W \in N_0> \) have \( x \in \int(W+x) + (\int(W)) \)
using elem_in_int_rtrans(2) by simp
moreover note \( W \)
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 \( \{ U \in \Pow(G). \forall x \in U. U \in \{ \{ t, \{ V \{ t \} . V \in \roelckeUniformity \} \} . t \in G \} \} \subseteq T \)
by auto
moreover
{ fix \( U \) assume op: \( U \in T \)
}
fix \( x \) assume \( x : x \in U \)
with op have \( xg : x \in G \) by auto
have \( (-x) + U \in N_0 \) using open_trans_neigh op x by auto
then obtain \( W \) where \( W : W \in N_0 \) \( W + W \subseteq (-x) + U \) using exists_procls_zerohood
by blast
let \( V = x + (W + (-x)) \cap W \)
from \( W(1) \) have \( W \subseteq G \) by auto
from \( xWx \) \( W(1) \) have \( 0 \in \int(x + (W + (-x))) \cap \int(W) \) by auto
have \( \int(x + (W + (-x))) \cap \int(W) \in T \)
using Top_2_L2 topSpaceAssum unfolding IsATopology_def by auto
have \( \int(x + (W + (-x))) \subseteq (x + (W + (-x))) \cap W \) using Top_2_L1 by auto
with \( \int \) have \( \int(x + (W + (-x))) \cap \int(W) \subseteq \int(x + (W + (-x))) \cap W \)
using Top_2_L5 by auto
moreover note xWx W(1)
ultimately have V_NEIG: V ∈ N₀ unfolding zerohoods_def by auto
{
fix z assume z: z ∈ (V+x)+V
from W(1) have VG: V ⊆ G by auto
with <x∈G> have VxG: V+x ⊆ G using lrtrans_in_group_add(2) by auto

from z VG VxG W(1) obtain a₁ b₁ where ab: z = a₁ + b₁ a₁ ∈ V+x b₁ ∈ V
using elements_in_set_sum by blast
from ab(2) xg VG obtain c₁ where c: c₁ = c₁ + x c₁ ∈ V using elements_in_rtrans
by blast
from ab(3) c(2) have bc: b₁ ∈ W c₁ ∈ x+(W+(-x)) by auto
from <x∈G> have x+(W+(-x)) = {x+y. y ∈ (W+(-x))}
using neg_in_tgroup lrtrans_in_group_add lrtrans_image by auto
with <c₁ ∈ x+(W+(-x))> obtain d where d: c₁ = x+d d ∈ W+(-x)
by auto
from <x∈G> <e∈N₀> <d ∈ W+(-x)> obtain e where e: d = e+(-x) e ∈ W
using neg_in_tgroup lrtrans_in_group_add lrtrans_image(2) by auto
from e(2) WG have eG: e ∈ G by auto
from <e∈W> <W⊆G> <b₁ ∈ W> have eG b₁ ∈ G by auto
from <z = a₁ + b₁> <a₁ = c₁ + x> <c₁ = x+d> <d = e+(-x)>
have z = x+(e+(-x)) + x + b₁ by simp
with <x∈G> <e∈G> have z = (x+e)+b₁ using cancel_middle(4) by simp
with <x∈G> <e∈G> <b₁ ∈ G> have z = x+(e+b₁) using group_oper_assoc
by simp
moreover from e(2) ab(3) WG have e+b₁ ∈ W+W using elements_in_set_sum_inv
by auto
moreover note xg WG
ultimately have z ∈ x+(W+W) using elements_in_ltrans_inv interval_add(1)
by auto
moreover
from <W⊆G> <U∈T> have W + W ⊆ G and U ⊆ G using interval_add(1)
by auto
with <W + W ⊆ (-x)+U> <x∈G> have x+(W+W) ⊆ U using trans_subset
by simp
ultimately have z ∈ U by auto
}
then have sub: (V+x)+V ⊆ U by auto
moreover from V_NEIG have unif: {⟨s,t⟩ ∈ G×G. t: (V+s)+V} ∈ roelckeUniformity
unfolding roelckeUniformity_def by auto
moreover from xg have

1177
∀q. q ∈ {(s, t) ∈ G × G. t ∈ (V+s)+V} x = (V+x)+V}

by auto
then have \{q ∈ G × G. t ∈ (V+s)+V\} x = ((V+x)+V) ∩ G
by auto
ultimately have basic: \{q ∈ G × G. t ∈ (V+s)+V\} x ⊆ U using op
by auto
have add: \{(x) × U\} x = U by auto
from basic add have \{(s, t) ∈ G × G. t ∈ (V+s)+V\} x = (V+x)+V) ∩ G
by auto
ultimately have basic: \{(s, t) ∈ G × G. t : (V+s)+V\} x ⊆ U using op
by auto
ultimately have \{(s, t) ∈ G × G. t : (V+s)+V\} x = ((V+x)+V)
by auto
ultimately have basic: \{(s, t) ∈ G × G. t ∈ (V+s)+V\} x = U using op
by auto
moreover have R: ∀B ∈ roelckeUniformity. (∀C ∈ Pow(G × G). B ⊆ C
→ C ∈ roelckeUniformity)
using roelcke_uniformity unfolding IsUniformity_def IsFilter_def
by auto
moreover from op xg have GG: \{(s, t) ∈ G × G. t ∈ (V+s)+V\} x = Pow(G × G)
by auto
ultimately have \{(s, t) ∈ G × G. t ∈ (V+s)+V\} x = U using op
by auto
moreover have \{(s, t) ∈ G × G. t ∈ (V+s)+V\} x = U using op
by auto
moreover from R unif GG have
(\{(s, t) ∈ G × G. t ∈ (V+s)+V\} x = U using op
by auto
moreover from R unif GG have
ultimately have \exists V ∈ roelckeUniformity. V x = U by auto
then have U ∈ \{(V \{t\} . V ∈ roelckeUniformity) . t ∈ G\} x by auto
with xg fun have U ∈ \{(t, \{V \{t\} . V ∈ roelckeUniformity\}) . t ∈ G\} x by auto
with xg fun have U ∈ \{(t, \{V \{t\} . V ∈ roelckeUniformity\}) . t ∈ G\} x by auto
ultimately have \{U ∈ Pow(G). ∀x ∈ U. U ∈ \{(t, \{V \{t\} . V ∈ roelckeUniformity\}) . t ∈ G\} x = T by auto
then show thesis unfolding UniformTopology_def by auto
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 = ProdFunction(GroupInv(G,f), GroupInv(G,f))

have L: GroupInv(G,f):G\rightarrow G
using groupAssum group0_2_T2 roelcke_uniformity by auto

have \(\forall V \in \text{roelckeUniformity} \) . P-(V) \(\in \text{roelckeUniformity}\)
proof
fix V assume v:V \(\in \text{roelckeUniformity}\)
then obtain U where U \(\in \mathbb{N}_0\) and \(\{(s,t) \in G \times G . t \in U + s + U\}\) \(\subseteq V\)
unfolding roelckeUniformity_def by auto
with \(<V \in \text{roelckeUniformity}>\) have
as: V \(\subseteq G \times G\) U \(\in \mathbb{N}_0\) \(\{(s,t) \in G \times G . t \in U + s + U\}\) \(\subseteq V\)
unfolding roelckeUniformity_def by auto
from as(2) obtain W where w:W \(\in \mathbb{N}_0\) W \(\subseteq U (-W) = W\)
using exists_sym_zerohood
by blast
from w(1) have wg:W\(\subseteq G\) by auto
{
fix z assume z:z \(\in \{(s,t) \in G \times G . t \in U + s + W\}\)
then obtain s t where st:z=(s,t) s \(\in G\) t \(\in U + s + W\) by auto
with \(<W \in \mathbb{N}_0\) st(2) obtain u v where uv:t=u+v u\(\in W+s\) v\(\in W\)
using interval_add(4) lrtrans_in_group_add(2) by auto
from \(<W \subseteq G\) \(<s \in G\) \(<u \in W+s\>) obtain q where q:q\(\in W\) u=q+s using elements_in_rtrans
by blast
from w(2) as(2) q st(2) have u\(\in U+s\) using lrtrans_image(2) by auto
with w(2) uv(1,3) as(2) st(2) have t\(\in U + s + U\) using interval_add(4)
lrtrans_in_group_add(2) by auto
with st have z: z \(\in \{(s,t) \in G \times G . t \in U + s + U\}\) by auto
}
then have
sub: \{(s,t) \in G \times G . t \in U + s + W\} \(\subseteq \{(s,t) \in G \times G . t \in U + s + U\}\)
by auto
{
fix z assume z:z \(\in \{(s,t) \in G \times G . t \in U + s + W\}\)
then obtain s t where st:z=(s,t) s \(\in G\) t \(\in U + s + W\) by auto
with \(<W \in \mathbb{N}_0\) obtain u v where uv:t=u+v u\(\in W+s\) v\(\in W\)
using interval_add(4) lrtrans_in_group_add(2) st(2) by auto
from \(<W \subseteq G\) \(<s \in G\) \(<u \in W+s\>) obtain q where q:q\(\in W\) u=q+s using elements_in_rtrans
by blast
from \(<W \subseteq G\) \(<q \in W\) \(<v \in W\>) have qG vG \(\in G\) by auto
with \(<qG \subseteq G\) \(<v \in G\>) \(<u \in W+s\>) st(2) uv(1) q(2) have t=q+(s+v)
using group_op_closed_add group_op_add_zero group_inv_of_two
with st(2) \(<qG \subseteq G\) \(<v \in G\>) have minust:(-t) = (-v)+(-s)+(-q)
using group_op_closed_add group_op_add_zero group_inv_of_two
from q(1) wg have (-q) \(\in \mathbb{N}_0\) using ginv_image_add(2) by auto
with \( w(3) \) have minusq: \((-q) \in W \) by auto

from \( uv(3) \) \( W \) have \((-v) \in W \) using \( \text{ginv_image_add}(2) \) by auto

with \( w(3) \) have minusv: \((-v) \in W \) by auto

using \( \text{lrtrans_image}(2) \) \( \text{inverse_in_group} \) by auto

with \( \text{minust minusq st(2)} \) \( W \) have \((-t) \in (W + (-s)) + W \)

by auto

moreover

from \( \text{st groupAssum} \) have \( P(z) = \langle \text{GroupInv}(G,f)(s), \text{GroupInv}(G,f)(t) \rangle \)

using \( \text{prodFunctionApp group0_2_T2} \) by blast

with \( \text{st(2,3)} \) have \( P(z) = \langle -s, -t \rangle \) by auto

ultimately have \( P(z) \in \{ (s,t) \in G \times G . t \in W + s + W \} \)

using \( \text{st(2,3)} \) \( \text{inverse_in_group} \) by auto

with \( \text{sub have P(z) \in \{ (s,t) \in G \times G . t \in U + s + U \} by force} \)

with \( \text{as(3) have P(z) \in V by force} \)

with \( z L \) have \( z \in P-(V) \) using \( \text{prodFunction func1_1_L5A vimage_iff} \) by blast

\]

with \( w(1) \) have \( \exists U \in N_0. \{ (s,t) \in G \times G . t \in U + s + U \} \subseteq P-(V) \)

by blast

with \( L \) show \( P-(V) \in \text{roelckeUniformity} \)

unfolding \( \text{roelckeUniformity_def} \) using \( \text{prodFunction func1_1_L6A by blast} \)

qed

with \( L R \) show thesis using \( \text{IsUniformlyCont_def by auto} \)

qed

end

80 Topological groups 2

theory TopologicalGroup_ZF_2 imports Topology_ZF_8 TopologicalGroup_ZF Group_ZF_2

begin

This theory deals with quotient topological groups.

80.1 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 \( \in T \)

defines \( r \equiv \text{QuotientGroupRel}(G,f,H) \)

shows \( \{ \langle b, r(b) \rangle . \langle b, r(b) \rangle \in \langle T(\text{quotient by}) r \rangle \} \)

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}-(∪{b,r{b}). b∈∪T}A)=H+A
proof
{
  fix t assume t∈{(b,r{b}). b∈∪T}-(∪{(b,r{b}). b∈∪T}A)
  then have ∃m∈{(b,r{b}). b∈∪T}A). (t,m)∈{(b,r{b}). b∈∪T} using
image_iff by auto
  then obtain m where (m∈{(b,r{b}). b∈∪T}A) A∈{(b,r{b}). b∈∪T}
  by auto
  then obtain b where b,t ∈ {(b,r{b}). b∈∪T} t∈G 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 _ eqT] 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⟩<A∈T> by auto
  then have (-b)∈G using neg_in_tgroup by auto
  from prd have h=t+(-b) by simp
  with ⟨t,G⟩ ⟨b,G⟩ have t = h+b using inv_cancel_two_add(1) by simp
  then have ⟨(h,b),t⟩∈f using apply_Pair[OF topgroup_f_binop] ⟨h∈G⟩ ⟨b∈G⟩ by auto
  moreover from ⟨h,H⟩ ⟨b,A⟩ have ⟨h,b⟩∈H×A by auto
  ultimately have t∈f(H×A) using image_iff by auto
  with subA subH have t∈H+A using interval_add(2) by auto
}
then show ((∪{b,r{b}). b∈∪T})-(∪{(b,r{b}). b∈∪T}A)⊆H+A by force
{
  fix t assume t∈H+A
  with subA subH have t ∈ f(H×A) using interval_add(2) by auto
  then obtain ha where ha∈H×A ⟨ha,t⟩∈f using image_iff by auto
  then obtain h aa where ha=(h,aa)h∈Haa by auto
  then have h∈Gaa using subH subA by auto
  from ⟨(h,t)⟩∈f have t∈G using topgroup_f_binop unfolding Pi_def
  by auto
  from ⟨ha=(h,aa)⟩ ⟨(h,t)⟩∈f have t=ha using apply_equality topgroup_f_binop
    by auto
  with ⟨h,G⟩ ⟨aa,G⟩ have t+(-aa) = h using inv_cancel_two_add(2)
  by simp
  with ⟨h∈H⟩ ⟨t∈G⟩ ⟨aa∈G⟩ have {t,aa}∈r unfolding r_def QuotientGroupRel_def
  by auto

1181
then have \( r(t) = r(aa) \) using \( \text{eqT} \equiv \text{equiv_class_eq} \) by auto

with \( \langle aa,G \rangle \) have \( \langle aa,r(t) \rangle \in \{(b,r(b)) \mid b \in \bigcup T \} \) by auto

with \( \langle aa,A \rangle \) have \( A_1 : r(t) \in \{(b,r(b)) \mid b \in \bigcup T \} \) using \( \text{image_iff} \) by auto

from \( \langle t \in G \rangle \) have \( \langle t,r(t) \rangle \in \{(b,r(b)) \mid b \in \bigcup T \} \) by auto

ultimately have \( H + A \subseteq \{(b,r(b)) \mid b \in \bigcup T \} \) by auto

A quotient of a topological group is just a quotient group with an appropriate topology that makes product and inverse continuous.

**Theorem (in topgroup)** \( \text{quotient_top_group_F_cont} \):

assumes \( \text{IsAnormalSubgroup}(G,f,H) \)

defines \( r \equiv \text{QuotientGroupRel}(G,f,H) \)

defines \( F \equiv \text{QuotientGroupOp}(G,f,H) \)

shows \( \text{IsContinuous}(\text{ProductTopology}(T\{\text{quotient by} r\},T\{\text{quotient by} r\}),T\{\text{quotient by} r\}) \)

**Proof**

have \( \text{equiv}(\bigcup T,r) \) and \( \text{eqG} : \text{equiv}(G,r) \) using \( \text{group0.Group_ZF.2.4_L3 assms(1) unfolding r_def IsAnormalSubgroup_def} \)

using \( \text{group0_valid_in_tgroup} \) by auto

have \( \text{fun} : \{(b,c),(r(b),r(c)) \mid (b,c) \in \bigcup T \times \bigcup T : G \times G \rightarrow (G//r) \times (G//r) \} \)

using \( \text{product_equiv_rel_fun unfolding G_def by auto} \)

have \( \text{C:Congruent2(r,f)} \) using \( \text{Group_ZF.2.4_L5A(OF Group assms(1)] unfolding r_def.} \)

with \( \text{eqT} \equiv \text{IsContinuous}(\text{ProductTopology}(T,T),\text{ProductTopology}(T\{\text{quotient by} r\},T\{\text{quotient by} r\})) \)

using \( \text{product_quo_fun by auto} \)

have \( \text{tprod:topology0}(\text{ProductTopology}(T,T)) \)

using \( \text{Top_1.1_4.T1(1) [OF topology0_def unfolding topology0_def using Top_1.1_4.T1(1) [OF topSpaceAssum topSpaceAssum].} \)

have \( \text{Hfun : \{(b,c),(r(b),r(c)) \mid (b,c) \in \bigcup T \times \bigcup T \} \) surj \( \text{ProductTopology}(T,T),\bigcup \{(\text{quotient topology in})(\bigcup T//r) \times (\bigcup T//r)\} \)

using \( \text{product_topo_def by auto} \)

1182
using prod_equiv_rel_surj
total_quo_equip[of eqT] topology0.total_quo_func[of tprod prod_equiv_rel_surj]
unfolding F_def QuotientGroupOp_def r_def by auto
have Ffun:⋃(⟨⟨b,c⟩,⟨r{b},r{c}⟩⟩.
⟨b,c⟩∈⋃T×⋃T){by}{⟨⟨b,c⟩,⟨r{b},r{c}⟩⟩}.
⟨b,c⟩∈⋃T×⋃T}{from}(ProductTopology(T,T)))→⋃(T{quotient by}r)
using EquivClass_1_T1[of eqG C] using 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⟩,⟨r{b},r{c}⟩⟩}.
⟨b,c⟩∈⋃T×⋃T):G×G→G//r
using comp_fun[of EquivClass_1_T1[of eqG C]] unfolding F_def QuotientGroupOp_def r_def by auto
then have (F O {⟨⟨b,c⟩,⟨r{b},r{c}⟩⟩}.
⟨b,c⟩∈⋃T×⋃T):⋃(ProductTopology(T,T))→⋃(T{quotient by}r)
using Top_1_4_T1[of topSpaceAssum topSpaceAssum]
total_quo_equi[of eqT] by auto
then have two:two_top_spaces0(ProductTopology(T,T),T{quotient by}r,(F O {⟨⟨b,c⟩,⟨r{b},r{c}⟩⟩}.
⟨b,c⟩∈⋃T×⋃T}))
unfolding two_top_spaces0_def
using Top_1_4_T1[of topSpaceAssum topSpaceAssum] equiv_quo_is_top[of eqT] by auto
have IsContinuous(ProductTopology(T,T),T,f)
using fcon prodtop_def by auto
moreover have IsContinuous(T,T{quotient by}r,⟨b,r{b}⟩. b∈⋃T)
using quotient_func_cont[of quotient_proj_surj]
unfolding equiv_quo_is_top[of eqT] by auto
ultimately have cont:IsContinuous(ProductTopology(T,T),T{quotient by}r,⟨b,r{b}⟩.
b∈⋃T) O f)
using comp_cont by auto
{
  fix A assume A:A∈G×G
  then obtain g1 g2 where A_def:A=(g1,g2) g1∈G g2∈G by auto
  then have f=g1+g2 and p:g1+g2∈⋃T unfolding grop_def using
  apply_type[of topgroup_f_binop] by auto
  then have {⟨b,r{b}⟩. b∈⋃T}⟨fa⟩={⟨b,r{b}⟩. b∈⋃T}{g1+g2} by auto
  with p have {⟨b,r{b}⟩. b∈⋃T}⟨fa⟩=r{g1+g2} using apply_equality[of _ quotient_proj_fun]
  by auto
  then have Pri:{⟨b,r{b}⟩. b∈⋃T} O fa=A=r{g1+g2} using comp_fun_apply[of topgroup_f_binop A] by auto
  from A_def(2,3) have {g1,g2}∈⋃T×⋃T by auto
  then have ⟨⟨g1,g2⟩,⟨r{g1},r{g2}⟩⟩∈{(⟨b,c⟩,⟨r{b},r{c}⟩⟩).
⟨b,c⟩∈⋃T×⋃T}
by auto
  then have {⟨⟨b,c⟩,⟨r{b},r{c}⟩⟩}.
⟨b,c⟩∈⋃T×⋃T}A=A=r{g1},r{g2}) using
A_def(1) apply_equality[of _ product_equiv_rel_fun]
by auto
  then have F(⟨⟨b,c⟩,⟨r{b},r{c}⟩⟩). ⟨b,c⟩∈⋃T×⋃T)A=r{g1},r{g2}) by auto
  then have F(⟨⟨b,c⟩,⟨r{b},r{c}⟩⟩). ⟨b,c⟩∈⋃T×⋃T)A=r{g1+g2}) using
group0.Group_ZF_2_2_L2[of group0_valid_in_tgroup eqG C]
  _ A_def(2,3)) unfolding F_def QuotientGroupOp_def r_def by auto

1183
moreover

note fun ultimately have \( (F \circ O \{\langle b,c \rangle,\langle r\{b\},r\{c\} \rangle \}) = (\{b,r\} \cup \{b,c \}) \) using comp_fun_apply[OF _ A] by auto

then have \( (F \circ O \{\langle b,c \rangle,\langle r\{b\},r\{c\} \rangle \}) = (\{b,r\} \cup \{b,c \}) \) using Pr1 by auto

then have \( (F \circ O \{\langle b,c \rangle,\langle r\{b\},r\{c\} \rangle \}) = (\{b,r\} \cup \{b,c \}) \) 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 \( (F \circ O \{\langle b,c \rangle,\langle r\{b\},r\{c\} \rangle \}) = (\{b,r\} \cup \{b,c \}) \) using cont by auto

have IsAsubgroup(H,f) using assms(1) unfolding IsAnormalSubgroup_def by auto

then have \( \forall A \in T. \{\langle b, r\{b\} \rangle \cdot b \in \bigcup T \} A \in \{\text{quotient by}r\} \) using quotient_map_topgroup_open unfolding r_def by auto

with eqT have ProductTopology(\{quotient by}r\},\{quotient by}r\}) = (\{Qtopology in\}(\bigcup T \mathbin//r) \times (\bigcup T \mathbin//r))\{by\}(\{b,c \},\{r\{b\},r\{c\} \}) \cdot (b,c) \in \bigcup T \times \bigcup T \) from \( \{\text{quotient by}r\} \) using prod_quotient by auto

with A show IsContinuous(ProductTopology(T, T), \{quotient by}r\}, F \circ O \{\langle b,c \rangle,\langle r\{b\},r\{c\} \rangle \}) = (\{b,r\} \cup \{b,c \}) \) 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 \equiv QuotientGroupRel(G,P,H)
and F \equiv QuotientGroupOp(G,P,H)
shows GroupInv(G//r,F):G//r \rightarrow 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 \equiv QuotientGroupRel(G,f,H)
and F \equiv QuotientGroupOp(G,f,H)
shows IsContinuous(T\{quotient by}r\}, T\{quotient by}r\}, GroupInv(G//r,F))

proof-

have eqT: equiv(\bigcup 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 two:two_top_spaces0(T, T\{quotient by}r\}, \{\langle b, r\{b\} \rangle \cdot b \in 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

1184
{fix $g$ assume $G: g \in G$
  then have $\text{GroupInv}(G, f) g = -g$ using $\text{grinv_def}$ by auto
  then have $\{\{b, r(b)\}. b \in G\}(\text{GroupInv}(G, f) g) = \text{GroupInv}(G//r, F)(\{\{b, r(b)\}. b \in G\})$
    using $\text{group0.GroupZF_2_4_L7}$
    $\{\text{OF group0_valid_in_tgroup assms(1) G}\}$ unfolding $\text{r_def F_def}$ by auto
  then have $\{\{b, r(b)\}. b \in G\} \text{GroupInv}(G, f) g = \text{GroupInv}(G//r, F) \{\{b, r(b)\}. b \in G\}$
    using $\text{comp_fun_apply[OF quotient_proj_fun G] comp_fun_apply[OF group0_2_T2[OF Ggroup] G]}$ by auto
  then have $A1: \{\{b, r(b)\}. b \in G\} \text{GroupInv}(G, f) = \text{GroupInv}(G//r, F) \{\{b, r(b)\}. b \in G\}$
    unfolding $\text{r_def F_def IsAnormalSubgroup_def}$
    $\text{quotient_top_group}$ by auto
  then have $A1: \{\{b, r(b)\}. b \in G\} \text{GroupInv}(G, f) = \text{GroupInv}(G//r, F) \{\{b, r(b)\}. b \in G\}$
    using $\text{fun_extension[OF quotient_proj_fun group0.GroupZF_2_4_L8[OF group0_valid_in_tgroup assms(1)]]}$
    $\text{comp_fun[OF group0_2_T2[OF Ggroup] quotient_proj_fun[of Gr]]}$ unfolding $\text{r_def F_def}$
    $\text{quotient_top_group_INV_cont quotient_top_group_F_cont}$
    $\text{group0.GroupZF_2_4_L3 group0_valid_in_tgroup assms(1)}$
    using $\text{total_quo_equi[OF eqT]}$ unfolding $\text{EquivQuo_def[OF eqT]}$ by auto
  ultimately have $\text{IsContinuous}(T, T(\text{quotient by})r, \{\{b, r(b)\}. b \in \bigcup T\}) (\text{GroupInv}(G, f))$
    using $\text{quotient_func_cont[OF quotient_proj_surj]}$
    unfolding $\text{EquivQuo_def[OF eqT]}$ by auto
  with $A1$ have $\text{IsContinuous}(T, T(\text{quotient by})r, \text{GroupInv}(G//r, F) \{\{b, r(b)\}. b \in G\}$
    by auto
  then have $\text{IsContinuous}((\{\text{quotient topology in}\} \bigcup T) // r(\{b, r \{b\} \in \bigcup T\}) \{\text{quotient by})r, T(\text{quotient by})r, \text{GroupInv}(G//r, F))$
    using $\text{two_top_spaces0.cont_quotient_top[OF two_quotient_proj_surj,}$
    $\text{GroupInv}(G//r, F) T(\text{quotient by})r, \text{GroupZF_2_4_L8[OF group0_valid_in_tgroup assms(1)]}$
    using $\text{total_quo_equi[OF eqT]}$ unfolding $\text{r_def F_def}$ by auto
  then show thesis unfolding $\text{EquivQuo_def[OF eqT]}$.
  qed
}

Finally we can prove that quotient groups of topological groups are topological groups.

\text{theorem}(in toppr) \text{quotient_top_group:}$
  assumes $\text{IsAnormalSubgroup}(G, f, H)$
  defines $r \equiv \text{QuotientGroupRel}(G, f, H)$
  defines $F \equiv \text{QuotientGroupOp}(G, f, H)$
  shows $\text{IsAtopologicalGroup}((\{\text{quotient by})r, F)$
    unfolding $\text{IsAtopologicalGroup_def using total_quo_equi equiv_quo_is_top}$
    $\text{GroupZF_2_4_T1 Group assms(1) quotient_top_group_INV_cont quotient_top_group_F_cont}$
    $\text{group0.GroupZF_2_4_L3 group0_valid_in_tgroup assms(1)}$ unfolding $\text{r_def F_def}$
    $\text{IsAnormalSubgroup_def}$
This theory deals with topological properties of subgroups, quotient groups and relations between group theoretical properties and topological properties.

81.1 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_top_spaces0(ProductTopology(T,T),T,f) unfolding two_top_spaces0_def
  - unfolding two_top_spaces0_def
  - using topSpaceAssum Top_1_4_T1(1,3) topgroup_f_binop by auto
  - from fcon have cont:IsContinuous(ProductTopology(T,T),T,f) by auto
  - then have closed:∀ D. D{is closed in}T −→ f-D{is closed in}τ using
    - two_top_spaces0.TopZF_2_1_L1
    - two by auto
  - then have closure:∀ A∈Pow(∪τ). f(Closure(A,τ))⊆cl(fA) using two_top_spaces0.Top_ZF_2_1_L2
    - two by force
  - have sub1:H⊆G using group0.group0_3_L2 group0_valid_in_tgroup assms by force
    - then have sub:(H)×(H)⊆∪τ using prod_top_on_G(2) by auto
    - from sub1 have clHG:cl(H)⊆G using Top_3_L11(1) by auto
    - then have clHsub1:cl(H)×cl(H)⊆G×G by auto
    - have Closure(H×H,ProductTopology(T,T))=cl(H)×cl(H) using cl_product
      - unfolding two_top_spaces0_TopZF_2_1_L1
      - topSpaceAssum group0.group0_3_L2 group0_valid_in_tgroup assms by auto
      - then have f(Closure(H×H,ProductTopology(T,T)))=f(cl(H)×cl(H)) by auto
        - with closure sub have clcl:f(cl(H)×cl(H))⊆cl(f(H×H)) by force
        - from assms have fun:restrict(f,H×H):H×H→H unfolding IsAsubgroup_def
          - using
          - group0.group_oper_fun unfolding group0_def by auto
          - then have restrict(f,H×H)(H×H)=f(H×H) using restrict_image by auto
            - moreover from fun have restrict(f,H×H)(H×H)⊆H using func1_1_L6(2)
              - by blast
              - ultimately have f(H×H)⊆H by auto
with sub1 have \( f(H \times H) \subseteq Hf(H \times H) \subseteq GH \subseteq G \) by auto
then have \( \text{cl}(f(H \times H)) \subseteq \text{cl}(H) \) using top_closure_mono by auto
with \( \text{cl} \) have \( \text{img}: f(\text{cl}(H) \times \text{cl}(H)) \subseteq \text{cl}(H) \) by auto

\[
\{ \text{fix } x \text{ y assume } x \in \text{cl}(H) y \in \text{cl}(H) \}
\]
then have \( (x, y) \in \text{cl}(H) \times \text{cl}(H) \) by auto moreover
have \( f(\text{cl}(H) \times \text{cl}(H)) = \{ f(t) : t \in \text{cl}(H) \times \text{cl}(H) \} \) using func_imagedef topgroup_f_binop

\( \text{cl} \text{Hsub}1 \) by auto ultimately
have \( f(x, y) \in f(\text{cl}(H) \times \text{cl}(H)) \) by auto
with \( \text{img} \) have \( f(x, y) \in \text{cl}(H) \) by auto

\[
\text{Fix } x \text{ y assume } x \in \text{cl}(H) y \in \text{cl}(H) \}
\]
then have \( A_1: \text{cl}(H) \{ \text{is closed under} \} f \) unfolding IsOpClosed_def by auto
have two: two_top_spaces0(T, T, GroupInv(G, f)) unfolding two_top_spaces0_def using
topSpaceAssum Group group0_2_T2 by auto
from inv_cont have cont: IsContinuous(T, T, GroupInv(G, f)) by auto
then have closed: \( \forall D. D \{ \text{is closed in} \} T \longrightarrow \text{GroupInv}(G, f) - D \{ \text{is closed in} \} T \) using two_top_spaces0.TopZF_2_1_L1
two by auto
then have closure: \( \forall A \in \text{Pow}(\bigcup T). \text{GroupInv}(G, f)(\text{cl}(A)) \subseteq \text{cl}(\text{GroupInv}(G, f) A) \) using two_top_spaces0.Top_ZF_2_1_L2
two by force
with sub1 have Inv: \( \text{GroupInv}(G, f)(\text{cl}(H)) \subseteq \text{cl}(\text{GroupInv}(G, f) H) \) by auto
moreover
have \( \text{GroupInv}(H, \text{restrict}(f, H \times H)) : H \rightarrow H \) using assms unfolding IsAsubgroup_def using group0_2_T2 by auto
have \( \text{GroupInv}(H, \text{restrict}(f, H \times H)) H \subseteq H \) using func1_1_L6(2) by auto
then have restrict: \( \text{GroupInv}(G, f), H \subseteq H \) using group0.group0_3_T1 assms
then have sss: \( \text{GroupInv}(G, f) H \subseteq H \) using restrict_image by auto
then have \( H \subseteq G \) GroupInv(G, f) H \subseteq G using sub1 by auto
with sub1 sss have \( \text{cl}(\text{GroupInv}(G, f) H) \subseteq \text{cl}(H) \) using top_closure_mono by auto
ultimately
have \( \text{img}: \text{GroupInv}(G, f)(\text{cl}(H)) \subseteq \text{cl}(H) \) by auto

\[
\{ \text{fix } x \text{ assume } x \in \text{cl}(H) \}
\]
then have \( A_2: \forall x \in \text{cl}(H). \text{GroupInv}(G, f) x \in \text{cl}(H) \) by auto
from assms have \( H \neq 0 \) using group0.group0_3_L5 group0_valid_in_tgroup by auto
moreover
have \( H \subseteq \text{cl}(H) \) using cl_contains_set sub1 by auto ultimately
have \( \text{cl}(H) \neq 0 \) by auto
with \( \text{cl} \text{HG} \) have \( A_2 \) and \( A_1 \) show \( \text{thesis} \) using group0.group0_3_T3 group0_valid_in_tgroup

1187
The closure of a normal subgroup is normal.

**Theorem (in topgroup) normal_subg:**

- **Assumes** IsAnormalSubgroup(G,f,H)
- **Shows** IsAnormalSubgroup(G,f,cl(H))

**Proof:**

- **Have** A:IsASubgroup(cl(H),f) using closure_subgroup assms unfolding IsAnormalSubgroup_def
- **Have** sub1:H G using group0.group0_3_L2 group0_valid_in_tgroup assms unfolding IsAnormalSubgroup_def by auto
- **Then** have sub2:cl(H) G using Top_3_L11(1) by auto

\[
\begin{align*}
\text{fix } & g \text{ assume } g \in G \\
\text{then have } & cl1:cl(g+H)=g+cl(H) \text{ using trans_closure sub1 by auto} \\
\text{have ss:} & g+cl(H) \subseteq G \text{ unfolding ltrans_def LeftTranslation_def by auto} \\
\text{have } & g+H \subseteq G \text{ unfolding ltrans_def LeftTranslation_def by auto} \\
\text{moreover from } & g \text{ have } (-g) \in G \text{ using neg_in_tgroup by auto} \\
\text{ultimately have } & cl2:cl((g+H)+(-g))=cl(g+H)+(-g) \text{ using trans_closure2} \\
\text{by auto} \\
\text{with } & cl1 \text{ have c1con:} cl((g+H)+(-g))=(g+(cl(H)))+(-g) \text{ by auto} \\
\text{fix } & r \text{ assume } r \in (g+H)+(-g) \text{ by auto} \\
\text{then obtain } & q \text{ where } q:q \in G \text{ using assms unfolding IsAnormalSubgroup_def} \\
\text{grinv_def grop_def by auto} \\
\text{then have } & (g+H)+(-g) \subseteq H \text{ by auto} \\
\text{moreover then have } & (g+H)+(-g) \subseteq G \text{ using sub1 by auto} \\
\text{ultimately have } & cl((g+H)+(-g)) \subseteq cl(H) \text{ using top_closure_mono by auto} \\
\text{with } & clcon \text{ have } (g+(cl(H)))+(-g) \subseteq cl(H) \text{ by auto} \\
\text{moreover } \text{fix } & b \text{ assume } b \in \{g+(d-g). d \in cl(H)\} \\
\text{then obtain } & d \text{ where } d:d \in cl(H) =g+(d-g) \text{ by auto moreover} \\
\text{then have } & d \in G \text{ using sub2 by auto} \\
\text{then have } & g+d \in G \text{ using group0.group_op_closed[OF group0_valid_in_tgroup} \\
\text{<g\in G>} \text{ by auto} \\
\text{from } & d(2) \text{ have } b:b=(g+d)-g \text{ using group0.group_oper_assoc[OF group0_valid_in_tgroup} \\
\text{<g\in G>} <d\in G><(-g)\in G> \text{ unfolding grsub_def grop_def grinv_def by blast} \\
\text{have } & (g+d)=LeftTranslation(G,f,g)d \text{ using group0.group0_5_L2(2)[OF} \\
\text{group0_valid_in_tgroup]} \\
\text{<g\in G>-d\in G> by auto} \\
\text{with } & d\in cl(H) \text{ have } g+d \in G+cl(H) \text{ unfolding ltrans_def using func_imagedef[OF} \\
\end{align*}
\]
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 `assms(1)` have `sub:H⊆G` using `group0.group0_valid_in_tgroup` `group0.group0_5_L2(1)` by force

  - Fix `t` assume `t∈G-H` then have `tnH:t∉H` and `tG:t∈G` by auto
    - From `assms(1)` have `sub:H⊆G` using `group0.group0_5_L2(1)` `group0_valid_in_tgroup` by force
      - From `assms(1)` have `nSubG:0∈H` using `group0.group0_valid_in_tgroup` `group0.group0_3_L5` by force
        - From `assms(2)` `tG` have `P:t+H∈T` using `open_tr_open(1)` by auto
          - From `nSubG sub tG` have `tp:t∈t+H` using `group0_valid_in_tgroup` `group0.neut_trans_elem` 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∈Cx∈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∈H>` have `x+(−u)∈H` unfolding `grop_def` using `assms(1)` `group0.group0_valid_in_tgroup` by auto
                        - With `<x+(−u)=t>` have False using `tnH` by auto
                      - Then have `(t+H)∩H=0` by auto moreover
Any subgroup with non-empty interior is open.

**Theorem (in topgroup) clopen_or_emptyInt:**

- **Assumptions:** \( \text{IsAsubgroup}(H,f) \) \( \text{int}(H) \neq 0 \)
- **Shows:** \( H \in T \)

**Proof:**

1. From the assumptions (1) have \( \text{sub}:H \subseteq G \) using group0.group0_3_L2 group0_valid_in_tgroup by force

   \[
   \begin{align*}
   \text{fix } h & \quad \text{assume } h \in H \\
   \text{have } \text{intsub:} \text{int}(H) \subseteq H & \quad \text{using Top_2_L1 by auto} \\
   \text{from assm's (2) obtain } u & \quad \text{where } u \in \text{int}(H) \text{ by auto} \\
   \text{with intsub have } u \in H & \quad \text{by auto} \\
   \text{then have } (-u) \in H & \quad \text{unfolding grinv_def using assm's (1) group0.group0_3_T3A} \\
   \text{group0_valid_in_tgroup} & \quad \text{by auto} \\
   \text{with } <h> & \quad \text{have } h-u \in H \text{ unfolding grop_def using assm's (1) group0.group0_3_L6} \\
   \text{group0_valid_in_tgroup} & \quad \text{by auto} \\
   \end{align*}
   \]

2. Fix \( t \) assume \( t \in (h-u)+(\text{int}(H)) \)

   Then obtain \( r \) where \( r \in \text{int}(H) t=(h-u)+r \) unfolding grsub_def grinv_def grop_def

   \[
   \begin{align*}
   \text{trans_def LeftTranslation_def by auto} \\
   \text{then have } r \in H & \quad \text{using intsub by auto} \\
   \text{with } <h-u> & \quad \text{have } (h-u)+r \in H \text{ unfolding grop_def using assm's (1) group0.group0_3_L6} \\
   \text{group0_valid_in_tgroup} & \quad \text{by auto} \\
   \text{with } <t=(h-u)+r> & \quad \text{have } t \in H \text{ by auto} \\
   \end{align*}
   \]

3. Then have \( ss:(h-u)+(\text{int}(H)) \subseteq H \) by auto

   Have \( P:(h-u)+(\text{int}(H)) \subseteq T \) using open_tr_open(1) \( <h-u> \) Top_2_L2 sub by blast

   From \( <h-u> \in \text{sub have } (h-u) \subseteq G \) \( u \in \text{Gh} \in G \) by auto

   Have \( \text{int}(H) \subseteq G \) using sub intsub by auto moreover

   LeftTranslation\((G,f,(h-u)) \subseteq G \) using group0.group0_5_L1(2) group0_valid_in_tgroup

   \( <(h-u) \subseteq G \)

   By auto ultimately

   Have \( \text{LeftTranslation}(G,f,(h-u))(\text{int}(H)) = \{\text{LeftTranslation}(G,f,(h-u))r. r \in \text{int}(H)\} \)
using func_imagedef by auto moreover
from \((h-u) \in G\) have \(\text{LeftTranslation}(G,f,(h-u))u=(h-u)+u\) using group0.group0_5_L2(2) group0_valid_in_tgroup by auto
with \(u \in \text{int}(H)\) show \((h-u)+u \in \text{LeftTranslation}(G,f,(h-u))r. \ r \in \text{int}(H)\) by force ultimately have \((h-u)+u \in (\text{int}(H))\) unfolding ltrans_def by auto moreover have \((h-u)+u = h\) using group0.inv_cancel_two(1) group0_valid_in_tgroup
with \(u \in \text{int}(H)\) have \((h-u)+u \in (\text{int}(H))\) unfolding Bex_def by auto } then show thesis using open_neigh_open by auto qed

In conclusion, a subgroup is either open or has empty interior.
corollary (in topgroup) emptyInterior_xor_op:
  assumes IsAsubgroup(H,f)
  shows \((\text{int}(H)=0) \text{xor } (H \in T)\)
  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
  { assume H \in T moreover then have \(H\text{is closed in}T\) using open_subgroup_closed assms(1) by auto ultimately have \(H=0\text{V}H=G\) 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 } 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.
thm (in topgroup) loc_compact_T0_closed:
  assumes IsAsubgroup(H,f) \((T\text{restricted to}H)\{is locally-compact\} \ T{is} \ T_0\)
  shows \(H\{is closed in}T\)
proof-

1191
from assms(1) have clsub:IsAsubgroup(cl(H), f) using closure_subgroup by auto
then have subcl: cl(H) ⊆ G using group0.group0_3_L2 group0_valid_in_tgroup by force
from assms(1) have sub: H ⊆ G using group0.group0_3_L2 group0_valid_in_tgroup by force
from assms(3) have T{is T}{restricted to}H using T1_imp_T2 neu_closed_imp_T1 T0_imp_neu_closed by auto
then have sub: H ⊆ G using group0.group0_3_L2 group0_valid_in_tgroup by force
from assms(1) have is T2 using T2_here sub by auto
then have (T{restricted to}H){is compact in} (T{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 ⊆ H K{is compact in} (T{restricted to}H)0 ∈ Interior(K, (T{restricted to}H)) using group0.group0_3_L5 group0_valid_in_tgroup assms(1) unfolding gzero_def by force
K(1,2) have K: K{is closed in} T using compact_subspace_imp_compact by auto
with assms(2) have ∀ x ∈ H. ∃ A ∈ Pow(H). A{is compact in} (T{restricted to}H) ∧ x ∈ Interior(A, (T{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 U where U: U ∈ Interior(K, (T{restricted to}H)) using group0.group0_3_L5 group0_valid_in_tgroup assms(1) unfolding topology0_def by auto
then have U: U ∈ Interior(K, (T{restricted to}H)) = H ∩ U unfolding RestrictedTo_def by auto
then have H ∩ U ⊆ K using topology0.Top_2_L1[of T{restricted to}H] unfolding topology0_def using Top_1_L4 by force
moreover have U2: U ⊆ U ∪ K by auto
have ksub: K ⊆ H using tot K(2) unfolding IsCompact_def by auto
ultimately have int: H ∩ (U ∪ K) = K by auto
from U(2) K(3) have 0 ∈ U by auto
with U(1) U2 have 0 ∈ int(U ∪ K) using Top_2_L6 by auto
then have U ∪ K ∈ N_0 unfolding zerohoods_def using U(1) ksub sub by auto
then obtain V where V: V ⊆ U ∪ K V ∈ N_0 V ∪ V ∪ K(- V) = V using exists_procls_zerohood[of U ∪ K]
by auto

{ fix h assume AS: h ∈ cl(H)
  with clsub have (-h) ∈ cl(H) using group0.group0_3_T3A group0_valid_in_tgroup by auto
  moreover have (-h) ∈ G using subcl by auto
  with V(2) have (-h) ∈ int((-h) + V) using elem_in_int_ltrans by auto
  ultimately have (-h) ∈ cl(H) ∩ (int((-h) + V)) by auto
  have int((-h) + V) ∈ T using Top_2_L2 by auto

1192
note sub ultimately
have \( H \cap (\text{int}(-h)+V)) \neq 0 \) using \( \text{cl\_inter\_neigh} \) by auto moreover
from \((-h) \in G\) \( V(2) \) have \( \text{int}((-h)+V)=(-h)+\text{int}(V) \) unfolding \( \text{zeroshoods\_def} \)
  using \( \text{ltrans\_interior} \) by force
ultimately have \( H \cap ((-h)+\text{int}(V)) \neq 0 \) by auto
then obtain \( y \) where \( y:y \in H \) \( y \in (-h)+\text{int}(V) \) by blast
then obtain \( v \) where \( v:v \in \text{int}(V) \) \( y=(-h)+v \) unfolding \( \text{ltrans\_def} \) \( \text{LeftTranslation\_def} \)
by auto
with \((-h) \in G\) \( V(2) \) \( y(1) \) sub have \( v \in G (-h) \in G y \in G \) using \( \text{Top\_2\_L1[of V]} \) unfolding \( \text{zeroshoods\_def} \) by auto
with \( v(2) \) have \((-h))y=v \) using \( \text{group0.group0\_2\_L18(2) group0\_valid\_in\_tgroup} \)
  unfolding \( \text{grop\_def} \) \( \text{grinv\_def} \) by auto moreover
have \( h \in G \) using \( \text{AS \_subcl} \) by auto
then have \((-h))=h \) using \( \text{group0.group\_inv\_of\_inv group0\_valid\_in\_tgroup} \)
by auto ultimately
have \( h+y=v \) by auto
with \( v(1) \) have \( h+y \in \text{int}(V) \) by auto
have \( y \in \text{cl}(H) \) using \( y(1) \) \( \text{cl\_contains\_set} \) sub by auto
with \( \text{AS} \) have \( \text{hycl\_h+ y} \in \text{cl}(H) \) using \( \text{cl\_sub group0.group0\_3\_L6 group0\_valid\_in\_tgroup} \)
by auto
\{ 
  fix \( W \) assume \( W:W \in \text{Th+y} \in W \)
with \( h+y \in \text{int}(V) \cap W \) by auto moreover
from \( W(1) \) have \( \text{Top\_2\_L2 topSpaceAssum unfolding IsATopology\_def} \) by auto moreover
note \( \text{hycl\_sub} \)
ultimately have \( \text{(int}(V)\cap W) \cap H \neq 0 \) using \( \text{cl\_inter\_neigh[of Hint}(V)\cap W)h+y] \)
by auto
then have \( \text{V} \cap W \cap H \neq 0 \) using \( \text{Top\_2\_L1} \) by auto
with \( V(1) \) have \( (U \cup K) \cap W \cap H \neq 0 \) by auto
then have \( (H \cap (U \cup K)) \cap W \neq 0 \) 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 \( K \cap G \) \( h+y \in G \) using \( \text{kaub sub hycl subcl} \) by auto ultimately
have \( h+y \in \text{cl}(K) \) using \( \text{inter\_neigh\_cl[of Kh+y]} \) unfolding \( G\_def \) by force
then have \( h+y \in K \) using \( \text{Kcl Top\_3\_L8 <K\_G> by auto} \)
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 \( \text{group0.group0\_3\_L6} \)
\( \text{group0\_valid\_in\_tgroup} \)
assms(1) by auto
moreover
have \( (-y) \in G \) using \( <(-y) \in H> \) sub by auto
then have \( h+(y-y) = (h+y) - y \) using \( <y \in G<-h \in G> \) \( \text{group0\_group\_op\_assoc} \)
\( \text{group0\_valid\_in\_tgroup} \) unfolding \( \text{grsub\_def} \) by auto

1193
then have $h+0=(h+y)-y$ using group0.group0_2_L6 group0_valid_in_tgroup 
unfolding gsub_def grinv_def grop_def 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 cl_contains_set sub by auto
then show thesis using Top_3_L8 sub by auto
qed

We can always consider a factor group which is $T_2$.

**Theorem (in topgroup) factor_haus:**

shows $(T\{\text{quotient by}\}\text{QuotientGroupRel}(G,f,\text{cl}(\{0\})))\{\text{is } T_2\}$

**Proof**-

let $r=\text{QuotientGroupRel}(G,f,\text{cl}(\{0\}))$
let $f=\text{QuotientGroupOp}(G,f,\text{cl}(\{0\}))$
let $i=\text{GroupInv}(G//r,f)$

have IsAnormalSubgroup$(G,f,\{0\})$ using group0.trivial_normal_subgroup

Ggroup unfolding group0_def
by auto
then have normal:IsAnormalSubgroup$(G,f,\{0\})$ using normal_subg by auto
then have eq:equiv$(\bigcup T,r)$ using group0.GroupZF_2_4_L3[OF group0_valid_in_tgroup]
unfolding IsAnormalSubgroup_def by auto
then have tot:$\bigcup (T\{\text{quotient by}\}r)=G//r$ using total_quo_equi by auto
have neu:$r(0)=\text{TheNeutralElement}(G//r,f)$ using GroupZF_2_4_L5B[OF Ggroup normal] by auto
then have $r(0)\in G//r$ using group0.group0_2_L2 GroupZF_2_4_T1[OF Ggroup normal]
unfolding group0_valid_in_tgroup by auto
then have sub1:$\{r(0)\}\subseteq G//r$ by auto
then have sub:$\{r(0)\}\subseteq \bigcup (T\{\text{quotient by}\}r)$ using tot by auto
have $zG:0\in \bigcup T$ using group0.group0_2_L2[OF group0_valid_in_tgroup] by auto

from $zG$ have $\text{cla}:r(0)\in G//r$ unfolding quotient_def by auto
let $x=G//r-\{r(0)\}$

\{ fix $s$ assume $A:s\in (G//r-\{r(0)\})$
then obtain $U$ where $s\in U \in G//r-\{r(0)\}$ by auto
then have $U\in G//r$ $U\neq r(0)$ $s\in U$ by auto
then have $s\in G//r$ $s\in U \notin s\notin r(0)$ using cla quotient_disj[OF eq] 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 $s\in U \in G//r$ $s\notin r(0)$ by auto
then have $s\in U \in G//r-\{r(0)\}$ 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

\[
\text{fix } s \text{ assume } \begin{cases} A: s \in r(0) \\ \end{cases} \text{ then have } (0,s) \in r \text{ by auto} \\
\text{then have } (s,0) \in r \text{ using eq unfolding equiv_def sym_def by auto} \]

unfolding QuotientGroupRel_def by auto

moreover 

\[
\text{fix } s \text{ assume } \begin{cases} A: s \in cl(\{0\}) \\ \end{cases} \text{ then have } s \in G \text{ using Top_3_L11(1) zG by auto} \\
\text{then have } (s,0) \in r \text{ using group0.Group_ZF_2_4_L5C[OF group0_valid_in_tgroup]} \]

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 \( cl(\{0\}) \text{ is closed in } T \) 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 eq] by auto

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 \( (T\{\text{quotient by } r\})\{0\} \text{ is } T_1 \) using topgroup.neu_closed_imp_T1[OF topGroupLocale[OF quotient_top_group[OF normal]]] 

total_quo_equi[OF eq] by auto

then show thesis using topgroup.T1_imp_T2[OF topGroupLocale[OF quotient_top_group[OF normal]]] by auto

qed

end

82 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 Is-
arMathLib 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.

82.1 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_theorems.thy, new_deps.thy and new_known_theorems.txt.
The file new_theorems.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 au-
tomatically by Isabelle. However, some manual work has to be done for the
dependencies that Isabelle can not 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.

### 82.2 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.

```plaintext
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 `caddset` that represents the binary function, and the second one called `ca` which defines the $a + b$ notation for `caddset(a,b)`. The same applies to multiplication of real numbers.

Another difficulty is that Metamath allows to define sets with syntax \{x | 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 ∈ 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 can not be imported.

```plaintext
locale MMIsar0 =
  fixes real (R)
  fixes complex (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 \cdot 71)
defines cm\_def: a \cdot b \equiv \langle a,b \rangle
fixes sub (infixl - 69)
defines sub\_def: a - b \equiv \bigcup \{ x \in 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 C. b \cdot x = a \}
fixes cpnf (+\infty)
defines cpnf\_def: +\infty \equiv \{ C \}
fixes cmnf (-\infty)
defines cmnf\_def: -\infty \equiv \{ C \}
fixes cxr (R')
defines cxr\_def: R' \equiv R \cup \{ +\infty, -\infty \}
fixes cxn (N)
defines cxn\_def: N \equiv \bigcap \{ \forall n \in N \rightarrow n+1 \in N \}
fixes less (infix < 68)
defines less\_def: a <_R b \equiv \langle a,b \rangle \in <\_R
fixes cltrrset (<)
defines cltrrset\_def: < \equiv (<_R \cap R \times R) \cup \{ -\infty, +\infty \} \cup (R \times \{ +\infty \}) \cup \{ -\infty \} \times R \}
fixes cltrr (infix < 68)
defines cltrr\_def: a < b \equiv \langle a,b \rangle \in <
fixes convcltrr (infix > 68)
defines convcltrr\_def: a > b \equiv \langle a,b \rangle \in converse(<)
fixes lsq (infix \leq 68)
defines lsq\_def: a \leq b \equiv \neg (b < a)
fixes two (2)
defines two\_def: 2 \equiv 1+1
fixes three (3)
defines three\_def: 3 \equiv 2+1
fixes four (4)
defines four\_def: 4 \equiv 3+1
fixes five (5)
defines five\_def: 5 \equiv 4+1
fixes six (6)
defines six\_def: 6 \equiv 5+1
fixes seven (7)
defines seven\_def: 7 \equiv 6+1
fixes eight (8)
defines eight_def: 8 ≡ 7+1
fixes nine (9)
defines nine_def: 9 ≡ 8+1

assumes MMI_pre_axlttrri:
A ∈ R ∧ B ∈ R → (A < R B ←→ ¬(A=B ∨ B < R A))
assumes MMI_pre_axlttrn:
A ∈ R ∧ B ∈ R ∧ C ∈ R → ((A < R B ∧ B < C) → A < R C)
assumes MMI_pre_axltadd:
A ∈ R ∧ B ∈ R ∧ C ∈ R → (A < R B → C+A < R C+B)
assumes MMI_pre_axmulgt0:
A ∈ R ∧ B ∈ R → (0 < R A ∧ 0 < R B → 0 < R A·B)
assumes MMI_pre_axsup:
A ⊆ R ∧ A ≠ 0 ∧ (∃x∈R. ∀y∈A. y < R x) → (∃x∈R. ∀y∈A. ¬(x < R y)) ∧ (∀y∈R. (y < R x → (∃z∈A. y < R z)))
assumes MMI_axresscn: R ⊆ ℳ
assumes MMI_ax1ne0: 1 ≠ 0
assumes MMI_axcnex: ℳ is Aset
assumes MMI_axaddopr: + : (C × C) → C
assumes MMI_axmulopr: · : (C × C) → C
assumes MMI_axmulcom: A ∈ C ∧ B ∈ C → A·B = B·A
assumes MMI_axaddcl: A ∈ C ∧ B ∈ C → A + B ∈ C
assumes MMI_axmulcl: A ∈ C ∧ B ∈ C → A·B ∈ C
assumes MMI_axdistr:
A ∈ C ∧ B ∈ C ∧ C ∈ C → A·(B + C) = A·B + A·C
assumes MMI_axaddcom: A ∈ C ∧ B ∈ C → A + B = B + A
assumes MMI_axaddass:
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_axire: 1 ∈ R
assumes MMI_axi2m1: 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_axrecrex: A ∈ C ∧ A ≠ 0 → (∃ x ∈ C. A · x = 1)
assumes MMI_axlid: 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_axrnegeq: A ∈ R → (∃ x ∈ R. A + x = 0)
assumes MMI_axrrecex: A ∈ R ∧ A ≠ 0 → (∃ x ∈ R. A · x = 1)

end

83 Logic and sets in Metamatah

theory MMI_logic_and_sets imports MMI_prelude
83.1 Basic Metamath theorems

This section contains Metamath theorems that the more advanced theorems from MMIsar.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 \( \varphi \) and \( \varphi \rightarrow \psi \) shows \( \psi \)
using assms by auto

**lemma** MMI_sseli: assumes A1: \( A \subseteq B \)
shows \( C \in A \rightarrow C \in B \)
using assms by auto

**lemma** MMI_sselii: assumes A1: \( A \subseteq B \) and A2: \( C \in A \)
shows \( C \in B \)
using assms by auto

**lemma** MMI_syl: assumes A1: \( \varphi \rightarrow \psi \) and A2: \( \psi \rightarrow \chi \)
shows \( \varphi \rightarrow \chi \)
using assms by auto

**lemma** MMI_elimhyp: assumes A1: \( A = \text{if} ( \varphi , A , B ) \rightarrow ( \varphi \leftrightarrow \psi ) \) and A2: \( B = \text{if} ( \varphi , A , B ) \rightarrow ( \chi \leftrightarrow \psi ) \) and A3: \( \chi \)
shows \( \psi \)
proof -
{ assume \( \varphi \) with A1 have \( \psi \) by simp }
moreover
{ assume \( \neg \varphi \) with A2 A3 have \( \psi \) by simp }
ultimately show \( \psi \) by auto
qed

**lemma** MMI_neeq1: shows \( A = B \rightarrow ( A \neq C \leftrightarrow B \neq C ) \)
by auto

**lemma** MMI_mp2: assumes A1: \( \varphi \) and A2: \( \psi \) and A3: \( \varphi \rightarrow ( \psi \rightarrow \chi ) \)
saves \( \chi \)
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 MMIisar0) 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

1201
lemma MMI_mp2an: assumes A1: \( \varphi \) and
A2: \( \psi \)
A3: \( ( \varphi \land \psi ) \rightarrow \chi \)
show \( \chi \)
using assms by auto

lemma MMI_mp3an: assumes A1: \( \varphi \) and
A2: \( \psi \)
A3: \( \chi \)
A4: \( ( \varphi \land \psi \land \chi ) \rightarrow \vartheta \)
shows \( \vartheta \)
using assms by auto

lemma MMI_eqeltrr: assumes A1: \( A = B \) and
A2: \( A \in C \)
shows \( B \in C \)
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: \( \varphi \rightarrow \psi \) and
A2: \( \psi \rightarrow \varphi \)
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: \( \chi \) and
A2: \( ( \varphi \land \psi \land \chi ) \rightarrow \vartheta \)
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 \( \varphi \rightarrow ( A = C \leftrightarrow B = D ) \)
using assms by auto

lemma MMI_mpan2: assumes A1: \( \psi \) and
A2: \( ( \varphi \land \psi ) \rightarrow \chi \)
shows \( \varphi \rightarrow \chi \)
using assms by auto
lemma (in MMIsar0) MMI_opreq2:
  shows
  A = B → ( C + A ) = ( C + B )
  A = B → ( C · A ) = ( C · B )
  A = B → ( C - A ) = ( C - B )
  A = B → ( C / A ) = ( C / B )
by auto

lemma MMI_syl5bir: assumes A1: φ → ( ψ ↔ ch ) and
  A2: ϑ → ch
  shows φ → ( ϑ → ψ )
  using assms by auto

lemma MMI_adantr: assumes A1: φ → ψ
  shows ( φ ∧ ch ) → ψ
  using assms by auto

lemma MMI_mpan: assumes A1: φ and
  A2: ( φ ∧ ψ ) → ch
  shows ψ → ch
  using assms by auto

lemma MMI_eqeq1d: assumes A1: φ → A = B
  shows φ → ( A = C ↔ B = C )
  using assms by auto

lemma (in MMIsar0) MMI_opreq1:
  shows
  A = B → ( A · C ) = ( B · C )
  A = B → ( A + C ) = ( B + C )
  A = B → ( A - C ) = ( B - C )
  A = B → ( A / C ) = ( B / C )
by auto

lemma MMI_syl6eq: assumes A1: φ → A = B and
  A2: B = C
  shows φ → A = C
  using assms by auto

lemma MMI_syl6bi: assumes A1: φ → ( ψ ↔ ch ) and
  A2: ch → ϑ
  shows φ → ( ψ → ϑ )
  using assms by auto

lemma MMI_imp: assumes A1: φ → ( ψ → ch )
  shows ( φ ∧ ψ ) → ch
  using assms by auto

lemma MMI_sylibd: assumes A1: φ → ( ψ → ch ) and
A2: $\varphi \rightarrow (\text{ch} \leftrightarrow \vartheta )$
shows $\varphi \rightarrow (\psi \rightarrow \vartheta )$
using asms by auto

lemma MMI_ex: assumes A1: $(\varphi \land \psi ) \rightarrow \text{ch}$
shows $\varphi \rightarrow (\psi \rightarrow \text{ch})$
using asms 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 asms by auto

lemma MMI_bitr: assumes A1: $\varphi \leftrightarrow \psi$ and
A2: $\psi \leftrightarrow \text{ch}$
shows $\varphi \leftrightarrow \text{ch}$
using asms by auto

lemma MMI_eeq12i: assumes A1: $A = B$ and
A2: $C = D$
shows $A = C \leftrightarrow B = D$
using asms 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 asms 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 asms by auto

lemma MMI_eeq1:
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 asms by auto

lemma MMI_eeq2d: assumes A1: $\varphi \rightarrow A = B$
shows $\varphi \rightarrow (C = A \leftrightarrow C = B)$
using asms by auto

lemma MMI_eeq2:
shows $A = B \rightarrow (C = A \leftrightarrow C = B)$
by auto

lemma MMI_elim1: assumes A1: B ∈ C
  shows if ( A ∈ C , A , B ) ∈ C
  using assms by auto

lemma MMI_3adant3: assumes A1: ( φ ∧ ψ ) → ch
  shows ( φ ∧ ψ ∧ ϑ ) → ch
  using assms by auto

lemma MMI_bitr3d: assumes A1: φ → ( ψ ↔ ch ) and
  A2: φ → ( ψ ↔ ϑ )
  shows φ → ( ch ↔ ϑ )
  using assms by auto

lemma MMI_3eqtr3d: assumes A1: φ → A = B and
  A2: φ → A = C and
  A3: φ → B = D
  shows φ → C = D
  using assms by auto

lemma (in MMIIsar0) MMI_opreq1d: assumes A1: φ → A = B
  shows
  φ → ( A + C ) = ( B + C )
  φ → ( A - C ) = ( B - C )
  φ → ( A · C ) = ( B · C )
  φ → ( A / C ) = ( B / C )
  using assms by auto

lemma MMI_3com12: assumes A1: ( φ ∧ ψ ∧ ch ) → ϑ
  shows ( ψ ∧ φ ∧ ch ) → ϑ
  using assms by auto

lemma (in MMIIsar0) MMI_opreq2d: assumes A1: φ → A = B
  shows
  φ → ( C + A ) = ( C + B )
  φ → ( C - A ) = ( C - B )
  φ → ( C · A ) = ( C · B )
  φ → ( C / A ) = ( C / B )
  using assms by auto

lemma MMI_3com23: assumes A1: ( φ ∧ ψ ∧ ch ) → ϑ
  shows ( φ ∧ ch ∧ ψ ) → ϑ
  using assms by auto

lemma MMI_3expa: assumes A1: ( φ ∧ ψ ∧ ch ) → ϑ
  shows ( ( φ ∧ ψ ) ∧ ch ) → ϑ

1205
using assms by auto

lemma MMI_adantrr: assumes A1: \( \varphi \land \psi \) \( \rightarrow \) ch
   shows \( \varphi \land ( \psi \land \vartheta ) \) \( \rightarrow \) 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: \( \varphi \land \psi \land \text{ch} \) \( \rightarrow \) \( \vartheta \)
   shows \( \varphi \land ( \psi \land \text{ch} ) \) \( \rightarrow \) \( \vartheta \)
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: \( \varphi \land \psi \land \text{ch} \) \( \rightarrow \) ch
   shows \( ( \vartheta \land \varphi ) \land ( \tau \land \psi ) \) \( \rightarrow \) ch
using assms by auto

lemma MMI_pm3_2i: assumes A1: \( \varphi \land \psi \land \text{ch} \) \( \rightarrow \) \( \vartheta \land \psi \land \text{ch} \)
   shows \( \varphi \land ( \psi \land \text{ch} ) \) \( \rightarrow \) \( \vartheta \land \psi \land \text{ch} \)
using assms by auto

lemma (in MMIisar0) MMI_opreq2i: assumes A1: A = B
   shows \( ( C + A ) = ( C + B ) \)
            \( ( C - A ) = ( C - B ) \)
            \( ( C \cdot A ) = ( C \cdot B ) \)
using assms by auto

lemma MMI_mpbir2an: assumes A1: \( \varphi \leftrightarrow ( \psi \land \text{ch} ) \) and
            A2: \( \psi \land \text{ch} \)
   shows \( \varphi \land \psi \land \text{ch} \)
using assms by auto

lemma MMI_reu4: assumes A1: \( \forall x \ y. \ x = y \rightarrow ( \varphi(x) \leftrightarrow \psi(y) ) \)
   shows \( \exists ! \ x. \ x \in A \land \varphi(x) \)
            \( ( \exists x \in A. \varphi(x) ) \land ( \forall x \in A. \forall y \in A. \)
            \( ( \varphi(x) \land \psi(y) ) \rightarrow x = y ) \)
using assms by auto
lemma MMI_risset:
  shows A ∈ B ←→ ( ∃ x ∈ B . x = A )
  by auto

lemma MMI_sylib: assumes A1: φ → ψ and
  A2: ψ ←→ ch
  shows φ → ch
  using assms by auto

lemma MMI_mp3an13: assumes A1: φ and
  A2: ch
  A3: ( φ ∧ ψ ∧ ch ) → ϑ
  shows ψ → ϑ
  using assms by auto

lemma MMI_eqcomd: assumes A1: φ → A = B
  shows φ → B = A
  using assms by auto

lemma MMI_sylan9eqr: assumes A1: φ → A = B and
  A2: ψ → B = C
  shows ( ψ ∧ φ ) → A = C
  using assms by auto

lemma MMI_exp32: assumes A1: ( φ ∧ ( ψ ∧ ch ) ) → ϑ
  shows φ → ( ψ → ( ch → ϑ ) )
  using assms by auto

lemma MMI_impcom: assumes A1: φ → ( ψ → ch )
  shows ( ψ ∧ φ ) → ch
  using assms by auto

lemma MMI_a1d: assumes A1: φ → ψ
  shows φ → ( ch → ψ )
  using assms by auto

lemma MMI_r19_21aiv: assumes A1: ∀ x. φ → ( x ∈ A → ψ(x) )
  shows φ → ( ∀ x ∈ A . ψ(x) )
  using assms by auto

lemma MMI_r19_22:
  shows ( ∀ x ∈ A . ( φ(x) → ψ(x) ) ) →
  ( ( ∃ x ∈ A . φ(x) ) → ( ∃ x ∈ A . ψ(x) ) )
  by auto

lemma MMI_syl6: assumes A1: φ → ( ψ → ch ) and
  A2: ch → ϑ
  shows φ → ( ψ → ϑ )
  using assms by auto

1207
lemma MMI_mpid: assumes A1: $\varphi \rightarrow ch$ and 
A2: $\varphi \rightarrow ( \psi \rightarrow ( ch \rightarrow \vartheta ) )$
shows $\varphi \rightarrow ( \psi \rightarrow \vartheta )$
using assms by auto

lemma MMI_eqtr3t: 
shows ( A = C \land B = C ) \rightarrow A = B
by auto

lemma MMI_syl5bi: assumes A1: $\varphi \rightarrow ( \psi \leftrightarrow ch )$ and 
A2: $\vartheta \rightarrow \psi$
shows $\varphi \rightarrow ( \vartheta \rightarrow ch )$
using assms by auto

lemma MMI_mp3an1: assumes A1: $\varphi$ and 
A2: $( \varphi \land \psi \land ch ) \rightarrow \vartheta$
shows $( \psi \land ch ) \rightarrow \vartheta$
using assms by auto

lemma MMI_rgen2: assumes A1: $\forall x y. ( x \in A \land y \in A ) \rightarrow \varphi(x,y)$
shows $\forall x \in A . \forall y \in A . \varphi(x,y)$
using assms by auto

lemma MMI_ax_17: shows $\varphi \rightarrow (\forall x. \varphi)$ 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: ch $\leftrightarrow \varphi$ and 
A3: $\vartheta \leftrightarrow \psi$
shows ch $\rightarrow \vartheta$
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 = if ( \( \varphi \), A , C ) \rightarrow ( ch \leftrightarrow \tau )
  and
  A2: B = if ( \( \psi \), B , D ) \rightarrow ( \tau \leftrightarrow \vartheta )
  and
  A3: \( \tau \)
  shows \( \varphi \land \psi \) \rightarrow ch
  using assms by auto

lemma MMI_eleq1d: assumes A1: \( \varphi \rightarrow A = B \)
  shows \( \varphi \rightarrow ( A \in C \leftarrow\rightarrow B \in C ) \)
  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_eeuni: 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_eeuni by simp

lemma MMI_eqeq1i: assumes A1: A = B
shows $A = C \iff B = C$
using assms by auto

lemma MMI_syl6rbr: assumes $A1: \forall x. \varphi(x) \longrightarrow (\psi(x) \longleftrightarrow ch(x))$ and
$A2: \forall x. \vartheta(x) \longleftrightarrow ch(x)$
shows $\forall x. \varphi(x) \longrightarrow (\vartheta(x) \longleftrightarrow \psi(x))$
using assms by auto

lemma MMI_syl6rbrA: assumes $A1: \varphi \longrightarrow (\psi \longleftrightarrow ch)$ and
$A2: \vartheta \longleftrightarrow ch$
shows $\varphi \longrightarrow (\vartheta \longleftrightarrow \psi)$
using assms by auto

lemma MMI_vtoclga: assumes $A1: \forall x. x = A \longrightarrow (\varphi(x) \longleftrightarrow \psi)$ and
$A2: \forall x. x \in B \longrightarrow \varphi(x)$
shows $A \in B \longrightarrow \psi$
using assms by auto

lemma MMI_3bitr4: assumes $A1: \varphi \longleftrightarrow \psi$ and
$A2: ch \longleftrightarrow \varphi$ and
$A3: \vartheta \longleftrightarrow \psi$
shows $ch \longleftrightarrow \vartheta$
using assms by auto

lemma MMI_mpbiir: assumes $A_{min}: \psi$ and
$A_{maj}: \varphi \longrightarrow (\psi \longleftrightarrow ch)$
shows $\varphi \longrightarrow ch$
using assms by auto

lemma MMI_eqid:
shows $A = A$
by auto

lemma MMI_pm3_27:
shows $(\varphi \land \psi) \longrightarrow \psi$
by auto

lemma MMI_pm3_26:
shows $(\varphi \land \psi) \longrightarrow \varphi$
by auto

lemma MMI_ancoms: assumes $A1: (\varphi \land \psi) \longrightarrow ch$
shows $(\psi \land \varphi) \longrightarrow ch$
using assms by auto
lemma MMI_syl3anc: assumes $A_1: (\varphi \land \psi \land ch) \rightarrow \vartheta$ and
  $A_2: \tau \rightarrow \varphi$ and
  $A_3: \tau \rightarrow \psi$ and
  $A_4: \tau \rightarrow ch$
  shows $\tau \rightarrow \vartheta$
  using assms by auto

lemma MMI_syl5eq: assumes $A_1: \varphi \rightarrow A = B$ and
  $A_2: C = A$
  shows $\varphi \rightarrow C = B$
  using assms by auto

lemma MMI_eqcomi: assumes $A_1: A = B$
  shows $B = A$
  using assms by auto

lemma MMI_3eqtr: assumes $A_1: A = B$ and
  $A_2: B = C$ and
  $A_3: C = D$
  shows $A = D$
  using assms by auto

lemma MMI_mpbir: assumes $A_{\text{min}}: \psi$ and
  $A_{\text{maj}}: \varphi \leftrightarrow \psi$
  shows $\varphi$
  using assms by auto

lemma MMI_syl3an3: assumes $A_1: (\varphi \land \psi \land ch) \rightarrow \vartheta$ and
  $A_2: \tau \rightarrow ch$
  shows $(\varphi \land \psi \land \tau) \rightarrow \vartheta$
  using assms by auto

lemma MMI_3eqtrd: assumes $A_1: \varphi \rightarrow A = B$ and
  $A_2: \varphi \rightarrow B = C$ and
  $A_3: \varphi \rightarrow C = D$
  shows $\varphi \rightarrow A = D$
  using assms by auto

lemma MMI_syl5: assumes $A_1: \varphi \rightarrow (\psi \rightarrow ch)$ and
  $A_2: \vartheta \rightarrow \psi$
  shows $\varphi \rightarrow (\vartheta \rightarrow ch)$
  using assms by auto

lemma MMI_exp3a: assumes $A_1: \varphi \rightarrow ((\psi \land ch) \rightarrow \vartheta)$
  shows $\varphi \rightarrow ((\psi \land ch) \rightarrow ch \rightarrow \vartheta)$
  using assms by auto

lemma MMI_com12: assumes $A_1: \varphi \rightarrow (\psi \rightarrow ch)$
  shows $\psi \rightarrow (\varphi \rightarrow ch)$
lemma MMI_3imp: assumes A1: \( \varphi \rightarrow ( \psi \rightarrow ( \text{ch} \rightarrow \theta ) ) \)
shows \( ( \varphi \land \psi \land \text{ch} ) \rightarrow \theta \)
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 MMI_isar0) MMI_opreq1i: assumes A1: A = B
shows
( A + C ) = ( B + C )
( 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 ( \( \varphi \), A , B ) \( \rightarrow \) ( \( \psi \leftrightarrow \text{ch} \))
and
A2: ch
shows \( \varphi \rightarrow \psi \)
using assms by auto

lemma MMI_id:
shows \( \varphi \rightarrow \varphi \)
by auto

lemma MMI_eqtr3d: assumes A1: \( \varphi \rightarrow A = B \) and
A2: \( \varphi \rightarrow A = C \)
shows \( \varphi \rightarrow B = C \)
using assms by auto

lemma MMI_sylan2: assumes A1: ( \( \varphi \land \psi \) ) \( \rightarrow \) ch and
A2: \( \vartheta \rightarrow \psi \)
shows ( \( \varphi \land \vartheta \) ) \( \rightarrow \) ch
using assms by auto

lemma MMI_adantl: assumes A1: \( \varphi \rightarrow \psi \)
shows ( \( \text{ch} \land \varphi \) ) \( \rightarrow \psi \)
using assms by auto

lemma (in MMIIsar0) MMI_opreq12:
  shows
  \(( \text{A} = \text{B} \land \text{C} = \text{D} ) \rightarrow ( \text{A} + \text{C} ) = ( \text{B} + \text{D} )\)
  \(( \text{A} = \text{B} \land \text{C} = \text{D} ) \rightarrow ( \text{A} - \text{C} ) = ( \text{B} - \text{D} )\)
  \(( \text{A} = \text{B} \land \text{C} = \text{D} ) \rightarrow ( \text{A} \cdot \text{C} ) = ( \text{B} \cdot \text{D} )\)
  \(( \text{A} = \text{B} \land \text{C} = \text{D} ) \rightarrow ( \text{A} / \text{C} ) = ( \text{B} / \text{D} )\)
  by auto

lemma MMI_anidms: assumes A1: \(( \varphi \land \varphi ) \rightarrow \psi\)
  shows \(\varphi \rightarrow \psi\)
  using assms by auto

lemma MMI_anabsan2: assumes A1: \(( \varphi \land ( \psi \land \psi ) ) \rightarrow \text{ch}\)
  shows \(( \varphi \land \psi ) \rightarrow \text{ch}\)
  using assms by auto

lemma MMI_3simp2:
  shows \(( \varphi \land \psi \land \text{ch} ) \rightarrow \psi\)
  by auto

lemma MMI_3simp3:
  shows \(( \varphi \land \psi \land \text{ch} ) \rightarrow \text{ch}\)
  by auto

lemma MMI_sylbir: assumes A1: \(\psi \leftrightarrow \varphi\) and
  A2: \(\psi \rightarrow \text{ch}\)
  shows \(\varphi \rightarrow \text{ch}\)
  using assms by auto

lemma MMI_3eqtr3g: assumes A1: \(\varphi \rightarrow \text{A} = \text{B}\) and
  A2: \(\text{A} = \text{C}\) and
  A3: \(\text{B} = \text{D}\)
  shows \(\varphi \rightarrow \text{C} = \text{D}\)
  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 \( A = B \leftrightarrow B = A \)
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: \( \varphi \rightarrow ( \text{ch} \leftrightarrow \tau ) \)
shows \( \varphi \rightarrow ( \vartheta \leftrightarrow \tau ) \)
using assms by auto

lemma MMI_syl3an2: assumes A1: ( \( \varphi \land \psi \land \text{ch} \) ) \( \rightarrow \vartheta \) and
A2: \( \tau \rightarrow \psi \)
shows ( \( \varphi \land \tau \land \text{ch} \) ) \( \rightarrow \vartheta \)
using assms by auto

lemma MMI_df_rex:
shows ( \( \exists x \in A . \varphi(x) \) ) \( \leftrightarrow \exists x . ( x \in A \land \varphi(x) ) \) \( \) by auto

lemma MMI_mpbi: assumes Amin: \( \varphi \) and
Amaj: \( \varphi \leftrightarrow \psi \)
shows \( \psi \)
using assms by auto

lemma MMI_mp3an12: assumes A1: \( \varphi \) and
A2: \( \psi \) and
A3: ( \( \varphi \land \psi \land \text{ch} \) ) \( \rightarrow \vartheta \)
shows \( \text{ch} \rightarrow \vartheta \)
using assms by auto

lemma MMI_syl5bb: assumes A1: \( \varphi \rightarrow ( \psi \leftrightarrow \text{ch} ) \) and
A2: \( \vartheta \leftrightarrow \psi \)
shows \( \varphi \rightarrow ( \vartheta \leftrightarrow \text{ch} ) \)
using assms by auto

lemma MMI_eleq1a:
shows \( A \in B \leftrightarrow ( C = A \rightarrow C \in B ) \)
by auto
lemma MMI_sylbird: assumes A1: $\varphi \rightarrow (ch \leftrightarrow \psi)$ and
    A2: $\varphi \rightarrow (ch \rightarrow \vartheta)$
    shows $\varphi \rightarrow (\psi \rightarrow \vartheta)$
    using assms by auto

lemma MMI_19_23aiv: assumes A1: $\forall x. \varphi(x) \rightarrow \psi$
    shows $(\exists x. \varphi(x)) \rightarrow \psi$
    using assms by auto

lemma MMI_eqeltrrd: assumes A1: $\varphi \rightarrow A = B$ and
    A2: $\varphi \rightarrow A \in C$
    shows $\varphi \rightarrow B \in C$
    using assms by auto

lemma MMI_syl2an: assumes A1: $(\varphi \land \psi) \rightarrow ch$ and
    A2: $\vartheta \rightarrow \varphi$ and
    A3: $\tau \rightarrow \psi$
    shows $(\vartheta \land \tau) \rightarrow ch$
    using assms by auto

lemma MMI_adantrl1: assumes A1: $(\varphi \land \psi) \rightarrow ch$
    shows $(\varphi \land (\vartheta \land \psi)) \rightarrow ch$
    using assms by auto

lemma MMI_ad2ant2r: assumes A1: $(\varphi \land \psi) \rightarrow ch$
    shows $(\varphi \land \vartheta \land (\psi \land \tau)) \rightarrow ch$
    using assms by auto

lemma MMI_adantll1: assumes A1: $(\varphi \land \psi) \rightarrow ch$
    shows $(\vartheta \land \varphi \land \psi) \rightarrow ch$
    using assms by auto

lemma MMI_anandirs: assumes A1: $(\varphi \land \psi) \land (ch \land \vartheta \land \psi) \rightarrow \tau$
    shows $(\varphi \land \psi \land ch) \rightarrow \tau$
    using assms by auto

lemma MMI_adantlr: assumes A1: $(\varphi \land \psi) \rightarrow ch$
    shows $(\varphi \land \vartheta \land \psi) \rightarrow ch$
    using assms by auto

lemma MMI_an42s: assumes A1: $(\varphi \land \psi) \land (ch \land \vartheta \land \psi) \rightarrow \tau$
    shows $(\varphi \land \psi \land (\vartheta \land \psi)) \rightarrow \tau$
    using assms by auto
lemma MMI_mp3an2: assumes A1: \( \psi \) and
  A2: (\( \varphi \land \psi \land \text{ch} \)) \( \rightarrow \) \( \vartheta \)
shows (\( \varphi \land \text{ch} \)) \( \rightarrow \) \( \vartheta \)
using asms by auto

lemma MMI_3simpl:
  shows (\( \varphi \land \psi \land \text{ch} \)) \( \rightarrow \) \( \varphi \)
by auto

lemma MMI_3impb: assumes A1: (\( \varphi \land (\psi \land \text{ch}) \)) \( \rightarrow \) \( \vartheta \)
  shows (\( \varphi \land \psi \land \text{ch} \)) \( \rightarrow \) \( \vartheta \)
using asms by auto

lemma MMI_mpbird: assumes Amin: \( \varphi \rightarrow \text{ch} \) and
  Amaj: \( \varphi \rightarrow (\psi \leftrightarrow \text{ch}) \)
  shows \( \varphi \rightarrow \psi \)
  using asms by auto

lemma (in MMIIsar0) 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 asms by auto

lemma MMI_3eqtr4: assumes A1: A = B and
  A2: C = A and
  A3: D = B
shows C = D
using asms by auto

lemma MMI_eqtr4d: assumes A1: \( \varphi \rightarrow A = B \) and
  A2: \( \varphi \rightarrow C = B \)
shows \( \varphi \rightarrow A = C \)
using asms by auto

lemma MMI_3eqtr3d: assumes A1: \( \varphi \rightarrow A = B \) and
  A2: \( \varphi \rightarrow A = C \) and
  A3: \( \varphi \rightarrow B = D \)
shows \( \varphi \rightarrow D = C \)
using asms by auto
lemma MMI_sylanc: assumes A1: \((\varphi \land \psi) \rightarrow \text{ch}\) and  
A2: \(\vartheta \rightarrow \varphi\) and  
A3: \(\vartheta \rightarrow \psi\)  
shows \(\vartheta \rightarrow \text{ch}\)  
using assms by auto

lemma MMI_anim12i: assumes A1: \(\varphi \rightarrow \psi\) and  
A2: \(\text{ch} \rightarrow \vartheta\)  
shows \((\varphi \land \text{ch}) \rightarrow (\psi \land \vartheta)\)  
using assms by auto

lemma (in MMIIsar0) MMI_opreqan12d: assumes A1: \(\varphi \rightarrow A = B\) and  
A2: \(\psi \rightarrow C = D\)  
shows  
\[\begin{align*}  
(\varphi \land \psi) & \rightarrow (A + C) = (B + D) 
(\varphi \land \psi) & \rightarrow (A - C) = (B - D) 
(\varphi \land \psi) & \rightarrow (A \cdot C) = (B \cdot D) 
\end{align*}\]  
using assms by auto

lemma MMI_sylanr2: assumes A1: \((\varphi \land (\psi \land \text{ch})) \rightarrow \vartheta\) and  
A2: \(\tau \rightarrow \text{ch}\)  
shows \((\varphi \land (\psi \land \tau)) \land \text{ch}) \rightarrow \vartheta\)  
using assms by auto

lemma MMI_sylan2: assumes A1: \((\varphi \land (\psi \land \text{ch})) \rightarrow \vartheta\) and  
A2: \(\tau \rightarrow \psi\)  
shows \((\varphi \land (\psi \land \tau)) \land \text{ch}) \rightarrow \vartheta\)  
using assms by auto

lemma MMI_ancom2s: assumes A1: \((\varphi \land (\psi \land \text{ch})) \rightarrow \vartheta\)  
shows \((\varphi \land (\psi \land \text{ch})) \rightarrow \vartheta\)  
using assms by auto

lemma MMI_anandis: 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_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: \(A = \text{if} (\varphi, A, B) \rightarrow (\psi \leftarrow \vartheta)\) and  
A2: \(B = \text{if} (\varphi, A, B) \rightarrow (\text{ch} \leftarrow \vartheta)\) 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 \(A = B \rightarrow (A \in C \iff B \in C)\)
by auto

lemma MMI_pm4_2i:
  shows \(\varphi \rightarrow (\psi \iff \psi)\)
by auto

lemma MMI_3anbi123d: assumes A1: \(\varphi \rightarrow (\psi \iff \text{ch})\) and
  A2: \(\varphi \rightarrow (\vartheta \iff \tau)\) and
  A3: \(\varphi \rightarrow (\eta \iff \zeta)\)
shows \(\varphi \rightarrow ((\psi \land \vartheta \land \eta) \iff (\text{ch} \land \tau \land \zeta))\)
using assms by auto

lemma MMI_imbi12d: assumes A1: \(\varphi \rightarrow (\psi \iff \text{ch})\) and
  A2: \(\varphi \rightarrow (\vartheta \iff \tau)\)
shows \(\varphi \rightarrow ((\psi \iff \vartheta) \iff (\text{ch} \iff \tau))\)
using assms by auto

lemma MMI_bitrd: assumes A1: \(\varphi \rightarrow (\psi \iff \text{ch})\) and
  A2: \(\varphi \rightarrow (\text{ch} \iff \vartheta)\)
shows \(\varphi \rightarrow (\psi \iff \vartheta)\)
using assms by auto

lemma MMI_df_ne:
  shows \((A \neq B \iff \neg (A = B))\)
by auto

lemma MMI_3pm3_2i: assumes A1: \(\varphi\) and
  A2: \(\psi\) and
  A3: \(\text{ch}\)
shows \(\varphi \land \psi \land \text{ch}\)
using assms by auto

lemma MMI_eqeq2i: assumes A1: \(A = B\)
shows \(C = A \iff C = B\)
using assms by auto
lemma MMI_syl5bbr: assumes A1: \( \varphi \rightarrow (\psi \leftrightarrow \text{ch}) \) and
A2: \( \psi \leftrightarrow \vartheta \)
shows \( \varphi \rightarrow (\vartheta \leftrightarrow \text{ch}) \)
using assms by auto

lemma MMI_bimpd: assumes A1: \( \varphi \rightarrow (\psi \leftrightarrow \text{ch}) \)
shows \( \varphi \rightarrow (\psi \rightarrow \text{ch}) \)
using assms by auto

lemma MMI_orrd: assumes A1: \( \varphi \rightarrow (\neg(\psi) \rightarrow \text{ch}) \)
shows \( \varphi \rightarrow (\psi \lor \text{ch}) \)
using assms by auto

lemma MMI_jaoi: assumes A1: \( \varphi \rightarrow \psi \) and
A2: \( \text{ch} \rightarrow \psi \)
shows \( (\varphi \lor \text{ch}) \rightarrow \psi \)
using assms by auto

lemma MMI_oridm:
shows \( (\varphi \lor \varphi) \leftrightarrow \varphi \)
by auto

lemma MMI_orbi1d: assumes A1: \( \varphi \rightarrow (\psi \leftrightarrow \text{ch}) \)
shows \( \varphi \rightarrow ((\psi \lor \vartheta) \leftrightarrow (\text{ch} \lor \vartheta)) \)
using assms by auto

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 \( \text{if} ( \varphi , A , B ) \in C \)
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: \( ( \varphi \land \psi ) \rightarrow ( \text{ch} \rightarrow \vartheta ) \)
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 
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 
A2: \( \varphi \rightarrow ( \vartheta \leftrightarrow \text{ch} ) \)
shows $\varphi \rightarrow (\psi \leftrightarrow \vartheta)$
using asms by auto

lemma MMI_neeq1d: assumes A1: $\varphi \rightarrow A = B$
shows $\varphi \rightarrow (A \neq C \leftrightarrow B \neq C)$
using asms by auto

lemma MMI_3anim123i: assumes A1: $\varphi \rightarrow \psi$ and
  A2: $\chi \rightarrow \vartheta$ and
  A3: $\tau \rightarrow \eta$
shows $(\varphi \land \chi \land \tau) \rightarrow (\psi \land \vartheta \land \eta)$
using asms by auto

lemma MMI_3exp: assumes A1: $(\varphi \land \psi \land \chi) \rightarrow \vartheta$
shows $\varphi \rightarrow (\psi \rightarrow (\chi \rightarrow \vartheta))$
using asms by auto

lemma MMI_exp4a: assumes A1: $\varphi \rightarrow (\psi \rightarrow ((\chi \land \vartheta) \rightarrow \tau))$
shows $\varphi \rightarrow (\psi \rightarrow (\chi \rightarrow (\vartheta \rightarrow \tau)))$
using asms by auto

lemma MMI_3imp1: assumes A1: $\varphi \rightarrow (\psi \rightarrow (\chi \rightarrow (\vartheta \rightarrow \tau)))$
shows $(\varphi \land \psi \land \chi) \rightarrow \tau$
using asms by auto

lemma MMI_anim1i: assumes A1: $\varphi \rightarrow \psi$
shows $(\varphi \land \chi) \rightarrow (\psi \land \chi)$
using asms by auto

lemma MMI_3adantl1: assumes A1: $(\varphi \land \psi) \land \chi \rightarrow \vartheta$
shows $(\tau \land \varphi \land \psi) \land \chi \rightarrow \vartheta$
using asms by auto

lemma MMI_3adantl2: assumes A1: $(\varphi \land \psi) \land \chi \rightarrow \vartheta$
shows $(\varphi \land \tau \land \psi) \land \chi \rightarrow \vartheta$
using asms by auto

lemma MMI_3comr: assumes A1: $(\varphi \land \psi \land \chi) \rightarrow \vartheta$
shows $(\chi \land \varphi \land \psi) \rightarrow \vartheta$
using asms by auto

lemma MMI_bitr3: assumes A1: $\psi \leftrightarrow \varphi$ and
  A2: $\psi \leftrightarrow \chi$
shows $\varphi \leftrightarrow \chi$
using assms by auto

lemma MMI_anbi12d: assumes A1: \( \varphi \rightarrow ( \psi \leftrightarrow \text{ch} ) \) and
A2: \( \varphi \rightarrow ( \theta \leftrightarrow \tau ) \)
shows \( \varphi \rightarrow ( ( \psi \land \theta ) \leftrightarrow ( \text{ch} \land \tau ) ) \)
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: \( ( \varphi \land \psi \land \text{ch} ) \rightarrow \theta \) and
A2: \( \tau \rightarrow \varphi \)
shows \( ( \tau \land \psi \land \text{ch} ) \rightarrow \theta \)
using assms by auto

lemma MMI_syl3an1l2: assumes A1: \( ( \varphi \land \psi \land \text{ch} ) \land \theta \) \rightarrow \( \tau \) and
A2: \( \eta \rightarrow \psi \)
shows \( ( ( \varphi \land \eta \land \text{ch} ) \land \theta ) \rightarrow \tau \)
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 \( ( \psi \land \theta \land \varphi ) \rightarrow \text{ch} \)
using assms by auto
lemma MMI_anin2i: assumes A1: \( \varphi \rightarrow \psi \)
    shows ( \( \text{ch} \land \varphi \) ) \( \rightarrow \) ( \( \text{ch} \land \psi \) )
    using assms by auto

lemma MMI_ancom:
    shows ( \( \varphi \land \psi \) ) \( \iff \) ( \( \psi \land \varphi \) )
    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 ( ( \( \varphi \land \psi \) ) \( \land \) ( \( \text{ch} \land \vartheta \) ) ) \( \leftrightarrow \)
    ( ( \( \varphi \land \text{ch} \) ) \( \land \) ( \( \vartheta \land \psi \) ) )
    by auto

lemma MMI_sylanb: assumes A1: ( \( \varphi \land \psi \) ) \( \rightarrow \text{ch} \) and
    A2: \( \vartheta \leftrightarrow \varphi \)
    shows ( \( \vartheta \land \psi \) ) \( \rightarrow \text{ch} \)
    using assms by auto

lemma MMI_an4:
    shows ( ( \( \varphi \land \psi \) ) \( \land \) ( \( \text{ch} \land \vartheta \) ) ) \( \leftrightarrow \)
    ( ( \( \varphi \land \text{ch} \) ) \( \land \) ( \( \vartheta \land \psi \) ) )
    by auto

lemma MMI_syl2anb: assumes A1: ( \( \varphi \land \psi \) ) \( \rightarrow \text{ch} \) and
    A2: \( \vartheta \leftrightarrow \varphi \) and
    A3: \( \tau \leftrightarrow \psi \)
    shows ( \( \vartheta \land \tau \) ) \( \rightarrow \text{ch} \)
    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: \( \varphi \rightarrow ( \psi \leftrightarrow \text{ch} ) \) and
    A2: \( \varphi \rightarrow ( \text{ch} \rightarrow \vartheta ) \)
    shows \( \varphi \rightarrow ( \psi \rightarrow \vartheta ) \)
    using assms by auto

lemma MMI_sylan1i: 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_sylan2b: assumes A1: ( \( \varphi \land \psi \) ) \( \rightarrow \text{ch} \) and
lemma MMI_pm3_22:
shows \( (\varphi \land \psi) \rightarrow (\psi \land \varphi) \)
by auto

lemma MMI_ancli: assumes A1: \( \varphi \rightarrow \psi \)
such that \( \varphi \rightarrow \varphi \)
using asms by auto

lemma MMI_ad2antlr: assumes A1: \( \varphi \rightarrow \psi \)
such that \( \varphi \rightarrow \psi \)
using asms by auto

lemma MMI_bimpa: assumes A1: \( \varphi \rightarrow (\psi \leftrightarrow \text{ch}) \)
such that \( \varphi \rightarrow \psi \)
using asms by auto

lemma MMI_sylan2i: assumes A1: \( \varphi \rightarrow (\psi \wedge \text{ch} \rightarrow \vartheta ) \)
and A2: \( \tau \rightarrow \text{ch} \)
such that \( \varphi \rightarrow (\psi \wedge \tau \rightarrow \vartheta) \)
using asms by auto

lemma MMI_3jca: assumes A1: \( \varphi \rightarrow \psi \) and
A2: \( \varphi \rightarrow \text{ch} \) and
A3: \( \varphi \rightarrow \vartheta \)
such that \( \varphi \rightarrow (\psi \wedge \text{ch} \wedge \vartheta) \)
using asms by auto

lemma MMI_com34: assumes A1: \( \varphi \rightarrow (\psi \rightarrow (\text{ch} \rightarrow (\vartheta \rightarrow \tau)) ) \)
such that \( \varphi \rightarrow (\psi \rightarrow (\vartheta \rightarrow (\text{ch} \rightarrow \tau)) ) \)
using asms by auto

lemma MMI_imp43: assumes A1: \( \varphi \rightarrow (\psi \rightarrow (\text{ch} \rightarrow (\vartheta \rightarrow \tau)) ) \)
such that \( (\varphi \wedge \psi) \wedge (\text{ch} \wedge \vartheta)) \rightarrow \tau \)
using asms by auto

lemma MMI_3anass:
such that \( (\varphi \wedge \psi \wedge \text{ch} ) \leftrightarrow (\varphi \wedge (\psi \wedge \text{ch} )) \)
by auto

lemma MMI_3eqtr4r: assumes A1: \( A = B \) and
A2: \( C = A \) and
A3: D = B
shows D = C
using assms by auto

lemma MMI_jctl: assumes A1: ψ
shows ϕ → ( ψ ∧ ϕ )
using assms by auto

lemma MMI_sylibr: assumes A1: ϕ → ψ and
A2: ch ←→ ψ
shows ϕ → ch
using assms by auto

lemma MMI_mpan1i: assumes A1: ϕ and
A2: ( ( ϕ ∧ ψ ) ∧ ch ) → ϑ
shows ( ψ ∧ ch ) → ϑ
using assms by auto

lemma MMI_alli: assumes A1: ϕ
shows ϕ → ϕ
using assms by auto

lemma (in MMI_isar0) MMI_opreqan12rd: assumes A1: ϕ → A = B and
A2: ψ → C = D
shows
( ψ ∧ ϕ ) → ( A + C ) = ( B + D )
( ψ ∧ ϕ ) → ( A · C ) = ( B · 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_eleq1i: assumes A1: A = B
shows A ∈ C ↔ B ∈ C
using assms by auto

1225
lemma MMI_mtbir: assumes A1: \( \neg (\psi) \) and
A2: \( \varphi \leftrightarrow \psi \)
shows \( \neg (\varphi) \)
using assms by auto

lemma MMI_mto: assumes A1: \( \neg (\psi) \) and
A2: \( \varphi \rightarrow \psi \)
shows \( \neg (\varphi) \)
using assms by auto

lemma MMI_df_nel:
shows \( (A \notin B \leftrightarrow \neg (A \in B)) \)
by auto

lemma MMI_snid: assumes A1: A isASet
shows A \in \{A\}
using assms by auto

lemma MMI_en21p:
shows \( \neg (A \in B \land B \in A) \)
proof
assume A1: A \in B \land B \in A
then have A \in B by simp
moreover
\{ assume \( \neg (A \in B \land B \in A) \) 
then have B \in 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 \( (\varphi \rightarrow \neg (\psi)) \leftrightarrow \neg ((\varphi \land \psi)) \)
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: ¬ ( ch ) and
    Amaj: ϕ → ( ψ ↔ ch )
shows ϕ → ¬ ( ψ )
using assms by auto

lemma MMI_con2i: assumes Aa: ϕ → ¬ ( ψ )
shows ψ → ¬ ( ϕ )
using assms by auto

lemma MMI_intnand: assumes A1: ϕ → ¬ ( ψ )
shows ϕ → ¬ ( ( ch ∧ ψ ) )
using assms by auto

lemma MMI_intnanrd: assumes A1: ϕ → ¬ ( ψ )
shows ϕ → ¬ ( ( ψ ∧ ch ) )
using assms by auto

lemma MMI_biorf:
    shows ¬ ( ϕ ) → ( ψ ↔ ( ϕ ∨ ψ ) )
by auto

lemma MMI_bitr2d: assumes A1: ϕ → ( ψ ↔ ch ) and
    A2: ϕ → ( ch ↔ ϑ )
shows ϕ → ( ϑ ↔ ψ )
using assms by auto

lemma MMI_orass:
    shows ( ( ϕ ∨ ψ ) ∨ ch ) ↔ ( ϕ ∨ ( ψ ∨ ch ) )
by auto

lemma MMI_orcom:
    shows ( ϕ ∨ ψ ) ↔ ( ψ ∨ ϕ )
by auto

lemma MMI_3bitr4d: assumes A1: ϕ → ( ψ ↔ ch ) and
    A2: ϕ → ( ϑ ↔ ψ ) and
    A3: ϕ → ( τ ↔ ch )
shows ϕ → ( ϑ → τ )
using assms by auto

lemma MMI_3imtr4d: assumes A1: ϕ → ( ψ → ch ) and
    A2: ϕ → ( ϑ → ψ ) and
    A3: ϕ → ( τ → ch )
shows ϕ → ( ϑ → τ )
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} ) and  
\begin{align*}
A2: \vartheta \rightarrow ( \tau \leftrightarrow \eta ) 
\end{align*}
shows ( \( \varphi \land \vartheta \) ) \rightarrow ( ( \psi \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} ) ) \quad 
shows \( \varphi \rightarrow ( \text{ch} \leftrightarrow \neg ( \psi ) ) \)  
using assms by auto

lemma MMI_so: assumes  
\begin{align*}
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 ) 
\end{align*}
shows R Orders A  
using assms StrictOrder_def by auto

lemma MMI_con1bid: assumes A1: \varphi \rightarrow ( \neg ( \psi ) \leftrightarrow \text{ch} ) \quad 
shows \( \varphi \rightarrow ( \neg ( \text{ch} ) \leftrightarrow \psi ) \) 

1228
using assms by auto

lemma MMI_sotrieq:
  shows ((R Orders A) ∧ ( B ∈ A ∧ C ∈ A )) →
  ( B = C ←→ ¬ ( ⟨B,C⟩ ∈ R ∨ ⟨C, B⟩ ∈ R ))
proof -
  { assume A1: R Orders A and A2: B ∈ A ∧ C ∈ A
    from A1 have ∀ x y z. (x ∈ A ∧ y ∈ A ∧ z ∈ A) →
      ⟨(x,y) ∈ R ↔ ¬(x=y ∨ (y,x) ∈ R)⟩ ∧
      ⟨(x,y) ∈ R ∧ ⟨y,z⟩ ∈ R → ⟨x,z⟩ ∈ R⟩
      by (unfold StrictOrder_def)
    then have ∀ x y. x ∈ A ∧ y ∈ A →
      ⟨(x,y) ∈ R ↔ ¬(x=y ∨ ⟨y,x⟩ ∈ R)⟩ ∧
      ⟨(x,y) ∈ R ∧ ⟨y,z⟩ ∈ R → ⟨x,z⟩ ∈ R⟩
      by auto
    with A2 have I: ⟨B,C⟩ ∈ R ↔ ¬(B=C ∨ ⟨C,B⟩ ∈ R)
      by blast
    then have B = C ↔ ¬ ( ⟨B,C⟩ ∈ R ∨ (C, B) ∈ R )
      by auto
  } then show ((R Orders A) ∧ ( B ∈ A ∧ C ∈ A )) →
  ( B = C ←→ ¬ ( ⟨B,C⟩ ∈ R ∨ ⟨C, B⟩ ∈ R )) by simp
qed

lemma MMI_bicomd: assumes A1: ϕ → ( ψ ↔ ch )
  shows ϕ → ( ch ↔ ψ )
  using assms by auto

lemma MMI_sotrieq2:
  shows ((R Orders A) ∧ ( B ∈ A ∧ C ∈ A )) →
  ( B = C ←→ ¬ ( ⟨B,C⟩ ∈ R ) ∧ ¬ ( ⟨C, B⟩ ∈ R ))
  using MMI_sotrieq by auto

lemma MMI_orc:
  shows ϕ → ( ϕ ∨ ψ )
  by auto

lemma MMI_syl6bbr: assumes A1: ϕ → ( ψ ↔ ch ) and
  A2: ϑ ↔ ch
  shows ϕ → ( ψ ↔ ϑ )
  using assms by auto

lemma MMI_orbii1: assumes A1: ϕ ↔ ψ
  shows ( ϕ ∨ ch ) ↔ ( ψ ∨ ch )
  using assms by auto

lemma MMI_syl5rbbr: assumes A1: ϕ → ( ψ ↔ ch ) and
  A2: ψ ↔ ϑ
  shows ϕ → ( ch ↔ ϑ )

1229
using assms by auto

lemma MMI_anbi2d: assumes A1: $\varphi \rightarrow (\psi \leftrightarrow \text{ch})$
  shows $\varphi \rightarrow ((\vartheta \land \psi) \leftrightarrow (\vartheta \land \text{ch}))$
  using assms by auto

lemma MMI_ord: assumes A1: $\varphi \rightarrow (\psi \lor \text{ch})$
  shows $\varphi \rightarrow (\neg (\psi) \rightarrow \text{ch})$
  using assms by auto

lemma MMI_impbid: assumes A1: $\varphi \rightarrow (\psi \rightarrow \text{ch})$
  A2: $\varphi \rightarrow (\text{ch} \rightarrow \vartheta)$
  shows $\varphi \rightarrow (\psi \rightarrow (\text{ch} \land \vartheta))$
  using assms by blast

lemma MMI_jcad: assumes A1: $\varphi \rightarrow (\psi \rightarrow \text{ch})$
  A2: $\varphi \rightarrow (\psi \rightarrow \vartheta)$
  shows $\varphi \rightarrow (\psi \rightarrow (\text{ch} \land \vartheta))$
  using assms by auto

lemma MMI_ax_1: shows $\varphi \rightarrow (\psi \rightarrow \varphi)$
  by auto

lemma MMI_pm2_24: shows $\varphi \rightarrow (\neg (\varphi) \rightarrow \psi)$
  by auto

lemma MMI_imp3a: assumes A1: $\varphi \rightarrow (\psi \rightarrow (\text{ch} \rightarrow \vartheta))$
  shows $\varphi \rightarrow ((\psi \land \text{ch}) \rightarrow \vartheta)$
  using assms by auto

lemma (in MMIasar0) 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_bimprd: assumes A1: $\varphi \rightarrow (\psi \leftrightarrow \text{ch})$
  shows $\varphi \rightarrow (\text{ch} \rightarrow \psi)$
  using assms by auto

lemma MMI_jaod: assumes A1: $\varphi \rightarrow (\psi \rightarrow \text{ch})$
  A2: $\varphi \rightarrow (\vartheta \rightarrow \text{ch})$
  shows $\varphi \rightarrow ((\psi \lor \vartheta) \rightarrow \text{ch})$
  using assms by auto

lemma MMI_com23: assumes A1: $\varphi \rightarrow (\psi \rightarrow (\text{ch} \rightarrow \vartheta))$
  shows $\varphi \rightarrow (\text{ch} \rightarrow (\psi \rightarrow \vartheta))$

1230
lemma (in MMIsar0) MMI_breq2:
  shows
  A = B  →  ( C ≤ A ↔ C ≤ B )
  A = B  →  ( C < A ↔ C < B )
  by auto

lemma MMI_sylD: assumes A1: φ  →  ( ψ  →  ch ) and
         A2: φ  →  ( ch  →  θ )
  shows φ  →  ( ψ  →  θ )
  using assms by auto

lemma MMI_bimpC: assumes A1: φ  →  ( ψ  ↔  ch )
  shows ψ  →  ( φ  →  ch )
  using assms by auto

lemma MMI_mp2AND: assumes A1: φ  →  ψ and
         A2: φ  →  ch and
         A3: φ  →  ( ( ψ ∧ ch )  →  θ )
  shows φ  →  θ
  using assms by auto

lemma MMI_SOnr:
  shows ( R Orders A ∧ B ∈ A )  →  ¬ ( (B,B) ∈ R )
  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 MMIsar0) MMI_pnFxr:
  shows +∞ ∈ R^∗
  using cxr_def by simp

1231
lemma MMI_elisseti: assumes A1: A ∈ B
    shows A isASet
    using assms by auto

lemma (in MMIIsar0) MMI_mmfxr:
    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 → ( C ∈ A ↔ C ∈ B )
    by auto

lemma MMI_nelneq:
    shows ( A ∈ C ∧ ¬ ( B ∈ C ) ) → ¬ ( A = B )
    by auto

lemma MMI_df_pr:
    shows { A , B } = ( { A } ∪ { B } )
    by auto

lemma MMI_ineq2i: assumes A1: A = B
                shows ( C ∩ A ) = ( C ∩ B )
                using assms by auto

lemma MMI_mt2: assumes A1: ψ and
         A2: φ → ¬ ( ψ )
    shows ¬ ( φ )
    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 \(( ( A \cap B ) = 0 \longrightarrow ( A \subseteq ( B \cup C ) \iff A \subseteq C ) )\)
  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 \( A1: \varphi \longrightarrow ( \psi \longrightarrow ch ) \) and 
  \( A2: \psi \iff \vartheta \)
  shows \( \varphi \longrightarrow ( \vartheta \longrightarrow ch ) \)
  using assms by auto

lemma MMI_con3d: assumes \( A1: \varphi \longrightarrow ( \psi \longrightarrow ch ) \)
  shows \( \varphi \longrightarrow ( \neg ( ch ) \longrightarrow \neg ( \psi ) ) \)
  using assms by auto

lemma MMI_dfrex2:
  shows \(( \exists x \in A . \varphi(x) ) \iff \neg ( \forall x \in A . \neg \varphi(x) )\)
  by auto

lemma MMI_visset:
  shows \( x \in ASet\)
  by auto

lemma MMI_elpr: assumes \( A1: A \in \{ B , C \} \)
  shows \( ( A = B \vee A = C )\)
using assms by auto

lemma MMI_rexbii: assumes A1: \( \forall x. \varphi(x) \iff \psi(x) \)  
shows ( \( \exists x \in A. \varphi(x) \) ) \iff ( \( \exists x \in A. \psi(x) \) )  
using assms by auto

lemma MMI_r19_43:  
shows ( \( \exists x \in A. (\varphi(x) \or \psi(x)) \) ) \iff ( ( \( \exists x \in A. \varphi(x) \) \or ( \( \exists x \in A. \psi(x) \) ) )  
by auto

lemma MMI_exancom:  
shows ( \( \exists x. (\varphi(x) \and \psi(x)) \) ) \iff ( \( \exists x. (\psi(x) \and \varphi(x)) \) )  
by auto

lemma MMI_ceqsexv: assumes A1: A isASet and  
A2: \( \forall x. x = A \iff (\varphi(x) \iff \psi(x)) \)  
shows ( \( \exists x. (x = A \and \varphi(x)) \) ) \iff \( \psi(A) \)  
using assms by auto

lemma MMI_orbi12i_orig: assumes A1: \( \varphi \iff \psi \) and  
A2: \( ch \iff \vartheta \)  
shows ( \( \exists x. \varphi(x) \or ch \) ) \iff ( \( \psi \or \vartheta \) )  
using assms by auto

lemma MMI_orbi12i: assumes A1: (\( \exists x. \varphi(x) \) \iff \( \psi \)) and  
A2: (\( \exists x. ch(x) \) \iff \( \vartheta \))  
shows ( \( \exists x. \varphi(x) \) \or (\( \exists x. ch(x) \) ) \iff ( \( \psi \or \vartheta \) )  
using assms by auto

lemma MMI_syl6ib: assumes A1: \( \varphi \rightarrow (\psi \rightarrow ch) \) and  
A2: \( ch \leftrightarrow \vartheta \)  
shows \( \varphi \rightarrow (\psi \rightarrow \vartheta) \)  
using assms by auto

lemma MMI_intnan: assumes A1: \( \neg (\varphi) \)  
shows \( \neg (\psi \and \varphi) \)  
using assms by auto

lemma MMI_intnanr: assumes A1: \( \neg (\varphi) \)  
shows \( \neg (\psi \and \varphi) \)  
using assms by auto

lemma MMI_pm3_2ni: assumes A1: \( \neg (\varphi) \) and  
A2: \( \neg (\psi) \)  
shows \( \neg (\varphi \or \psi) \)  
using assms by auto
lemma (in MMIsar0) MMI_breq12:
  shows
  ( A = B ∧ C = D ) −→ ( A < C ↔ B < D )
  ( A = B ∧ C = D ) −→ ( A ≤ C ↔ B ≤ D )
by auto

lemma MMI_necom:
  shows A ≠ B ↔ B ≠ A
by auto

lemma MMI_3jaoi: assumes A1: φ → ψ and
  A2: ch → ψ and
  A3: θ → ψ
  shows ( φ ∨ ch ∨ θ ) → ψ
  using assms by auto

lemma MMI_jctr: assumes A1: ψ
  shows φ → ( φ ∧ ψ )
  using assms by auto

lemma MMI_olc:
  shows φ → ( ψ ∨ φ )
by auto

lemma MMI_3syl: assumes A1: φ → ψ and
  A2: ψ → ch and
  A3: ch → θ
  shows φ → θ
  using assms by auto

lemma MMI_mtbird: assumes Amin: φ → ¬ ( ch ) and
  Amaj: φ → ( ψ ↔ ch )
  shows φ → ¬ ( ψ )
  using assms by auto

lemma MMI_pm2_21d: assumes A1: φ → ¬ ( ψ )
  shows φ → ( ψ → ch )
  using assms by auto

lemma MMI_3jaodan: assumes A1: ( φ ∧ ψ ) → ch and
  A2: ( φ ∧ θ ) → ch and
  A3: ( φ ∧ τ ) → ch
  shows ( φ ∧ ( ψ ∨ θ ∨ τ ) ) → ch
  using assms by auto

lemma MMI_sylan2br: assumes A1: ( φ ∧ ψ ) → ch and
  A2: ψ ↔ θ
shows \((\varphi \land \psi) \rightarrow \text{ch}\)
using assms by auto

**lemma MML_3jaoin:** assumes \(A1: (\varphi \land \psi) \rightarrow \text{ch}\) and
\(A2: (\theta \land \psi) \rightarrow \text{ch}\) and
\(A3: (\tau \land \psi) \rightarrow \text{ch}\)
shows \(((\varphi \lor \theta \lor \tau) \land \psi) \rightarrow \text{ch}\)
using assms by auto

**lemma MML_mtbid:** assumes \(A\text{min}: \varphi \rightarrow (\neg (\psi) \rightarrow \text{ch})\) and
\(A\text{maj}: \varphi \rightarrow (\psi \leftrightarrow \text{ch})\)
shows \(\varphi \rightarrow (\neg (\text{ch}) \rightarrow \psi)\)
using assms by auto

**lemma MML_conld:** assumes \(A1: \varphi \rightarrow (\neg (\psi) \rightarrow \text{ch})\)
shows \(\varphi \rightarrow ((\neg (\text{ch}) \rightarrow \psi)\)
using assms by auto

**lemma MML_pm2_21nd:** assumes \(A1: \varphi \rightarrow \psi\)
shows \(\varphi \rightarrow ((\neg (\psi) \rightarrow \text{ch})\)
using assms by auto

**lemma MML_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

**lemma MML_adantld:** assumes \(A1: \varphi \rightarrow (\psi \rightarrow \text{ch})\)
shows \(\varphi \rightarrow ((\theta \land \psi) \rightarrow \text{ch})\)
using assms by auto

**lemma MML_adantrd:** assumes \(A1: \varphi \rightarrow (\psi \rightarrow \text{ch})\)
shows \(\varphi \rightarrow ((\psi \land \vartheta) \rightarrow \text{ch})\)
using assms by auto

**lemma MML_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 MML_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 MML_mpbid:** assumes \(A\text{min}: \varphi \rightarrow \psi\) and
\(A\text{maj}: \varphi \rightarrow (\psi \leftrightarrow \text{ch})\)
shows \( \varphi \rightarrow \chi \)
using assms by auto

**lemma MMI_orbi12d:** assumes A1: \( \varphi \rightarrow (\psi \leftrightarrow \chi) \) and
A2: \( \varphi \rightarrow (\vartheta \leftrightarrow \tau) \)
shows \( \varphi \rightarrow ((\psi \vee \vartheta) \leftrightarrow (\chi \vee \tau)) \)
using assms by auto

**lemma MMI_ianor:**
\[
\neg(\varphi \land \psi) \leftrightarrow \neg\varphi \lor \neg\psi
\]
by auto

**lemma MMI_bitr2:** assumes A1: \( \varphi \leftrightarrow \psi \) and
A2: \( \psi \leftrightarrow \chi \)
shows \( \chi \leftrightarrow \varphi \)
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: \( \varphi \rightarrow \chi \) and
A2: \( \varphi \rightarrow ((\psi \land \chi) \rightarrow \vartheta) \)
shows \( \varphi \rightarrow (\psi \rightarrow \vartheta) \)
using assms by auto

**lemma MMI_ad2antrr:** assumes A1: \( \varphi \rightarrow \psi \)
shows (\( (\varphi \land \chi) \land \vartheta \)) \rightarrow \psi
using assms by auto

**lemma MMI_bimpac:** assumes A1: \( \varphi \rightarrow (\psi \leftrightarrow \chi) \)
shows \( (\psi \land \varphi) \rightarrow \chi \)
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: \( \varphi \leftrightarrow (\psi \land \chi) \)
shows \( \varphi \rightarrow \psi \)
using assms by auto

**lemma MMI_bimpl:** assumes A1: \( \varphi \leftrightarrow \psi \)
shows \( \psi \rightarrow \varphi \)
using assms by auto
lemma (in MMIas0) 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 MMIas0) MMI_breq12i: assumes A1: A = B and 
A2: C = D 
shows 
A < C \iff B < D 
A \leq C \iff B \leq D 
using assms by auto 

lemma MMI_negbii: assumes Aa: \( \varphi \iff \psi \) 
shows \( \neg \varphi \iff \neg \psi \) 
using assms by auto 

lemma (in MMIas0) MMI_breq1i: assumes A1: A = B 
shows 
A < C \iff B < C 
A \leq C \iff 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 MMIas0) MMI_breq2d: assumes A1: \( \varphi \rightarrow A = B \) 
shows 
\( \varphi \rightarrow C < A \iff C < B \) 
\( \varphi \rightarrow C \leq A \iff C \leq B \) 
using assms by auto 

lemma MMI_ccase: assumes A1: \( \varphi \land \psi \rightarrow \tau \) and 
A2: \( \chi \land \psi \rightarrow \tau \) and 
A3: \( \varphi \land \vartheta \rightarrow \tau \) and 
A4: \( \chi \land \vartheta \rightarrow \tau \) 
shows \( (\varphi \lor \chi) \land (\psi \lor \vartheta) \rightarrow \tau \) 
using assms by auto 

lemma MMI_pm3_27bd: assumes A1: \( \varphi \iff \psi \land \chi \) 
shows \( \varphi \rightarrow \chi \) 
using assms by auto
lemma MMI_nsyl3: 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_jctred: assumes A1: $\varphi \land \psi$ $\rightarrow$ $\tau$ and 
A2: ch $\rightarrow$ $\tau$ and 
A3: $\vartheta$ $\rightarrow$ $\tau$
shows $(\varphi \lor \text{ch}) \land (\psi \lor \vartheta) \rightarrow \tau$
using assms by auto

lemma MMI_3bitr3r: assumes A1: $\varphi \leftrightarrow \psi$ and 
A2: $\varphi$ $\leftrightarrow$ ch and 
A3: $\psi$ $\leftrightarrow$ $\vartheta$
shows $\vartheta$ $\leftrightarrow$ 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_61i: 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_61d: assumes A1: \( \varphi \rightarrow \psi \rightarrow \text{ch} \) and 
A2: \( \varphi \rightarrow \neg \psi \rightarrow \text{ch} \)
shows \( \varphi \rightarrow \text{ch} \)
using asms 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 asms 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 asms 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 asms by auto

lemma MMI_bibi2d: assumes A1: \( \varphi \rightarrow \psi \leftrightarrow \text{ch} \)
shows \( \varphi \rightarrow (\vartheta \leftrightarrow \psi) \leftrightarrow \vartheta \leftrightarrow \text{ch} \)
using asms by auto

lemma MMI_con4bid: assumes A1: \( \varphi \rightarrow \neg \psi \leftrightarrow \neg \text{ch} \)
shows \( \varphi \rightarrow \psi \rightarrow \text{ch} \)
using asms 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 asms by auto

lemma MMI_3bitr3rd: assumes A1: \( \varphi \rightarrow \psi \leftrightarrow \text{ch} \) and 
A2: \( \varphi \rightarrow \psi \leftrightarrow \vartheta \) and 
A3: \( \varphi \rightarrow \)
lemma MMI_3imtr4g: assumes A1: \( \varphi \rightarrow \psi \rightarrow ch \) and
A2: \( \vartheta \leftrightarrow \psi \) and
A3: \( \tau \leftrightarrow ch \)
shows \( \varphi \rightarrow \)
\( \vartheta \rightarrow \tau \)
using assms by auto

lemma MMI_expcom: assumes A1: \( \varphi \land \psi \rightarrow ch \)
shows \( \psi \rightarrow \varphi \rightarrow ch \)
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: \( ch \leftrightarrow \psi \) and
A3: \( ch \leftrightarrow \vartheta \)
shows \( \vartheta \leftrightarrow \varphi \)
using assms by auto

lemma MMI_dedth4h: assumes A1: A = \( if(\varphi, A, R) \rightarrow \)
\( \tau \leftrightarrow \eta \) and
A2: B = \( if(\psi, B, S) \rightarrow \)
\( \eta \leftrightarrow \zeta \) and
A3: C = \( if(ch, C, F) \rightarrow \)
\( \zeta \leftrightarrow si \) and
A4: D = \( if(\vartheta, D, G) \rightarrow si \leftrightarrow rh \) and
A5: rh
shows \( (\varphi \land \psi) \land ch \land \vartheta \rightarrow \tau \)
using assms by auto

lemma MMI_anbi1d: assumes A1: \( \varphi \rightarrow \)
\( \psi \leftrightarrow ch \)
sends \( \varphi \rightarrow \)
\( \psi \land \vartheta \leftrightarrow \psi \land \vartheta \)
using assms by auto
lemma (in MMIsar0) 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: \( \varphi \land \psi \land \text{ch} \rightarrow \theta \) and
  A2: \( \tau \rightarrow \varphi \) and
  A3: \( \eta \rightarrow \psi \) and
  A4: \( \zeta \rightarrow \text{ch} \)
shows \( \tau \land \eta \land \zeta \rightarrow \theta \)
using assms by auto

lemma MMI_3bitrd: assumes A1: \( \varphi \rightarrow \psi \leftrightarrow \text{ch} \) and
  A2: \( \varphi \rightarrow \text{ch} \leftrightarrow \varphi \) and
  A3: \( \varphi \rightarrow \text{ch} \leftrightarrow \varphi \)
shows \( \varphi \rightarrow \psi \leftrightarrow \tau \)
using assms by auto

lemma (in MMIsar0) MMI_breqtr: assumes A1: A < B and
  A2: B = C
shows A < C
using assms by auto

lemma MMI_mpi: assumes A1: \( \psi \) and
  A2: \( \varphi \rightarrow \psi \rightarrow \text{ch} \)
shows \( \varphi \rightarrow \text{ch} \)
using assms by auto

lemma MMI_eqtr2: assumes A1: A = B and
  A2: B = C
shows C = A
using assms by auto

lemma MMI_eqneqi: assumes A1: A = B \( \leftrightarrow \) C = D
shows A \( \neq \) B \( \leftrightarrow \) C \( \neq \) D
using assms by auto

lemma (in MMIsar0) MMI_eqbrtrrd: assumes A1: \( \varphi \rightarrow A = B \) and
  A2: \( \varphi \rightarrow A < C \)
shows $\varphi \rightarrow B < C$
using assms by auto

lemma MMI_mpd: assumes A1: $\varphi \rightarrow \psi$ and
A2: $\varphi \rightarrow \psi \rightarrow \text{ch}$
shows $\varphi \rightarrow \text{ch}$
using assms by auto

lemma MMI_mpdan: assumes A1: $\varphi \rightarrow \psi$ and
A2: $\varphi \land \psi \rightarrow \text{ch}$
shows $\varphi \rightarrow \text{ch}$
using assms by auto

lemma (in MMI_isar0) MMI_breqtrd: assumes A1: $\varphi \rightarrow A < B$ and
A2: $\varphi \rightarrow B = C$
shows $\varphi \rightarrow A < C$
using assms by auto

lemma MMI_mpand: assumes A1: $\varphi \rightarrow \psi$ and
A2: $\varphi \rightarrow 
\psi \land \text{ch} \rightarrow \vartheta$
shows $\varphi \rightarrow \text{ch} \rightarrow \vartheta$
using assms by auto

lemma MMI_imbi1d: assumes A1: $\varphi \rightarrow 
\psi \leftrightarrow \text{ch}$
shows $\varphi \rightarrow
(\psi \rightarrow \vartheta) \leftrightarrow
(\text{ch} \rightarrow \vartheta)$
using assms by auto

lemma MMI_sylan2d: assumes A1: $\varphi \rightarrow 
\psi \land \text{ch} \rightarrow \vartheta$ and
A2: $\varphi \rightarrow \tau \rightarrow \text{ch}$
shows $\varphi \rightarrow
\psi \land \tau \rightarrow \vartheta$
using assms by auto

1243
lemma MMI_imp32: assumes A1: $\varphi \rightarrow$
  $\psi \rightarrow \chi \rightarrow \vartheta$
  shows $\varphi \land \psi \land \chi \rightarrow \vartheta$
  using assms by auto

lemma (in MMIisar0) MMI_breqan12d: assumes A1: $\varphi \rightarrow A = B$ and
  A2: $\psi \rightarrow C = D$
  shows $\varphi \land \psi \rightarrow A < C \iff B < D$
  $\varphi \land \psi \rightarrow A \leq C \iff B \leq D$
  using assms by auto

lemma MMI_a1dd: assumes A1: $\varphi \rightarrow \psi \rightarrow \chi$
  shows $\varphi \rightarrow \psi \rightarrow \vartheta \rightarrow \chi$
  using assms 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 assms by auto

lemma MMI_ad2antll: assumes A1: $\varphi \rightarrow \psi$
  shows $\chi \land \vartheta \land \varphi \rightarrow \psi$
  using assms 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 \rightarrow $ $\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 $ $\vartheta$
  shows $\varphi \rightarrow $

1244
\begin{verbatim}

lemma MMI_mpani: assumes A1: \( \psi \) and  
    A2: \( \varphi \rightarrow \psi \land \text{ch} \rightarrow \theta \)  
    shows \( \varphi \rightarrow \text{ch} \rightarrow \theta \)  
    using assms by auto

lemma MMI_sylidan: assumes A1: \( \varphi \land \psi \rightarrow \text{ch} \) and  
    A2: \( \varphi \land \text{ch} \rightarrow \theta \)  
    shows \( \varphi \land \psi \rightarrow \theta \)  
    using assms by auto

lemma MMI_mp3anl1: assumes A1: \( \varphi \land \psi \rightarrow \text{ch} \) and  
    A2: \( \varphi \land \psi \land \text{ch} \land \theta \rightarrow \tau \)  
    shows \( \varphi \land \psi \land \text{ch} \land \theta \rightarrow \tau \)  
    using assms by auto

lemma MMI_3ad2anti: assumes A1: \( \varphi \rightarrow \text{ch} \)  
    shows \( \varphi \land \psi \land \varphi \rightarrow \psi \)  
    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: \( \text{ch} \)  
    shows \( \varphi \rightarrow \text{ch} \land \psi \)  
    using assms by auto

lemma MMI_mpanl12: assumes A1: \( \varphi \) and  
    A2: \( \psi \) and  
    A3: \( \varphi \land \psi \land \text{ch} \rightarrow \theta \)  
    shows \( \text{ch} \rightarrow \theta \)  
    using assms by auto

lemma MMI_mpanr1: assumes A1: \( \psi \) and

\end{verbatim}
lemma MMI_ad2antrl: assumes A1: \( \varphi \rightarrow \psi \)
  shows ch \& \varphi \& \theta \rightarrow \psi 
  using assms by auto

lemma MMI_3adant3r: assumes A1: \( \varphi \& \psi \& ch \rightarrow \theta \)
  shows \( \varphi \& \psi \& ch \& \tau \rightarrow \theta \)
  using assms by auto

lemma MMI_3adant1l: assumes A1: \( \varphi \& \psi \& ch \rightarrow \theta \)
  shows \( \varphi \& \psi \& \tau \& ch \rightarrow \theta \)
  using assms by auto

lemma MMI_3adant2r: assumes A1: \( \varphi \& \psi \& ch \rightarrow \theta \)
  shows \( \varphi \& (\psi \& \tau) \& ch \rightarrow \theta \)
  using assms by auto

lemma MMI_3bitr4rd: assumes A1: \( \varphi \rightarrow \)
  \( \psi \leftrightarrow ch \) and
  A2: \( \varphi \rightarrow \)
  \( \theta \leftrightarrow \psi \) and
  A3: \( \varphi \rightarrow \)
  \( \tau \leftrightarrow ch \)
  shows \( \varphi \rightarrow \)
  \( \tau \leftrightarrow \theta \)
  using assms by auto

lemma MMI_3anrev:
  shows \( \varphi \& \psi \& ch \leftrightarrow ch \& \psi \& \varphi \)
  by auto

lemma MMI_eqtr4: assumes A1: A = B and
  A2: C = B
  shows A = C
  using assms by auto

lemma MMI_anidm:
  shows \( \varphi \& \varphi \leftrightarrow \varphi \)
  by auto

lemma MMI_bi2anan9r: assumes A1: \( \varphi \rightarrow \)
  \( \psi \leftrightarrow ch \) and
  A2: \( \theta \rightarrow \)
  \( \tau \leftrightarrow \eta \)
\[
\begin{align*}
\text{shows } & \vartheta \land \varphi \rightarrow \\
& \psi \land \tau \leftrightarrow \text{ch} \land \eta \\
\text{using asms by auto} \\
\text{lemma } & 
\text{MMI}_3\text{imtr3g}: \text{assumes } A1: \varphi \rightarrow \psi \rightarrow \text{ch} \text{ and } \\
& A2: \psi \leftrightarrow \vartheta \text{ and } \\
& A3: \text{ch} \leftrightarrow \tau \\
\text{shows } & \varphi \rightarrow \\
& \vartheta \rightarrow \tau \\
\text{using asms by auto} \\
\text{lemma } & 
\text{MMI}_a3d: \text{assumes } A1: \varphi \rightarrow \\
& \neg\psi \rightarrow \neg\text{ch} \\
\text{shows } & \varphi \rightarrow \text{ch} \rightarrow \psi \\
\text{using asms by auto} \\
\text{lemma } & 
\text{MMI}_sylan9bbr: \text{assumes } A1: \varphi \rightarrow \\
& \psi \leftrightarrow \text{ch} \text{ and } \\
& A2: \vartheta \rightarrow \\
& \text{ch} \leftrightarrow \tau \\
\text{shows } & \vartheta \land \varphi \rightarrow \\
& \psi \leftrightarrow \tau \\
\text{using asms by auto} \\
\text{lemma } & 
\text{MMI}_sylan9bb: \text{assumes } A1: \varphi \rightarrow \\
& \psi \leftrightarrow \text{ch} \text{ and } \\
& A2: \vartheta \rightarrow \\
& \text{ch} \leftrightarrow \tau \\
\text{shows } & \varphi \land \vartheta \rightarrow \\
& \psi \leftrightarrow \tau \\
\text{using asms by auto} \\
\text{lemma } & 
\text{MMI}_3\text{bitr3g}: \text{assumes } A1: \varphi \rightarrow \\
& \psi \leftrightarrow \text{ch} \text{ and } \\
& A2: \psi \leftrightarrow \vartheta \text{ and } \\
& A3: \text{ch} \leftrightarrow \tau \\
\text{shows } & \varphi \rightarrow \\
& \vartheta \leftrightarrow \tau \\
\text{using asms by auto} \\
\text{lemma } & 
\text{MMI}_\text{pm5}.21: \\
\text{shows } & \neg\varphi \land \neg\psi \rightarrow \\
& \varphi \leftrightarrow \psi \\
\text{by auto} \\
\text{lemma } & 
\text{MMI}_\text{an6}: \\
\text{shows } & (\varphi \land \psi \land \text{ch}) \land \vartheta \land \tau \land \eta \leftrightarrow \\
\end{align*}
\]

1247
\[(\varphi \land \vartheta) \land (\psi \land \tau) \land \text{ch} \land \eta\]
by auto

**lemma** MMI_syl3anl1: assumes A1: \((\varphi \land \psi \land \text{ch}) \land \vartheta \rightarrow \tau\) and

A2: \(\eta \rightarrow \varphi\)

shows \((\eta \land \psi \land \text{ch}) \land \vartheta \rightarrow \tau\)
using assms by auto

**lemma** MMI_imp4a: assumes A1: \(\varphi \rightarrow\)

\(\psi \rightarrow\)

\(\text{ch} \rightarrow\)

\(\vartheta \rightarrow \tau\)

shows \(\varphi \rightarrow\)

\(\psi \rightarrow\)

\(\text{ch} \land \vartheta \rightarrow \tau\)
using assms by auto

**lemma** (in MMIIsar0) MMI_breqan12rd: assumes A1: \(\varphi \rightarrow A = B\) and

A2: \(\psi \rightarrow C = D\)

shows

\(\psi \land \varphi \rightarrow A < C \leftrightarrow B < D\)

\(\psi \land \varphi \rightarrow A \leq C \leftrightarrow B \leq D\)
using assms by auto

**lemma** (in MMIIsar0) 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 assms by auto

**lemma** MMI_adantrrr: 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_adantrlr: 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_imdistani: assumes A1: \(\varphi \rightarrow \psi \rightarrow \text{ch}\)

shows \(\varphi \land \psi \rightarrow \text{ch}\)
using assms by auto

**lemma** MMI_anabss3: assumes A1: \(\varphi \land \psi \rightarrow \text{ch}\)

shows \(\varphi \land \psi \rightarrow \text{ch}\)
using assms by auto

**lemma** MMI_mp3anl2: assumes A1: \(\psi\) and
\( A2: (\varphi \land \psi \land \text{ch}) \land \vartheta \rightarrow \tau \)

shows \( (\varphi \land \text{ch}) \land \vartheta \rightarrow \tau \)

using assms by auto

lemma MMI_mpanl2: assumes A1: \( \psi \) and
A2: \( (\varphi \land \psi) \land \text{ch} \rightarrow \vartheta \)
shows \( \varphi \land \text{ch} \rightarrow \vartheta \)
using assms by auto

lemma MMI_mpancom: assumes A1: \( \psi \rightarrow \varphi \) and
A2: \( \varphi \land \psi \rightarrow \text{ch} \)
shows \( \psi \rightarrow \text{ch} \)
using assms by auto

lemma MMI_or12:
shows \( \varphi \lor \psi \lor \text{ch} \leftrightarrow \psi \lor \varphi \lor \text{ch} \)

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 assms by auto

lemma MMI_jctir: assumes A1: \( \varphi \rightarrow \psi \) and
A2: \( \text{ch} \)
shows \( \varphi \rightarrow \psi \land \text{ch} \)
using assms by auto

lemma MMI_iffalse:
shows \( \neg \varphi \rightarrow \text{if}(\varphi, A, B) = B \)
by auto

lemma MMI_iftrue:
shows \( \varphi \rightarrow \text{if}(\varphi, A, B) = A \)
by auto

lemma MMI_pm2_61d2: assumes A1: \( \varphi \rightarrow \neg \psi \rightarrow \text{ch} \) and
A2: \( \psi \rightarrow \text{ch} \)
shows \( \varphi \rightarrow \text{ch} \)
using assms by auto

lemma MMI_pm2_61dan: assumes A1: \( \varphi \land \neg \psi \rightarrow \text{ch} \) and
A2: \( \varphi \land \neg \psi \rightarrow \text{ch} \)
shows \( \varphi \rightarrow \text{ch} \)
using assms by auto

lemma MMI_orcanai: assumes A1: \( \varphi \rightarrow \psi \lor \text{ch} \)
shows $\varphi \land \neg \psi \rightarrow \text{ch}$
using assms by auto

lemma MMI_ifcl:
shows $A \in C \land B \in C \rightarrow \text{if}(\varphi, A, B) \in C$
by auto

lemma MMI_imim2i: assumes $A1: \varphi \rightarrow \psi$
shows $(\text{ch} \rightarrow \varphi) \rightarrow \text{ch} \rightarrow \psi$
using assms by auto

lemma MMI_com13: assumes $A1: \varphi \rightarrow$
\[
\psi \rightarrow \text{ch} \rightarrow \vartheta
\]
shows $\text{ch} \rightarrow \psi \rightarrow \varphi \rightarrow \vartheta$
using assms by auto

lemma MMI_rcla4v: assumes $A1: \forall x. x = A \rightarrow \varphi(x) \leftrightarrow \psi$
shows $A \in B \rightarrow (\forall x \in B. \varphi(x)) \rightarrow \psi$
using assms by auto

lemma MMI_syl5d: assumes $A1: \varphi \rightarrow$
\[
\psi \rightarrow \text{ch} \rightarrow \vartheta \text{ and } A2: \varphi \rightarrow \tau \rightarrow \text{ch}
\]
shows $\varphi \rightarrow$
\[
\psi \rightarrow \tau \rightarrow \vartheta
\]
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 MMI_isar0) 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 ⊆ B
shows A = A
  using assms by auto

end

84 Complex numbers in Metamatah - introduction

theory MMI_Complex_ZF imports MMI_logic_and_sets

begin

This theory contains theorems (with proofs) about complex numbers imported 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 “div1lt”.

  lemma (in MMIar0) MMI_recnt:
    shows A ∈ R −→ A ∈ C
proof -
  have S1: R ⊆ C by (rule MMI_axresscn)
  from S1 show A ∈ R −→ A ∈ C by (rule MMI_sseli)
  qed

lemma (in MMIar0) 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 MMIar0) 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 MMIar0) 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_axine0)
from S1 S2 S3 show if ( A ≠ 0 , A , 1 ) ≠ 0 by (rule MMI_elimhyp)
qed

lemma (in MMIar0) 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 MMIar0) 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)
qed

lemma (in MMIar0) 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)
from S3 S4 have S5: ( ( A ∈ C ∧ B ∈ C ) ∧ C ∈ C ) →
      ( ( A + B ) · C ) = ( C · ( A + B ) ) by (rule MMI_axaddcl)
from S3 S4 S5 have S6: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
      ( ( A + B ) · C ) = ( C · ( A + B ) ) by (rule MMI_3impa)
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 )
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_axmulcom)

from S8 S10 have S11: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( ( A · C ) + ( B · C ) ) = ( ( C · A ) + ( C · B ) )
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 MMIsar0) 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 MMIsar0) 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 MMIar0) 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 MMIar0) 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.
  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 + 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 MMIsar0) 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_adddirt)
  from S1 S2 S3 S4 show ( ( A + B ) · C ) =
    ( ( A · C ) + ( B · C ) ) by (rule MMI_mp3an)
qed

lemma (in MMIsar0) 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 MMIsar0) 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 · i ) ∈ C by (rule MMI_mulcl)
  have S5: 1 ∈ C by (rule MMI_1cn)
  from S4 S5 have S6: ( ( i · i ) + 1 ) ∈ C by (rule MMI_addcl)
  from S1 S6 show 0 ∈ C by (rule MMI_eqeltrr)
qed

lemma (in MMIsar0) MMI_addid1: assumes A1: A ∈ C
    shows ( A + 0 ) = A
proof -
  from A1 have S1: A ∈ C.
  have S2: A ∈ C → ( A + 0 ) = A by (rule MMI_ax0id)
  from S1 S2 show ( A + 0 ) = A by (rule MMI_ax_imp)
qed

lemma (in MMIsar0) MMI_addid2: assumes A1: A ∈ C
    shows ( 0 + A ) = A
proof -
  have S1: 0 ∈ C by (rule MMI_0cn)
  from A1 have S2: A ∈ C.
  from S1 S2 have S3: ( 0 + A ) = ( A + 0 ) by (rule MMI_addcom)
  from A1 have S4: A ∈ 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 MMIIsar0) MMI_mulid1: assumes A1: A ∈ C
  shows ( A · 1 ) = A
proof -
  from A1 have S1: A ∈ C.
  have S2: A ∈ C → ( A · 1 ) = A by (rule MMI_ax1id)
  from S1 S2 show ( A · 1 ) = A by (rule MMI_ax_mp)
qed

lemma (in MMIIsar0) MMI_mulid2: assumes A1: A ∈ C
  shows ( 1 · A ) = A
proof -
  have S1: 1 ∈ C by (rule MMI_1cn)
  from A1 have S2: A ∈ C.
  from S1 S2 have S3: ( 1 · A ) = ( A · 1 ) by (rule MMI_mulcom)
  from A1 have S4: A ∈ C.
  from S4 have S5: ( A · 1 ) = A by (rule MMI_mulid1)
  from S3 S5 show ( 1 · A ) = A by (rule MMI_eqtr)
qed

lemma (in MMIIsar0) MMI_negex: assumes A1: A ∈ C
  shows ∃ x ∈ C . ( A + x ) = 0
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 MMIIsar0) MMI_readdcl: 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_axrecex)
  from S1 S2 S3 show ∃ x ∈ C . ( A · x ) = 1 by (rule MMI_mp2an)
qed

lemma (in MMIIsar0) MMI_readdcl: assumes A1: A ∈ R and
  A2: B ∈ R
  shows ( A + B ) ∈ R
proof -
from $A_1$ have $S_1: A \in R$.
from $A_2$ have $S_2: B \in R$.

have $S_3: ( A \in R \land B \in R ) \longrightarrow ( A + B ) \in R$ by (rule MMI_axaddrcl)
from $S_1$ $S_2$ $S_3$ show $( A + B ) \in R$ by (rule MMI_mp2an)

qed

lemma (in MMIIsar0) MMI_remulcl: assumes $A_1: A \in R$ and
$A_2: B \in R$
shows $( A \cdot B ) \in R$
proof -
from $A_1$ have $S_1: A \in R$.
from $A_2$ have $S_2: B \in R$.

have $S_3: ( A \in R \land B \in R ) \longrightarrow ( A \cdot B ) \in R$ by (rule MMI_axmulrcl)
from $S_1$ $S_2$ $S_3$ show $( A \cdot B ) \in R$ by (rule MMI_mp2an)

qed

lemma (in MMIIsar0) MMI_addcan: assumes $A_1: A \in C$ and
$A_2: B \in C$ and
$A_3: C \in C$
shows $( A + B ) = ( A + C ) \iff B = C$
proof -
from $A_1$ have $S_1: A \in C$.
from $S_1$ have $S_2: \exists x \in C . ( A + x ) = 0$ by (rule MMI_negex)
from $A_1$ have $S_3: A \in C$.
from $A_2$ have $S_4: B \in C$.

{ fix $x$
  have $S_5: ( x \in C \land A \in C \land B \in C ) \longrightarrow ( ( x + A ) + B ) =
    ( x + ( A + B ) )$ by (rule MMI_axaddass)
  from $S_4$ $S_5$ have $S_6: ( x \in C \land A \in C ) \longrightarrow ( ( x + A ) + B ) =
    ( x + ( A + B ) )$ by (rule MMI_mp3an3)
  from $A_3$ have $S_7: C \in C$.
  have $S_8: ( x \in C \land A \in C \land C \in C ) \longrightarrow ( ( x + A ) + C ) =
    ( x + ( A + C ) )$ by (rule MMI_axaddass)
  from $S_7$ $S_8$ have $S_9: ( x \in C \land A \in C ) \longrightarrow ( ( x + A ) + C ) =
    ( x + ( A + C ) )$ by (rule MMI_mp3an3)
  from $S_6$ $S_9$ have $S_{10}: ( x \in C \land A \in C ) \longrightarrow
    ( ( x + A ) + B ) = ( ( x + A ) + C ) \iff
    ( x + ( A + B ) ) = ( x + ( A + C ) )$ by (rule MMI_eqeq12d)
  from $S_3$ $S_{10}$ have $S_{11}: x \in C \longrightarrow ( ( x + A ) + B ) =
    ( ( x + A ) + C ) \iff ( x + ( A + B ) ) =
    ( x + ( A + C ) )$ by (rule MMI_mp2an)
  have $S_{12}: ( A + B ) = ( A + C ) \longrightarrow ( x + ( A + B ) ) =
    ( x + ( A + C ) )$ by (rule MMI_opreq2)
  from $S_{11}$ $S_{12}$ have $S_{13}: x \in C \longrightarrow ( ( A + B ) = ( A + C ) \longrightarrow
    ( ( x + A ) + B ) = ( ( x + A ) + C ) )$
    by (rule MMI_syl5bir)
  from $S_{13}$ have $S_{14}: ( x \in C \land ( A + x ) = 0 ) \longrightarrow ( ( A + B ) =
\[(A + C) \rightarrow ((x + A) + B) =
((x + A) + C)\] by (rule MMI_adantr)

from A1 have S15: \(A \in C\).
from S16: \((A \in C \land x \in C) \rightarrow (A + x) = (x + A)\)
by (rule MMI_axaddcom)

from S15 S16 have S17: \(x \in C \rightarrow (A + x) = (x + A)\)
by (rule MMI_mpan)

have S19: \((x + A) = 0 \rightarrow ((x + A) + B) =
(0 + B)\) by (rule MMI_opreq1)

from A2 have S20: \(B \in C\).
from S21: \((0 + B) = B\) by (rule MMI_addid2)
from S19 S20 have S22: \((x + A) = 0 \rightarrow
type error\))
by (rule MMI_opeqld)

have S23: \((x + A) = 0 \rightarrow ((x + A) + C) =
(0 + C)\) by (rule MMI_opreq1)

from S18 S23 have S24: \(x \in C \rightarrow ((A + x) = 0 \rightarrow
type error\))
by (rule MMI_syl6eq)

from S25 have S26: \((x + A) = 0 \rightarrow
type error\))
by (rule MMI_syl6bi)

from S27 have S28: \((x + A) = 0 \rightarrow
type error\))
by (rule MMI_syl6eq)

from S25 S26 have S27: \((x + A) = 0 \rightarrow
type error\))
by (rule MMI_syl6eq)

from S28 have S29: \((x \in C \land (A + x) = 0) \rightarrow
type error\))
by (rule MMI_imp)

from S29 have S30: \((x \in C \land (A + x) = 0) \rightarrow
type error\))
by (rule MMI_sylibd)

from S30 have S31: \(\forall x. (x \in C \rightarrow ((A + x) = 0 \rightarrow
type error\))
by auto

from S31 have S32: \(\exists x \in C. (A + x) = 0\) \(\rightarrow
type error\))
by (rule MMI_r19_23aiv)

from S2 S32 have S33: \((A + B) = (A + C) \rightarrow B = C\)
by (rule MMI_ax_mp)

have S34: \(B = C \rightarrow (A + B) = (A + C)\) by (rule MMI_opreq2)
from S33 S34 have \((A + B) = (A + C) \leftrightarrow B = C\)
by (rule MMI_impbi)

qed

lemma (in MMIIsar0) MMI_addcan2: assumes A1: \(A \in C\) and

A2: \(B \in C\) and
A3: \(C \in C\)

shows \((A + C) = (B + C) \leftrightarrow A = B\)

1258
proof -
from A1 have S1: A ∈ C.
from A3 have S2: C ∈ C.
from S1 S2 have S3: ( A + C ) = ( C + A ) by (rule MMI_addcom)
from A2 have S4: B ∈ C.
from A3 have S5: C ∈ C.
from S4 S5 have S6: ( B + C ) = ( C + B ) by (rule MMI_addcom)
from S3 S6 have S7: ( A + C ) = ( B + C ) ←→
( C + A ) = ( C + B ) by (rule MMI_eqeq12i)
from A3 have S8: C ∈ C.
from A1 have S9: A ∈ C.
from A2 have S10: B ∈ C.
from S8 S9 S10 have S11: ( C + A ) = ( C + B ) ←→
A = B by (rule MMI_addcan)
from S7 S11 show ( A + C ) = ( B + C ) ←→
A = B by (rule MMI_bitr)
qed

lemma (in MMIar0) MMI_addcant:
shows ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( ( A + B ) = ( A + C ) ←→ B = C )
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 + C ) = ( if ( A ∈ C , A , 0 ) + C ) by (rule MMI_opreq1)
  from S1 S2 have S3: A = if ( A ∈ C , A , 0 ) →
( ( A + B ) = ( A + C ) ←→
( if ( A ∈ C , A , 0 ) + B ) = ( if ( A ∈ C , A , 0 ) + C )
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 ) + ( 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 ) + ( 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 ) + B ) =
( if ( A ∈ C , A , 0 ) + C ) ←→ if ( B ∈ C , B , 0 ) = C )
by (rule MMI_bibi1d2)
  have S9: C = if ( C ∈ C , C , 0 ) → ( if ( A ∈ C , A , 0 ) + C

1259
\[ ( \text{if (} A \in C, A, 0 \text{) + if (} C \in C, C, 0 \text{)} \) \]

by (rule MMI_opreq2)

from S9 have S10: \( C = \text{if (} C \in C, C, 0 \text{)} \) \( \rightarrow \)

\( ( ( \text{if (} A \in C, A, 0 \text{) + if (} B \in C, B, 0 \text{)} ) = \)

\( ( \text{if (} A \in C, A, 0 \text{) + if (} C \in C, C, 0 \text{)} \) \leftrightarrow \)

\( ( \text{if (} A \in C, A, 0 \text{) + if (} B \in C, B, 0 \text{)} ) = \)

\( ( \text{if (} A \in C, A, 0 \text{) + if (} C \in C, C, 0 \text{)} \) \rightarrow \)

by (rule MMI_eqeq2d)

have S11: \( C = \text{if (} C \in C, C, 0 \text{)} \) \( \rightarrow \) ( \( \text{if (} B \in C, B, 0 \text{)} = C \) \)

\( \leftrightarrow \)

\( \text{if (} B \in C, B, 0 \text{)} = \text{if (} C \in C, C, 0 \text{)} \) \( \) by (rule MMI_eqeq2d)

have S13: 0 \( \in C \) by (rule MMI_0cn)

from S10 S11 have S12: \( C = \text{if (} C \in C, C, 0 \text{)} \) \( \rightarrow \)

\( ( ( \text{if (} A \in C, A, 0 \text{) + if (} B \in C, B, 0 \text{)} ) = \)

\( ( \text{if (} A \in C, A, 0 \text{) + if (} C \in C, C, 0 \text{)} \) \leftrightarrow \)

\( \text{if (} B \in C, B, 0 \text{)} = \text{if (} C \in C, C, 0 \text{)} \) \) by (rule MMI_bibi12d)

have S14: if ( \( A \in C, A, 0 \) ) \( \in C \) by (rule MMI_elim)

have S15: 0 \( \in C \) by (rule MMI_0cn)

from S10 S15 have S16: if ( \( B \in C, B, 0 \) ) \( \in C \) by (rule MMI_elim)

have S17: 0 \( \in C \) by (rule MMI_0cn)

from S10 S15 S17 have S18: if ( \( C \in C, C, 0 \) ) \( \in C \) by (rule MMI_elim)

from S14 S16 S18 have S19:

\( ( \text{if (} A \in C, A, 0 \text{) + if (} B \in C, B, 0 \text{)} ) = \)

\( ( \text{if (} A \in C, A, 0 \text{) + if (} C \in C, C, 0 \text{)} \) \leftrightarrow \)

\( \text{if (} B \in C, B, 0 \text{)} = \text{if (} C \in C, C, 0 \text{)} \) \) by (rule MMI_addcan)

from S4 S8 S12 S19 show ( \( A \in C \land B \in C \land C \in C \) ) \( \rightarrow \)

\( ( A + B ) = ( A + C ) \leftrightarrow B = C \) by (rule MMI_dedth3h)

qed

\begin{align*}
\text{lemma (in MMIIsar0) MMI_addcan2t:} \\
\text{shows (} A \in C \land B \in C \land C \in C \text{) } & \rightarrow \text{ (} A + C \) = ( B + C ) \rightarrow \\
A = B \\
\) \end{align*}

proof -

have S1: \( C \in C \land A \in C \) \( \rightarrow \) ( \( C + A \) ) = ( \( A + C \) )

by (rule MMI_axaddcom)

from S1 have S2: \( C \in C \land A \in C \land B \in C \) \( \rightarrow \) ( \( C + A \) ) = ( \( A + C \) ) by (rule MMI_3adant3)

have S3: \( C \in C \land B \in C \) \( \rightarrow \) ( \( C + B \) ) = ( \( B + C \) )

by (rule MMI_axaddcom)

from S3 have S4: \( C \in C \land A \in C \land B \in C \) \( \rightarrow \) ( \( C + B \) ) = ( \( B + C \) ) by (rule MMI_3adant2)

from S2 S4 have S5: \( C \in C \land A \in C \land B \in C \) \( \rightarrow \)

\( ( ( C + A ) = ( C + B ) \rightarrow ( A + C ) = ( B + C ) ) \)

by (rule MMI_eqeq12d)

1260
have S6: (C ∈ f ∧ A ∈ f ∧ B ∈ f ) → (C + A ) = (C + B ) by (rule MMI_addcant)
from S5 S6 have S7: (C ∈ f ∧ A ∈ f ∧ B ∈ f ) → (A + C ) = (B + C ) by (rule MMI_bitr3d)
from S7 show (A ∈ f ∧ B ∈ f ∧ C ∈ f ) → (A + C ) = (B + C ) by (rule MMI_3coml)
qed

lemma (in MMIar0) MMI_add12t:
shows (A ∈ f ∧ B ∈ f ∧ C ∈ f ) → (A + (B + C ) ) = (B + (A + C ))
proof -
  have S1: (A ∈ f ∧ B ∈ f ) → (A + B ) = (B + A )
    by (rule MMI_axaddcom)
  from S1 have S2: (A ∈ f ∧ B ∈ f ) → ((A + B ) + C ) = (B + A ) + C ) by (rule MMI_opreq1d)
  from S2 have S3: (A ∈ f ∧ B ∈ f ∧ C ∈ f ) → ((A + B ) + C ) = ((B + A ) + C )
    by (rule MMI_3adant3)
  have S4: (A ∈ f ∧ B ∈ f ∧ C ∈ f ) → (((A + B ) + C ) = (B + (A + C ))
    by (rule MMI_axaddass)
  have S5: (B ∈ f ∧ C ∈ f ∧ A ∈ f ) → ((B + A ) + C ) = (B + A ) + C ) by (rule MMI_addass)
  from S5 have S6: (A ∈ f ∧ B ∈ f ∧ C ∈ f ) → ((B + A ) + C ) = (B + (A + C ))
    by (rule MMI_3com12)
  from S3 S4 S6 show (A ∈ f ∧ B ∈ f ∧ C ∈ f ) → (A + (B + C ) ) = (B + (A + C ))
    by (rule MMI_3eqtr3d)
qed

lemma (in MMIar0) MMI_add23t:
shows (A ∈ f ∧ B ∈ f ∧ C ∈ f ) → ((A + B ) + C ) = ((A + C ) + B )
proof -
  have S1: (B ∈ f ∧ C ∈ f ) → (B + C ) = (C + B )
    by (rule MMI_axaddcom)
  from S1 have S2: (A ∈ f ∧ C ∈ f ) → (A + (B + C ) ) = (B + (C + B )) by (rule MMI_opreq2d)
  from S2 have S3: (B ∈ f ∧ B ∈ f ∧ C ∈ f ) → (B + (C + B ) ) = (A + (B + C ))
    by (rule MMI_3adant1)
  have S4: (A ∈ f ∧ B ∈ f ∧ C ∈ f ) → ((A + B ) + C ) = (A + (B + C ))
    by (rule MMI_axaddass)
  have S5: (A ∈ f ∧ C ∈ f ∧ B ∈ f ) → ((A + C ) + B ) = (A + (C + B )) by (rule MMI_addass)
  have S6: (A ∈ f ∧ B ∈ f ∧ C ∈ f ) → (A + (B + C ))
    by (rule MMI_3com12)
( ( A + C ) + B ) = ( A + ( C + B ) ) by (rule MMI_3com23)
from S3 S4 S6 show ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( ( A + B ) + C ) = ( ( A + C ) + B )
by (rule MMI_3eqtr4d)

qed

lemma (in MMIsar0) MMI_add4t:
shows ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →
( ( A + B ) + ( C + D ) ) = ( ( A + C ) + ( B + D ) )

proof -
  have S1: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( ( A + B ) + C ) = ( ( A + C ) + B ) by (rule MMI_add23t)
  from S1 have S2: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( ( ( A + B ) + C ) + D ) =
( ( ( A + C ) + B ) + D ) by (rule MMI_opreq1d)
  from S2 have S3: ( ( A ∈ C ∧ B ∈ C ) ∧ C ∈ C ) →
( ( ( A + B ) + C ) + D ) =
( ( ( A + C ) + B ) + D ) by (rule MMI_3expa)
  from S3 have S4: ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →

( ( ( A + B ) + C ) + D ) =
( ( ( A + C ) + B ) + D ) by (rule MMI_adantrr)
  have S5: ( ( A + B ) ∈ C ∧ C ∈ C ∧ D ∈ C ) →
( ( ( A + B ) + C ) + D ) =
( ( A + B ) + ( C + D ) ) by (rule MMI_axaddass)
  from S5 have S6: ( ( A + B ) ∈ C ∧ ( C ∈ C ∧ D ∈ C ) ) →
( ( ( A + B ) + C ) + D ) =
( ( A + B ) + ( C + D ) ) by (rule MMI_3expb)
  have S7: ( A ∈ C ∧ B ∈ C ) → ( A + B ) ∈ C by (rule MMI_axaddcl)
  from S6 S7 have S8: ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →

( ( ( A + B ) + C ) + D ) =
( ( A + B ) + ( C + D ) ) by (rule MMI_sylan)
  have S9: ( ( A + C ) ∈ C ∧ B ∈ C ∧ D ∈ C ) →
( ( ( A + C ) + B ) + D ) =
( ( A + C ) + ( B + D ) ) by (rule MMI_axaddass)
  from S9 have S10: ( ( A + C ) ∈ C ∧ ( B ∈ C ∧ D ∈ C ) ) →
( ( ( A + C ) + B ) + D ) =
( ( A + C ) + ( B + D ) ) by (rule MMI_3expb)
  have S11: ( A ∈ C ∧ C ∈ C ) → ( A + C ) ∈ C by (rule MMI_axaddcl)
  from S10 S11 have S12: ( ( A ∈ C ∧ C ∈ C ) ∧ ( B ∈ C ∧ D ∈ C ) ) →

( ( ( A + C ) + B ) + D ) =
( ( A + C ) + ( B + D ) ) by (rule MMI_sylan)
  from S12 have S13: ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →

( ( ( A + C ) + B ) + D ) =
( ( A + C ) + ( B + D ) ) by (rule MMI_an4s)
  from S4 S8 S13 show ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →
\[( ( A + B ) + ( C + D ) ) = ( ( A + C ) + ( B + D ) ) \] by (rule MMI_3eqtr3d)

qed

lemma (in MMIsar0) MMI_add42t:

shows \(( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \longrightarrow ( ( A + B ) + ( C + D ) ) = ( ( A + C ) + ( D + B ) )\)

proof -

have S1: \(( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \longrightarrow ( ( A + B ) + ( C + D ) ) = ( ( A + C ) + ( B + D ) ) \) by (rule MMI_add4t)

have S2: \(( B \in C \land D \in C ) \longrightarrow ( B + D ) = ( D + B ) \) by (rule MMI_axaddcom)

from S2 have S3: \(( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \longrightarrow ( B + D ) = ( D + B ) \) by (rule MMI_axaddcom)

from S1 S3 have S4: \(( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \longrightarrow ( ( A + B ) + ( C + D ) ) = ( ( A + C ) + ( D + B ) ) \) by (rule MMI_opreq2d)

from S1 S4 show \(( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \longrightarrow ( ( A + B ) + ( C + D ) ) = ( ( A + C ) + ( D + B ) ) \) by (rule MMI_eqtrd)

qed

lemma (in MMIsar0) MMI_add12: assumes A1: \( A \in C \) and

A2: \( B \in C \) and

A3: \( C \in C \)

shows \(( A + ( B + C ) ) = ( B + ( A + C ) )\)

proof -

from A1 have S1: \( A \in C \).

from A2 have S2: \( B \in C \).

from A3 have S3: \( C \in C \).

have S4: \(( A \in C \land B \in C \land C \in C ) \longrightarrow ( A + ( B + C ) ) = ( B + ( A + C ) ) \) by (rule MMI_add12t)

from S1 S2 S3 S4 show \(( A + ( B + C ) ) = ( B + ( A + C ) ) \) by (rule MMI_mp3an)

qed

lemma (in MMIsar0) MMI_add23: 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: \(( A \in C \land B \in C \land C \in C ) \longrightarrow ( ( A + B ) + C ) = ( ( A + C ) + B ) \) by (rule MMI_mp2an)

qed

1263
( ( 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 MMIsar0) 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_add4)
  from S3 S6 S7 show ( ( A + B ) + ( C + D ) ) =
( ( A + C ) + ( B + D ) ) by (rule MMI_mp2an)

qed

lemma (in MMIsar0) MMI_add42: assumes A1: A ∈ C and
  A2: B ∈ C and
  A3: C ∈ C and
  A4: D ∈ C
shows ( ( A + B ) + ( C + D ) ) =
( ( A + C ) + ( D + B ) )

proof -
  from A1 have S1: A ∈ C.
  from A2 have S2: B ∈ C.
  from A3 have S3: C ∈ C.
  from A4 have S4: D ∈ C.
  from S1 S2 S3 S4 have S5: ( ( A + B ) + ( C + D ) ) =
( ( A + C ) + ( B + D ) ) by (rule MMI_add4)
  from A2 have S6: B ∈ C.
  from A4 have S7: D ∈ C.
  from S6 S7 have S8: ( B + D ) = ( D + B ) by (rule MMI_addcom)
  from S8 have S9: ( ( A + C ) + ( B + 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 ∈ C → ( 0 + A ) = A

1264
proof -
  have S1: 0 ∈ C by (rule MMI_0cn)
  have S2: ( 0 ∈ C ∧ A ∈ C ) → ( 0 + A ) = ( A + 0 )
    by (rule MMI_axaddcom)
  from S1 S2 have S3: A ∈ C → ( 0 + A ) = ( A + 0 )
    by (rule MMI_mpan)
  have S4: A ∈ C → ( A + 0 ) = A by (rule MMI_ax0id)
  from S3 S4 show A ∈ C → ( 0 + A ) = A by (rule MMI_eqtrd)
qed

lemma (in MMIsar0) MMI_peano2cn:
  shows A ∈ C → ( A + 1 ) ∈ C
proof -
  have S1: 1 ∈ C by (rule MMI_1cn)
  have S2: ( A ∈ C ∧ 1 ∈ C ) → ( A + 1 ) ∈ C by (rule MMI_axaddcl)
  from S1 S2 show A ∈ C → ( A + 1 ) ∈ C by (rule MMI_mpan2)
qed

lemma (in MMIsar0) MMI_peano2re:
  shows A ∈ R → ( A + 1 ) ∈ R
proof -
  have S1: 1 ∈ R by (rule MMI_ax1re)
  have S2: ( A ∈ R ∧ 1 ∈ R ) → ( A + 1 ) ∈ R by (rule MMI_axaddcl)
  from S1 S2 show A ∈ R → ( A + 1 ) ∈ R by (rule MMI_mpan2)
qed

lemma (in MMIsar0) 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: \( y \in C \rightarrow ( \exists x \in C . x = ( y + B ) ) \)
by (rule MMI_sylib)
\{ fix x \\
  have S11: x = ( y + B ) \rightarrow ( A + x ) = ( A + ( y + B ) ) by (rule MMI_opreq2)
  from A1 have S12: A \in C.
  from A2 have S13: B \in C.
  have S14: ( A \in C \wedge y \in C \wedge B \in C ) \rightarrow ( ( A + y ) + B ) = ( A + ( y + B ) )
by (rule MMI_axaddass)
from S12 S13 S14 have S15: y \in C \rightarrow ( ( A + y ) + B ) = ( A + ( y + B ) )
by (rule MMI_mp3an13)
from S11 S15 have S16: y \in C \rightarrow ( A + ( y + B ) ) = ( A + y )
by (rule MMI_eqcomd)
from S11 S16 have S17: ( A + y ) = ( y + B )
by (rule MMI_sylan9eqr)
from S17 have S18: ( A + y ) = 0 
by (rule MMI_opreq1)
from A2 have S19: B \in C.
from S19 have S20: ( 0 + B ) = B by (rule MMI_addid2)
from S18 S20 have S21: ( A + y ) = 0 
by (rule MMI_syl6eq)
from S17 S21 have S22: ( ( A + y ) = 0 \wedge ( y \in C \wedge x = ( y + B ) ) ) 
by (rule MMI_impcom)
from S24 have ( \forall x \in C . ( x = ( y + B ) \rightarrow ( A + x ) = B ) )
by (rule MMI_a1d)
\} then have S25: \( \forall x . ( y \in C \wedge ( A + y ) = 0 ) \rightarrow ( x \in C \rightarrow ( x = ( y + B ) \rightarrow ( A + x ) = B ) ) \)
by (rule MMI_exp32)
from S25 have S26: ( y \in C \wedge ( A + y ) = 0 ) 
by (rule MMI_r19_21aiv)
from S26 have S27: y \in C \rightarrow ( ( A + y ) = 0 
by (rule MMI_ext)
have S28: ( \forall x \in C . ( x = ( y + B ) \rightarrow ( A + x ) = B ) ) 
by (rule MMI_r19_22)
from S27 S28 have S29: y \in C \rightarrow ( ( A + y ) = 0 
by (rule MMI_syl6)
from S10 S29 have y \in C \rightarrow ( ( A + y ) = 0 
by (rule MMI_mpid)
} then have S30: \( \forall y. y \in C \rightarrow ( (A + y) = 0 \rightarrow \exists x \in C. (A + x) = B ) \) by simp

from S30 have S31: \( (\exists y \in C. (A + y) = 0) \rightarrow (\exists x \in C. (A + x) = B) \) by (rule MMI_r19_23aiv)

from S5 S31 have S32: \( \exists x \in C. (A + x) = B \) by (rule MMI_ax_mp)

from A1 have S33: \( A \in C \).

\{ fix \ x \ y \n
have S34: \( (A \in C \land x \in C \land y \in C) \rightarrow ((A + x) = (A + y) \leftrightarrow x = y) \) by (rule MMI_addcant)

have S35: \( (A + x) = B \land (A + y) = B) \rightarrow (A + x) = (A + y) \) by (rule MMI_eqtr3t)

from S34 S35 have S36: \( (A \in C \land x \in C \land y \in C) \rightarrow ((A + x) = B \land (A + y) = B) \rightarrow x = y \) by (rule MMI_syl5bi)

from S33 S36 have \( (x \in C \land y \in C) \rightarrow ((A + x) = B \land (A + y) = B) \rightarrow x = y \) by (rule MMI_mp3an1)

} then have S37: \( \forall x \ y. (x \in C \land y \in C) \rightarrow ((A + x) = B \land (A + y) = B) \rightarrow x = y \) by auto

from S37 have S38: \( \forall x \in C \land y \in C. ((A + x) = B \land (A + y) = B) \rightarrow x = y \) by (rule MMI_rgen2)

from S3 S32 S38 show \( \exists! x \in C. (A + x) = B \) by (rule MMI_mpbir2an)

qed

lemma (in MMIIsar0) MMI_subval: assumes A \( \in \) C B \( \in \) 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 \( \rightarrow \) (-A) = (-B)

proof -

have S1: \( A = B \rightarrow (0 - A) = (0 - B) \) 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 \( \rightarrow \) (-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 −→ (¬A) = (¬B) by (rule MMI_negeq)
  from S1 S2 show (¬A) = (¬B) by (rule MMI_ax_mp)
qed

lemma (in MMIsar0) MMI_negeqd: assumes A1: ϕ −→ A = B
  shows ϕ −→ (¬A) = (¬B)
proof -
  from A1 have S1: ϕ −→ A = B.
  have S2: A = B −→ (¬A) = (¬B) by (rule MMI_negeq)
  from S1 S2 show ϕ −→ (¬A) = (¬B) by (rule MMI_syl)
qed

lemma (in MMIsar0) MMI_hbneg: assumes A1: y ∈ A −→ ( ∀ x . y ∈ A )
  shows y ∈ ((¬ A)) −→ ( ∀ x . (y ∈ ((¬ A)) ) )
using assms by auto

lemma (in MMIsar0) MMI_minusex:
  shows ((¬ A)) isASet by auto

lemma (in MMIsar0) MMI_subcl: 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.
  from S1 S2 have S3: ( A - B ) = ∪ { x ∈ C . ( B + x ) = A } by (rule MMI_subval)
  from A2 have S4: B ∈ C.
  from A1 have S5: A ∈ C.
  from S4 S5 have S6: ∃! x . x ∈ C ∧ ( B + x ) = A by (rule MMI_negeu)
  have S7: ( ∃! x . x ∈ C ∧ ( B + x ) = A ) −→ ∪ { x ∈ C . ( B + x ) = A } ∈ C by (rule MMI_reucl)
  from S6 S7 have S8: ∪ { x ∈ C . ( B + x ) = A } ∈ C
  by (rule MMI_ax_mp)
  from S3 S8 show ( A - B ) ∈ C by simp
qed

lemma (in MMIsar0) MMI_subclt:
  shows ( A ∈ C ∧ B ∈ C ) −→ ( A - B ) ∈ C
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 ←→
( if ( A ∈ C , A , 0 ) - B ) ∈ C ) by (rule MMI_eleq1d)

have S3: B = if ( B ∈ C , B , 0 ) → ( if ( A ∈ C , A , 0 ) - B ) ∈ C
→← (if ( A ∈ C , A , 0 ) - ( B ∈ C , B , 0 )) ∈ C
by (rule MMI_eleq1d)

have S5: 0 ∈ C by (rule MMI_0cn)
from S5 have S6: if ( A ∈ C , A , 0 ) ∈ C by (rule MMI_elimel)

have S4: B = if ( B ∈ C , B , 0 ) - ( if ( A ∈ C , A , 0 ) - B ) ∈ C
by (rule MMI_opreq2)

have S7: 0 ∈ C by (rule MMI_0cn)
from S7 have S8: if ( B ∈ C , B , 0 ) ∈ C by (rule MMI_elimel)

have S9: ( if ( A ∈ C , A , 0 ) - if ( B ∈ C , B , 0 ) ) ∈ C
by (rule MMI_subcl)

from S2 S4 S9 show ( A ∈ C ∧ B ∈ C ) → ( A - B ) ∈ C
by (rule MMI_dedth2h)

qed

lemma (in MMIIsar0) MMI_negclt:
shows A ∈ C → ( - A ) ∈ C
proof -

have S1: 0 ∈ C by (rule MMI_0cn)
have S2: ( 0 ∈ C ∧ A ∈ C ) → ( 0 - A ) ∈ C by (rule MMI_subclt)
from S1 S2 have S3: A ∈ C → ( 0 - A ) ∈ C by (rule MMI_mpan)

have S4: ( - A ) = ( 0 - A ) by (rule MMI_df_neg)
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_eeq2)

have S3: x = C → ( B + x ) = ( B + C ) by (rule MMI_opreq2)
from S3 have S4: x = C \iff ( ( A - B ) = x \iff ( B + x ) = A )
  by (rule MMI_eqeq1d)
from S2 S4 have x = C \iff ( ( A - B ) = C \iff ( B + C ) = A )
  by (rule MMI_bibi12d)
} then have S5: \forall x. x = C \iff ( ( A - B ) = x \iff ( B + x ) = A )
  by (rule MMI_negeu)
from A2 have S6: B \in C.
from A1 have S7: A \in C.
from S6 S7 have S8: \exists ! x . x \in C \land ( B + x ) = A by (rule MMI_negeu)
  \{ fix x 
    have S9: ( x \in C \land ( \exists ! x . x \in C \land ( B + x ) = A ) \implies
      ( ( B + x ) = A ) \iff \bigcup \{ x \in C . ( B + x ) = A \} = x 
    by (rule MMI_reuuni1)
    from S8 S9 have x \in C \implies ( ( B + x ) = A \iff \bigcup \{ x \in C . ( B + x ) = A \} = x )
      by (rule MMI_mpan2)
  \} then have S10: \forall x. x \in C \implies ( ( B + x ) = A \iff \bigcup \{ x \in C . ( B + x ) = A \} = x )
    by blast
from A1 have S11: A \in C.
from A2 have S12: B \in C.
from S11 S12 have S13: ( A - B ) = \bigcup \{ x \in C . ( B + x ) = A \}
  by (rule MMI_subval)
from S13 have S14: \forall x. ( A - B ) = x \iff \bigcup \{ x \in C . ( B + x ) = A \} = x by simp
from S10 S14 have S15: \forall x. x \in C \implies ( ( A - B ) = x \iff
  ( B + x ) = A ) by (rule MMI_syl6rbbr)
from S5 S15 have S16: C \in C \implies ( ( A - B ) = C \iff
  ( B + C ) = A ) by (rule MMI_vtoclga)
from S1 S16 show ( A - B ) = C \iff ( B + C ) = A
  by (rule MMI_ax_mp)
qed

lemma (in MMIar0) MMI_subsub23: assumes A1: A \in C and
  A2: B \in C and
  A3: C \in C
shows ( A - B ) = C \iff ( A - C ) = B
proof -
  from A2 have S1: B \in C.
  from A3 have S2: C \in C.
  from S1 S2 have S3: ( B + C ) = ( C + B ) by (rule MMI_addcom)
  from S3 have S4: ( B + C ) = A \iff ( C + B ) = A
    by (rule MMI_eqeq1i)
  from A1 have S5: A \in C.
  from A2 have S6: B \in C.
  from A3 have S7: C \in C.
from S5 S6 S7 have S8: \(( A - B ) = C \iff ( B + C ) = A \)
  by (rule MMI_subadd)
from A1 have S9: \( A \in C \).
from A3 have S10: \( C \in C \).
from A2 have S11: \( B \in C \).
from S9 S10 S11 have S12: \(( A - C ) = B \iff ( C + B ) = A \)
  by (rule MMI_subadd)
from S4 S8 S12 show \(( A - B ) = C \iff ( A - C ) = B \)
  by (rule MMI_3bitr4)
qed

lemma (in MMI0) MMI_subaddt:
  shows \(( A \in C \land B \in C \land C \in C ) \rightarrow (( A - B ) = C \iff ( B + C ) = A )\)
proof -
have S1: \( A = \text{if} ( A \in C , A , 0 ) \rightarrow ( A - B ) = \)
  \(( ( A \in C , A , 0 ) - B ) \) by (rule MMI_opreq1)
from S1 have S2: \( A = \text{if} ( A \in C , A , 0 ) \rightarrow ( ( A - B ) = C \iff ( B + C ) = A ) \)
  by (rule MMI_eqeq1d)
from S2 S3 have S4: \( A = \text{if} ( A \in C , A , 0 ) \rightarrow (( ( A - B ) = C \iff ( B + C ) = A ) \)
  by (rule MMI_opreq2)
from S4 S5 have S6: \( B = \text{if} ( B \in C , B , 0 ) \rightarrow (( ( A \in C , A , 0 ) - B ) = C \iff ( B + C ) = A ) \)
  by (rule MMI_bibi12d)
from S6 S7 have S8: \( B = \text{if} ( B \in C , B , 0 ) \rightarrow ( ( ( A \in C , A , 0 ) - B ) = C \iff ( B + C ) = A ) \)
  by (rule MMI_opreq1)
from S7 S8 have S9: \( B = \text{if} ( B \in C , B , 0 ) \rightarrow ( ( ( A \in C , A , 0 ) - B ) = C \iff ( B + C ) = A ) \)
  by (rule MMI_bibi12d)
from S8 S9 have S10: \( C = \text{if} ( C \in C , C , 0 ) \rightarrow (( ( A \in C , A , 0 ) - B ) = C \iff ( B + C ) = A ) \)
  by (rule MMI_opreq2)

1271
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 ( B ∈ C, B, 0 ) + C ) = if ( A ∈ C, A, 0 ) )
( 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 ) + 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 ) 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_mpbii)
qed

lemma (in MMIsar0) MMI_pncan3: assumes A1: A ∈ C and
A2: B ∈ C

1272
shows \(( A + ( B - A ) ) = B\)

proof -
from A1 have S1: A \in C.
from A2 have S2: B \in C.
have S3: \(( A \in C \land B \in C ) \rightarrow ( A + ( B - A ) ) = B\)
by (rule MMI_pncan3t)
from S1 S2 S3 show \(( A + ( B - A ) ) = B\) by (rule MMI_mp2an)
qed

lemma (in MMIIsar0) MMI_negidt:
shows A \in C \rightarrow ( A + ( - A ) ) = 0
proof -
have S1: 0 \in C by (rule MMI_0cn)
have S2: \(( A \in C \land 0 \in C ) \rightarrow ( A + ( 0 - A ) ) = 0\)
by (rule MMI_pncan3t)
from S1 S2 have S3: A \in C \rightarrow ( A + ( 0 - A ) ) = 0
by (rule MMI_mpan2)
have S4: \((- A ) ) = ( 0 - A ) by (rule MMI_df_neg)
from S4 have S5: \(( A + ( - A ) ) ) = ( A + ( 0 - A ) )
by (rule MMI_cpreq2i)
from S3 S5 show A \in C \rightarrow ( A + ( - A ) ) = 0 by (rule MMI_syl5eq)
qed

lemma (in MMIIsar0) MMI_negid: assumes A1: A \in C
shows ( A + ( (-A) ) ) = 0
proof -
from A1 have S1: A \in C.
have S2: A \in C \rightarrow ( A + ( (-A) ) ) = 0 by (rule MMI_negidt)
from S1 S2 show ( A + ( (-A) ) ) = 0 by (rule MMI_ax_mp)
qed

lemma (in MMIIsar0) MMI_negsub: assumes A1: A \in C and
A2: B \in C
shows ( A + ( (-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 S4 have S11: ( (-B) ) \in C .
from A2 have S12: B \in C.
from S11 S12 have S13: ( ( (- B) ) + B ) = ( B + ( (- B) ) ) by (rule MMI_addcom)
from A2 have S14: B ∈ C.
from S14 have S15: ( B + ( (- B) ) ) = 0 by (rule MMI_negid)
from S13 S15 have S16: ( ( (- B) ) + B ) = 0 by (rule MMI_eqtr)
by (rule MMI_opreq21)
from A1 have S18: A ∈ C.
from S18 have S19: ( A + 0 ) = A by (rule MMI_addid1)
from S10 S17 have S20: ( ( A + ( (- B) ) ) + B ) = ( A + ( (- B) ) ) = A by (rule MMI_eqcomi)
by (rule MMI_3eqtr)
from A1 have S22: A ∈ C.
from A2 have S23: B ∈ C.
from S5 have S24: ( A + ( (- B) ) ) ∈ C.
from S22 S23 S24 have S25: ( A - B ) = ( A + ( (- B) ) ) ←→ ( B + ( A + ( (- B) ) ) ) = A by (rule MMI_eqtr)
by (rule MMI_mpbir)
from S26 show ( A + ( (- B) ) ) = ( A - B ) by (rule MMI_eqcomi)
qed
lemma (in MMIar0) MMI_negsubt:
shows ( A ∈ C ∧ B ∈ C ) → ( 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)
have S2: A = if ( A ∈ C , A , 0 ) → ( A - B ) = ( if ( A ∈ C , A , 0 ) - B ) by (rule MMI_opreq1)
from S1 S2 have S3: A = if ( A ∈ C , A , 0 ) → ( ( A + ( (- B) ) ) ) = ( A - B ) ←→ ( if ( A ∈ C , A , 0 ) + ( (- B) ) ) = ( if ( A ∈ C , A , 0 ) - B ) by (rule MMI_eqeq12d)
have S4: B = if ( B ∈ C , B , 0 ) → ( (- B) ) = ( - if ( B ∈ C , B , 0 ) ) by (rule MMI_negeq)
from S4 have S5: B = if ( B ∈ C , B , 0 ) → ( if ( A ∈ C , A , 0 ) + ( (- B) ) ) = ( if ( A ∈ C , A , 0 ) - B ) by (rule MMI_opreq2d)
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 S5 S6 have S7: 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 ) - if ( B ∈ C , B , 0 ) )

1274
by (rule MMI_eqeq12d)

have S8: 0 ∈ C by (rule MMI_0cn)

from S8 have S9: if ( A ∈ C , A , 0 ) ∈ C by (rule MMI_elimel)

have S10: 0 ∈ C by (rule MMI_0cn)

from S10 have S11: if ( B ∈ C , B , 0 ) ∈ C by (rule MMI_elimel)

from S9 S11 have S12:
  ( if ( A ∈ C , A , 0 ) + ( - if ( B ∈ C , B , 0 ) ) ) =
  ( if ( A ∈ C , A , 0 ) - if ( B ∈ C , B , 0 ) )

by (rule MMI_negsub)

from S3 S7 S12 show ( A ∈ C ∧ B ∈ C ) → ( A + ( - B ) ) =
  ( A - B ) by (rule MMI_dedth2h)

qed

lemma (in MMIasar0) MMI_addsubasst:
  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 + ( B + ( - C ) ) ) by (rule MMI_axaddass)

have S2: C ∈ C → ( - C ) ∈ C by (rule MMI_negclt)

from S1 S2 have S3: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
  ( ( A + B ) + ( - C ) ) =
  ( A + ( B + ( - C ) ) ) by (rule MMI_syl3an3)

have S4: ( ( A + B ) ∈ C ∧ C ∈ C ) →
  ( ( A + B ) + ( - C ) ) = ( ( A + B ) - C )

by (rule MMI_negsubt)

have S5: ( A ∈ C ∧ B ∈ C ) → ( A + B ) ∈ C 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 MMIasar0) 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_opreq1d)
   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_addsubasst)
   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 ∈ 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_exp3a)
   from S9 have S10: A ∈ C → ( B ∈ C →
     ( C ∈ C → ( B + ( A - C ) ) = ( ( A - C ) + B ) ) )
     by (rule MMI_com12)
   from S10 have S11: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
     ( B + ( A - C ) ) = ( ( A - C ) + B ) by (rule MMI_3eqtrd)
   qed

lemma (in MMIsr0) MMI_addsub12t:
   shows ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) → ( A + ( B - C ) ) =
     ( B + ( A - C ) )
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_opreq1d)
   from S2 have S3: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
     ( ( A + B ) - C ) = ( ( B + A ) - C )
     by (rule MMI_3adant3)
   have S4: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) → ( ( A + B ) - C ) =
     ( A + ( B - C ) ) by (rule MMI_addsubasst)
   have S5: ( B ∈ C ∧ A ∈ C ∧ C ∈ C ) → ( ( B + A ) - C ) =
     ( B + ( A - C ) ) by (rule MMI_addsubasst)
   from S5 have S6: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
(B + A) - C = (B + (A - C)) by (rule MMI_3com12)

from S3 S4 S6 show (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
  (A + (B - C)) = (B + (A - C))

by (rule MMI_3eqtr3d)

qed

lemma (in MMI2ar0) MMI_addsubass: 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_addsubass)
  from S1 S2 S3 S4 show ((A + B) - C) =
    (A + (B - C)) by (rule MMI_mp3an)

qed

lemma (in MMI2ar0) MMI_addsub: 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_addsub)
  from S1 S2 S3 S4 show ((A + B) - C) =
    ((A - C) + B) by (rule MMI_mp3an)

qed

lemma (in MMI2ar0) MMI_2addsubt:

shows ((A ∈ C ∧ B ∈ C) ∧ (C ∈ C ∧ D ∈ C)) →
  ((A + B) + C) - D =
  ((A + C) + B) by (rule MMI_add23)

proof -
  have S1: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
    ((A + B) + C) =
    (A + (B + C)) by (rule MMI_add23t)
  from S1 have S2: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
    ((A + B) + C) =
    (A + C) + B) by (rule MMI_3expa)
  from S2 have S3: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
    ((A + B) + C) =
    (A + C) + B) by (rule MMI_adantrr)
  from S3 have S4: (A ∈ C ∧ B ∈ C ∧ C ∈ C) ∧
    ((A + B) + C) - D =

1277
( (( A + C ) + B ) - D ) by (rule MMI_opreq1d)
have S5: ( ( A + C ) ∈ C ∧ B ∈ C ∧ D ∈ C ) →
    ( ( ( A + C ) + B ) - D ) =
    ( ( ( A + C ) - D ) + B ) by (rule MMI_addsubt)
from S5 have S6: ( ( ( A + C ) ∈ C ∧ ( B ∈ C ∧ D ∈ C ) ) ) →
    ( ( ( A + C ) + B ) - D ) =
    ( ( ( A + C ) - D ) + B ) by (rule MMI_3expb)
have S7: ( A ∈ C ∧ C ∈ C ) → ( A + C ) ∈ C by (rule MMI_axaddcl)
from S6 S7 have S8: ( ( A ∈ C ∧ C ∈ C ) ∧ ( B ∈ C ∧ D ∈ C ) ) →

( ( ( A + C ) + B ) - D ) =
( ( ( A + C ) - D ) + B ) by (rule MMI_sylan)
from S8 have S9: ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →

( ( ( A + C ) + B ) - D ) =
( ( ( A + C ) - D ) + B ) by (rule MMI_an4s)
from S4 S9 show ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →

( ( ( A + B ) + C ) - D ) =
( ( ( A + C ) - D ) + B ) by (rule MMI_eqtrd)

qed

lemma (in MMIsar0) MMI_negneg: assumes A1: A ∈ C
    shows ( - ( ( - A ) ) ) = A
proof -
    from A1 have S1: A ∈ C.
    from S1 have S2: ( ( - A ) ) ∈ C by (rule MMI_negcl)
    from S2 have S3: ( ( ( ( - A ) ) + ( - ( ( - A ) ) ) ) ) = 0
        by (rule MMI_negid)
    from S3 have S4: ( A + ( ( ( - A ) ) + ( - ( ( - A ) ) ) ) ) =
        ( A + 0 ) by (rule MMI_opreq2i)
    from A1 have S5: A ∈ 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 ∈ C.
    from S2 have S9: ( ( - A ) ) ∈ C .
    from S2 have S10: ( ( - A ) ) ∈ C .
    from S10 have S11: ( ( - A ) ) ∈ 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 ) ) ) ∈ 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 ∈ C.

1278
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 MMIar0) MMI_subid: assumes A1: A ∈ C
shows ( A - A ) = 0
proof -
from A1 have S1: A ∈ C.
from A1 have S2: A ∈ C.
from S1 S2 have S3: ( A + ( ( - A ) ) ) = ( A - A )
by (rule MMI_negsub)
from A1 have S4: A ∈ 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 MMIar0) MMI_subid1: assumes A1: A ∈ C
shows ( A - 0 ) = A
proof -
from A1 have S1: A ∈ C.
from S1 have S2: ( 0 + A ) = A by (rule MMI_addid2)
from A1 have S3: A ∈ C.
have S4: 0 ∈ C by (rule MMI_0cn)
from A1 have S5: A ∈ C.
from S3 S4 S5 have S6: ( A - 0 ) = A ↔ ( 0 + A ) = A
by (rule MMI_subadd)
from A1 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 by (rule MMI_dedth)
qed

lemma (in MMIar0) MMI_negnegt:
shows A ∈ C −→ ( - ( ( - A ) ) ) = A
proof -
have S1: A = if ( A ∈ C , A , 0 ) → ( ( - A ) ) =
( - if ( A ∈ C , A , 0 ) ) by (rule MMI_negeq)
from S1 have S2: A = if ( A ∈ C , A , 0 ) → ( ( - ( ( - A ) ) ) =
( - ( - if ( A ∈ C , A , 0 ) ) ) by (rule MMI_negeq)
from S2 have S3: A = if ( A ∈ C , A , 0 ) → A = if ( A ∈ C , A , 0 )
by (rule MMI_id)
from S3 S2 have S4: A = if ( A ∈ C , A , 0 ) →
( ( - ( ( - A ) ) ) = A ←→
( - if ( A ∈ C , A , 0 ) ) ) = if ( A ∈ C , A , 0 )
by (rule MMI_eqeq12d)
from S4 S3 have S5: 0 ∈ C by (rule MMI_0cn)
from S5 have S6: if ( A ∈ C , A , 0 ) ∈ C by (rule MMI_elimel)
from S6 have S7: ( - ( - if ( A ∈ C , A , 0 ) ) ) =
if ( A ∈ C , A , 0 ) by (rule MMI_negneg)
from S4 S7 show A ∈ C → ( ( ( ( - A ) ) ) = A by (rule MMI_dedth)
qed
lemma (in MMIIsar0) MMI_subnegt:
  shows ( A ∈ C ∧ B ∈ C ) −→ ( A - (- B) ) = ( A + B )
proof -
  have S1: ( A ∈ C ∧ (- B) ∈ C ) −→
    ( A + (- (- B)) ) = ( A - (- B) )
    by (rule MMI_negsubt)
  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: B ∈ C −→ (- (- B)) = B by (rule MMI_negnegt)
  from S4 have S5: B ∈ C −→ ( A + (- (- B)) ) =
    ( A + B ) by (rule MMI_opreq2d)
  from S5 have S6: ( A ∈ C ∧ B ∈ C ) −→
    ( A + (- (- B)) ) = ( A + B ) by (rule MMI_adantl)
  from S3 S6 show ( A ∈ C ∧ B ∈ C ) −→
    ( A - (- B) ) = ( A + B ) by (rule MMI_eqtr3d)
qed

lemma (in MMIIsar0) MMI_subidt:
  shows A ∈ C −→ ( A - A ) = 0
proof -
  have S1: ( A = if ( A ∈ C , A , 0 ) ∧ A = if ( A ∈ C , A , 0 ) )
    −→
    ( A - A ) = ( if ( A ∈ C , A , 0 ) - if ( A ∈ C , A , 0 ) )
    by (rule MMI_opreq12)
  from S1 have S2: A = if ( A ∈ C , A , 0 ) −→
    ( A - A ) = ( if ( A ∈ C , A , 0 ) - if ( A ∈ 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_eqeq1d)
  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_subidit:
  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 MMIar0) MMI_pncant:
shows ( A ∈ C ∧ B ∈ C ) → ( ( A + B ) - B ) = A

proof -
  have S1: ( A ∈ 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_pncant)
  from S2 S3 S7 show ( A ∈ C ∧ B ∈ C ) → ( ( A + B ) - B ) = A
  by (rule MMI_eqtr3d)

qed

lemma (in MMIar0) MMI_pncan2t:
shows ( A ∈ C ∧ B ∈ C ) → ( ( A + B ) - A ) = B

proof -
  have S1: ( B ∈ C ∧ A ∈ C ) → ( B + A ) = ( A + B )
  by (rule MMI_axaddcom)
  from S1 have S2: ( B ∈ C ∧ A ∈ C ) → ( ( B + A ) - A ) =
( ( A + B ) - A ) by (rule MMI_opreq1d)
  have S3: ( B ∈ C ∧ A ∈ C ) → ( ( B + A ) - A ) = B
  by (rule MMI_pncant)
  from S2 S3 have S4: ( B ∈ C ∧ A ∈ C ) →
( ( A + B ) - A ) = B by (rule MMI_eqtr3d)
  from S4 show ( A ∈ C ∧ B ∈ C ) → ( ( A + B ) - A ) = B
  by (rule MMI_ancoms)

qed

lemma (in MMIar0) MMI_npcant:
shows ( A ∈ C ∧ B ∈ C ) → ( ( A - B ) + B ) = A

proof -
  have S1: ( A ∈ C ∧ B ∈ C ) →
( ( A + B ) - B ) = ( ( A - B ) + B )
by (rule MMI_addsubt)
from S1 have S2: \(( A \in C \land ( B \in C \land B \in C ) ) \rightarrow \\
\(( ( A + B ) - B ) = ( ( A - B ) + B ) \) by (rule MMI_3expb)
from S2 have S3: \(( A \in C \land B \in C ) \rightarrow \\
\(( ( A + B ) - B ) = ( ( A - B ) + B ) \)
by (rule MMI_anabssan2)
have S4: \(( A \in C \land B \in C ) \rightarrow ( ( A + B ) - B ) = A \\
by (rule MMI_pncant)
from S3 S4 show \(( A \in C \land B \in C ) \rightarrow ( ( A - B ) + B ) = A \\
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 ) + ( B - C ) ) = ( A - C )
proof -
  have S1: \(( ( A - B ) \in C \land B \in C \land C \in C ) \rightarrow \\
  \(( ( A - B ) + B ) - C ) = \\
  \(( ( A - B ) + ( B - C ) ) \) by (rule MMI_addsubasst)
  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 B \in C \\
  by (rule MMI_3simp2)
  have S5: \(( A \in C \land B \in C \land C \in C ) \rightarrow C \in C \\
  by (rule MMI_3simp3)
  from S1 S3 S4 S5 have S6: \(( A \in C \land B \in C \land C \in C ) \rightarrow \\
  \(( ( A - B ) + B ) - C ) = \\
  \(( ( A - B ) + ( B - C ) ) \\
  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_cpreqtd)
  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 - B ) + B ) + 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 

1282
by (rule MMI_3adant3)

have S4: (A \in \mathbb{C} \land B \in \mathbb{C} \land C \in \mathbb{C}) \implies C \in \mathbb{C} by (rule MMI_3simp3)

have S5: (A \in \mathbb{C} \land B \in \mathbb{C} \land C \in \mathbb{C}) \implies B \in \mathbb{C} by (rule MMI_3simp2)

from S1 S3 S4 S5 have S6: (A \in \mathbb{C} \land B \in \mathbb{C} \land C \in \mathbb{C}) \implies 

( ( (A - B) + C ) + B ) = 

( ( (A - B) + B ) + C ) by (rule MMI_syl3anc)

have S7: (A \in \mathbb{C} \land B \in \mathbb{C}) \implies ( (A - B) + B ) = A 

by (rule MMI_npcant)

from S7 have S8: (A \in \mathbb{C} \land B \in \mathbb{C}) \implies 

( ( (A - B) + B ) + C ) = (A + C) 

by (rule MMI_opreqid)

from S8 have S9: (A \in \mathbb{C} \land B \in \mathbb{C} \land C \in \mathbb{C}) \implies 

( ( (A - B) + B ) + C ) = (A + C) 

by (rule MMI_3adant3)

from S6 S9 show (A \in \mathbb{C} \land B \in \mathbb{C} \land C \in \mathbb{C}) \implies 

( ( (A - B) + C ) + B ) = (A + C) by (rule MMI_eqtrd)

qed

lemma (in MMIIsar0) MMI_subneg: assumes A1: A \in \mathbb{C} and 

A2: B \in \mathbb{C} 

shows (A - ( (-B))) = (A + B) 

proof - 

from A1 have S1: A \in \mathbb{C}.

from A2 have S2: B \in \mathbb{C}.

have S3: (A \in \mathbb{C} \land B \in \mathbb{C}) \implies (A - ( (-B))) = (A + B) 

by (rule MMI_subnegt)

from S1 S2 S3 show (A - ( (-B))) = (A + B) 

by (rule MMI_mp2an)

qed

lemma (in MMIIsar0) MMI_subeq0: assumes A1: A \in \mathbb{C} and 

A2: B \in \mathbb{C} 

shows (A - B) = 0 \iff A = B 

proof - 

from A1 have S1: A \in \mathbb{C}.

from A2 have S2: B \in \mathbb{C}.

from S1 S2 have S3: (A + ( (-B))) = (A - B) 

by (rule MMI_negsub)

from S3 have S4: (A + ( (-B))) = 0 \iff (A - B) = 0 

by (rule MMI_eqeq1i)

have S5: (A + ( (-B))) = 0 \implies 

( (A + ( (-B))) + B ) = (0 + B) by (rule MMI_opreq1)

from S4 S5 have S6: (A - B) = 0 \implies 

( (A + ( (-B))) + B ) = (0 + B) by (rule MMI_sylb1r)

from A1 have S7: A \in \mathbb{C}.

from A2 have S8: B \in \mathbb{C}.

from S8 have S9: ( (-B)) \in \mathbb{C} by (rule MMI_negcl)

from A2 have S10: B \in \mathbb{C}.

from S7 S9 S10 have S11: ( (A + ( (-B))) + B ) = 

1283
\(( ( A + B ) + ( - ( - B ) ) )\) by (rule MMI_add23)

from A1 have S12: A ∈ C.
from A2 have S13: B ∈ C.
from S9 have S14: ( - B ) ∈ C.
from S12 S13 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_opreq2i)
from A1 have S19: A ∈ C.
from S19 have S20: ( A + 0 ) = A by (rule MMI_addid1)
from S18 S20 have S21: ( ( A + ( B + ( - B ) ) ) ) = A
by (rule MMI_opreq2i)
from S15 have S22: ( ( A + ( - B ) ) + B ) = A
by (rule MMI_eqtr)
from S11 S15 have S23: ( A + ( - B ) ) = A
by (rule MMI_addid2)
from S23 have S24: ( A - B ) = 0 by (rule MMI_subid)
from S6 S22 have S25: ( A - B ) = 0 → A = B
by (rule MMI_eqeq12i)
from S4 have S26: 0 ∈ C by (rule MMI_eqeq12i)
from S26 S25 have S27: ( 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 ) ← 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)
from S4 have S5: 0 ∈ C by (rule MMI_0cn)
from S5 have S6: 0 ∈ C by (rule MMI_0cn)
from S6 have S7: B ∈ C.
from S4 S5 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 S9 have S10: ( - B ) = ( 0 - B )
from S10 have S11: ( A + ( - B ) ) = ( A + ( 0 - B ) )
by (rule MMI_opreq2i)
from S11 have S12: A ∈ C.
from S12 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_bitr)
from S3 S9 S20 show ( ( - A ) ) = ( ( - B ) ) ←→ A = B by (rule MMI_3bitr)
qed

lemma (in MMIIsar0) 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_eqcom)
  have S8: A = ( ( - B ) ) ←→ ( ( - B ) ) = A by (rule MMI_eqcom)
  from S3 S7 S8 show ( ( - A ) ) = B ←→ ( ( - B ) ) = A by (rule MMI_3bitr3)
qed

lemma (in MMIIsar0) MMI_negcon2: assumes A1: A ∈ C and
  A2: B ∈ C
  shows A = ( ( - B ) ) ←→ B = ( ( - A ) )
proof -
  from A2 have S1: B ∈ C.
  from A1 have S2: A ∈ C.
  from S1 S2 have S3: ( ( - B ) ) = A ←→ ( ( - A ) ) = B
    by (rule MMI_negcon1)
  have S4: A = ( ( - B ) ) ←→ ( ( - B ) ) = A by (rule MMI_eqcom)
  have S5: B = ( ( - A ) ) ←→ ( ( - A ) ) = B by (rule MMI_eqcom)
  from S3 S4 S5 show A = ( ( - B ) ) ←→ B = ( ( - A ) ) by (rule MMI_3bitr4)
qed

lemma (in MMIIsar0) MMI_neg11t:
  shows ( A ∈ C ∧ B ∈ C ) ←→ ( ( ( - A ) ) = ( ( - B ) ) ←→ A = B )
proof -
  have S1: A = if ( A ∈ C , A , 0 ) → ( ( - A ) ) =
( - if ( A ∈ C , A , 0 ) ) by (rule MMI_negeq)
from S1 have S2: A = if ( A ∈ C , A , 0 ) → ( A = B ) by (rule MMI_negeq)
from S2 S3 have S4: if ( A ∈ C , A , 0 ) = ( ( - A ) = ( - B ) ) = ( - if ( A ∈ C , A , 0 ) ) = ( - if ( A ∈ C , A , 0 ) ) = ( - B ) ) = ( - if ( A ∈ C , A , 0 ) ) by (rule MMI_bibi12d)
have S5: if ( A ∈ C , A , 0 ) = ( - if ( A ∈ C , A , 0 ) ) = ( - if ( A ∈ C , A , 0 ) ) = ( - B ) ) by (rule MMI_negeq)
from S5 have S6: A = if ( B ∈ C , B , 0 ) = ( ( - if ( A ∈ C , A , 0 ) ) = ( - if ( A ∈ C , A , 0 ) ) = ( - if ( B ∈ C , B , 0 ) ) )
by (rule MMI_eqeq2d)
from S6 S7 have S8: A = if ( B ∈ C , B , 0 ) = ( if ( A ∈ C , A , 0 ) = ( - 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 ) ) = ( - B ) ) = ( 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 S1: 0 ∈ C by (rule MMI_0cn)
from S11 have S12: B ∈ 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 ) ) = if ( B ∈ C , B , 0 ) by (rule MMI_neg11)
from S4 S8 S13 show ( A ∈ C ∧ B ∈ C ) → ( ( - A ) = A ) by (rule MMI_dedth2h)

lemma (in MMIar0) 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 ∧ B ∈ C ) → ( - ( - A ) ) = A by (rule MMI_adantr)
from S5 have S6: ( A ∈ C ∧ B ∈ C ) → ( ( - ( - A ) ) = ( - B ) ) = ( ( - A ) ) = B ) by (rule MMI_eqeq2d)
from S3 S6 have S7: ( A ∈ C ∧ B ∈ C ) → ( ( - A ) = B ↔ A

1286
lemma (in MMIar0) MMI_negcon2t:
  shows \((A \in \mathcal{F} \land B \in \mathcal{F}) \rightarrow (A = (-B) \leftrightarrow B = (-A))\)
proof -
  have S1: \((A \in \mathcal{F} \land B \in \mathcal{F}) \rightarrow ((-A) = B \leftrightarrow (-B) = A)\)
    by (rule MMI_negcon1t)
  have S2: \(A = (-B) \leftrightarrow (-B) = A\)
    by (rule MMI_eqcom)
  from S1 S2 have S3: \((A \in \mathcal{F} \land B \in \mathcal{F}) \rightarrow (A = (-B) \leftrightarrow (-B) = A)\)
    by (rule MMI_syl6rbbr)
  have S4: \((-A) = B \leftrightarrow B = (-A)\)
    by (rule MMI_eqcom)
  from S3 S4 have \((A \in \mathcal{F} \land B \in \mathcal{F}) \rightarrow (A = (-B) \leftrightarrow B = (-A))\)
    by (rule MMI_syl6bb)
qed
lemma (in MMIar0) MMI_subcant:
  shows \((A \in \mathcal{F} \land B \in \mathcal{F} \land C \in \mathcal{F}) \rightarrow ((A - B) = (A - C) \leftrightarrow B = C)\)
proof -
  have S1: \((A \in \mathcal{F} \land (-B) \in \mathcal{F} \land (-C) \in \mathcal{F}) \rightarrow ((A + (-B)) = (A + (-C)) \leftrightarrow (-B) = (-C))\)
    by (rule MMI_addcant)
  have S2: \(C \in \mathcal{F} \rightarrow (-C) \in \mathcal{F}\)
    by (rule MMI_negclt)
  from S1 S2 have S3: \((A \in \mathcal{F} \land (-B) \in \mathcal{F} \land (-C) \in \mathcal{F}) \rightarrow ((A + (-B)) = (A + (-C)) \leftrightarrow (-B) = (-C))\)
    by (rule MMI_syl3an3)
  have S4: \(B \in \mathcal{F} \rightarrow (-B) \in \mathcal{F}\)
    by (rule MMI_negclt)
  from S3 S4 have S5: \((A \in \mathcal{F} \land B \in \mathcal{F} \land C \in \mathcal{F}) \rightarrow ((A + (-B)) = (A + (-C)) \leftrightarrow (-B) = (-C))\)
    by (rule MMI_syl3an2)
  have S6: \((A \in \mathcal{F} \land B \in \mathcal{F}) \rightarrow (A + (-B)) = (A - B)\)
    by (rule MMI_negsubt)
  from S6 have S7: \((A \in \mathcal{F} \land B \in \mathcal{F} \land C \in \mathcal{F}) \rightarrow ((A + (-B)) = (A - B) \leftrightarrow A + (-C) = (A - C))\)
    by (rule MMI_3adant3)
  have S8: \((A \in \mathcal{F} \land C \in \mathcal{F}) \rightarrow (A + (-C)) = (A - C)\)
    by (rule MMI_negsubt)
  from S8 have S9: \((A \in \mathcal{F} \land B \in \mathcal{F} \land C \in \mathcal{F}) \rightarrow ((A + (-C)) = (A - C) \leftrightarrow (A - B) = (A - C))\)
    by (rule MMI_3adant2)
  from S7 S9 have S10: \((A \in \mathcal{F} \land B \in \mathcal{F} \land C \in \mathcal{F}) \rightarrow ((A + (-C)) = (A - C) \leftrightarrow (A - B) = (A - C))\)
    by (rule MMI_eqeq12d)
  have S11: \((B \in \mathcal{F} \land C \in \mathcal{F}) \rightarrow ((-B) = (-C) \leftrightarrow B = C)\)
by (rule MMI_neg11t)
from S11 have S12: \((A \in f \land B \in f \land C \in f) \rightarrow \(( - B) = ( - C) \iff B = C)\) by (rule MMI_3adant1)
from S5 S10 S12 show \((A \in f \land B \in f \land C \in f) \rightarrow \((A - B) = (A - C) \iff B = C)\) by (rule MMI_3bitr3d)

qed

lemma (in MMIIsar0) MMI_subcan2t:
shows \((A \in f \land B \in f \land C \in f) \rightarrow \((A - C) = (B - C) \iff A = B)\)
proof -
  have S1: \((A \in f \land C \in f) \rightarrow (A + (- C)) = (A - C)\)
    by (rule MMI_negsubt)
  from S1 have S2: \((A \in f \land B \in f \land C \in f) \rightarrow (A + (- C)) = (A - C)\) by (rule MMI_3adant2)
  have S3: \((B \in f \land C \in f) \rightarrow (B + (- C)) = (B - C)\)
    by (rule MMI_negsubt)
  from S3 have S4: \((A \in f \land B \in f \land C \in f) \rightarrow (B + (- C)) = (B - C)\) by (rule MMI_3adant1)
  from S2 S4 have S5: \((A \in f \land B \in f \land C \in f) \rightarrow ((A + (- C)) = (B + (- C)) \iff (A - C) = (B - C))\)
    by (rule MMI_eqeq12d)
  have S6: \((A \in f \land B \in f \land (- C) \in f) \rightarrow ((A + (- C)) = (B + (- C)) \iff A = B)\)
    by (rule MMI_addcan2t)
  have S7: \(C \in f \rightarrow (- C) \in f\) by (rule MMI_negclt)
  from S6 S7 have S8: \((A \in f \land B \in f \land C \in f) \rightarrow ((A + (- C)) = (B + (- C)) \iff A = B)\)
    by (rule MMI_syl3an3)
  from S5 S8 show \((A \in f \land B \in f \land C \in f) \rightarrow ((A - C) = (B - C) \iff A = B)\) by (rule MMI_bitr3d)
qed

lemma (in MMIIsar0) MMI_subcan: assumes A1: \(A \in f\) and
A2: \(B \in f\) and
A3: \(C \in f\)
shows \((A - B) = (A - C) \iff B = C)\)
proof -
  from A1 have S1: \(A \in f\).
  from A2 have S2: \(B \in f\).
  from A3 have S3: \(C \in f\).
  have S4: \((A \in f \land B \in f \land C \in f) \rightarrow ((A - B) = (A - C) \iff B = C)\)
    by (rule MMI_subcant)
  from S1 S2 S3 S4 show \((A - B) = (A - C) \iff B = C)\)
    by (rule MMI_mp3an)
qed

lemma (in MMIIsar0) MMI_subcan2: assumes A1: \(A \in f\) and
A2: \(B \in f\) and

1288
A3: \( C \subseteq C \)
shows \( (A - C) = (B - C) \longleftrightarrow A = B \)

proof -
from A1 have S1: \( A \in C \).
from A2 have S2: \( B \in C \).
from A3 have S3: \( C \subseteq C \).
have S4: \((A \in C \land B \in C \land C \subseteq C) \rightarrow ((A - C) = (B - C) \leftrightarrow A = B)\) by (rule MMI_subcan2t)
from S1 S2 S3 S4 show \((A - C) = (B - C) \leftrightarrow A = B\)
by (rule MMI_mp3an)
qed

lemma (in MMIIsar0) MMI_subeq0t:
shows \((A \in C \land B \in C) \rightarrow ((A - B) = 0 \leftrightarrow A = B)\)

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)
from S1 have S2: \(A = \text{if}(A \in C, A, 0) \rightarrow (A = B \rightarrow \)
\((\text{if}(A \in C, A, 0) - B) = 0 \leftrightarrow \)
\((\text{if}(A \in C, A, 0) = B)\) by (rule MMI_bibi12d)

have S3: \(A = \text{if}(A \in C, A, 0) \rightarrow (A = B \rightarrow \)
\((\text{if}(A \in C, A, 0) - B) = \)
\((\text{if}(A \in C, A, 0) - B) \leftrightarrow \)
\((\text{if}(A \in C, A, 0) = B)\) by (rule MMI_elimel)

have S4: \(A = \text{if}(A \in C, A, 0) \rightarrow (A = B \rightarrow \)
\((\text{if}(A \in C, A, 0) - B) = 0 \leftrightarrow \)
\((\text{if}(A \in C, A, 0) = B)\) by (rule MMI_eqeq1d)

have S5: \(B = \text{if}(B \in C, B, 0) \rightarrow (B = B \rightarrow \)
\((\text{if}(B \in C, B, 0) - B) = \)
\((\text{if}(B \in C, B, 0) - B) \leftrightarrow \)
\((\text{if}(B \in C, B, 0) = B)\) by (rule MMI_elimel)

have S6: \(B = \text{if}(A \in C, A, 0) \rightarrow (B = B \rightarrow \)
\((\text{if}(A \in C, A, 0) - B) = 0 \leftrightarrow \)
\((\text{if}(A \in C, A, 0) = B)\) by (rule MMI_opreq2)

have S7: \(B = \text{if}(A \in C, A, 0) \rightarrow (B = B \rightarrow \)
\((\text{if}(A \in C, A, 0) - B) = 0 \leftrightarrow \)
\((\text{if}(A \in C, A, 0) = B)\) by (rule MMI_elimel)

have S8: \(B = \text{if}(A \in C, A, 0) \rightarrow (B = B \rightarrow \)
\((\text{if}(A \in C, A, 0) - B) = 0 \leftrightarrow \)
\((\text{if}(A \in C, A, 0) = B)\) by (rule MMI_bibi12d)

have S9: \(0 \in C\) by (rule MMI_0cn)

have S10: \(A \in C \rightarrow \text{if}(A \in C, A, 0) \in C\) by (rule MMI_eqeq1d)

have S11: \(0 \in C\) by (rule MMI_0cn)
from S10 S12 have S13:
\((\text{if}(A \in C, A, 0) - \text{if}(B \in C, B, 0)) = 0 \leftrightarrow \)
if (A ∈ f, A, 0) = if (B ∈ f, 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 MMIIsar0) 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 MMIIsar0) 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_axrnegex)
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
have S6: x ∈ R → x ∈ C by (rule MMI_recnt)
have S7: 0 ∈ C by (rule MMI_0cn)
from A1 have S8: A ∈ R.
from S8 have S9: A ∈ C by (rule MMI_recn)
have S10: (0 ∈ C ∧ A ∈ C ∧ x ∈ C) → ((0 - A) = x ↔
(A + x) = 0) by (rule MMI_subaddt)
from S7 S9 S10 have S11: x ∈ C → ((0 - A) = x ↔
(A + x) = 0) by (rule MMI_mp3an12)
from S6 S11 have S12: x ∈ R → ((0 - A) = x ↔
(A + x) = 0) by (rule MMI_syl)
have S13: ((- A)) = (0 - A) by (rule MMI_df_neg)
from S13 have S14: ((- A)) = x ↔ (0 - A) = x
by (rule MMI_eqeq1)
from S12 S14 have S15: x ∈ R → ((- A) = x ↔
(A + x) = 0) by (rule MMI_sy15bb)
have S16: x ∈ R → ((- A) = x → ((- A) ∈ R)
by (rule MMI_eleq1)
from S15 S16 have S17: x ∈ R → ((A + x) = 0 →
((- A) ∈ R) by (rule MMI_sylbird)
from S17 have (x ∈ R ∧ (A + x) = 0) → ((- A) ∈ R
by (rule MMI_imp)

1290
\( \forall x . ( x \in \mathbb{R} \land ( A + x ) = 0 ) \longrightarrow ( ( - A ) ) \in \mathbb{R} \) by auto

from S18 have S19: \( ( \exists x . ( x \in \mathbb{R} \land ( A + x ) = 0 ) ) \longrightarrow ( ( - A ) ) \in \mathbb{R} \) by (rule MMI_19_23aiv)

from S5 S19 show \( ( ( - A ) ) \in \mathbb{R} \) by (rule MMI_ax_mp)

qed

lemma (in MMIsar0) MMI_renegclt:
shows \( A \in \mathbb{R} \longrightarrow ( ( - A ) ) \in \mathbb{R} \)
proof -
  have S1: \( A = \text{if} ( A \in \mathbb{R} , A , 1 ) \longrightarrow ( ( - A ) ) = ( - \text{if} ( A \in \mathbb{R} , A , 1 ) ) \) by (rule MMI_negeq)
  from S1 have S2: \( A = \text{if} ( A \in \mathbb{R} , A , 1 ) \longrightarrow ( ( - A ) ) \in \mathbb{R} \leftrightarrow \)
    \( ( - \text{if} ( A \in \mathbb{R} , A , 1 ) ) \in \mathbb{R} \) by (rule MMI_eleq1d)
  have S3: \( 1 \in \mathbb{R} \) by (rule MMI_ax1re)
  from S3 have S4: \( \text{if} ( A \in \mathbb{R} , A , 1 ) \in \mathbb{R} \) by (rule MMI_elimel)
  from S4 have S5: \( ( - \text{if} ( A \in \mathbb{R} , A , 1 ) ) \in \mathbb{R} \) by (rule MMI_renegcl)
  from S2 S5 show \( A \in \mathbb{R} \longrightarrow ( ( - A ) ) \in \mathbb{R} \) by (rule MMI_dedth)

qed

lemma (in MMIsar0) MMI_resubclt:
shows \( A \in \mathbb{R} \land B \in \mathbb{R} \longrightarrow ( A - B ) \in \mathbb{R} \)
proof -
  have S1: \( ( A + ( ( - B ) ) ) = ( A - B ) \)
  by (rule MMI_negsubt)
  have S2: \( A \in \mathbb{R} \longrightarrow A \in \mathbb{C} \) by (rule MMI_recnt)
  have S3: \( B \in \mathbb{R} \longrightarrow B \in \mathbb{C} \) by (rule MMI_recnt)
  from S1 S2 S3 have S4: \( A \in \mathbb{R} \land B \in \mathbb{R} \longrightarrow ( A + ( ( - B ) ) ) = ( A - B ) \)
  by (rule MMI_syl2an)
  have S5: \( ( A \in \mathbb{R} \land ( ( - B ) ) \in \mathbb{R} ) \longrightarrow ( A + ( ( - B ) ) ) \in \mathbb{R} \)
  by (rule MMI_axaddrc1)
  have S6: \( B \in \mathbb{R} \longrightarrow ( ( - B ) ) \in \mathbb{R} \) by (rule MMI_renegcl)
  from S5 S6 have S7: \( A \in \mathbb{R} \land B \in \mathbb{R} \longrightarrow ( A + ( ( - B ) ) ) \in \mathbb{R} \)
  by (rule MMI_sylan2)
  from S4 S7 show \( A \in \mathbb{R} \land B \in \mathbb{R} \longrightarrow ( A - B ) \in \mathbb{R} \)
  by (rule MMI_eqeltrrd)

qed

lemma (in MMIsar0) MMI_resubcl: assumes A1: \( A \in \mathbb{R} \) and A2: \( B \in \mathbb{R} \)
shows \( ( A - B ) \in \mathbb{R} \)
proof -
  from A1 have S1: \( A \in \mathbb{R} \).
  from A2 have S2: \( B \in \mathbb{R} \).
  have S3: \( A \in \mathbb{R} \land B \in \mathbb{R} \longrightarrow ( A - B ) \in \mathbb{R} \) by (rule MMI_resubclt)
from S1 S2 S3 show \((A - B) \in \mathbb{R}\) by (rule MMI_mp2an)

qed

lemma (in MMIsarO) MMI_0re:
shows \(0 \in \mathbb{R}\)
proof -
  have S1: \(1 \in \mathbb{C}\) by (rule MMI_1cn)
  from S1 have S2: \((1 - 1) = 0\) by (rule MMI_subid)
  have S3: \(1 \in \mathbb{R}\) by (rule MMI_axire)
  have S4: \(1 \in \mathbb{R}\) by (rule MMI_axire)
  from S3 S4 have S5: \((1 - 1) \in \mathbb{R}\) by (rule MMI_resubcl)
  from S2 S5 show \(0 \in \mathbb{R}\) by (rule MMI_eqeltrr)
qed

lemma (in MMIsarO) MMI_mulid2t:
shows \(A \in \mathbb{F} \rightarrow (1 \cdot A) = A\)
proof -
  have S1: \(1 \in \mathbb{F}\) by (rule MMI_1cn)
  have S2: \((1 \in \mathbb{F} \land A \in \mathbb{F}) \rightarrow (1 \cdot A) = (A \cdot 1)\)
    by (rule MMI_axmulcom)
  from S1 S2 have S3: \(A \in \mathbb{F} \rightarrow (1 \cdot A) = (A \cdot 1)\) by (rule MMI_mpan)
  have S4: \(A \in \mathbb{F} \rightarrow (A \cdot 1) = A\) by (rule MMI_ax1id)
  from S3 S4 show \(A \in \mathbb{F} \rightarrow (1 \cdot A) = A\) by (rule MMI_eqtrd)
qed

lemma (in MMIsarO) MMI_mul12t:
shows \((A \in \mathbb{F} \land B \in \mathbb{F} \land C \in \mathbb{F}) \rightarrow (A \cdot (B \cdot C)) = (B \cdot (A \cdot C))\)
proof -
  have S1: \((A \in \mathbb{C} \land B \in \mathbb{C}) \rightarrow (A \cdot B) = (B \cdot A)\)
    by (rule MMI_axmulcom)
  from S1 have S2: \((A \in \mathbb{C} \land B \in \mathbb{C}) \rightarrow ((A \cdot B) \cdot C) = ((B \cdot A) \cdot C)\)
    by (rule MMI_opreq1d)
  from S2 have S3: \((A \in \mathbb{C} \land B \in \mathbb{C} \land C \in \mathbb{C}) \rightarrow ((A \cdot B) \cdot C) = ((B \cdot A) \cdot C)\)
    by (rule MMI_3adant3)
  have S4: \((A \in \mathbb{C} \land B \in \mathbb{C} \rightarrow (A \cdot B) \cdot C) \rightarrow (A \cdot (B \cdot C))\) by (rule MMI_axmulass)
  have S5: \((B \in \mathbb{C} \land A \in \mathbb{C} \land C \in \mathbb{C}) \rightarrow ((B \cdot A) \cdot C) = (B \cdot (A \cdot C))\)
  from S5 have S6: \((A \in \mathbb{C} \land B \in \mathbb{C} \land C \in \mathbb{C}) \rightarrow ((B \cdot A) \cdot C) = (B \cdot (A \cdot C))\)
  from S3 S4 S6 show \((A \in \mathbb{C} \land B \in \mathbb{C} \land C \in \mathbb{C}) \rightarrow (A \cdot (B \cdot C)) = (B \cdot (A \cdot C))\)
    by (rule MMI_3eqtr3d)
qed

lemma (in MMIsarO) MMI_mul23t:
shows \((A \in \mathbb{C} \land B \in \mathbb{C} \land C \in \mathbb{C}) \rightarrow (A \cdot (B \cdot C)) = (B \cdot (A \cdot C))\)

1292
}\)
\textbf{proof} -
\begin{enumerate}
\item \textbf{have S1:} \(( A \in C \land C \in C ) \rightarrow ( B \cdot C ) = ( C \cdot B )\)
  
  by \textbf{(rule MMI\_axmulcom)}

\item \textbf{from S1 have S2:} \(( B \in C \land C \in C ) \rightarrow ( A \cdot ( B \cdot C ) ) = ( A \cdot ( C \cdot B ) )\)
  
  by \textbf{(rule MMI\_opreq2d)}

\item \textbf{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 \textbf{(rule MMI\_3adant1)}

\item \textbf{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 \textbf{(rule MMI\_axmulass)}

\item \textbf{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 \textbf{(rule MMI\_axmulass)}

\item \textbf{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 \textbf{(rule MMI\_3com23)}

\item \textbf{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 \textbf{(rule MMI\_3eqtr4d)}
\end{enumerate}
\textbf{qed}

\textbf{lemma \textit{(in MMI\_isar0)} MMI\_mul4t:}
\begin{enumerate}
\item \textbf{shows} \(( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \rightarrow ( ( A \cdot B ) \cdot ( C \cdot D ) ) = ( ( A \cdot C ) \cdot ( B \cdot D ) )\)
\end{enumerate}
\textbf{proof} -
\begin{enumerate}
\item \textbf{have S1:} \(( A \in C \land B \in C \land C \in C ) \rightarrow ( ( A \cdot B ) \cdot C ) = ( ( A \cdot C ) \cdot B )\)
  
  by \textbf{(rule MMI\_mul23t)}

\item \textbf{from S1 have S2:} \(( A \in C \land B \in C \land C \in C ) \rightarrow ( ( ( A \cdot B ) \cdot C ) \cdot D ) = ( ( ( A \cdot C ) \cdot B ) \cdot D )\)
  
  by \textbf{(rule MMI\_opreqid)}

\item \textbf{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 \textbf{(rule MMI\_3expa)}

\item \textbf{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 \textbf{(rule MMI\_adantrr)}

\item \textbf{have S5:} \(( ( A \cdot B ) \in C \land C \in C \land D \in C ) \rightarrow ( ( ( A \cdot B ) \cdot C ) \cdot D ) = ( ( A \cdot C ) \cdot ( C \cdot D ) )\)
  
  by \textbf{(rule MMI\_axmulass)}

\item \textbf{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 \textbf{(rule MMI\_3expb)}

\item \textbf{have S7:} \(( A \in C \land B \in C ) \rightarrow ( A \cdot B ) \in C\)
  
  by \textbf{(rule MMI\_axmulcl)}

\item \textbf{from S6 S7 have S8:} \(( ( A \in C \land B \in C ) \land ( ( A \in C \land C \land D \in C ) ) \rightarrow ( ( A \cdot B ) \cdot ( C \cdot D ) ) = ( ( A \cdot C ) \cdot ( B \cdot D ) )\)
\end{enumerate}
\textbf{qed}
by (rule MMI_3expb)
have S11: \(( A \in \mathcal{F} \land C \in \mathcal{F} \) \) \arrow \(( A \cdot C ) \in \mathcal{F}\) by (rule MMI_axmulcl)
from S10 S11 have S12: \(( ( A \in \mathcal{F} \land C \in \mathcal{F} \) \land ( B \in \mathcal{F} \land D \in \mathcal{F} \) \) \arrow
\(( ( A \cdot C ) \cdot B ) \cdot D ) = ( ( A \cdot C ) \cdot ( B \cdot D ) )
by (rule MMI_sylan)
from S12 have S13: \(( ( A \in \mathcal{F} \land C \in \mathcal{F} \) \land ( B \in \mathcal{F} \land D \in \mathcal{F} \) \) \arrow
\(( ( A \cdot C ) \cdot B ) \cdot D ) = ( ( A \cdot C ) \cdot ( B \cdot D ) )
by (rule MMI_3eqtr3d)
qed

lemma (in MMIsar0) MMI_muladdt:
shows \(( ( A \in \mathcal{F} \land B \in \mathcal{F} \) \land ( C \in \mathcal{F} \land D \in \mathcal{F} \) \) \arrow
\(( ( A + B ) \cdot ( C + D ) = ( ( A \cdot C ) + ( D \cdot B ) + ( A \cdot D ) \cdot ( C \cdot B ) ) \)
proof -
have S1: \(( ( A + B ) \in \mathcal{F} \land C \in \mathcal{F} \land D \in \mathcal{F} \) \arrow
\(( ( A + B ) \cdot ( C + D ) ) = ( ( A \cdot C ) + ( A \cdot D ) \cdot ( C \cdot B ) ) \)
by (rule MMI_axdistr)
have S2: \(( A \in \mathcal{F} \land B \in \mathcal{F} \) \arrow \(( A + B ) \in \mathcal{F}\) by (rule MMI_axaddcl)
from S2 have S3: \(( ( A \in \mathcal{F} \land B \in \mathcal{F} \) \land ( C \in \mathcal{F} \land D \in \mathcal{F} \) \) \arrow
\(( A + B ) \in \mathcal{F}\) by (rule MMI_adantr)
have S4: \(( C \in \mathcal{F} \land D \in \mathcal{F} \) \arrow \(( C \in \mathcal{F}\) by (rule MMI_pm3_26)
from S4 have S5: \(( ( A \in \mathcal{F} \land B \in \mathcal{F} \) \land ( C \in \mathcal{F} \land D \in \mathcal{F} \) \) \arrow
\(( C \in \mathcal{F}\) by (rule MMI_adantl)
have S6: \(( C \in \mathcal{F} \land D \in \mathcal{F} \) \arrow \(( D \in \mathcal{F}\) by (rule MMI_pm3_27)
from S6 have S7: \(( ( A \in \mathcal{F} \land B \in \mathcal{F} \) \land ( C \in \mathcal{F} \land D \in \mathcal{F} \) \) \arrow
\(( D \in \mathcal{F}\) by (rule MMI_adantl)
from S1 S3 S5 S7 have S8:
\(( ( A \in \mathcal{F} \land B \in \mathcal{F} \) \land ( C \in \mathcal{F} \land D \in \mathcal{F} \) \) \arrow
\(( ( A + B ) \cdot ( C + D ) ) = ( ( A \cdot C ) + ( A \cdot D ) \cdot ( C \cdot B ) ) \)
by (rule MMI_syl3anc)
have S9: \(( A \in \mathcal{F} \land B \in \mathcal{F} \land C \in \mathcal{F} \) \arrow
\(( ( A + B ) \cdot C ) = ( ( A \cdot C ) + ( B \cdot C ) ) \)
by (rule MMI_adddirt)
from S9 have S10: \(( ( A \in \mathcal{F} \land B \in \mathcal{F} \) \land ( C \in \mathcal{F} \) \) \arrow
\(( ( A + B ) \cdot C ) = ( ( A \cdot C ) + ( B \cdot C ) ) \)
by (rule MMI_3expa)
from S10 have S11: \(( ( A \in \mathcal{F} \land B \in \mathcal{F} \) \land ( C \in \mathcal{F} \land D \in \mathcal{F} \) \) \arrow
1294
( \( A + B \) \cdot C \) = ( \( A \cdot C \) + \( B \cdot C \) )

by (rule MMI_adantrr)

have S12: \( A \in C \land B \in C \land D \in C \) \( \rightarrow \)
( \( A + B \) \cdot D \) = ( \( A \cdot D \) + \( B \cdot D \) )

by (rule MMI_adddirt)

from S12 have S13: \( ( A \in C \land B \in C ) \land D \in C \) \( \rightarrow \)
( \( A + B \) \cdot D \) = ( \( A \cdot D \) + \( B \cdot D \) )

by (rule MMI_3expd)

from S13 have S14: \( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) \) \( \rightarrow \)

( \( A + B \) \cdot D \) = ( \( A \cdot D \) + \( B \cdot D \) )

by (rule MMI_adantrl)

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 \land
( ( A \cdot C ) + ( B \cdot C ) ) + ( ( A \cdot D ) + ( B \cdot D ) ) =
( ( A \cdot C ) + ( ( A \cdot D ) + ( B \cdot D ) ) ) + ( B \cdot C )

by (rule MMI_add23t)

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_adant1l)

have S19: \( B \in C \land C \in C \) \( \rightarrow \) ( B \cdot C ) \in C by (rule MMI_axmulcl)
from S19 have S20: \( B \in C \land ( C \in C \land D \in C ) \) \( \rightarrow \)

( B \cdot C ) \in C by (rule MMI_adantrr)

from S20 have S21: \( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) \) \( \rightarrow \)

( B \cdot C ) \in C by (rule MMI_adant1l)

have S22: \( ( ( A \cdot D ) \in C \land ( B \cdot D ) \in C ) \) \( \rightarrow \)

( ( A \cdot D ) + ( B \cdot D ) ) \in C by (rule MMI_axaddcl)

have S23: \( A \in C \land D \in C \) \( \rightarrow \) ( A \cdot D ) \in C by (rule MMI_axmulcl)

have S24: \( B \in C \land D \in C \) \( \rightarrow \) ( B \cdot D ) \in C by (rule MMI_axmulcl)

from S22 S23 S24 have S25:

( ( A \in C \land D ) \in C ) \land ( ( B \in C \land D ) \in C ) \land ( ( A \cdot D ) + ( B \cdot D ) ) \in C by (rule MMI_syl2an)

from S25 have S26: \( ( A \in C \land B \in C ) \land D \in C \) \( \rightarrow \)

( ( A \cdot D ) + ( B \cdot D ) ) \in C by (rule MMI_anandirs)

from S26 have S27: \( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) \) \( \rightarrow \)

( ( A \cdot D ) + ( B \cdot D ) ) \in C by (rule MMI_adantrl)

from S16 S18 S21 S27 have S28:

( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \( \rightarrow \)

( ( ( A \cdot C ) + ( B \cdot C ) ) + ( ( A \cdot D ) + ( B \cdot D ) ) ) =

1295
( ( ( A \cdot C ) + ( ( A \cdot D ) + ( B \cdot D ) ) ) + ( B \cdot C ) )

by (rule MMI_syl3anc)

have S29: ( B \in C \land D \in C ) \rightarrow ( B \cdot D ) = ( D \cdot B )

by (rule MMI_axmulcom)

from S29 have S30: ( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \rightarrow

( B \cdot D ) = ( D \cdot B ) by (rule MMI_ad2ant2l)

from S30 have S31: ( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \rightarrow

( ( ( A \cdot C ) + ( A \cdot D ) ) + ( B \cdot D ) ) =

( ( ( A \cdot C ) + ( A \cdot D ) ) + ( D \cdot B ) )

by (rule MMI_opreq2d)

have S32: ( ( A \cdot C ) \in C \land ( A \cdot D ) \in C \land ( B \cdot D ) \in C ) \rightarrow

( ( ( A \cdot C ) + ( A \cdot D ) ) + ( B \cdot D ) ) =

( ( A \cdot C ) + ( ( A \cdot D ) + ( B \cdot D ) ) )

by (rule MMI_axaddass)

from S18 have S33:

( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \rightarrow ( A \cdot C ) \in C .

from S23 have S34: ( A \in C \land D \in C ) \rightarrow ( A \cdot D ) \in C .

from S34 have S35: ( A \in C \land ( C \in C \land D \in C ) ) \rightarrow

( A \cdot D ) \in C by (rule MMI_adantlr)

from S35 have S36: ( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \rightarrow

( A \cdot D ) \in C by (rule MMI_adantlr)

from S32 S33 S36 S38 have S39:

( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \rightarrow

( ( ( A \cdot C ) + ( A \cdot D ) ) + ( B \cdot D ) ) =

( ( A \cdot C ) + ( ( A \cdot D ) + ( B \cdot D ) ) ) by (rule MMI_syl3anc)

have S40: ( ( A \cdot C ) \in C \land ( A \cdot D ) \in C \land ( D \cdot B ) \in C ) \rightarrow

( ( ( A \cdot C ) + ( A \cdot D ) ) + ( D \cdot B ) ) =

( ( ( A \cdot C ) + ( D \cdot B ) ) + ( A \cdot D ) ) by (rule MMI_ad23t)

from S18 have S41:

( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \rightarrow ( A \cdot C ) \in C .

from S36 have S42: ( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \rightarrow

( A \cdot D ) \in C .

have S43: ( D \in C \land B \in C ) \rightarrow ( D \cdot B ) \in C by (rule MMI_axmulcl)

from S43 have S44: ( B \in C \land D \in C ) \rightarrow ( D \cdot B ) \in C

by (rule MMI_ancom)

from S44 have S45: ( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \rightarrow

( D \cdot B ) \in C by (rule MMI_ad2ant2l)

from S40 S41 S42 S45 have S46:

( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \rightarrow

( ( ( A \cdot C ) + ( A \cdot D ) ) + ( D \cdot B ) ) =

1296
( ( ( A \cdot C ) + ( D \cdot B ) ) + ( A \cdot D ) ) \text{ by (rule MMI_syl3anc)}

from S31 S39 S46 have S47:
( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \to
( ( A \cdot C ) + ( ( A \cdot D ) + ( B \cdot D ) ) ) =
( ( ( A \cdot C ) + ( B \cdot D ) ) + ( A \cdot D ) ) \text{ by (rule MMI_3eqtr3d)}

have S48: ( B \in C \land C \in C ) \to ( B \cdot C ) = ( C \cdot B )
by (rule MMI_axmulcom)
from S48 have S49: ( ( A \in C \land D \in C ) \land ( B \in C \land C \in C ) ) \to
( B \cdot C ) = ( C \cdot B ) \text{ by (rule MMI_adantl)}
from S49 have S50: ( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \to
( ( A \cdot C ) + ( A \cdot D ) + ( C \cdot B ) ) \text{ by (rule MMI_opreq12d)}

have S52: ( ( ( A \cdot C ) + ( D \cdot B ) ) \in C \land ( A \cdot D ) \in C \land
( C \cdot B ) \in C ) \to
( ( ( ( A \cdot C ) + ( D \cdot B ) ) + ( A \cdot D ) ) + ( C \cdot B ) ) =
( ( ( A \cdot C ) + ( D \cdot B ) ) + ( A \cdot D ) + ( C \cdot B ) ) \text{ by (rule MMI_axaddass)}

have S53: ( ( A \cdot C ) \in C \land ( D \cdot B ) \in C ) \to
( ( A \cdot C ) + ( D \cdot B ) ) \in C \text{ by (rule MMI_axaddcl)}
from S17 have S54: ( A \in C \land C \in C ) \to ( A \cdot C ) \in C .
from S44 have S55: ( B \in C \land D \in C ) \to ( D \cdot B ) \in C .
from S53 S54 S55 have S56:
( ( ( A \in C \land C \in C ) \land ( B \in C \land D \in C ) ) \to
( ( A \cdot C ) + ( D \cdot B ) ) \in C \text{ by (rule MMI_syl2an)}
from S56 have S57: ( ( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \to
( ( A \cdot C ) + ( D \cdot B ) ) \in C \text{ by (rule MMI_an4s)}
from S36 have S58: ( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \to
( A \cdot D ) \in C .
have S59: ( C \in C \land B \in C ) \to ( C \cdot B ) \in C \text{ by (rule MMI_axmulcl)}
from S59 have S60: ( B \in C \land C \in C ) \to ( C \cdot B ) \in C
by (rule MMI_ancoms)
from S60 have S61: ( ( A \in C \land D \in C ) \land ( B \in C \land C \in C ) ) \to
( C \cdot B ) \in C \text{ by (rule MMI_adantl)}
from S61 have S62: ( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \to
( C \cdot B ) \in C \text{ by (rule MMI_an42s)}
from S52 S57 S58 S62 have S63:
( ( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \to
( ( ( A \cdot C ) + ( D \cdot B ) ) + ( A \cdot D ) ) + ( C \cdot B ) ) =

1297
( ( ( A \cdot C ) + ( D \cdot B ) ) + ( ( A \cdot D ) + ( C \cdot B ) ) )
by (rule MMI_syl3anc)
from S28 S51 S63 have S64:
( ( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \rightarrow 
( ( ( A \cdot C ) + ( B \cdot C ) ) + ( ( A \cdot D ) + ( B \cdot D ) ) ) =
( ( ( A \cdot C ) + ( D \cdot B ) ) + ( ( A \cdot D ) + ( C \cdot B ) ) )
by (rule MMI_3eqtrd)
from S8 S15 S64 show ( ( 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_3eqtrd)
qed

lemma (in MMIIsar0) MMI_muladd11t:
shows ( A \in C \land B \in C ) \rightarrow ( ( 1 + A ) \cdot ( 1 + B ) ) =
( ( 1 + A ) + ( B + ( A \cdot B ) ) )
proof -
  have S1: 1 \in C by (rule MMI_1cn)
  have S2: ( ( 1 + A ) \in C \land 1 \in C \land B \in C ) \rightarrow
( ( ( 1 + A ) \cdot ( 1 + B ) ) ) =
( ( ( 1 + A ) \cdot 1 ) + ( ( 1 + A ) \cdot B ) )
   by (rule MMI_axdistr)
  from S1 S2 have S3: ( ( 1 + A ) \in C \land B \in C ) \rightarrow
( ( 1 + A ) \cdot ( 1 + B ) ) =
( ( ( 1 + A ) \cdot 1 ) + ( ( 1 + A ) \cdot B ) )
   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 + B ) ) =
( ( ( 1 + A ) \cdot 1 ) + ( ( 1 + A ) \cdot B ) ) by (rule MMI_sylan)
  from S6 have S8: A \in C \rightarrow ( 1 + A ) \in C .
  have S9: ( 1 + A ) \in C \rightarrow ( ( 1 + A ) \cdot 1 ) = ( 1 + A )
   by (rule MMI_ax1id)
  from S8 S9 have S10: 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: ( 1 \in C \land B \in C ) \rightarrow ( ( 1 + A ) \cdot B ) =
( ( 1 \cdot B ) + ( A \cdot B ) ) by (rule MMI_mp3an1)
  have S15: B \in C \rightarrow ( 1 \cdot B ) = B by (rule MMI_mulid2t)
  from S15 have S16: ( A \in C \land B \in C ) \rightarrow ( 1 \cdot B ) = B
by (rule MMI_adantl)
from S16 have S17:
  ( A ∈ C ∧ B ∈ C ) → ( ( 1 · B ) + ( A · B ) ) =
  ( B + ( A · B ) ) by (rule MMI_opreq1d)
from S14 S17 have S18:
  ( A ∈ C ∧ B ∈ C ) → ( ( 1 + A ) · B ) =
  ( B + ( A · B ) ) by (rule MMI_eqtrd)
from S11 S18 have S19:
  ( A ∈ C ∧ B ∈ C ) → ( ( ( 1 + A ) · 1 ) + (( 1 + A ) · B ) ) =
  ( ( 1 + A ) + ( B + ( A · B ) ) ) by (rule MMI_opreq12d)
from S7 S19 show ( A ∈ C ∧ B ∈ C ) →
  ( ( 1 + A ) · ( 1 + B ) ) =
  ( ( 1 + A ) + ( B + ( A · B ) ) )
  by (rule MMI_eqtrd)
qed

lemma (in MMIsar0) MMI_mul12: assumes A1: A ∈ C and
  A2: B ∈ 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 ) = ( A · ( B · C ) )
    by (rule MMI_mulass)
  from S2 S3 have S5: ( A · B ) = ( B · A ) by (rule MMI_mulcom)
  from S4 have S6: ( ( A · B ) · C ) = ( ( B · A ) · C )
    by (rule MMI_opreq1i)
  from S1 have S7: ( A · B ) = ( B · A ) by (rule MMI_mulass)
  from S4 S7 show ( A · ( B · C ) ) = ( B · ( A · C ) )
    by (rule MMI_3eqtr3)
qed

lemma (in MMIsar0) MMI_mul23: 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 ) =
\[(A \cdot C) \cdot B\] 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 MMIsar0)** MMI_mul4: assumes
A1: \(A \in C\) and
A2: \(B \in C\) and
A3: \(C \in C\) and
A4: \(D \in C\)
shows \((A \cdot B) \cdot (C \cdot D) = (A \cdot C) \cdot (B \cdot D)\)
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: \((A \in C \land B \in C) \land (C \in C \land D \in C)\) \(\longrightarrow\)
\((A \cdot B) \cdot (C \cdot D) = (A \cdot C) \cdot (B \cdot D)\)
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 MMIsar0)** MMI_muladd: assumes
A1: \(A \in C\) and
A2: \(B \in C\) and
A3: \(C \in C\) 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: \((A \in C \land B \in C) \land (C \in C \land D \in C)\) \(\longrightarrow\)
\((A + B) \cdot (C + D) = (A \cdot C) + (D \cdot B) + (A \cdot D) + (C \cdot B)\)
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 MMIar0) 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_anoms)
  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: \((A \in C \land B \in C \land C \in C) \rightarrow (A \cdot (B - C)) = ((A \cdot B) \land (A \cdot (B - C)) \in C \) by (rule MMI_opreq2d)
  from S6 S10 have S11: \((A \in C \land B \in C \land C \in C) \rightarrow ((A \cdot B) \land (A \cdot (B - C)) \in C \) by (rule MMI_eqtr3d)
  have S12: \(((A \cdot B) \land (A \cdot (B - C)) \in C \) = (A \cdot (B - C)) \in C \) by (rule MMI_3adant3)
  from S4 have S18: \((B \in C \land C \in C) \rightarrow (B - C) \in C \). from S17 have S19: \((A \in C \land (B \in C \land C \in C)) \rightarrow (A \cdot (B - C)) \in C \) by (rule MMI_syl3anc)
  from S19 have S20: \((A \in C \land B \in C \land C \in C) \rightarrow (A \cdot (B - C)) \in C \) by (rule MMI_syl3anc)
  from S12 S14 S16 S20 have S21: \((A \in C \land B \in C \land C \in C) \rightarrow ((A \cdot B) - (A \cdot C)) = ((A \cdot B) - ((A \cdot B) - (A \cdot C)) \) by (rule MMI_eqcomd)
  qed
lemma (in MMIsar0) MMI_subdirt:
  shows \(( A \in C \land B \in C \land C \in C ) \rightarrow \(( A - B ) \cdot C ) = ( ( A \cdot C ) - ( B \cdot C ) ) \)
proof -
  have S1: ( C \in C \land A \in C \land B \in C ) \rightarrow ( C \cdot ( A - B ) ) = ( ( C \cdot A ) - ( C \cdot B ) ) by (rule MMI_subdit)
  from S1 have S2: ( A \in C \land B \in C \land C \in C ) \rightarrow ( C \cdot ( A - B ) ) = ( ( C \cdot A ) - ( C \cdot B ) ) by (rule MMI_3coml)
  have S3: ( ( A - B ) \in C \land C \in C ) \rightarrow ( ( A - B ) \cdot C ) = ( C \cdot ( A - B ) ) by (rule MMI_axmulcom)
  from S3 S4 have S5: ( A \in C \land B \in C \land C \in C ) \rightarrow ( ( A - B ) \cdot C ) = ( C \cdot ( A - B ) ) by (rule MMI_3impa)
  have S6: ( A \in C \land B \in C \land C \in C ) \rightarrow ( ( A - B ) \cdot C ) = ( C \cdot ( A - B ) ) by (rule MMI_sylan)
  from S6 have S7: ( A \in C \land B \in C \land C \in C ) \rightarrow ( ( A - B ) \cdot C ) = ( C \cdot ( A - B ) ) by (rule MMI_opreq12d)
  from S2 S6 S7 have S8: ( A \in C \land B \in C \land C \in C ) \rightarrow ( B \cdot C ) = ( C \cdot B ) by (rule MMI_mp3an)
qed

lemma (in MMIsar0) MMI_subdir: assumes A1: A \in C and
  A2: B \in C 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 have S5: ( A \cdot ( B - C ) ) = ( ( A \cdot B ) - ( A \cdot C ) ) by (rule MMI_mp3an)
qed
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 MMIIsar0) **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)
from \(S4\) \(S7\) \(S11\) show \((A \cdot 0) = 0\) by (rule MMI_3eqtr3)
qed

**lemma** (in MMIIsar0) **MMI_mul02**: assumes \(A1: A \in C\)
shows \((0 \cdot A) = 0\)

**proof** -

have \(S1: 0 \in C\) by (rule MMI_0cn)
from \(A1\) have \(S2: A \in C\).
from \(S1\) \(S2\) have \(S3: (0 \cdot A) = (A \cdot 0)\) by (rule MMI_mulcom)
from \(A1\) have \(S4: A \in C\).
from \(S4\) have \(S5: (A \cdot 0) = 0\) by (rule MMI_mul01)
from \(S3\) \(S5\) show \((0 \cdot A) = 0\) by (rule MMI_eqtr)
qed

**lemma** (in MMIIsar0) **MMI_1p1times**: assumes \(A1: A \in C\)
shows \((1 + 1) \cdot A) = (A + A)\)

**proof** -

have \(S1: 1 \in C\) by (rule MMI_1cn)
have \(S2: 1 \in C\) by (rule MMI_1cn)
from \(A1\) have \(S3: A \in C\).
from \(S1\) \(S2\) \(S3\) have \(S4: ((1 + 1) \cdot A) = ((1 \cdot A) + (1 \cdot A))\)
by (rule MMI_adddir)
from A1 have S5: A ∈ C.
from S5 have S6: (1 · A) = A by (rule MMI_mulid2)
from S6 have S7: (1 · A) = A.
from S6 S7 have S8: ((1 · A) + (1 · A)) = (A + A)
  by (rule MMI_opreq12i)
from S4 S8 show ((1 + 1) · A) = (A + A)
  by (rule MMI_eqtr)
qed

lemma (in MMIsar0) MMI_mul01t:
  shows A ∈ C → (A · 0) = 0
proof -
  have S1: A = if (A ∈ C, A, 0) →
    (A · 0) = (if (A ∈ C, A, 0) · 0) by (rule MMI_opreq1)
  from S1 have S2: A = if (A ∈ C, A, 0) →
    (A · 0) = 0 ←→ (if (A ∈ C, A, 0) · 0) = 0) by (rule MMI_eqeq1d)
  have S3: 0 ∈ C by (rule MMI_0cn)
  from S3 have S4: if (A ∈ C, A, 0) ∈ C by (rule MMI_elimel)
  from S4 have S5: if (A ∈ C, A, 0) · 0 = 0 by (rule MMI_mul01)
  from S2 S5 show A ∈ C → (A · 0) = 0 by (rule MMI_dedth)
qed

lemma (in MMIsar0) MMI_mul02t:
  shows A ∈ C → (0 · A) = 0
proof -
  have S1: 0 = if (A ∈ C, A, 0) →
    (0 · A) = (if (A ∈ C, A, 0) · 0) by (rule MMI_axmulcom)
  from S1 have S2: A = if (A ∈ C, A, 0) →
    (A · 0) = 0 ←→ (if (A ∈ C, A, 0) · 0) = 0) by (rule MMI_eqeq1d)
  have S3: A ∈ C by (rule MMI_0cn)
  from S3 have S4: if (A ∈ C, A, 0) ∈ C by (rule MMI_elimel)
  from S4 have S5: if (A ∈ C, A, 0) · 0 = 0 by (rule MMI_mul01)
  from S3 S4 show A ∈ C → (0 · A) = 0 by (rule MMI_eqtrd)
qed

lemma (in MMIsar0) MMI_mulneg1: assumes A1: A ∈ C and
  A2: B ∈ C
  shows ((-A) · B) = (- (A · B))
proof -
  from A2 have S1: B ∈ C.
  from A1 have S2: (B · 0) = 0 by (rule MMI_mul01)
  from A2 have S3: B ∈ C.
  from A1 have S4: A ∈ C.
  from S3 S4 have S5: (B · A) = (A · B) by (rule MMI_mulcom)
  from S2 S5 have S6: (B · 0) - (B · A) = (0 - (A · B))
    by (rule MMI_opreq12i)
  have S7: ( -A ) = (0 - A) by (rule MMI_df_neg)
  from S7 have S8: (((-A) · B)) = ((0 - A) · B)
    by (rule MMI_opreq1i)
  have S9: 0 ∈ C by (rule MMI_0cn)
from A1 have S10: A ∈ C.
from S9 S10 have S11: (0 - A) ∈ C by (rule MMI_subcl)
from A2 have S12: B ∈ C.
from S11 S12 have S13: (0 - A) · B = (B · (0 - A))
  by (rule MMI_mulcom)
from A2 have S14: B ∈ C.
have S15: 0 ∈ C by (rule MMI_0cn)
from A1 have S16: A ∈ C.
from S14 S15 S16 have
  S17: (B · (0 - A)) = ((B · 0) - (B · A))
  by (rule MMI_subdi)
from S8 S13 S17 have
  S18: ((-(A)) · B) = ((B · 0) - (B · A)) by (rule MMI_3eqtr)
have S19: (-(A · B)) = (0 - (A · B)) by (rule MMI_df_neg)
from S6 S18 S19 show ((-(A)) · B) = (-(A · B))
  by (rule MMI_3eqtr4)
qed

lemma (in MMIar0) MMI_mulneg2: 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 S2 have S3: (-(B)) ∈ C by (rule MMI_negcl)
  from S1 S3 have S4: (A · (-(B))) =
  ((-(B)) · A) by (rule MMI_mulcom)
  from A2 have S5: B ∈ C.
  from A1 have S6: A ∈ C.
  from S5 S6 have S7: ((-(B)) · A) =
  (-(B · A)) by (rule MMI_mulneg1)
  from A2 have S8: B ∈ C.
  from A1 have S9: A ∈ C.
  from S8 S9 have S10: (B · A) = (A · B) by (rule MMI_mulcom)
  from S10 have S11: (-(B · A)) =
  ((-(A · B))) by (rule MMI_negeqi)
  from S4 S7 S11 show (A · (-(B))) =
  (-(A · B)) by (rule MMI_3eqtr)
qed

lemma (in MMIar0) MMI_mul2neg: 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 S2 have S3: ( ( - B ) ) ∈ C by (rule MMI_negcl)
from S1 S3 have S4: ( ( ( - A ) ) · ( ( - B ) ) ) =
( ( - ( A · ( ( - B ) ) ) ) ) by (rule MMI_mulneg1)
from A1 have S5: A ∈ C.
from S3 have S6: ( ( - B ) ) ∈ C.
from S5 S6 have S7: ( A · ( ( - B ) ) ) =
( ( ( - B ) ) · A ) by (rule MMI_mulcom)
from A2 have S8: B ∈ C.
from A1 have S9: A ∈ C.
from S8 S9 have S10: ( ( ( - B ) ) · A ) =
( ( ( - B ) ) · A ) by (rule MMI_mulneg1)
from S7 S10 have S11: ( A · ( ( - B ) ) ) =
( ( ( - B ) ) · A ) by (rule MMI_eqtr)
from S11 have S12: ( ( ( - A ) ) · ( ( - B ) ) ) =
( ( ( - B ) ) · A ) by (rule MMI_negeqi)
from A2 have S13: B ∈ C.
from A1 have S14: A ∈ C.
from S13 S14 have S15: ( B · A ) ∈ C by (rule MMI_mulcl)
from S15 have S16: ( ( ( - ( B · A ) ) ) ) =
( B · A ) by (rule MMI_negneg)
from S4 S12 S16 have S17: ( ( ( - A ) ) · ( ( - B ) ) ) =
( ( ( - A ) ) · ( ( - B ) ) ) by (rule MMI_3eqtr)
from A2 have S18: B ∈ C.
from A1 have S19: A ∈ C.
from S18 S19 have S20: ( B · A ) = ( A · B ) by (rule MMI_mulcom)
from S17 S20 show ( ( ( - A ) ) · ( ( - B ) ) ) =
( ( ( - A ) ) · ( ( - B ) ) ) by (rule MMI_eqtr)
qed

lemma (in MMIasar0) 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 ) ) ) ) =
( ( ( 1 · ( 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 ) + ( ( ( - 1 ) ) · B ) ) by (rule MMI_adddi)
have S11: 1 ∈ C by (rule MMI_1cn)
from S3 have S12: \(( A + B ) \in C \).
from S11 S12 have S13: \(( ( - 1 ) \cdot ( A + B ) ) =
\(( ( - 1 ) \cdot ( A + B ) ) \) by (rule MMI_mulneg1)
have S14: \(1 \in C \) by (rule MMI_1cn)
from A1 have S15: \(A \in C \).
from S14 S15 have S16: \(( ( - 1 ) \cdot A ) =
\(( ( - 1 ) \cdot A ) \) by (rule MMI_mulneg1)
from A1 have S17: \(A \in C \).
from S17 have S18: \(( 1 \cdot A ) = A \) by (rule MMI_mulid2)
from S18 have S19: \(( - ( 1 \cdot A ) ) = ( ( - A ) \) by (rule MMI_negeqi)
from S16 S19 have S20: \(( ( - 1 ) \cdot A ) = ( ( - A ) \) by (rule MMI_eqtr)
have S21: \(1 \in C \) by (rule MMI_1cn)
from A2 have S22: \(B \in C \).
from S21 S22 have S23: \(( ( - 1 ) \cdot B ) =
\(( ( - 1 ) \cdot B ) \) by (rule MMI_mulneg1)
from A2 have S24: \(B \in C \).
from S24 have S25: \(( 1 \cdot B ) = B \) by (rule MMI_mulid2)
from S25 have S26: \(( - ( 1 \cdot B ) ) = ( ( - B ) \) by (rule MMI_negeqi)
from S23 S26 have S27: \(( ( - 1 ) \cdot B ) = ( ( - B ) \) by (rule MMI_eqtr)
from S20 S27 have S28: \(( ( ( - 1 ) \cdot A ) + ( ( - 1 ) \cdot B ) ) =
\(( ( ( - A ) ) + ( ( - B ) ) \) by (rule MMI_opreq12i)
from S10 S13 S28 have S29: \(( - ( 1 \cdot ( A + B ) ) ) =
\(( ( - A ) ) + ( ( - B ) ) \) by (rule MMI_3eqtr3)
from S5 S29 show \(( - ( A - B ) ) =
\(( ( ( - A ) ) + ( - ( - B ) ) ) \) by (rule MMI_eqtr3)
qed

lemma (in MMIar0) MMI_negsubdi: assumes A1: \(A \in C \) and
A2: \(B \in C \)
shows \(( - ( A - B ) ) =
\(( ( - A ) ) + B \) 
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 + ( ( - B ) ) ) ) =
\(( ( ( - A ) ) + ( ( - B ) ) ) \) by (rule MMI_negdi)
from A1 have S5: \(A \in C \).
from A2 have S6: \(B \in C \).
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: \(B \in C \).
from S9 have S10: \(( ( ( - B ) ) ) = B \) by (rule MMI_negneg)
from S10 have S11: \(( ( ( - A ) ) + ( ( ( - B ) ) ) ) =
\(( ( ( - A ) ) ) + B \) by (rule MMI_opreq2i)
from S4 S8 S11 show \(( ( A - B ) ) =
\(( ( A ) ) + B \) by (rule MMI_3eqtr3)
qed
lemma (in MMIar0) MMI_negsubdi2: assumes A1: A ∈ ƒ and A2: B ∈ ƒ
shows ( - ( A - B ) ) = ( B - A )
proof -
  from A1 have S1: A ∈ ƒ.
  from A2 have S2: B ∈ ƒ.
  from S1 S2 have S3: ( - ( A - B ) ) =
    ( ( ( - A ) ) + B ) by (rule MMI_negsubdi)
  from A1 have S4: A ∈ ƒ.
  from S4 have S5: ( ( - A ) ) ∈ C by (rule MMI_negcl)
  from A2 have S6: B ∈ ƒ.
  from S5 S6 have S7: ( ( ( - A ) ) + B ) =
    ( B + ( ( - A ) ) ) by (rule MMI_addcom)
  from A2 have S8: B ∈ ƒ.
  from A1 have S9: A ∈ ƒ.
  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 MMIar0) MMI_mulneg1t:
shows ( A ∈ ƒ ∧ B ∈ ƒ ) −→
( ( ( - A ) ) · B ) =
( - ( A · B ) )
proof -
  have S1: A =
    if ( A ∈ ƒ , A , 0 ) −→
    ( ( - A ) ) =
    ( - if ( A ∈ ƒ , A , 0 ) ) by (rule MMI_negeq)
  from S1 have S2: A =
    if ( A ∈ ƒ , A , 0 ) −→
    ( ( ( - A ) ) · B ) =
    ( ( - if ( A ∈ ƒ , A , 0 ) ) · B ) by (rule MMI_opreq1d)
  have S3: A =
    if ( A ∈ ƒ , A , 0 ) −→
    ( A · B ) =
    ( if ( A ∈ ƒ , A , 0 ) · B ) by (rule MMI_opreq1)
  from S3 have S4: A =
    if ( A ∈ ƒ , A , 0 ) −→
    ( ( A · B ) ) =
    ( - ( if ( A ∈ ƒ , A , 0 ) · B ) ) by (rule MMI_negeqd)
  from S2 S4 have S5: A =
    if ( A ∈ ƒ , A , 0 ) −→
    ( ( ( - A ) ) · B ) =
    ( ( - ( A · B ) ) ) ←→
    ( ( - if ( A ∈ ƒ , A , 0 ) ) · B ) =
    ( - ( if ( A ∈ ƒ , A , 0 ) · B ) ) by (rule MMI_eqeq12d)
  have S6: B =
    if ( B ∈ ƒ , B , 0 ) −→
\[
\begin{align*}
&((- \text{if}(A \in C, A, 0)) \cdot B) = \\
&(( - \text{if}(A \in C, A, 0)) \cdot \text{if}(B \in C, B, 0)) \text{ by (rule MMI_opreq2)}
\end{align*}
\]

have S7: B = 

\[
\begin{align*}
&\text{if}(B \in C, B, 0) \quad \text{by (rule MMI_optreq2)}
\end{align*}
\]

from S7 have S8: B = 

\[
\begin{align*}
&((- \text{if}(A \in C, A, 0) \cdot B) = \\
&(- (\text{if}(A \in C, A, 0)) \cdot \text{if}(B \in C, B, 0)) \text{ by (rule MMI_ngeqeqd)}
\end{align*}
\]

from S6 S8 have S9: B = 

\[
\begin{align*}
&\text{if}(B \in C, B, 0) \quad \text{by (rule MMI_0cn)}
\end{align*}
\]

from S9 have S10: B = 

\[
\begin{align*}
&\text{if}(B \in C, B, 0) \quad \text{by (rule MMI_elimel)}
\end{align*}
\]

have S11: \(0 \in F\) by (rule MMI_0cn)

from S11 have S12: \(0 \in F\) by (rule MMI_elimel)

have S13: \(0 \in F\) by (rule MMI_0cn)

from S12 S13 have S14: \(( - \text{if}(A \in C, A, 0)) \cdot \text{if}(B \in C, B, 0)) = \\
\text{by (rule MMI_mulneg1)}
\]

from S5 S9 S14 show \((A \in C \land B \in C) \quad \text{by (rule MMI_dedth2h)}
\]

lemma (in MMIar0) MMI_mulneg2t:

shows \((A \in C \land B \in C) \quad \text{by (rule MMI_dedth2h)}
\]

proof - 

have S1: \((B \in C) \land \text{A} \in C) \quad \text{by (rule MMI_dedth2h)}

from S1 have S2: \((A \in C \land B \in C) \quad \text{by (rule MMI_dedth2h)}

have S3: \((A \in C) \land (\text{B} \in C) \quad \text{by (rule MMI_dedth2h)}

have S4: \((\text{B} \in C) \land \text{A} \in C) \quad \text{by (rule MMI_dedth2h)}

have S5: \((A \in C \land B \in C) \quad \text{by (rule MMI_dedth2h)}

have S6: \((A \in C \land B \in C) \quad \text{by (rule MMI_dedth2h)}

from S6 have S7: \((A \in C \land B \in C) \quad \text{by (rule MMI_dedth2h)}

1309
(- (A \cdot B)) =
(- (B \cdot A)) \text{ by (rule MMI_negeqd)}
\begin{align*}
\text{from S2 S5 S7 show } & (A \in C \land B \in C) \rightarrow \\
(A \cdot (-B)) =
(- (A \cdot B)) \text{ by (rule MMI_3eqtr4d)}
\end{align*}
\text{qed}

\begin{align*}
\text{lemma (in MMIar0) MMI_mulneg12t:} \\
\text{shows } & (A \in C \land B \in C) \rightarrow \\
& ((-A) \cdot B) = \\
& (A \cdot (-B))
\end{align*}
\text{proof -}
\begin{align*}
\text{have S1: } & (A \in C \land B \in C) \rightarrow \\
& ((-A) \cdot B) = \\
& (- (A \cdot B)) \text{ by (rule MMI_mulneg1it)}
\end{align*}
\text{have S2: } (A \in C \land B \in C) \rightarrow
\begin{align*}
(A \cdot (-B)) =
(- (A \cdot B)) \text{ by (rule MMI_mulneg2t)}
\end{align*}
\text{from S1 S2 show } (A \in C \land B \in C) \rightarrow
\begin{align*}
& ((-A) \cdot B) = \\
& (A \cdot (-B)) \text{ by (rule MMI_eqtr4d)}
\end{align*}
\text{qed}

\begin{align*}
\text{lemma (in MMIar0) MMI_mul2negt:} \\
\text{shows } & (A \in C \land B \in C) \rightarrow \\
& ((-A) \cdot (-B)) = \\
& (A \cdot B)
\end{align*}
\text{proof -}
\begin{align*}
\text{have S1: } & A = \\
& \text{if } (A \in C, A, 0) \rightarrow \\
& (-A) = \\
& (- \text{if } (A \in C, A, 0) \text{ by (rule MMI_negeq)} \\
\text{from S1 have S2: } & A = \\
& \text{if } (A \in C, A, 0) \rightarrow \\
& ((-A) \cdot (-B)) = \\
& (- \text{if } (A \in C, A, 0) \cdot (-B)) \text{ by (rule MMI_opreq1d)}
\end{align*}
\text{have S3: } (A \in C, A, 0) \rightarrow
\begin{align*}
& (A \cdot B) = \\
& \text{if } (A \in C, A, 0) \cdot B \text{ by (rule MMI_opreq1)} \\
\text{from S2 S3 have S4: } & A = \\
& \text{if } (A \in C, A, 0) \rightarrow \\
& ((-A) \cdot (-B)) = \\
& (A \cdot B) \leftrightarrow \\
& (- \text{if } (A \in C, A, 0) \cdot (-B)) \text{ by (rule MMI_eqeq12d)}
\end{align*}
\text{have S5: } B =
\begin{align*}
& \text{if } (B \in C, B, 0) \rightarrow \\
& (-B)
\end{align*}
\[
(- \text{if (} B \in \mathbb{C}, B, 0 \text{)} ) \text{ by (rule MMI_negeq)}
\]

from S5 have S6: \(B = \text{if (} B \in \mathbb{C}, B, 0 \text{)} \to \text{if (} A \in \mathbb{C}, A, 0 \text{)} \cdot \text{if (} B \in \mathbb{C}, B, 0 \text{)} \) by (rule MMI_opreq2d)

have S7: \(B = \text{if (} B \in \mathbb{C}, B, 0 \text{)} \to \text{if (} A \in \mathbb{C}, A, 0 \text{)} \cdot B \) by (rule MMI_opreq)

from S6 S7 have S8: \(B = \text{if (} B \in \mathbb{C}, B, 0 \text{)} \to \text{if (} A \in \mathbb{C}, A, 0 \text{)} \cdot \text{if (} B \in \mathbb{C}, B, 0 \text{)} \) by (rule MMI_eqeq12d)

have S9: \(0 \in \mathbb{C} \) by (rule MMI_0cn)

from S9 have S10: \(\text{if (} A \in \mathbb{C}, A, 0 \text{)} \in \mathbb{C} \) by (rule MMI_elimel)

have S11: \(0 \in \mathbb{C} \) by (rule MMI_0cn)

from S11 have S12: \(\text{if (} B \in \mathbb{C}, B, 0 \text{)} \in \mathbb{C} \) by (rule MMI_elimel)

from S10 S12 have S13: \((- \text{if (} A \in \mathbb{C}, A, 0 \text{)} \cdot (- B) \) ) = \(\text{if (} A \in \mathbb{C}, A, 0 \text{)} \cdot \text{if (} B \in \mathbb{C}, B, 0 \text{)} \) \) by (rule MMI_mul2neg)

from S4 S8 S13 show \((A \in \mathbb{C} \land B \in \mathbb{C}) \to (- A) \cdot (- B) = \)

\((A \cdot B) \) by (rule MMI_dedth2h)

qed
from S2 S4 have S5: A =
( ( - ( A + B ) ) =
( ( ( - A ) ) + ( ( - B ) ) ) ←→
( ( - if ( A ∈ C , A , 0 ) + B ) ) =
( ( - if ( A ∈ C , A , 0 ) ) + ( ( - B ) ) ) by (rule MMI_eqeq12d)
have S6: B =
if ( B ∈ C , B , 0 ) →
( if ( A ∈ C , A , 0 ) + B ) =
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 ) ) ) =
( ( ( 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_negeq)
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 ) ) ) =
( ( ( - 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: ( ( ( 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 ) ) =
( ( ( - A ) ) + ( ( - B ) ) ) by (rule MMI_dedth2h)
qed

lemma (in MMIIsar0) MMI_negdi2t:
shows ( A ∈ C ∧ B ∈ C ) →
( ( A + B ) ) = ( ( ( - A ) ) - B )
proof -
have S1: (A ∈ f ∧ B ∈ f) →
( - (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 ∈ f → ((-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 ∈ C ∧ B ∈ C) →
  (- (A + (-B))) =
  (- (A - B)) by (rule MMI_negeqd)
  have S6: B ∈ C → ((-B)) = B by (rule MMI_negnegt)
  from S6 have S7: (A ∈ C ∧ B ∈ C) → (- (-B)) = B
  by (rule MMI_adantl)
  from S7 have S8: (A ∈ C ∧ B ∈ C) →
  ((-A)) + ((-B))) =
  ((-A)) + B) by (rule MMI_opreq2d)
  from S3 S5 S8 show (A ∈ C ∧ B ∈ C) →
  ( (-A - B)) = (((-A)) + B) by (rule MMI_3eqtr3d)
  qed

lemma (in MMIsar0) MMI_negsubdi2t:
shows (A ∈ C ∧ B ∈ C) →
( - (A - B)) = (B - A)

proof -
  have S1: (A ∈ C ∧ B ∈ C) →
  ( - (A - B)) = (((-A)) + B) by (rule MMI_negsubdit)
  have S2: ((-A) ∈ C ∧ B ∈ C) →

1313
\[( ( \neg A ) + B ) = ( B + ( \neg A ) ) \] by (rule MMI_axaddcom)

have \( S3: A \in C \rightarrow ( \neg A ) \in C \) by (rule MMI_negclt)

from \( S2 \) \( S3 \) have \( S4: ( A \in C \land B \in C ) \rightarrow \)

\[( ( \neg A ) + B ) = ( B + ( \neg A ) ) \] by (rule MMI_sylan)

have \( S5: ( B \in C \land A \in C ) \rightarrow \)

\[( B + ( \neg A ) ) = ( B - A ) \] by (rule MMI_ancoms)

from \( S1 \) \( S4 \) \( S6 \) show \( ( A \in C \land B \in C ) \rightarrow \)

\[ - ( A - B ) = ( B - A ) \]

by (rule MMI_3eqtrd)

qed

lemma (in MMIsar0) MMI_subsub2t:

shows \( ( A \in C \land B \in C \land C \in C ) \rightarrow \)

\[ ( A - ( B - C ) ) = ( A + ( C - B ) ) \]

proof -

have \( S1: ( A \in C \land ( B - C ) \in C ) \rightarrow \)

\[ ( A + ( \neg ( B - C ) ) ) = \]

\[ ( A - ( B - C ) ) \] by (rule MMI_negsubt)

have \( S2: ( B \in C \land C \in C ) \rightarrow ( B - C ) \in C \) by (rule MMI_subclt)

from \( S1 \) \( S2 \) have \( S3: ( A \in C \land ( B \in C \land C \in C ) ) \rightarrow \)

\[ ( A + ( \neg ( B - C ) ) ) = \]

\[ ( A - ( B - C ) ) \] by (rule MMI_sylan2)

have \( S5: ( B \in C \land C \in C ) \rightarrow \)

\[ ( - ( B - C ) ) = ( C - B ) \] by (rule MMI_negsubdi2t)

from \( S5 \) have \( S6: ( B \in C \land C \in C ) \rightarrow \)

\[ ( A + ( \neg ( B - C ) ) ) = \]

\[ ( A - ( B - C ) ) \] by (rule MMI_opreq2d)

from \( S6 \) have \( S7: ( A \in C \land B \in C \land C \in C ) \rightarrow \)

\[ ( A + ( \neg ( B - C ) ) ) = \]

\[ ( A + ( C - B ) ) \] by (rule MMI_3adant1)

from \( S4 \) \( S7 \) show \( ( A \in C \land B \in C \land C \in C ) \rightarrow \)

\[ ( A - ( B - C ) ) = ( A + ( C - B ) ) \]

by (rule MMI_eqtr3d)

qed

lemma (in MMIsar0) MMI_subsubt:

shows \( ( A \in C \land B \in C \land C \in C ) \rightarrow \)

\[ ( A - ( B - C ) ) = ( ( A - B ) + C ) \]

proof -

have \( S1: ( A \in C \land B \in C \land C \in C ) \rightarrow \)

\[ ( A - ( B - C ) ) = ( A + ( C - B ) ) \] by (rule MMI_subsub2t)

have \( S2: ( A \in C \land C \in C \land B \in C ) \rightarrow \)

\[ ( A + C ) - B = ( A + ( C - B ) ) \] by (rule MMI_addsubasst)

have \( S3: ( A \in C \land C \in C \land B \in C ) \rightarrow \)

1314
( ( 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 + ( C - B ) ) = ( ( A - B ) + C ) by (rule MMI_3com23)
from S1 S5 show ( 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 + ( C - B ) ) by (rule MMI_3com23)
from S1 S3 show ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( A - ( B - C ) ) = ( ( A + C ) - B )
by (rule MMI_eqtr4d)
qed

lemma (in MMIar0) MMI_subsub4t:
  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 - ( B + C ) ) by (rule MMI_subsubt)
  have S2: C ∈ C → ( - C ) ∈ C by (rule MMI_negclt)
from S1 S2 have S3: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( A - ( B - ( - C ) ) = ( A - ( B + C ) )
  have S4: ( B ∈ C ∧ C ∈ C ) →
( B - ( - C ) ) = ( B + C ) by (rule MMI_subnegt)
from S4 have S5: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( B - ( - C ) ) = ( B + C ) by (rule MMI_3adant1)
from S5 have S6: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( A - ( B - ( - C ) ) ) =
  have S7: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( A - ( B + C ) ) by (rule MMI_opreq2d)
  have S8: ( A ∈ C ∧ B ∈ C ) → ( A - B ) ∈ C by (rule MMI_subc1t)
from S7 S8 have S9: ( ( A ∈ C ∧ B ∈ C ) ∧ C ∈ C ) →
( ( A - B ) + ( - C ) ) =
\[(A - B) - C\] by (rule MMI_sylan)

from S9 have S10: \((A \in C \land B \in C \land C \in C) \rightarrow (A - B) + (-C) = 0\)

\[(A - B) - C\] by (rule MMI_3impa)

from S3 S6 S10 show \((A \in C \land B \in C \land C \in C) \rightarrow (A - B) - C = (A - (B + C))\)

by (rule MMI_3eqtr3rd)

qed

lemma (in MMI_sar0) MMI_sub23t:

shows \((A \in C \land B \in C \land C \in C) \rightarrow (A - B) - C = (A - C) - B\)

proof -

have S1: \((B \in C \land C \in C) \rightarrow (B + C) = (C + B)\) by (rule MMI_axaddcom)

from S1 have S2: \((A \in C \land B \in C \land C \in C) \rightarrow (B + C) = (C + B)\) 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: \((A \in C \land B \in C \land C \in C) \rightarrow (A - C) - B = (A - (B + C))\) by (rule MMI_subsub4t)

have S5: \((A \in C \land C \in C \land B \in C) \rightarrow (A - C) - B = (A - (B + C))\) by (rule MMI_3com23)

from S3 S4 S6 show \((A \in C \land B \in C \land C \in C) \rightarrow (A - (B - C) - C) = (A - B)\)

by (rule MMI_3eqtr4d)

qed

lemma (in MMI_sar0) MMI_nnncant:

shows \((A \in C \land B \in C \land C \in C) \rightarrow (A - (B - C) - C) = (A - B)\)

proof -

have S1: \((A \in C \land (B - C) \in C \land C \in C) \rightarrow (A - B - C) = (A - (B - C))\)

((A - (B - C)) - C) =

(A - ((B - C) + C)) by (rule MMI_subsub4t)

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: \((A \in C \land B \in C \land C \in C) \rightarrow (B - C) \in C\)

(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 have S6: \((A \in C \land B \in C \land C \in C) \rightarrow (A - (B - C) - C) = (A - (B - C))\)

((A - (B - C)) + C) =

(A - ((B - C) + C)) by (rule MMI_syl13anc)

have S7: \((B \in C \land C \in C) \rightarrow (B - C) + C\) by (rule MMI_npcant)

from S7 have S8: \((B \in C \land C \in C) \rightarrow (A - (B - C) + C) = (A - B)\) by (rule MMI_opreq2d)
from S8 have S9: \(( A \in \mathcal{C} \land B \in \mathcal{C} \land C \in \mathcal{C} ) \rightarrow \)
\(( A - ( ( B - C ) + C ) ) = ( A - B )\) by (rule MMI_3adant1)
from S6 S9 show \(( A \in \mathcal{C} \land B \in \mathcal{C} \land C \in \mathcal{C} ) \rightarrow \)
\(( ( A - ( B - C ) ) - C ) = ( A - B )\) by (rule MMI_eqtrd)
qed

lemma (in MMIIsar0) MMI_mnnncan1t:
shows \(( A \in \mathcal{C} \land B \in \mathcal{C} \land C \in \mathcal{C} ) \rightarrow \)
\(( ( A - B ) - ( A - C ) ) = ( C - B )\)
proof -
have S1: \(( ( A - B ) \in \mathcal{C} \land ( A - C ) \in \mathcal{C} ) \rightarrow \)
\(( ( A - B ) + ( - ( A - C ) ) ) = \)
\(( ( A - B ) - ( A - C ) )\) by (rule MMI_negsubt)
have S2: \(( ( A - B ) \in \mathcal{C} \land ( - ( A - C ) ) \in \mathcal{C} ) \rightarrow \)
\(( ( A - B ) + ( - ( A - C ) ) ) = \)
\(( - ( A - C ) ) + ( A - B )\) by (rule MMI_axaddcom)
have S3: \(( A - C ) \in \mathcal{C} \rightarrow \)
\(( - ( A - C ) ) \in \mathcal{C}\) by (rule MMI_negclt)
from S2 S3 have S4: \(( ( A - B ) \in \mathcal{C} \land ( A - C ) \in \mathcal{C} ) \rightarrow \)
\(( ( A - B ) + ( - ( A - C ) ) ) = \)
\(( ( - ( A - C ) ) + ( A - B ) )\) by (rule MMI_sylan2)
from S1 S4 have S5: \(( ( A - B ) \in \mathcal{C} \land ( A - C ) \in \mathcal{C} ) \rightarrow \)
\(( ( A - B ) - ( A - C ) ) = \)
\(( ( - ( A - C ) ) + ( A - B ) )\) by (rule MMI_eqtr3d)
from S6 have S7: \(( A \in \mathcal{C} \land B \in \mathcal{C} \land C \in \mathcal{C} ) \rightarrow \)
\(( A - B ) \in \mathcal{C}\) by (rule MMI_subclt)
have S8: \(( A \in \mathcal{C} \land C \in \mathcal{C} ) \rightarrow \)
\(( A - C ) \in \mathcal{C}\) by (rule MMI_3adant3)
from S8 have S9: \(( A \in \mathcal{C} \land B \in \mathcal{C} \land C \in \mathcal{C} ) \rightarrow \)
\(( A - C ) \in \mathcal{C}\) by (rule MMI_3adant2)
from S5 S7 S9 have S10: \(( A \in \mathcal{C} \land B \in \mathcal{C} \land C \in \mathcal{C} ) \rightarrow \)
\(( ( A - B ) - ( A - C ) ) = \)
\(( ( - ( A - C ) ) + ( A - B ) )\) by (rule MMI_sylanc)
have S11: \(( A \in \mathcal{C} \land C \in \mathcal{C} ) \rightarrow \)
\(( - ( A - C ) ) = ( C - A )\) by (rule MMI_negsubd1t)
from S11 have S12: \(( A \in \mathcal{C} \land B \in \mathcal{C} \land C \in \mathcal{C} ) \rightarrow \)
\(( - ( A - C ) ) = ( C - A )\) by (rule MMI_3adant2)
from S12 have S13: \(( A \in \mathcal{C} \land B \in \mathcal{C} \land C \in \mathcal{C} ) \rightarrow \)
\(( - ( A - C ) ) + ( A - B )\) by (rule MMI_opreqid)
have S14: \(( C \in \mathcal{C} \land A \in \mathcal{C} \land B \in \mathcal{C} ) \rightarrow \)
\(( ( C - A ) + ( A - B ) ) = ( C - B )\) by (rule MMI_npnccant)
from S14 have S15: \(( A \in \mathcal{C} \land B \in \mathcal{C} \land C \in \mathcal{C} ) \rightarrow \)
\(( ( C - A ) + ( A - B ) ) = ( C - B )\) by (rule MMI_3coml)
from S10 S13 S15 show \(( A \in \mathcal{C} \land B \in \mathcal{C} \land C \in \mathcal{C} ) \rightarrow \)
\(( ( A - B ) - ( A - C ) ) = ( C - B )\) by (rule MMI_3eqtrd)
qed
lemma (in MMIsar0) MMI_nncan2t:
  shows ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) −→
  ( ( A - C ) - ( B - C ) ) = ( A - B )
proof -
  have S1: ( A ∈ C ∧ ( B - C ) ∈ C ∧ C ∈ C ) −→
  ( ( A - ( B - C ) ) - C ) =
  ( ( A - C ) - ( B - C ) ) by (rule MMI_sub23t)
  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 ) −→
  ( B - C ) ∈ C by (rule MMI_3simp3)
  from S1 S2 S4 S5 have S6: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) −→
  ( ( A - ( B - C ) ) - C ) = ( ( A - C ) - ( B - C ) ) by (rule MMI_nnncan1t)
  have S7: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) −→
  ( ( A - ( B - C ) ) - C ) = ( A - B ) by (rule MMI_nncant)
  from S6 S7 have S8: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) −→
  ( ( A - C ) - ( B - C ) ) = ( A - B ) by (rule MMI_eqtr3d)
qed

lemma (in MMIsar0) MMI_nncant:
  shows ( A ∈ C ∧ B ∈ C ) −→
  ( A - ( A - B ) ) = B
proof -
  have S1: 0 ∈ C by (rule MMI_0cn)
  have S2: ( A ∈ C ∧ 0 ∈ C ∧ B ∈ C ) −→
  ( ( A - 0 ) - ( A - B ) ) = ( B - 0 ) by (rule MMI_nncan1t)
  from S1 S2 have S3: ( A ∈ C ∧ B ∈ C ) −→
  ( ( A - 0 ) - ( A - B ) ) = ( B - 0 ) by (rule MMI_mp3an2)
  have S4: A ∈ C −→ ( A - 0 ) = A by (rule MMI_subid1)
  from S4 have S5: ( A ∈ C ∧ B ∈ C ) −→ ( A - 0 ) = A
  by (rule MMI_adantr)
  from S5 have S6: ( A ∈ C ∧ B ∈ C ) −→
  ( ( A - 0 ) - ( A - B ) ) =
  ( A - ( A - B ) ) by (rule MMI_opreq1d)
  have S7: B ∈ C −→ ( B - 0 ) = B by (rule MMI_subid1)
  from S7 have S8: ( A ∈ C ∧ B ∈ C ) −→ ( B - 0 ) = B
  by (rule MMI_adant1)
  from S3 S6 S8 show ( A ∈ C ∧ B ∈ C ) −→
  ( A - ( A - B ) ) = B by (rule MMI_3eqtr3d)
qed

lemma (in MMIsar0) MMI_nppcan2t:
  shows ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) −→
  ( ( A - ( B + C ) ) + C ) = ( A - B )
proof -

have S1: \((A \in C \land (B + C) \in C \land C \in C)\) \rightarrow 
\((A - ((B + C) - C)) = \)
\((A - (B + C)) + C)\) by (rule MMI_subsubt)

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_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 have S6: \((A \in C \land B \in C \land C \in C)\) \rightarrow 
\((A - ((B + C) - C)) = \)
\((A - (B + C)) + C)\) by (rule MMI_syl3anc)

have S7: \((B \in C \land C \in C)\) \rightarrow \(((B + C) - C) = B\) by (rule MMI_pncant)

from S7 have S8: \((A \in C \land B \in C \land C \in C)\) \rightarrow 
\(((B + C) - C) = (A - B)\) 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 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 MMIsar0) MMI_mulm1t:
shows \(A \in C\) \rightarrow \((-1) \cdot A) = (\neg A)\)
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_muneg1t)

from S1 S2 have S3: \(A \in C\) \rightarrow 
\((-1) \cdot A) = (-1 \cdot A)\) by (rule MMI_mpan)

have S4: \(A \in C\) \rightarrow \((1 \cdot A) = A\) by (rule MMI_mulid2t)

from S4 have S5: \(A \in C\) \rightarrow \((-1 \cdot A) = (-A)\)

by (rule MMI_negeqd)

from S3 S5 show \(A \in C\) \rightarrow 
\((-1) \cdot A) = (-A)\) by (rule MMI_eqtrd)
qed

lemma (in MMIsar0) MMI_mulm1: assumes \(A1: A \in C\)
shows \((-1) \cdot A) = (-A)\)
proof -

from A1 have S1: \(A \in C\).

have S2: \(A \in C\) \rightarrow \((-1) \cdot A) = (-A)\) by (rule MMI_mulm1t)

from S1 S2 show \((-1) \cdot A) = (-A)\) by (rule MMI_ax_mp)
qed

lemma (in MMIsar0) MMI_sub4t:
shows \((A \in C \land B \in C) \land (C \in C \land D \in C)\) \rightarrow 
\((A + B) - (C + D)) = \)
\((A - C) + (B - D))\)
proof -
    have S1: \(( C \in C \land D \in C ) \rightarrow \)
    \((- ( C + D )) = \)
    \((- ( - C ) + ( - D ))\) by (rule MMI_negdit)
    from S1 have S2: \(( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \rightarrow \)
    \((- ( C + D )) = \)
    \((- ( - C ) + ( - D ))\) by (rule MMI_adantl)
    from S2 have S3: \(( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \rightarrow \)
    \(( ( A + B ) + ( ( - C ) + ( - D )) ) = \)
    \(( ( A + ( - C )) + ( B + ( - D )) )\) by (rule MMI_opreq2d)
    have S4: \(( ( B \in C \land ( C \in C \land D \in C ) ) \land ( C \in C \land D \in C ) ) \rightarrow \)
    \(( ( A + B ) + ( ( - C ) + ( - D )) ) = \)
    \(( ( A + ( - C )) + ( B + ( - D )) )\) by (rule MMI_eqtrd)
    have S10: \(( ( A + B ) \in C \land ( C + D ) \in C ) \rightarrow \)
    \(( ( A + B ) + ( ( - C ) + ( - D )) ) = \)
    \(( ( A + ( - C )) + ( B + ( - D )) )\) by (rule MMI_eqvrd)
    have S11: \(( A \in C \land B \in C ) \rightarrow \)
    \(( ( A + B ) \in C ) \land ( C + D ) \in C ) \rightarrow \)
    \(( ( A + B ) + ( ( - C ) + ( - D )) ) = \)
    \(( ( A + ( - C )) + ( B + ( - D )) )\) by (rule MMI_axaddcl)
    have S12: \(( C \in C \land D \in C ) \rightarrow \)
    \(( C + D ) \in C ) \rightarrow \)
    \(( ( A + B ) + ( C + D ) ) ) \rightarrow \)
    \(( ( A + ( - C )) + ( B + ( - D )) )\) by (rule MMI_axaddcl)
    from S10 S11 S12 have S13:
    \(( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \rightarrow \)
    \(( ( A + B ) + ( ( - C ) + ( - D )) ) = \)
    \(( ( A + ( - C )) + ( B + ( - D )) )\) by (rule MMI_axaddct)
    have S14: \(( A \in C \land C \in C ) \rightarrow \)
    \(( A + ( ( - C ) ) = ( A - C ) )\) by (rule MMI_negsubt)
    from S14 have S15: \(( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \rightarrow \)
    \(( ( A + ( ( - C ) ) = ( A - C ) )\) by (rule MMI_ad2ant2r)
    have S16: \(( B \in C \land D \in C ) \rightarrow \)
    \(( B + ( ( - D ) ) = ( B - D ) )\) by (rule MMI_negsubt)
    from S16 have S17: \(( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \rightarrow \)
    \(( ( B + ( ( - D ) ) = ( B - D ) )\) by (rule MMI_ad2ant2l)
    from S15 S17 have S18: \(( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \rightarrow \)
( ( A + ( - C ) ) + ( B + ( - D ) ) ) =
( ( A - C ) + ( 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 MMIar0) MMI_sub4: assumes
A1: A ∈ ƒ
A2: B ∈ ƒ
A3: C ∈ ƒ
A4: D ∈ ƒ
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

lemma (in MMIar0) MMI_mulsubt:
shows ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →
( ( A - B ) · ( C - D ) ) =
( ( ( A · C ) + ( D · B ) ) - ( ( A · D ) + ( C · B ) ) )
proof -
have S1: ( A ∈ C ∧ B ∈ C ) →
( A + ( - B ) ) = ( A - B ) by (rule MMI_negsubt)
have S2: ( C ∈ C ∧ D ∈ C ) →
( C + ( - D ) ) = ( C - D ) by (rule MMI_negsubt)
from S1 S2 have S3: ( ( A ∈ C ∧ B ∈ C ) ∧ ( C ∈ C ∧ D ∈ C ) ) →

( ( A + ( - B ) ) · ( C + ( - D ) ) ) =
( ( A - B ) · ( C - D ) ) by (rule MMI_opreqan12d)
have S4: ( ( A ∈ C ∧ ( - B ) ∈ C ) ∧ ( C ∈ C ∧ ( - D ) ∈ C ) ) →

( ( A + ( - B ) ) · ( C + ( - D ) ) ) =
( ( ( A · C ) + ( - D ) · ( - B ) ) ) + ( ( A · ( - D ) ) + ( C ·
( - B ) ) ) ) by (rule MMI_muladdt)
have S5: D ∈ C → ( - D ) ∈ C by (rule MMI_negclt)
from S4 S5 have S6: ( ( A ∈ C ∧ ( - B ) ∈ C ) ∧ ( C ∈ C ∧ D ∈ 1321)
\[(A + (-B)) \cdot (C + (-D)) = (A \cdot C) + (-(D) \cdot (-B)) + ((A \cdot (-D)) + (C \cdot (-B)))\] by (rule MMI_sylanr2)

have S7: \(B \in C \implies (-B) \in C\) by (rule MMI_negclt)

from S6 S7 have S8: \((A \in C \land B \in C) \land (C \in C \land D \in C)\) \(-->

\[(A + (-B)) \cdot (C + (-D)) = (A \cdot C) + (-(D) \cdot (-B)) + ((A \cdot (-D)) + (C \cdot (-B)))\] by (rule MMI_sylan12)

have S9: \((D \in C \land B \in C)\) \(-->

\[(-D) \cdot (-B) = (D \cdot B)\] by (rule MMI_mul2negt)

from S9 have S10: \((B \in C \land D \in C)\) \(-->

\[(A \cdot C) + ((-D) \cdot (B))\) by (rule MMI_opreq2d)

from S11 have S12: \((A \in C \land D \in C) \land (C \in C \land D \in C)\) \(-->

\[(A \cdot C) + ((-D) \cdot (-B))\] \(-->

\[(A \cdot D) \cdot (C \cdot B)\] by (rule MMI_mulneg2t)

have S13: \((A \in C \land D \in C)\) \(-->

\[A \cdot (-D) = (-A \cdot D)\] by (rule MMI_mulneg2t)

have S14: \((C \in C \land B \in C)\) \(-->

\[(C \cdot (-B)) = (-C \cdot B)\] by (rule MMI_mulneg2t)

from S13 S14 have S15: \((A \in C \land D \in C) \land (C \in C \land B \in C)\) \(-->

\[(A \cdot (-D)) + (C \cdot (-B))\] \(-->

\[(-A \cdot D) + (-C \cdot B)\] by (rule MMI_negdit)

have S16: \((A \in C \land D \in C) \land (C \in C \land B \in C)\) \(-->

\[(-A \cdot D) + (C \cdot B)\] \(-->

\[(-A \cdot D) + (-C \cdot B)\] by (rule MMI_opreqan12d)

have S17: \((A \in C \land D \in C)\) \(-->

\[(A \cdot D) \in C\) by (rule MMI_axmulc1)

have S18: \((C \in C \land B \in C)\) \(-->

\[(C \cdot B) \in C\) by (rule MMI_axmulc1)

from S16 S17 S18 have S19: \((A \in C \land D \in C) \land (C \in C \land B \in C)\) \(-->

\[(-A \cdot D) + (C \cdot B)\] \(-->

\[(-A \cdot D) + (-C \cdot B)\] by (rule MMI_syl12an)

from S15 S19 have S20: \((A \in C \land D \in C) \land (C \in C \land B \in C)\) \(-->

\[(A \cdot (-D)) + (C \cdot (-B))\] \(-->

\[(-A \cdot D) + (C \cdot B)\] by (rule MMI_eqtr4d)

from S20 have S21: \((A \in C \land D \in C) \land (C \in C \land B \in C)\) \(-->

\[(A \cdot (-D)) + (C \cdot (-B))\] \(-->

\[(-A \cdot D) + (C \cdot B)\] by (rule MMI_ancom2s)

from S21 have S22: \((A \in C \land B \in C) \land (C \in C \land D \in C)\) \(-->

1322
\[
( ( A \cdot ( - D ) ) + ( C \cdot ( - B ) ) ) = \\
( - ( ( A \cdot D ) + ( C \cdot B ) ) ) \text{ by (rule MMI_an42s)}
\]
from S12 S22 have S23: \(( ( A \in C \land B \in C ) \land ( C \in C \land D \in C ) ) \)
\(\rightarrow\)
\[
( ( ( A \cdot C ) + ( - D ) \cdot ( - B ) ) ) + \\
( ( A \cdot ( - D ) ) + ( C \cdot ( - B ) ) ) ) = \\
( ( ( A \cdot C ) + ( D \cdot B ) ) + ( ( A \cdot D ) + \\
( C \cdot B ) ) ) \text{ by (rule MMI_opreq12d)}
\]
have S24: \(( ( ( A \cdot C ) + ( D \cdot B ) ) ) \in C \land ( ( A \cdot D ) + \\
( C \cdot B ) ) ) \in C \)
\(\rightarrow\)
\[
( ( ( A \cdot C ) + ( D \cdot B ) ) ) + ( - ( ( A \cdot D ) + ( C \cdot B ) ) ) ) = \\
( ( ( A \cdot C ) + ( D \cdot B ) ) ) - ( ( ( A \cdot D ) + ( C \cdot B ) ) ) \text{ by (rule MMI_negsubt)}
\]
have S25: \(( ( A \cdot C ) ) \in C \land ( D \cdot B ) \in C ) \rightarrow \\
( ( A \cdot C ) + ( D \cdot B ) ) \in C \text{ by (rule MMI_axaddcl)}
\]
have S26: \(( A \in C \land C \in C ) \rightarrow ( A \cdot C ) ) \in C \text{ by (rule MMI_axmulcl)}
\]
have S27: \(( D \in C \land B \in C ) \rightarrow ( D \cdot B ) ) \in C \text{ by (rule MMI_axmulcl)}
\]
from S27 have S28: \(( B \in C \land D \in C ) \rightarrow ( D \cdot B ) ) \in C \\
by (rule MMI_ancoms)
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 \text{ by (rule MMI_syl12an)}
\]
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 \text{ by (rule MMI_an4s)}
\]
have S31: \(( ( A \cdot D ) ) \in C \land ( C \cdot B ) ) \in C ) \rightarrow \\
( ( A \cdot D ) + ( C \cdot B ) ) ) \in C \text{ by (rule MMI_axaddcl)}
\]
from S17 have S32: \(( A \in C \land D \in C ) \rightarrow ( A \cdot D ) ) \in C .
\]
from S18 have S33: \(( C \in C \land B \in C ) \rightarrow ( C \cdot B ) ) \in C .
\]
from S33 have S34: \(( B \in C \land C \in C ) \rightarrow ( C \cdot B ) ) \in C \\
by (rule MMI_ancoms)
from S31 S32 S34 have S35: \\
( ( A \in C \land D \in C ) ) \land ( B \in C \land C \in C ) ) \rightarrow \\
( ( A \cdot D ) + ( C \cdot B ) ) ) \in C \text{ by (rule MMI_syl12an)}
\]
from S35 have S36: \(( ( A \in C \land B \in C ) ) \land ( C \in C \land D \in C ) ) \rightarrow \\
( ( A \cdot D ) + ( C \cdot B ) ) ) \in C \text{ by (rule MMI_an42s)}
\]
from S24 S30 S36 have S37: \\
( ( 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 ) ) ) ) = \\
( ( ( A \cdot C ) + ( D \cdot B ) ) ) - ( ( ( A \cdot D ) + ( C \cdot B ) ) ) \text{ by (rule MMI_sylanc)}
\]
from S8 S23 S37 have S38: \(( ( 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 ) + ( D \cdot B ) ) ) - ( ( ( A \cdot D ) + ( C \cdot B ) ) ) \\
\text{ by (rule MMI_3eqtrd)}
\]
from S3 S38 show \(( ( A \in C \land B \in C ) ) \land ( C \in C \land D \in C ) ) \rightarrow \\
1323
((A - B) · (C - D)) =
((((A · C) + (D · B)) - ((A · D) + (C · B)))
by (rule MMI_eqtr3d)
qed

lemma (in MMIsar0) MMI_pnpcan1t:
  shows (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
  ((A + B) - (A + C)) = (B - C)
proof -
  have S1: (A ∈ C ∧ B ∈ C) ∧ (A ∈ C ∧ C ∈ C) →
  ((A + B) - (A + C)) =
  ((A - A) + (B - C)) by (rule MMI_sub4t)
  from S1 have S2: (A ∈ C ∧ (B ∈ C ∧ C ∈ C)) →
  ((A + B) - (A + C)) =
  ((A - A) + (B - C)) by (rule MMI_anandis)
  have S3: A ∈ C → (A - A) = 0 by (rule MMI_subidt)
  from S3 have S4: A ∈ C →
  ((A - A) + (B - C)) = (0 + (B - C)) by (rule MMI_opreq1d)
  have S5: (B ∈ C ∧ C ∈ C) → (B - C) ∈ C by (rule MMI_subclt)
  have S6: (B - C) ∈ C →
  (0 + (B - C)) = (B - C) by (rule MMI_addid2t)
  from S5 S6 have S7: (B ∈ C ∧ C ∈ C) →
  (0 + (B - C)) = (B - C) by (rule MMI_syl)
  have S8: (A ∈ C ∧ (B ∈ C ∧ C ∈ C)) →
  ((A - A) + (B - C)) = (B - C) by (rule MMI_sylan9eq)
  from S2 S8 have S9: (A ∈ C ∧ (B ∈ C ∧ C ∈ C)) →
  ((A + B) - (A + C)) = (B - C) by (rule MMI_eqtrd)
  from S9 show (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
  ((A + B) - (A + C)) = (B - C) by (rule MMI_3impb)
qed

lemma (in MMIsar0) MMI_pnpcan2t:
  shows (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
  ((A + C) - (B + C)) = (A - B)
proof -
  have S1: (A ∈ C ∧ C ∈ C) →
  (A + C) = (C + A) by (rule MMI_axaddcom)
  from S1 have S2: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
  (A + C) = (C + A) by (rule MMI_3adant2)
  have S3: (B ∈ C ∧ C ∈ C) →
  (B + C) = (C + B) by (rule MMI_axaddcom)
  from S3 have S4: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
  (B + C) = (C + B) by (rule MMI_3adant1)
  from S2 S4 have S5: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
  ((A + C) - (B + C)) =
  ((C + A) - (C + B)) by (rule MMI_opreq12d)
  have S6: (C ∈ C ∧ A ∈ C ∧ B ∈ C) →
  ((C + A) - (C + B)) = (A - B) by (rule MMI_pnpcant)

1324
from S6 have S7: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
((C + A) - (C + B)) = (A - B) by (rule MMI_3coml)
from S5 S7 show (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
((A + C) - (B + C)) = (A - B) by (rule MMI_eqtrd)
qed

lemma (in MMIasar0) MMI_pnncant:
  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_pnncant)
  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: (A ∈ C ∧ C ∈ C) →
  (A + (-C)) = (A - C) by (rule MMI_negsubt)
  from S4 have S5: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
  ((A + B) - (A + (-C))) =
  (A + (-C)) by (rule MMI_axaddcom)
  from S1 have S6: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
  (A + (C + B)) - (A + (-C)) =
  ((B - (-C)) by (rule MMI_3adant2)
  from S5 have S7: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
  ((A + B) - (A + (-C))) =
  (A + B) - (A - C) by (rule MMI_3adant3)
  from S7 have S8: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
  ((A + B) - (A - C)) = (B + C) by (rule MMI_3eqtr3d)
qed

lemma (in MMIasar0) MMI_ppncant:
  shows (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
  ((A + B) + (C - B)) = (A + C)
proof -
  have S1: (A ∈ C ∧ B ∈ C) →
  (A + B) = (B + A) by (rule MMI_axaddcom)
  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) →
  ((A + B) - (B - C)) =
  (B + A) - (B - C) by (rule MMI_opreq1d)
  have S4: ((A + B) ∈ C ∧ B ∈ C ∧ C ∈ C) →
  ((A + B) - (B - C)) =
  ((A + B) + (C - B)) by (rule MMI_subsub2t)
  have S5: (A ∈ C ∧ B ∈ C) → (A + B) ∈ C by (rule MMI_axaddcl)
  from S5 have S6: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
(A + B) ∈ C by (rule MMI_3adant3)

have S7: (A ∈ C ∧ B ∈ C ∧ C ∈ C) → B ∈ C by (rule MMI_3simp2)

have S8: (A ∈ C ∧ B ∈ C ∧ C ∈ C) → C ∈ C by (rule MMI_3simp3)

from S4 S6 S7 S8 have S9: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →

((A + B) - (B - C)) =

((A + B) + (C - B)) by (rule MMI_syl3anc)

have S10: (B ∈ C ∧ A ∈ C ∧ C ∈ C) →

((B + A) - (B - C)) = (A + C) by (rule MMI_pnncant)

from S10 have S11: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →

((B + A) - (B - C)) = (A + C) by (rule MMI_3com12)

from S11 have S12: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →

((B + A) - (B - C)) = (A + C) by (rule MMI_3eqtr3d)

qed

lemma (in MMIIsar0) MMI_pnncan: 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 A2 have S2: B ∈ C.
from A3 have S3: C ∈ C.

have S4: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →

((A + B) - (A - C)) = (B + C) by (rule MMI_pnncant)

from S4 S1 S2 S3 show ((A + B) - (A - C)) = (B + C) by (rule MMI_mp3an)

qed

lemma (in MMIIsar0) MMI_mulcan: assumes A1: A ∈ C and
A2: B ∈ C and
A3: C ∈ C and
A4: A ≠ 0
shows (A · B) = (A · C) ↔ B = C

proof -
from A1 have S1: A ∈ C.
from A4 have S2: A ≠ 0.
from S1 S2 have S3: ∃x ∈ C . (A · x) = 1 by (rule MMI_recex)
from A1 have S4: A ∈ C.
from A2 have S5: B ∈ C.

{ fix x have S6: (x ∈ C ∧ A ∈ C ∧ B ∈ C) →

  ((x · A) · B) = (x · (A · B)) by (rule MMI_axmulass)

  from S5 S6 have S7: (x ∈ C ∧ A ∈ C) →

  ((x · A) · B) = (x · (A · B)) by (rule MMI_mp3an3)

  from A3 have S8: C ∈ C.

  have S9: (x ∈ C ∧ A ∈ C ∧ C ∈ C) →

  ((x · A) · C) = (x · (A · C)) by (rule MMI_axmulass)

  from S8 S9 have S10: (x ∈ C ∧ A ∈ C) →

  ((x · A) · C) = (x · (A · C)) by (rule MMI_mp3an3)

1326
from S7 S10 have S11: \(( x \in C \land A \in C ) \rightarrow \)
\(( ( x \cdot A ) \cdot B ) = ( x \cdot ( A \cdot C ) ) \) by (rule MMI_eqeq12d)

from S4 S11 have S12: \( x \in C \rightarrow \)
\(( ( x \cdot A ) \cdot B ) = ( x \cdot ( A \cdot C ) ) \) by (rule MMI_mpan2)

have S13:
\(( A \cdot B ) = ( A \cdot C ) \rightarrow ( x \cdot B ) = ( x \cdot C ) \) by (rule MMI_opreq2)

from S12 S13 have S14: \( x \in C \rightarrow \)
\(( A \cdot B ) = ( A \cdot C ) \rightarrow ( x \cdot ( A \cdot C ) ) \) by (rule MMI_syl5bir)

from S14 have S15: \(( x \in C \land ( A \cdot x ) = 1 ) \rightarrow \)
\(( x \cdot ( A \cdot C ) ) = ( x \cdot A ) \) by (rule MMI_adantr)

have S17: \(( A \in C \land x \in C ) \rightarrow \)
\(( A \cdot x ) = ( x \cdot A ) \) by (rule MMI_axmulcom)

from S16 S17 have S18: \( x \in C \rightarrow ( A \cdot x ) = ( x \cdot A ) \)
by (rule MMI_mpan)

from S18 have S19: \( x \in C \rightarrow \)
\(( ( A \cdot x ) = 1 \leftrightarrow ( x \cdot A ) = 1 ) \) by (rule MMI_eqeq1id)

have S20: \( ( x \cdot A ) = 1 \rightarrow ( x \cdot A ) = ( 1 \cdot B ) \) 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: \( ( x \cdot A ) = 1 \rightarrow ( x \cdot A ) = B \)
by (rule MMI_syl6eq)

have S24: \( ( x \cdot A ) = 1 \rightarrow ( x \cdot A ) = ( 1 \cdot C ) \) 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: \( ( x \cdot A ) = 1 \rightarrow ( x \cdot A ) = C \)
by (rule MMI_syl6eq)

from S23 S27 have S28: \( ( x \cdot A ) = 1 \rightarrow \)
\(( ( x \cdot A ) \cdot B ) = ( x \cdot A ) \) by (rule MMI_epeq12d)

from S19 S28 have S29: \( x \in C \rightarrow \)
\(( ( x \cdot A ) \cdot B ) = ( x \cdot A ) \) by (rule MMI_syl6bi)

from S29 have S30:
\(( x \in C \land ( A \cdot x ) = 1 ) \rightarrow \)
\(( ( x \cdot A ) \cdot B ) = 1 \)

1327
\[( ( x \cdot A ) \cdot C ) \leftrightarrow B = C \] by (rule MMI_imp)

From S15 S30 have S31:
\[( x \in C \land ( A \cdot x ) = 1 ) \rightarrow ( ( A \cdot B ) = ( A \cdot C ) \rightarrow B = C )\]
by (rule MMI_sylibd)

From S31 have \( x \in C \rightarrow ( ( A \cdot x ) = 1 \rightarrow ( ( A \cdot B ) = ( A \cdot C ) \rightarrow B = C ) )\)
by (rule MMI_ex)

\} 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 show (A \cdot B ) = ( A \cdot C ) \leftrightarrow B = C by (rule MMI_impbi)

qed

lemma (in MMIar0) MMI_mulcant2: assumes A1: A \neq 0
shows ( A \in C \land B \in C \land C \in C \land ( 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
  ( A \cdot C ) =
  ( if ( A \in C , A , 1 ) \cdot C ) by (rule MMI_opreq1)
  from S1 S2 have S3: A =
  if ( A \in C , A , 1 ) \rightarrow
  ( ( A \cdot B ) = ( A \cdot C ) \leftrightarrow B = C )
  ( if ( A \in C , A , 1 ) \cdot B ) =
  ( if ( A \in C , A , 1 ) \cdot C ) \leftrightarrow
  B = C ) by (rule MMI_eqeq12d)
  from S3 have S4: A =
  if ( A \in C , A , 1 ) \rightarrow
  ( ( ( A \cdot B ) = ( A \cdot C ) \leftrightarrow B = C ) \leftrightarrow
  ( if ( A \in C , A , 1 ) \cdot B ) =
  ( if ( A \in C , A , 1 ) \cdot C ) \leftrightarrow
  B = C ) ) by (rule MMI_bibi1d)
  have S5: B =
  if ( B \in C , B , 1 ) \rightarrow
  ( if ( A \in C , A , 1 ) \cdot B ) =
  ( if ( A \in C , A , 1 ) \cdot if ( B \in C , B , 1 ) ) by (rule MMI_opreq2)
  from S5 have S6: B =
  if ( B \in C , B , 1 ) \rightarrow
  ( ( if ( A \in C , A , 1 ) \cdot B ) =

1328
( if ( A ∈ C , A , 1 ) · C ) ↔
( if ( A ∈ C , A , 1 ) · ( if ( B ∈ C , B , 1 ) ) =
( if ( A ∈ C , A , 1 ) · C ) ) by (rule MMI_eqeq1d)

have S7: B =
if ( B ∈ C , B , 1 ) →
( B = C ↔ if ( B ∈ C , B , 1 ) = C ) by (rule MMI_eqeq1)

from S6 S7 have S8: B =
if ( B ∈ C , B , 1 ) →
( ( ( if ( A ∈ C , A , 1 ) · B ) = ( if ( A ∈ C , A , 1 ) · C ) ↔
B = C ) ↔
( ( ( if ( A ∈ C , A , 1 ) · if ( B ∈ C , B , 1 ) ) =
( if ( A ∈ C , A , 1 ) · C ) ↔
if ( B ∈ C , B , 1 ) = C ) ) by (rule MMI_bibi12d)

have S9: C =
if ( C ∈ C , C , 1 ) →
( if ( A ∈ C , A , 1 ) · C ) =
( if ( A ∈ C , A , 1 ) · ( if ( C ∈ C , C , 1 ) ) ) by (rule MMI_opreq2)

from S9 have S10: C =
if ( C ∈ C , C , 1 ) →
( ( ( if ( A ∈ C , A , 1 ) · if ( B ∈ C , B , 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 , 1 ) ) ) ) by (rule MMI_eqeq2d)

have S11: C =
if ( C ∈ C , C , 1 ) →
( if ( B ∈ C , B , 1 ) =
C ↔
if ( B ∈ C , B , 1 ) =
if ( C ∈ C , C , 1 ) ) by (rule MMI_eqeq2)

from S10 S11 have S12: C =
if ( C ∈ C , C , 1 ) →
( ( ( if ( A ∈ C , A , 1 ) · if ( B ∈ C , B , 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 , 1 ) ) ) ↔
if ( B ∈ C , B , 1 ) =
if ( C ∈ C , C , 1 ) ) ) ) by (rule MMI_bibi12d)

have S13: 1 ∈ C by (rule MMI_1cn)

from S13 have S14: if ( A ∈ C , A , 1 ) ∈ C by (rule MMI_eliml)

have S15: 1 ∈ C by (rule MMI_1cn)

from S15 have S16: if ( B ∈ C , B , 1 ) ∈ C by (rule MMI_eliml)

have S17: 1 ∈ C by (rule MMI_1cn)

from S17 have S18: if ( C ∈ C , C , 1 ) ∈ C by (rule MMI_eliml)

have S19: A =
if ( A ∈ C , A , 1 ) →
( A ≠ 0 ↔ if ( A ∈ C , A , 1 ) ≠ 0 ) by (rule MMI_neeq1)

have S20: 1 =
if ( A ∈ C , A , 1 ) →
( 1 ≠ 0 ↔ if ( A ∈ C , A , 1 ) ≠ 0 ) by (rule MMI_neeq1)
from A1 have S21: $A \neq 0$.
have S22: $1 \neq 0$ by (rule MMI_ax1ne0)
from S19 S20 S21 S22 have S23: if $(A \in C, A, 1) \neq 0$ by (rule MMI_keephyp)
from S14 S16 S18 S23 have S24: $(A \in C \wedge A, 1) \cdot (B \in C, B, 1) = (C \in C, C, 1)$ by (rule MMI_mulcan)
from S4 S8 S12 S24 show $(A \in C \wedge B \in C \wedge C \in C) \rightarrow ((A \cdot B) = (A \cdot C) \leftrightarrow B = C)$ by (rule MMI_dedth3h)

qed

lemma (in MMIIsar0) MMI_mulcant:
shows $(A \in C \wedge B \in C \wedge C \in C) \wedge A \neq 0 \rightarrow ((A \cdot B) = (A \cdot C) \leftrightarrow B = C)$
proof -

have S1: $A =$
if $(A \neq 0, A, 1) \rightarrow
(A \in C \leftrightarrow (A \neq 0, A, 1) \in C)$ by (rule MMI_eleq1)
have S2: $A =$
if $(A \neq 0, A, 1) \rightarrow
(B \in C \leftrightarrow B \in C)$ by (rule MMI_pm4_2i)
have S3: $A =$
if $(A \neq 0, A, 1) \rightarrow
(C \in C \leftrightarrow C \in C)$ by (rule MMI_pm4_2i)
from S1 S2 S3 have S4: $A =$
if $(A \neq 0, A, 1) \rightarrow
((A \in C \wedge B \in C \wedge C \in C) \leftrightarrow
(if (A \neq 0, A, 1) \in C \wedge B \in C \wedge C \in C))$ by (rule MMI_3anbi123d)
have S5: $A =$
if $(A \neq 0, A, 1) \rightarrow
(A \cdot B) =
(if (A \neq 0, A, 1) \cdot B)$ by (rule MMI_opreq1)
have S6: $A =$
if $(A \neq 0, A, 1) \rightarrow
(A \cdot C) =
(if (A \neq 0, A, 1) \cdot C)$ by (rule MMI_opreq1)
from S5 S6 have S7: $A =$
if $(A \neq 0, A, 1) \rightarrow
((A \cdot B) = (A \cdot C) \leftrightarrow
(if (A \neq 0, A, 1) \cdot B) =
(if (A \neq 0, A, 1) \cdot C))$ by (rule MMI_eqeq12d)
from S7 have S8: $A =$
if $(A \neq 0, A, 1) \rightarrow
((A \cdot B) = (A \cdot C) \leftrightarrow B = C) \leftrightarrow
((if (A \neq 0, A, 1) \cdot B) =
(if (A \neq 0, A, 1) \cdot C) \leftrightarrow
1330
\[ B = C \] by (rule MMI_bibli1d)

from S4 S8 have S9: \( A = 0 \) \( A, 1 \) \( B = C \)

( ( ( A \( C \land B \in C \land C \in C \) ) \( A \cdot B = ( A \cdot C \land B \in C \land C \in C ) \) )

( ( if ( A \( A, 1 \) ) \( A \cdot B = ( A \cdot C \land B \in C \land C \in C ) \) )

( ( if ( A \( A, 1 \) ) \( A \cdot B = ( A \cdot C \land B \in C \land C \in C ) \) )

( if ( A \( A, 1 \) ) \( A \cdot B = ( A \cdot C \land B \in C \land C \in C ) \) )

( if ( A \( A, 1 \) ) \( A \cdot B = ( A \cdot C \land B \in C \land C \in C ) \) )

by (rule MMI_impcom)

qed

lemma (in MMIsar0) MMI_mulcan2t:

shows ( ( A \( A \cdot C \land B \in C \land C \in C ) \land C \neq 0 ) )

( ( A \cdot C ) = ( B \cdot C ) \land A = B )

proof -

have S1: ( ( A \( A \cdot C \land C \in C ) \land C \neq 0 ) )

( ( A \cdot C ) = ( B \cdot C ) \land A = B )

from S4 S8 have S9: ( ( A \( A \cdot C \land B \in C \land C \in C ) \land C \neq 0 ) )

( ( A \cdot C ) = ( B \cdot C ) \land A = B )

from S7 have S8: ( ( C \( C \cdot A \land B \in C \land B \in C \) \land C \neq 0 ) )

( ( C \cdot A ) = ( C \cdot B ) \land A = B )

from S8 have S9: ( ( C \( C \cdot A \land B \in C \land B \in C \) \land C \neq 0 ) )

( ( C \cdot A ) = ( C \cdot B ) \land A = B )

from S9 have S10: ( ( C \( C \cdot A \land B \in C \land C \in C ) \land C \neq 0 ) )

( ( C \cdot A ) = ( C \cdot B ) \land A = B )
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 MMI_isar0) 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_mpan)
  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_eqeq2i)
  from S7 S10 have S11: \( A \neq 0 \rightarrow ( ( A \cdot B ) = ( A \cdot 0 ) 
  \leftrightarrow B = 0 )\) by (rule MMI_syl5bbr)
  from S11 have S12: \( A \neq 0 \rightarrow ( ( A \cdot B ) = 0 
  \rightarrow B = 0 )\) by (rule MMI_biimpd)
  from S1 S12 have S13: \( \neg ( A = 0 ) 
  \rightarrow ( ( A \cdot B ) = 0 \rightarrow B = 0 )\) by (rule MMI_sylbir)
  from S13 have S14: \( ( A \cdot B ) = 0 
  \rightarrow ( \neg ( A = 0 ) \rightarrow B = 0 )\) 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 MMI_isar0) MMI_msq0: 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)

1332
have S4: ( A = 0 \lor A = 0 ) \iff A = 0 \text{ by (rule MMI_oridm)}
from S3 S4 show ( A \cdot A ) = 0 \iff A = 0 \text{ by (rule MMI_bitr)}
qed

lemma (in MMIisar0) MMI_mul0ort:
  shows ( A \in \mathbb{F} \land B \in \mathbb{F} ) \implies
  ( ( A \cdot B ) = 0 \iff ( A = 0 \lor B = 0 ) )
proof -
  have S1: A = if ( A \in \mathbb{F} , A , 0 ) \implies
  ( A \cdot B ) = if ( A \in \mathbb{F} , A , 0 ) \cdot B 
  by (rule MMI_opreq1)
  from S1 have S2: A = if ( A \in \mathbb{F} , A , 0 ) \implies
  ( A = 0 \iff ( A \in \mathbb{F} , A , 0 ) \cdot B ) = 0 
  by (rule MMI_eqeq1d)
  have S3: A = if ( A \in \mathbb{F} , A , 0 ) \implies
  ( A = 0 \iff ( A \in \mathbb{F} , A , 0 ) \cdot B ) =
  if ( A \in \mathbb{F} , A , 0 ) = 0 \lor B = 0 
  by (rule MMI_orbi1d)
  from S2 S4 have S5: A = if ( A \in \mathbb{F} , A , 0 ) \implies
  ( ( A \cdot B ) = 0 \iff ( A = 0 \lor B = 0 ) ) \iff
  ( ( if ( A \in \mathbb{F} , A , 0 ) \cdot B ) =
    0 \iff
    ( if ( A \in \mathbb{F} , A , 0 ) = 0 \lor B = 0 ) ) 
  by (rule MMI_bibi12d)
  have S6: B = if ( B \in \mathbb{F} , B , 0 ) \implies
  ( ( A \cdot B ) = 0 \iff ( A = 0 \lor B = 0 ) ) \iff
  ( ( if ( A \in \mathbb{F} , A , 0 ) \cdot B ) =
    0 \iff
    ( if ( A \in \mathbb{F} , A , 0 ) = 0 \lor B = 0 ) ) 
  by (rule MMI_eqeq1d)
  have S7: B = if ( B \in \mathbb{F} , B , 0 ) \implies
  ( if ( A \in \mathbb{F} , A , 0 ) \cdot if ( B \in \mathbb{F} , B , 0 ) ) =
  0 \iff
  ( if ( A \in \mathbb{F} , A , 0 ) \cdot if ( B \in \mathbb{F} , B , 0 ) ) =
  0 
  by (rule MMI_orbi2d)
  from S7 S9 have S10: B =
if \(( B \in C, B, 0 ) \rightarrow ( ( ( if \( A \in C, A, 0 ) \cdot B ) = 0 \leftrightarrow ( if \( A \in C, A, 0 ) = 0 \lor B = 0 )) \leftrightarrow ( if \( A \in C, A, 0 ) \cdot if \( B \in C, B, 0 ) ) = 0 \leftrightarrow ( if \( A \in C, A, 0 ) = 0 \lor if \( B \in C, B, 0 ) = 0 )) \) by (rule MMI_bibi12d)

have S11: 0 ∈ C by (rule MMI_0cn)

from S11 have S12: if \( A \in C, A, 0 ) \in C by (rule MMI_elimel)

have S13: 0 ∈ C by (rule MMI_0cn)

from S13 have S14: if \( B \in C, B, 0 ) \in C by (rule MMI_elimel)

from S12 S14 have S15: ( if \( A \in C, A, 0 ) \cdot if \( B \in C, B, 0 ) ) = 0 \leftrightarrow ( if \( A \in C, A, 0 ) = 0 \lor if \( B \in C, B, 0 ) = 0 )) \) by (rule MMI_mul0or)

from S5 S10 S15 show \( A \in C \land B \in C ) \rightarrow ( ( A \cdot B ) = 0 \leftrightarrow ( A = 0 \lor B = 0 ) ) by (rule MMI_dedth2h)

qed

lemma (in MMIar0) MMI_muln0bt:

shows \( ( A \in C \land B \in C ) \rightarrow ( ( A \neq 0 \land B \neq 0 ) \leftrightarrow ( A \cdot B ) \neq 0 ) \)

proof -

have S1: ( A ∈ C ∧ B ∈ C ) \rightarrow (( A \cdot B ) = 0 \leftrightarrow ( A = 0 \lor B = 0 )) by (rule MMI_mul0ort)

from S1 have S2: ( A ∈ C ∧ B ∈ C ) \rightarrow ( ( A \cdot B ) = 0 \leftrightarrow ( ( A = 0 \lor B = 0 ) ) \) by (rule MMI_negbid)

have S3: ( ( ( A = 0 \lor B = 0 ) ) \) by (rule MMI_ioran)

from S2 S3 have S4: ( A ∈ C ∧ B ∈ C ) \rightarrow ( ( A = 0 ) \land \neg ( B = 0 ) ) \leftrightarrow ( ( A = 0 ) \land \neg ( B = 0 ) ) \leftrightarrow

( ( A \cdot B ) = 0 ) \leftrightarrow ( ( A \cdot B ) = 0 ) \) by (rule MMI_syl6rbb)

have S5: A \neq 0 \leftrightarrow \neg ( A = 0 ) by (rule MMI_df_ne)

have S6: B \neq 0 \leftrightarrow \neg ( B = 0 ) by (rule MMI_df_ne)

from S5 S6 have S7: ( A \neq 0 \land B \neq 0 ) \leftrightarrow ( ( A = 0 ) \land \neg ( B = 0 ) ) \leftrightarrow ( ( A \cdot B ) \neq 0 ) \leftrightarrow ( ( A \cdot B ) = 0 ) by (rule MMI_anbii12i)

from S4 S7 S8 show ( A ∈ C ∧ B ∈ C ) \rightarrow ( ( A \neq 0 \land B \neq 0 ) \leftrightarrow ( ( A \cdot B ) \neq 0 ) by (rule MMI_anbii12i)

qed

lemma (in MMIar0) MMI_muln0: assumes A1: A ∈ C and

A2: B ∈ C and
A3: A \neq 0 and
A4: B \neq 0

1334
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 \(S6: (A \in C \land B \in C) \rightarrow (A \neq 0 \land B \neq 0)\) by (rule \(MMI_{muln0bt}\))

from \(S5 \ S6\) have \(S7: (A \in C \land B \in C) \rightarrow (A \cdot B) \neq 0\) by (rule \(MMI_{mpbii}\))

from \(S1 \ S2 \ S7\) show \((A \cdot B) \neq 0\) by (rule \(MMI_{mp2an}\))

qed

lemma (in \(MMIsar0\)) \(MMI\_receu\): 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 \(S2: x = y \rightarrow ((A \cdot x) = B \leftarrowarrow (A \cdot y) = B)\)

by (rule \(MMI_{eqeq1d}\))

\} then have \(S2: \forall x \ y. x = y \rightarrow ((A \cdot x) = B \leftarrowarrow (A \cdot y) = B)\)

by simp

from \(S2\) have \(S3:\)

\((\exists! x. x \in C \land (A \cdot x) = B) \leftarrowarrow\)

\((\exists x \in C. (A \cdot x) = B) \land\)

\((\forall x \in C. \forall y \in C. ((A \cdot x) = B \land (A \cdot y) = B) \rightarrow x = y)\)\)

by (rule \(MMI\_reu4\))

from \(A1\) have \(S4: A \in C\).
from \(A3\) have \(S5: A \neq 0\).
from \(S4 \ S5\) have \(S6: \exists y \in C. (A \cdot y) = 1\) by (rule \(MMI\_recex\))

from \(A2\) have \(S7: B \in C\).

\{ fix \(y\)

have \(S8: (y \in C \land B \in C) \rightarrow (y \cdot B) \in C\) by (rule \(MMI\_axmulcl\))

from \(S7 \ S8\) have \(S9: y \in C \rightarrow (y \cdot B) \in C\) by (rule \(MMI\_mp2an\))

have \(S10: (y \cdot B) \in C\) \leftarrowarrow\)

\((\exists x \in C. x = (y \cdot B))\) by (rule \(MMI\_riset\))

from \(S9 \ S10\) have \(S11: y \in C \rightarrow (\exists x \in C. x = (y \cdot B))\)

by (rule \(MMI\_sylib\))

\{ fix \(x\)

have \(S12: x = (y \cdot B)\) \rightarrowarrow\)

\((A \cdot x) = (A \cdot (y \cdot B))\) by (rule \(MMI\_opreq2\))

from \(A1\) have \(S13: A \in C\).
from \(A2\) have \(S14: B \in C\).

have \(S15: (A \in C \land y \in C \land B \in C)\) \rightarrowarrow\)

1335
((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 · y) · B) = (A · (y · B)) by (rule MMI_axmulass)

(1 → ((A · y) · B) = (1 · B)) by (rule MMI_mp3an13)

from S12 S17 have S18: (y ∈ C ∧ x = 1) →
((A · y) · B) = (A · (y · B)) by (rule MMI_eqcomd)

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 →
(x = (y · B) → (A · x) = B) by (rule MMI_exp32)

from S24 have S25:
(x = (y · B) → (A · x) = B) by (rule MMI_impcom)

from S25 have
\(\forall x. (y ∈ C ∧ (A · y) = 1) → (x ∈ C → (x = (y · B) → (A · x) = B))\) by (rule MMI_a1d)

\(\forall x. (y ∈ C ∧ (A · y) = 1 → (x ∈ C → (x = (y · B) → (A · x) = B)))\) by simp

(\(\forall x. (y ∈ C ∧ (A · y) = 1) → (x ∈ C → (x = (y · B) → (A · x) = B)))\) by (rule MMI_r19_21aiv)

from S27 have S28: \(y ∈ C → ((A · y) = 1 → \forall x ∈ C. (x = (y · B) → (A · x) = B))\) by (rule MMI_ex)

have S29: \(\forall x ∈ C. (x = (y · B) → (A · x) = B)\) by (rule MMI_syl6)

(\((\exists x ∈ C. x = (y · B)) → (\exists x ∈ C. (A · x) = B))\) by (rule MMI_mp2id)

(\((\exists x ∈ C. x = (y · B)) → (\exists x ∈ C. (A · x) = B))\) by (rule MMI_mp2id)

\(\forall y. y ∈ C → ((A · y) = 1 → (\exists x ∈ C. (A · x) = B))\) by (rule MMI_mpid)

\(\forall x. \forall y. x ∈ C → ((A · y) = 1 → (\exists x ∈ C. (A · x) = B))\) by (rule MMI_mpid)

1336
by simp
  from S31 have S32: \( \exists y \in C. (A \cdot y) = 1 \) \( \rightarrow \) \( \exists x \in C. (A \cdot x) = B \) by (rule MMI_r19_23aiv)
  from S6 S32 have S33: \( \exists x \in C. (A \cdot x) = B \) by (rule MMI_ax_mp)
  from A1 have S34: \( A \in \mathcal{F} \).
  from A3 have S35: \( A \neq 0 \).
  \{ fix \ x \ y \}
  from S35 have S36: \( \exists x \in \mathcal{F}. (A \cdot x) = B \) by (rule MMI_mulcant2)
  from S34 S36 have S37: \( \forall x \ y. (A \cdot x) = B \) \( \rightarrow \) \( (A \cdot x) = B \) by (rule MMI_mp3an1)
  from S3 S33 S40 show \( \exists! x. x \in \mathcal{F} \) \( \land (A \cdot x) = B \) by (rule MMI_mpbir2an)
  qed

lemma (in MMIasar0) MMI_divval: assumes \( A \in C \) \( B \in C \) \( B \neq 0 \)
  shows \( A / B = \bigcup \{ x \in C. B \cdot x = A \} \)
  using cdiv_def by simp

lemma (in MMIasar0) MMI_divmul: assumes \( A1: A \in C \) and
  \( A2: B \in C \) and
  \( A3: C \in C \) and
  \( A4: B \neq 0 \)
  shows \( (A / B) = C \longleftrightarrow (B \cdot C) = A \)
proof -
  from A3 have S1: \( C \in C \).
  \{ fix \ x \}
  have S2: \( x = C \leftrightarrow (A / B) = x \leftrightarrow (A / B) = C \) by (rule MMI_eqeq2)
  have S3: \( x = C \leftrightarrow (B \cdot x) = (B \cdot C) \) by (rule MMI_opreq2)
  from S3 have S4: \( x = C \leftrightarrow (B \cdot x) = A \leftrightarrow (B \cdot C) = A \) by (rule MMI_eqeq1d)
from S2 S4 have
\[ x = C \iff ((A / B) = x \iff (B \cdot x) = A) \iff ((A / B) = C \iff (B \cdot C) = A) \] by (rule MMI_bibi12d)
\}
then have S5: \[ \forall x. x = C \iff ((A / B) = x \iff (B \cdot x) = A) \iff ((A / B) = C \iff (B \cdot C) = A) \] by simp
from A2 have S6: B \in C.
from A1 have S7: A \in C.
from A4 have S8: B \neq 0.
from S6 S7 S8 have S9: \exists! x. x \in C \land (B \cdot x) = A by (rule MMI_receu)
\{ fix x
have S10: \(( x \in C \land (\exists! x. x \in C \land (B \cdot x) = A)) \iff
( (B \cdot x) = A \iff \bigcup \{ x \in C. (B \cdot x) = A \} = x) \) by (rule MMI_reuuni1)
from S9 S10 have
\begin{align*}
x \in C & \iff ( (B \cdot x) = A \iff \bigcup \{ x \in C. (B \cdot x) = A \} = x) \\
& \text{by (rule MMI_mpan2)}
\end{align*}
\} then have S11:
\[ \forall x. x \in C \iff ((B \cdot x) = A \iff \bigcup \{ x \in C. (B \cdot x) = A \} = x) \] by blast
from A1 have S12: A \in C.
from A2 have S13: B \in C.
from A4 have S14: B \neq 0.
from S12 S13 S14 have S15: \((A / B) = \bigcup \{ x \in C. (B \cdot x) = A \}) by (rule MMI_divval)
from S15 have S16: \forall x. (A / B) = x \iff \bigcup \{ x \in C. (B \cdot x) = A \} = x by simp
from S11 S16 have S17: \forall x. x \in C \iff \((A / B) = x \iff \bigcup \{ x \in C. (B \cdot x) = A \}) by (rule MMI_syl6rbbr)
from S5 S17 have S18: C \in C \iff 
( (A / B) = C \iff (B \cdot C) = A \) by (rule MMI_vtoclga)
from S1 S18 show \((A / B) = C \iff (B \cdot C) = A \) by (rule MMI_ax_mp)
qed

lemma (in MMIsar0) MMI_divmulz: assumes A1: A \in C and
A2: B \in C and
A3: C \in C
shows B \neq 0 \iff
\((A / B) = C \iff (B \cdot C) = A \)
proof -
have S1: B =
if (B \neq 0, B, 1) \iff
(A / B) =
(A / if (B \neq 0, B, 1)) by (rule MMI_opreq2)
from S1 have S2: B =

1338
if ( B ≠ 0 , B , 1 ) →

( ( A / B ) =
C ↦ ( A / if ( B ≠ 0 , B , 1 ) ) = C ) by (rule MMI_eqeq1d)

have S3: B =

if ( B ≠ 0 , B , 1 ) →

( B · C ) =

( if ( B ≠ 0 , B , 1 ) · C ) by (rule MMI_opreq1)

from S3 have S4: B =

if ( B ≠ 0 , B , 1 ) →

( ( B · C ) =
A ↦ ( if ( B ≠ 0 , B , 1 ) · C ) = A ) by (rule MMI_eqeq1d)

from S2 S4 have S5: B =

if ( B ≠ 0 , B , 1 ) →

( ( B · C ) =
A ↦ ( if ( B ≠ 0 , B , 1 ) · C ) = A ) by (rule MMI_eqeq1d)

from S1 S5 have S6: A =

if ( B ≠ 0 , B , 1 ) →

( ( A / B ) =
C ↦ ( A / if ( B ≠ 0 , B , 1 ) ) = C ) by (rule MMI_bibi12d)

from A1 have S6: A ∈ C.

from A2 have S7: B ∈ C.

have S8: 1 ∈ C by (rule MMI_1cn)

from S7 S8 have S9: if ( B ≠ 0 , B , 1 ) ∈ C by (rule MMI_keepel)

from A3 have S10: C ∈ C.

have S11: if ( B ≠ 0 , B , 1 ) ≠ 0 by (rule MMI_elimne0)

from S6 S9 S10 S11 have S12: ( A / if ( B ≠ 0 , B , 1 ) ) =
C ↦ ( if ( B ≠ 0 , B , 1 ) · C ) = A by (rule MMI_divmul)

from S5 S12 have B ≠ 0 →

( ( A / B ) = C ↦ ( B · C ) = A ) by (rule MMI_dedth)

show B ≠ 0 →

( ( A / B ) = C → ( B · C ) = A ) by (rule MMI_dedth)

qed

lemma (in MMI_isar0) MMI_divmult:

shows ( ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) ∧ B ≠ 0 ) →

( ( A / B ) = C ) → ( B · C ) = A

proof -

have S1: A =

if ( A ∈ C , A , 0 ) →

( A / B ) =

( ( 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_eqeq1d)

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 ) =

1339
C \leftarrow
\begin{align*}
( B \cdot C ) &= \text{if} ( A \in C, A, 0 ) \quad \text{by (rule MMI_bibi12d)} \\
&\quad \text{from S4 have S5: A =} \\
&\quad \text{if} ( A \in C, A, 0 ) \rightarrow \\
&\quad ( ( B \neq 0 \rightarrow ( ( A / B ) = C \leftrightarrow ( B \cdot C ) = A ) ) \leftrightarrow \\
&\quad ( B \neq 0 \rightarrow \\
&\quad ( ( \text{if} ( A \in C, A, 0 ) / B ) = \\
&\quad C \leftrightarrow \\
&\quad ( B \cdot C ) = \text{if} ( A \in C, A, 0 ) ) ) \quad \text{by (rule MMI_imbi12d)}
\end{align*}

have S6: B = \\
\begin{align*}
( B \in C, B, 0 ) \rightarrow \\
&\quad \text{if} ( B \in C, B, 0 ) \rightarrow \\
&\quad ( ( A \in C, A, 0 ) / ( B \in C, B, 0 ) ) \quad \text{by (rule MMI_opreq2)} \\
&\quad \text{from S7 have S8: B =} \\
&\quad ( ( A \in C, A, 0 ) / B ) = \\
&\quad C \leftarrow \\
&\quad ( ( A \in C, A, 0 ) / ( B \in C, B, 0 ) ) = \\
&\quad C \quad \text{by (rule MMI_eqeq1d)} \\
&\quad \text{have S9: B =} \\
&\quad ( B \in C, B, 0 ) \rightarrow \\
&\quad ( B \cdot C ) = \\
&\quad ( ( B \in C, B, 0 ) \cdot C ) \quad \text{by (rule MMI_opreq1)} \\
&\quad \text{from S9 have S10: B =} \\
&\quad ( ( B \in C, B, 0 ) \rightarrow \\
&\quad ( ( B \cdot C ) = \\
&\quad ( ( A \in C, A, 0 ) ) \leftrightarrow \\
&\quad ( ( B \in C, B, 0 ) \cdot C ) = \\
&\quad ( ( A \in C, A, 0 ) ) \quad \text{by (rule MMI_eqeq1d)} \\
&\quad \text{from S8 S10 have S11: B =} \\
&\quad ( B \in C, B, 0 ) \rightarrow \\
&\quad ( ( ( A \in C, A, 0 ) / B ) = C \leftrightarrow ( B \cdot C ) = \text{if} ( A \in C, A, 0 ) ) \leftrightarrow \\
&\quad ( ( A \in C, A, 0 ) / ( B \in C, B, 0 ) ) \quad \text{by (rule MMI_bibi12d)} \\
&\quad \text{from S6 S11 have S12: B =} \\
&\quad ( B \in C, B, 0 ) \rightarrow \\
&\quad ( ( B \neq 0 \rightarrow ( ( ( A \in C, A, 0 ) / B ) = C \leftrightarrow ( B \cdot C ) = \text{if} ( A \in C, A, 0 ) ) ) \leftrightarrow \\
&\quad ( ( B \in C, B, 0 ) ) \quad \text{by (rule MMI_imbi12d)}
\end{align*}
have $S_{13}$: $C = 
\begin{align*}
&\text{if ( } C \in C , C , 0 \) } \to \\
&\left( \left( \text{if ( } A \in C , A , 0 \) } / \text{if ( } B \in C , B , 0 \) \right) = \\
&C \hypothesis \hypothesis \\
&\left( \text{if ( } A \in C , A , 0 \) } / \text{if ( } B \in C , B , 0 \) \right) = \\
&\text{if ( } C \in C , C , 0 \) \text{ ) by (rule MMI_eqeq2) } \\
&\hypothesis \hypothesis \\
&\text{have $S_{14}$: $C = 
\begin{align*}
&\text{if ( } C \in C , C , 0 \) } \to \\
&\left( \left( \text{if ( } B \in C , B , 0 \right) \cdot C \right) = \\
&\left( \text{if ( } A \in C , A , 0 \text{ ) } \text{if ( } C \in C , C , 0 \) \right) = \\
&\text{if ( } A \in C , A , 0 \) \text{ ) by (rule MMI_eqeq1d) } \\
&\hypothesis \hypothesis \\
&\text{from $S_{13}$ $S_{15}$ have $S_{16}$: $C = 
\begin{align*}
&\text{if ( } C \in C , C , 0 \) } \to \\
&\left( \left( \text{if ( } A \in C , A , 0 \) } / \text{if ( } B \in C , B , 0 \) \right) = \\
&C \hypothesis \hypothesis \\
&\left( \text{if ( } A \in C , A , 0 \) } / \text{if ( } B \in C , B , 0 \) \right) = \\
&\text{if ( } C \in C , C , 0 \) \text{ ) by (rule MMI_bibi12d) } \\
&\hypothesis \hypothesis \\
&\text{from $S_{16}$ have $S_{17}$: $C = 
\begin{align*}
&\text{if ( } C \in C , C , 0 \) } \to \\
&\left( \text{if ( } A \in C , A , 0 \text{ ) } \text{if ( } B \in C , B , 0 \) \right) = \\
&C \hypothesis \hypothesis \\
&\text{if ( } A \in C , A , 0 \) \text{ ) by (rule MMI_bibi12d) } \\
&\hypothesis \hypothesis \\
&\text{have $S_{18}$: } 0 \in C \text{ by (rule MMI_0cn) } \\
&\text{from $S_{18}$ have $S_{19}$: if ( } A \in C , A , 0 \) \in C \text{ by (rule MMI_elim) } \\
&\text{have $S_{20}$: } 0 \in C \text{ by (rule MMI_0cn) } \\
&\text{from $S_{20}$ have $S_{21}$: if ( } B \in C , B , 0 \) \in C \text{ by (rule MMI_elim) } \\
&\text{have $S_{22}$: } 0 \in C \text{ by (rule MMI_0cn) } \\
&\text{from $S_{22}$ have $S_{23}$: if ( } C \in C , C , 0 \) \in C \text{ by (rule MMI_elim) } \\
&\text{from $S_{19}$ $S_{21}$ $S_{23}$ have $S_{24}$: if ( } B \in C , B , 0 \) \neq 0 \to \\
&\left( \left( \text{if ( } A \in C , A , 0 \text{ ) } / \text{if ( } B \in C , B , 0 \right) = \\
&C \hypothesis \hypothesis \\
&\left( \text{if ( } B \in C , B , 0 \text{ ) ) by (rule MMI_dedth3h) } \\
&\text{from $S_{25}$ show ( } ( A \in C \land B \in C \land C \in C \text{ ) } \land B \neq 0 \text{ ) } \to 
\end{align*}
\right)
\end{align*}

\[(A / B) = C \iff (B \cdot C) = A\] by (rule MMI_imp)

**qed**

**lemma (in MMIsar0) MMI_divmul2t:**

shows \((A \in C \land B \in C \land C \in C) \land B \neq 0\) \(\rightarrow\)

\[(A / B) = C \iff (B \cdot C) = A\]

**proof** -

- have S1: \((A \in C \land B \in C \land C \in C) \land B \neq 0\) \(\rightarrow\)
  \[(A / B) = C \iff (B \cdot C) = A\] by (rule MMI_divmul2t)
- have S2: \((B \cdot C) = A \iff A = (B \cdot C)\) by (rule MMI_eqcom)
- from S1 S2 show \((A \in C \land B \in C \land C \in C) \land B \neq 0\) \(\rightarrow\)
  \[(A / B) = C \iff A = (B \cdot C)\] by (rule MMI_syl6bb)

**qed**

**lemma (in MMIsar0) MMI_divmul3t:**

shows \((A \in C \land B \in C \land C \in C) \land B \neq 0\) \(\rightarrow\)

\[(A / B) = C \iff A = (C \cdot B)\]

**proof** -

- have S1: \((A \in C \land B \in C \land C \in C) \land B \neq 0\) \(\rightarrow\)
  \[(A / B) = C \iff A = (B \cdot C)\] by (rule MMI_divmul2t)
- have S2: \((B \in C \land C \in C) \rightarrow\)
  \[(B \cdot C) = (C \cdot B)\] by (rule MMI_axmulcom)
- from S2 have S3: \((B \in C \land C \in C) \rightarrow\)
  \[(A = (B \cdot C) \iff A = (C \cdot B))\] by (rule MMI_eqeq2d)
- from S4 have S5: \((A \in C \land B \in C \land C \in C) \land \land B \neq 0\) \(\rightarrow\)
  \[(A \in C \land B \in C \land C \in C) \land B \neq 0\) \(\rightarrow\)
  \[(A = (B \cdot C) \iff A = (C \cdot B))\] by (rule MMI_3adant1)
- from S5 have S6: \((A \in C \land B \in C \land C \in C) \land B \neq 0\) \(\rightarrow\)
  \[(A \in C \land B \in C \land C \in C) \land B \neq 0\) \(\rightarrow\)
  \[(A = (B \cdot C) \iff A = (C \cdot B))\] by (rule MMI_adantr)
- from S1 S5 show \((A \in C \land B \in C \land C \in C) \land B \neq 0\) \(\rightarrow\)
  \[(A / B) = C \iff A = (C \cdot B)\] by (rule MMI_bitrd)

**qed**

**lemma (in MMIsar0) 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)

1342
from S4 S10 show \(( A / B ) \in C\) by (rule MMI Eqeltr)

qed

lemma (in MMIsar0) 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 =
    \( \begin{cases} 
      B, & B \neq 0 \\
      1, & B = 0 
    \end{cases} \)
    by (rule MMI Eqeltr)
  from S1 have S2: B =
    \( \begin{cases} 
      B, & B \neq 0 \\
      1, & B = 0 
    \end{cases} \) \in C
  by (rule MMI Eqeltr)
  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 S8: ( 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 =
    \( \begin{cases} 
      A, & A \in C \\
      0, & A \notin C 
    \end{cases} \)
    by (rule MMI Eqeltr)
  from S1 have S2: A =
    \( \begin{cases} 
      A, & A \in C \\
      0, & A \notin C 
    \end{cases} \) / B \in C
    by (rule MMI keepel)
  from S2 have S3: A =
    \( \begin{cases} 
      A, & A \in C \\
      0, & A \notin C 
    \end{cases} \) / B \in C
    by (rule MMI Eqeltr)
  from S2 have S3: A =
    \( \begin{cases} 
      A, & A \in C \\
      0, & A \notin C 
    \end{cases} \) / B \in C
    by (rule MMI Eqeltr)
  have S4: B =
    \( \begin{cases} 
      B, & B \in C \\
      0, & B \notin C 
    \end{cases} \)
    by (rule MMI neeq1)
  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, B, 0)) ∈ C by (rule MMI_eleq1d)
   from S4 S6 have S7: B =
   if (B ∈ C, B, 0) →
   (B ≠ 0 → (if (A ∈ C, A, 0) / B) ∈ C)
   by (rule MMI_imbi12d)
   have S8: 0 ∈ C by (rule MMI_0cn)
   from S8 have S9: if (A ∈ C, A, 0) ∈ C by (rule MMI_elimel)
   have S10: 0 ∈ C by (rule MMI_0cn)
   from S10 have S11: if (B ∈ C, B, 0) ≠ 0 →
   (if (A ∈ C, A, 0) / if (B ∈ C, B, 0)) ∈ C by (rule MMI_divclz)
   from S3 S7 S12 have S13: (A ∈ C ∧ B ∈ C) →
   (B ≠ 0 → (A / B) ∈ C) by (rule MMI_days2h)
   from S13 show (A ∈ C ∧ B ∈ C ∧ B ≠ 0) →
   (A / B) ∈ C by (rule MMI_3impia)
   qed

lemma (in MMIasar0) MMI_reccl: assumes A1: A ∈ C and
   A2: A ≠ 0
   shows (1 / A) ∈ 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 MMIasar0) MMI_recclz: 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 MMIasar0) MMI_recclt:
   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)
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 S1 S2 S3 have S4: ( B / A ) ∈ C by (rule MMI_divcl)
  from A1 have S5: A ∈ C.
  from S4 S5 have S6: ( ( B / A ) · A ) = ( A · ( B / A ) ) by (rule MMI_mulcom)
  from A1 have S7: A ∈ C.
  from A2 have S8: B ∈ C.
  from A3 have S9: A ≠ 0.
  from S7 S8 S9 have S10: ( A · ( B / A ) ) = B by (rule MMI_divcan2)
  from S6 S10 show ( ( B / A ) · A ) = B by (rule MMI_eqtr)
qed

lemma (in MMIsar0) MMI_divcan1z: assumes A1: A ∈ C and A2: B ∈ C shows A ≠ 0 ←→ ( ( B / A ) · A ) = B
proof -
  have S1: A = if ( A ≠ 0 , A , 1 ) →
    ( B / A ) = ( B / if ( A ≠ 0 , A , 1 ) ) by (rule MMI_opreq2)
    have S2: A = if ( A ≠ 0 , A , 1 ) →
A = if ( A \neq 0 , A , 1 ) by (rule MMI_id)
from S1 S2 have S3: 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 S3 have S4: A =
if ( A \neq 0 , A , 1 ) \rightarrow
(( B / A ) \cdot A ) =
B \iff
(( B / if ( A \neq 0 , A , 1 ) ) \cdot if ( A \neq 0 , A , 1 ) ) = B
by (rule MMI_eqeq1d)
from A1 have S5: A \in C.
have S6: 1 \in C by (rule MMI_1cn)
from S5 S6 have S7: if ( A \neq 0 , A , 1 ) \in C by (rule MMI_1cn)
from A2 have S8: B \in C.
have S9: if ( A \neq 0 , A , 1 ) \neq 0 by (rule MMI_elimne0)
from S7 S8 S9 have S10: (( B / if ( A \neq 0 , A , 1 ) ) \cdot if ( A \neq 0 , A , 1 ) ) = B
by (rule MMI_divcan1)
from S4 S10 show A \neq 0 \rightarrow (( B / A ) \cdot A ) = B by (rule MMI_dedth)
qed

lemma (in MMI_id) MMI_divcan2z: assumes A1: A \in C and
A2: B \in C
shows A \neq 0 \rightarrow ( A \cdot ( B / A ) ) = B
proof -
have S1: A =
if ( A \neq 0 , A , 1 ) \rightarrow
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 \iff
( if ( A \neq 0 , A , 1 ) \cdot ( B / if ( A \neq 0 , A , 1 ) ) ) = B
by (rule MMI_eqeq1d)
from A1 have S5: A \in C.
have S6: 1 \in C by (rule MMI_1cn)
from S5 S6 have S7: if ( A \neq 0 , A , 1 ) \in C by (rule MMI_1cn)
from A2 have S8: B \in 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
lemma (in MMIars0) MMI_divcan1t:
  shows \(( A \in C \land B \in C \land A \neq 0 ) \longrightarrow \(( B / A ) \cdot A ) = B \) by (rule MMI_dedth)
proof -
  have S1: A = if ( A \in C , A , 0 ) \longrightarrow ( A \neq 0 \iff if ( A \in C , A , 0 ) \neq 0 ) by (rule MMI_neeq1)
  have S2: A = if ( A \in C , A , 0 ) \longrightarrow ( B / A ) = ( B / if ( A \in C , A , 0 ) ) by (rule MMI_opreq2)
  have S3: A = if ( A \in C , A , 0 ) \longrightarrow A = if ( A \in C , A , 0 ) by (rule MMI_id)
  from S2 S3 have S4: A = if ( A \in C , A , 0 ) \longrightarrow ( ( B / A ) \cdot A ) = if ( A \in C , A , 0 ) \neq 0 \longrightarrow ( ( B / if ( A \in C , A , 0 ) ) \cdot if ( A \in C , A , 0 ) ) = B ) by (rule MMI_opreq12d)
  from S4 have S5: A = if ( A \in C , A , 0 ) \longrightarrow ( ( ( B / A ) \cdot A ) \iff if ( A \in C , A , 0 ) \neq 0 \longrightarrow ( ( B / if ( A \in C , A , 0 ) ) \cdot if ( A \in C , A , 0 ) ) = B ) by (rule MMI_imbi12d)
  have S6: B = if ( B \in C , B , 0 ) \longrightarrow ( B / if ( A \in C , A , 0 ) ) = ( ( B / if ( A \in C , A , 0 ) ) \cdot if ( A \in C , A , 0 ) ) \iff if ( A \in C , A , 0 ) ) by (rule MMI_opreq1)
  from S7 have S8: B = if ( B \in C , B , 0 ) \longrightarrow (( B / if ( A \in C , A , 0 ) ) \cdot if ( A \in C , A , 0 ) ) = ( ( if ( B \in C , B , 0 ) / if ( A \in C , A , 0 ) ) \cdot if ( A \in C , A , 0 ) ) \cdot if ( A \in C , A , 0 ) ) by (rule MMI_opreq1d)
  have S9: B = if ( B \in C , B , 0 ) \longrightarrow B = if ( B \in C , B , 0 ) by (rule MMI_id)
  from S8 S9 have S10: B = if ( B \in C , B , 0 ) \longrightarrow
((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)
from S12 have S13: if (A ∈ C, A, 0) ∈ C by (rule MMI_elimel)
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 (A ∈ C, A, 0) = B) 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 MMIasar0) MMI_divcan2t:
  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 = if (A ∈ C, A, 0) by (rule MMI_id)
  have S3: A =
  if (A ∈ C, A, 0) →
  (B / A) =
  (B / if (A ∈ C, A, 0)) by (rule MMI_opreq2)
  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 ←→

1348
( if ( A ∈ C , A , 0 ) · ( B / if ( A ∈ C , A , 0 ) ) ) = B ) by (rule MMI_eeqeq1d)

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_eqeq1d)

have S7: B = if ( B ∈ C , B , 0 ) →
( ( B = if ( A ∈ C , A , 0 ) ) = if ( B ∈ C , B , 0 ) / if ( A ∈ C , A , 0 ) ) ) by (rule MMI_opreq1)

from S7 have S8: 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 ) ) ) ) by (rule MMI_opreq2d)

have S9: B = if ( B ∈ C , B , 0 ) →
( 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 ) ) = if ( B ∈ C , B , 0 ) / if ( A ∈ C , A , 0 ) ) ) = B )←→
( if ( A ∈ C , A , 0 ) · ( if ( B ∈ C , B , 0 ) / if ( A ∈ C , A , 0 ) ) ) ) = B ) by (rule MMI_divcan2z)

from S6 S11 S16 have S17: ( A ∈ C ∧ B ∈ C ) →
( A ≠ 0 → ( A · ( B / A ) ) = B ) by (rule MMI_3impia)

qed
lemma (in MMIsar0) 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_syl6rbbrA)
  from S4 have S5: ( A ∈ C ∧ B ∈ C ∧ B ≠ 0 ) →
  ( A = 0 ←→ ( B · 0 ) = A ) by (rule MMI_3ad2ant2)
  have S6: 0 ∈ C by (rule MMI_0cn)
  have S7: ( ( A ∈ C ∧ B ∈ C ∧ B ≠ 0 ) ∧ B ≠ 0 ) →
  ( ( A / B ) = 0 ←→ ( B · 0 ) = A ) by (rule MMI_mp3an13)
  from S6 S7 have S8: ( ( A ∈ C ∧ B ∈ C ) ∧ B ≠ 0 ) →
  ( ( A / B ) = 0 ←→ ( B · 0 ) = A ) by (rule MMI_3impa)
  from S8 have S9: ( A ∈ C ∧ B ∈ C ∧ B ≠ 0 ) →
  ( A = 0 ←→ ( A / B ) ≠ 0 ) by (rule MMI_bitr4d)
  from S9 have S10: ( A ∈ C ∧ B ∈ C ∧ B ≠ 0 ) →
  ( A ≠ 0 ←→ ( A / B ) ≠ 0 ) by (rule MMI_eqneqd)
  from S10 show ( A ∈ C ∧ B ∈ C ∧ B ≠ 0 ) →
  ( A ≠ 0 ←→ ( A / B ) ≠ 0 ) by (rule MMI_mp3an)
qed

lemma (in MMIsar0) MMI_divne0: assumes A1: A ∈ C and
  A2: B ∈ C and
  A3: A ≠ 0 and
  A4: B ≠ 0
  shows ( A / B ) ≠ 0
proof -
  from A1 have S1: A ∈ C.
  from A2 have S2: B ∈ C.
  from A4 have S3: B ≠ 0.
  from A3 have S4: A ≠ 0.
  have S5: ( A ∈ C ∧ B ∈ C ∧ B ≠ 0 ) →
  ( A ≠ 0 ←→ ( A / B ) ≠ 0 ) by (rule MMI_divne0bt)
  from S4 S5 have S6: ( A ∈ C ∧ B ∈ C ∧ B ≠ 0 ) →
  ( A / B ) ≠ 0 by (rule MMI_mp3an)
  from S1 S2 S3 S6 show ( A / B ) ≠ 0 by (rule MMI_mpbi)
qed

lemma (in MMIsar0) MMI_recne0z: assumes A1: A ∈ C
  shows A ≠ 0 → ( 1 / A ) ≠ 0
proof -
  have S1: A =
  if ( A ≠ 0 , A , 1 ) →
  ( 1 / A ) =
(1 / if (A ≠ 0, A, 1)) by (rule MMI_opreq2)
from S1 have S2: A =
if (A ≠ 0, A, 1) →
((1 / A) ≠ 0 ↔
(1 / if (A ≠ 0, A, 1)) ≠ 0) by (rule MMI_neeq1d)
have S3: 1 ∈ C by (rule MMI_1cn)
from A1 have S4: A ∈ C.
have S5: 1 ∈ C by (rule MMI_1cn)
from S4 S5 have S6: if (A ≠ 0, A, 1) ∈ C by (rule MMI_keepel)
have S7: 1 ≠ 0 by (rule MMI_ax1ne0)
have S8: if (A ≠ 0, A, 1) ≠ 0 by (rule MMI_elimne0)
from S3 S6 S7 S8 have S9: (1 / if (A ≠ 0, A, 1)) ≠ 0 by (rule MMI_divne0)
from S2 S9 show A ≠ 0 → (1 / A) ≠ 0 by (rule MMI_dedth)
qed

lemma (in MMIar0) MMI_recne0t:
shows (A ∈ C ∧ A ≠ 0) → (1 / A) ≠ 0
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) →
(1 / A) =
(1 / if (A ∈ C, A, 0)) by (rule MMI_opreq2)
from S2 have S3: A =
if (A ∈ C, A, 0) →
((1 / A) ≠ 0 ↔
(1 / if (A ∈ C, A, 0)) ≠ 0) by (rule MMI_neeq1d)
from S1 S3 have S4: A =
if (A ∈ C, A, 0) →
((A ≠ 0 → (1 / A) ≠ 0) ↔
(if (A ∈ C, A, 0) ≠ 0 →
(1 / if (A ∈ C, A, 0)) ≠ 0)) by (rule MMI_imbi12d)
have S5: 0 ∈ C by (rule MMI_0cn)
from S5 have S6: if (A ∈ C, A, 0) ∈ C by (rule MMI_elim)
from S6 have S7: if (A ∈ C, A, 0) ≠ 0 →
(1 / if (A ∈ C, A, 0)) ≠ 0 by (rule MMI_recne0z)
from S4 S7 have S8: A ∈ C → (A ≠ 0 → (1 / A) ≠ 0) by (rule MMI_dedth)
from S8 show (A ∈ C ∧ A ≠ 0) → (1 / A) ≠ 0 by (rule MMI_imp)
qed

lemma (in MMIar0) MMI_recid: assumes A1: A ∈ C and
A2: A ≠ 0
shows (A · (1 / A)) = 1
proof -
from A1 have S1: A ∈ 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 ) $\rightarrow$
    ( 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 ) $\rightarrow$
      A = if (A $\in C$, A , 0 ) by (rule MMI_id)
      have S3: $A =$
        if (A $\in C$, A , 0 ) $\rightarrow$
        ( 1 / A ) =
        ( 1 / if (A $\in C$, A , 0 ) ) by (rule MMI_opreq2)
      from S2 S3 have S4: $A =$
        if (A $\in C$, A , 0 ) $\rightarrow$
        (A $\cdot$ ( 1 / A ) ) =
        ( if (A $\in C$, A , 0 ) $\cdot$ ( 1 / if (A $\in C$, A , 0 ) ) ) by (rule MMI_opreq12d)
      from S4 have S5: $A =$
        if (A $\in C$, A , 0 ) $\rightarrow$
        ( (A $\cdot$ ( 1 / A ) ) =
        1 $\leftrightarrow$
        ( if (A $\in C$, A , 0 ) $\cdot$ ( 1 / if (A $\in C$, A , 0 ) ) ) =
        1 ) by (rule MMI_eqeq1d)
      from S1 S5 have S6: $A =$
        if (A $\in C$, A , 0 ) $\rightarrow$
        ( (A $\neq 0$ $\rightarrow$ (A $\cdot$ ( 1 / A ) ) ) $\leftrightarrow$
        ( if (A $\in C$, A , 0 ) $\neq 0$ $\rightarrow$
        ( if (A $\in C$, A , 0 ) $\cdot$ ( 1 / if (A $\in C$, A , 0 ) ) ) =
        1 ) $\leftrightarrow$
        ( (A $\in C$, A , 0 ) $\rightarrow$
        ( if (A $\in C$, A , 0 ) $\cdot$ ( 1 / if (A $\in C$, A , 0 ) ) ) =
        1 ) by (rule MMI_imbi12d)
      have S7: $0 \in C$ by (rule MMI_0cn)
      from S7 have S8: if (A $\in C$, A , 0 ) $\in C$ by (rule MMI_elimel)
      from S8 have S9: if (A $\in C$, A , 0 ) $\neq 0$ $\rightarrow$
        ( if (A $\in C$, A , 0 ) $\cdot$ ( 1 / if (A $\in C$, A , 0 ) ) ) =
        1 by (rule MMI_recidz)
      from S6 S9 have S10: $A \in C$
( \! \neq 0 \rightarrow (A \cdot (1/A)) = 1 ) \text{ by (rule MMI_dedth)}
from S10 show (\! A \in C \land A \neq 0 \rightarrow (A \cdot (1/A)) = 1 \text{ by (rule MMI_imp)}
qed

lemma (in MMIar0) MMI_recid2t:
shows (\! A \in C \land A \neq 0 \rightarrow (1/A) \cdot A = 1 \text{ by (rule MMI_imp)}
proof -
  have S1: ((1/A) \in C \land A \in C ) \rightarrow ((1/A) \cdot A ) = (A \cdot (1/A)) \text{ by (rule MMI_axmulcom)}
  have S2: (A \in C \land A \neq 0 ) \rightarrow (1/A) \in C \text{ by (rule MMI_recclt)}
  have S3: (A \in C \land A \neq 0 ) \rightarrow A \in C \text{ by (rule MMI_pm3_26)}
  from S1 S2 S3 have S4: (A \in C \land A \neq 0 ) \rightarrow (A \cdot (1/A) ) = 1 \text{ by (rule MMI_sylanc)}
  have S5: (A \in C \land A \neq 0 ) \rightarrow (A \cdot (1/A) ) = 1 \text{ by (rule MMI_recidt)}
from S4 S5 show (A \in C \land A \neq 0 ) \rightarrow (A \cdot (1/A) ) = 1 \text{ by (rule MMI_eqtrd)}
qed

lemma (in MMIar0) MMI_divrec: assumes A1: A \in C and
A2: B \in C and
A3: B \neq 0
shows (A/B) = (A \cdot (1/B))
proof -
  from A2 have S1: B \in C.
  from A1 have S2: A \in C.
  from A2 have S3: B \in C.
  from A3 have S4: B \neq 0.
  from S3 S4 have S5: (1/B) \in C \text{ by (rule MMI_reccl)}
  from S2 S5 have S6: (A \cdot (1/B)) \in C \text{ by (rule MMI_mulcl)}
  from S1 S6 have S7: (B \cdot (A \cdot (1/B))) = (A \cdot (1/B)) \cdot B \text{ by (rule MMI_mulcom)}
  from A1 have S8: A \in C.
  from S5 have S9: (1/B) \in C.
  from A2 have S10: B \in C.
  from S8 S9 S10 have S11: (A \cdot (1/B)) \cdot B = (A \cdot (1/B)) \cdot B \text{ by (rule MMI_mulass)}
  from A2 have S12: B \in C.
  have S13: 1 \in C \text{ by (rule MMI_1cn)}
  from A3 have S14: B \neq 0.
  from S12 S13 S14 have S15: ((1/B) \cdot B) = 1 \text{ by (rule MMI_divcan1)}
  from S15 have S16: (A \cdot ((1/B) \cdot B)) = (A \cdot 1) \text{ by (rule MMI_opreq2i)}
  from A1 have S17: A \in C.
  from S17 have S18: (A \cdot 1) = A \text{ by (rule MMI_mulid1)}
  from S16 S18 have S19: (A \cdot ((1/B) \cdot B)) = A \text{ by (rule MMI_eqtr)}
  from S7 S11 S19 have S20: (B \cdot (A \cdot (1/B))) = A \text{ by (rule MMI_3eqtr)}
  from A1 have S21: A \in 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 MMIasar0) 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 MMIasar0) MMI_divrect:
shows ( A ∈ C ∧ B ∈ C ∧ B ≠ 0 ) −→ ( A / B ) = ( A · ( 1 / B ) )
proof -
have S1: A =
if (A ∈ ƒ , A , 0 ) →
(A / B) =
(if (A ∈ ƒ , A , 0 ) / B) by (rule MMI_opreq1)

have S2: A =
if (A ∈ ƒ , A , 0 ) →
(A · (1 / B)) =
(if (A ∈ ƒ , A , 0 ) · (1 / B)) by (rule MMI_opreq1)

from S1 S2 have S3: A =
if (A ∈ C , A , 0 ) →
((A / B) = (if (A ∈ ƒ , A , 0 ) / B)) ←→
(B ≠ 0) by (rule MMI_opreq1)

have S4: A =
(if (A ∈ ƒ , A , 0 ) / B) =
(if (A ∈ ƒ , A , 0 ) · (1 / B)) by (rule MMI_opreq1)

from S3 have S5: A =
if (A ∈ C , B , 0 ) →
((B ≠ 0) → (A / B) = (A · (1 / B))) ←→
(B ≠ 0) by (rule MMI_opreq1)

have S6: B =
if (A ∈ C , A , 0 ) →
(if (A ∈ C , A , 0 ) / B) =
(if (A ∈ C , A , 0 ) · (1 / B)) by (rule MMI_imbi2d)

have S7: B =
if (B ∈ C , B , 0 ) →
(1 / B) =
(if (B ∈ C , B , 0 )) by (rule MMI_opreq2)

from S7 have S8: B =
if (B ∈ C , B , 0 ) →
(if (A ∈ C , B , 0 ) · (1 / B)) =
(if (A ∈ C , A , 0 ) · (1 / if (B ∈ C , B , 0 ))) by (rule MMI_opreq2)

from S6 S8 have S9: B =
if (B ∈ C , B , 0 ) →
((if (A ∈ C , A , 0 ) / B) =
(if (A ∈ C , A , 0 ) · (1 / if (B ∈ C , B , 0 ))) ←→
(if (A ∈ C , A , 0 ) · (1 / if (B ∈ C , B , 0 ))) by (rule MMI_opreq2)

from S5 S9 have S10: B =
if (B ∈ C , B , 0 ) →
((B ≠ 0) → (if (A ∈ C , A , 0 ) / B) = (if (A ∈ C , A , 0 ) · (1 / if (B ∈ C , B , 0 ))) ←→
(if (B ∈ C , B , 0 ) ≠ 0) by (rule MMI_0cn)

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 S12 have S14: if ( B ∈ C , B , 0 ) ∈ C by (rule MMI_elimel)
from S12 S14 have S15: if ( B ∈ C , B , 0 ) ≠ 0 →
(if ( A ∈ C , A , 0 ) / if ( B ∈ C , B , 0 ) ) =
(if ( A ∈ C , A , 0 ) · ( 1 / if ( B ∈ C , B , 0 ) ) ) by (rule MMI_divrecz)
from S4 S10 S15 have S16: ( A ∈ C ∧ B ∈ C ) →
( B ≠ 0 →
( A / B ) = ( A · ( 1 / B ) ) ) by (rule MMI_dedth2h)
from S16 show ( A ∈ C ∧ B ∈ C ∧ B ≠ 0 ) →
( A / B ) = ( A · ( 1 / B ) ) by (rule MMI_3impia)
qed

lemma (in MMIsar0) MMI_divrec2t:
shows ( A ∈ C ∧ B ∈ C ∧ B ≠ 0 ) →
( A / B ) = ( ( 1 / B ) · A )
proof -
have S1: ( A ∈ C ∧ B ∈ C ∧ B ≠ 0 ) →
( A / B ) = ( A · ( 1 / B ) ) by (rule MMI_divrect)
have S2: ( A ∈ C ∧ ( 1 / B ) ∈ C ) →
( A · ( 1 / B ) ) = ( ( 1 / B ) · A ) by (rule MMI_axmulcom)
have S3: ( A ∈ C ∧ B ∈ C ∧ B ≠ 0 ) → A ∈ C by (rule MMI_3simp1)
have S4: ( B ∈ C ∧ B ≠ 0 ) → ( 1 / B ) ∈ C by (rule MMI_recclt)
from S4 have S5: ( A ∈ C ∧ B ∈ C ∧ B ≠ 0 ) →
( 1 / B ) ∈ C by (rule MMI_3adant1)
from S2 S3 S5 have S6: ( A ∈ C ∧ B ∈ C ∧ B ≠ 0 ) →
( A · ( 1 / B ) ) = ( ( 1 / B ) · A ) by (rule MMI_sylanc)
from S1 S6 show ( A ∈ C ∧ B ∈ C ∧ B ≠ 0 ) →
( A / B ) = ( ( 1 / B ) · A ) by (rule MMI_eqtrd)
qed

lemma (in MMIsar0) MMI_divasst:
shows ( ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) ∧ C ≠ 0 ) →
( ( A · B ) / C ) = ( A · ( B / C ) )
proof -
have S1: A ∈ C → A ∈ C by (rule MMI_id)
have S2: B ∈ C → B ∈ C by (rule MMI_id)
have S3: ( C ∈ C ∧ C ≠ 0 ) → ( 1 / C ) ∈ C by (rule MMI_recclt)
from S1 S2 S3 have S4: ( A ∈ C ∧ B ∈ C ∧ ( C ∈ C ∧ C ≠ 0 ) ) →
( A ∈ C ∧ B ∈ C ∧ ( 1 / C ) ∈ C ) by (rule MMI_3anim123i)
from S4 have S5: A ∈ C →
( B ∈ C →
( ( C ∈ C ∧ C ≠ 0 ) →
( A ∈ C ∧ B ∈ C ∧ ( 1 / C ) ∈ C ) ) ) by (rule MMI_3exp)
from S5 have S6: A ∈ C →
( B ∈ C →
( C ∈ C →
( C ≠ 0 →

1356
( \( A \in C \land B \in C \land ( 1 / C ) \in C ) \) ) by (rule MMI_exp4a)
from S6 have S7: ( \( ( A \in C \land B \in C \land C \in C ) \land C \neq 0 \) ) \( \rightarrow \)
( \( A \in C \land B \in C \land ( 1 / C ) \in C ) \) by (rule MMI_3impl)
have S8: ( \( A \in C \land B \in C \land ( 1 / C ) \in C ) \) \( \rightarrow \)
( \( A \cdot B \cdot ( 1 / C ) ) = \)
( \( A \cdot ( B \cdot ( 1 / C ) ) ) \) by (rule MMI_axmulass)
from S7 S8 have S9: ( \( ( A \in C \land B \in C \land C \in C ) \land C \neq 0 \) ) \( \rightarrow \)
( \( A \cdot B \cdot ( 1 / C ) ) = \)
( \( A \cdot ( B \cdot ( 1 / C ) ) ) \) by (rule MMI_syl)
have S10: ( \( ( A \cdot B ) \in C \land C \neq 0 \) ) \( \rightarrow \)
( \( A \cdot B ) / C ) =
( \( ( A \cdot B ) \cdot ( 1 / C ) ) \) by (rule MMI_divrect)
from S10 have S11: ( \( ( ( A \cdot B ) \in C \land C \land C \neq 0 ) \) ) \( \rightarrow \)
( \( ( A \cdot B ) / C ) = \)
( \( ( A \cdot B ) \cdot ( 1 / C ) ) ) by (rule MMI_3expa)
have S12: ( \( A \in C \land B \in C ) \) \( \rightarrow \) \( ( A \cdot B ) \in C ) \) by (rule MMI_axmulcl)
from S12 have S13: ( \( ( A \in C \land B \in C ) \land C \in C ) \) \( \rightarrow \)
( \( ( A \cdot B ) \in C \land C \in C ) \) by (rule MMI_anim1i)
from S13 have S14: ( \( A \in C \land B \in C \land C \in C ) \) \( \rightarrow \)
( \( ( A \cdot B ) \in C \land C \in C ) \) by (rule MMI_3impa)
from S11 S14 have S15: ( \( ( ( A \cdot B ) \in C \land B \in C \land C \in C ) \land C \neq 0 ) \) \( \rightarrow \)
( \( ( A \cdot B ) / C ) = \)
( ( \( A \cdot B ) \cdot ( 1 / C ) ) \) by (rule MMI_sylan)
have S16: ( \( ( B \in C \land C \in C \land C \neq 0 ) \) ) \( \rightarrow \)
( \( B / C ) = ( B \cdot ( 1 / C ) ) \) by (rule MMI_divrect)
from S16 have S17: ( \( ( B \in C \land C \in C ) \land C \neq 0 ) \) \( \rightarrow \)
( \( B / C ) = ( B \cdot ( 1 / C ) ) \) by (rule MMI_3expa)
from S17 have S18: ( \( ( A \in C \land B \in C \land C \in C ) \land C \neq 0 ) \) \( \rightarrow \)
( \( B / C ) = ( B \cdot ( 1 / C ) ) \) by (rule MMI_3adant1)
from S18 have S19: ( \( ( A \in C \land B \in C \land C \in C ) \land C \neq 0 ) \) \( \rightarrow \)
( \( A \cdot ( B / C ) ) = \)
( \( A \cdot ( B \cdot ( 1 / C ) ) \) ) by (rule MMI_opreq2d)
from S9 S15 S19 show ( \( ( A \in C \land B \in C \land C \in C ) \land C \neq 0 ) \) \( \rightarrow \)
( \( ( A \cdot B ) / C ) = ( A \cdot ( B / C ) \) ) by (rule MMI_3eqtr4d)
qed

lemma (in MMI_isar0) MMI_div23t:
shows ( \( ( A \in C \land B \in C \land C \in C ) \land C \neq 0 ) \) \( \rightarrow \)
( \( ( A \cdot B ) / C ) = ( ( A / C ) \cdot B ) \)
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: ( \( A \in C \land B \in C \land C \in C ) \) \( \rightarrow \)
( \( A \cdot B ) = ( B \cdot A ) \) by (rule MMI_3adant3)
from S2 have S3: ( \( ( A \in C \land B \in C \land C \in C ) \land C \neq 0 ) \) \( \rightarrow \)
( \( A \cdot B ) = ( B \cdot A ) \) by (rule MMI_adantr)
from S3 have S4: ( \( ( A \in C \land B \in C \land C \in C ) \land C \neq 0 ) \) \( \rightarrow \)

1357
( ( 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_3simpl)
  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_3adant)
    from S9 S11 S14 have S15: ( ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) ∧ C ≠ 0 ) →
( B · ( A / C ) ) = ( ( A / C ) · B ) by (rule MMI_syl)
    from S4 S6 S15 show ( ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) ∧ C ≠ 0 ) →
( ( A · B ) / C ) = ( ( A / C ) · B ) by (rule MMI_3eqtrd)
lemma (in MMIIsar0) MMI_div13t:
  shows ( ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) ∧ B ≠ 0 ) →
( ( A / B ) · C ) = ( ( C / B ) · A )
proof -
  have S1: ( A ∈ C ∧ C ∈ C ) →
( A · C ) = ( ( C · A ) / B ) by (rule MMI_axmulcom)
    from S1 have S2: ( A ∈ C ∧ C ∈ C ) →
( ( A · C ) / B ) = ( ( C · A ) / B ) by (rule MMI_opreq1d)
      from S2 have S3: ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) →
( ( A · C ) / B ) = ( ( C · A ) / B ) by (rule MMI_3adant)
        from S3 have S4: ( ( A ∈ C ∧ B ∈ C ∧ C ∈ C ) ∧ B ≠ 0 ) →
( ( A · C ) / B ) = ( ( C · A ) / B ) by (rule MMI_adantr)
          have S5: ( ( A ∈ C ∧ C ∈ C ∧ B ∈ C ) ∧ B ≠ 0 ) →
( ( A · C ) / B ) = ( ( A / B ) · C ) by (rule MMI_div23t)
            from S5 have S6: ( A ∈ C ∧ C ∈ C ∧ B ∈ C ) →
( B ≠ 0 →
( ( A · C ) / B ) =
( ( A / B ) · C ) ) by (rule MMI_ex)
from S6 have S7: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
(B ≠ 0 →
((A · C) / B) =
((A / B) · C)) by (rule MMI_3com23)
from S7 have S8: ((A ∈ C ∧ B ∈ C ∧ C ∈ C) ∧ B ≠ 0) →
((A · C) / B) = ((A / B) · C) by (rule MMI_imp)
have S9: ((C ∈ C ∧ A ∈ C ∧ B ∈ C) ∧ B ≠ 0) →
((C · A) / B) = ((C / B) · A) by (rule MMI_div23t)
from S9 have S10: (C ∈ C ∧ A ∈ C ∧ B ∈ C) →
(B ≠ 0 →
((C · A) / B) =
((C / B) · A)) by (rule MMI_ex)
from S10 have S11: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
(B ≠ 0 →
((C / B) · A) =
((C / B) · A)) by (rule MMI_3com1)
from S11 have S12: ((A ∈ C ∧ B ∈ C ∧ C ∈ C) ∧ B ≠ 0) →
((C / B) · A) = ((C / B) · A) by (rule MMI_imp)
from S4 S8 S12 show ((A ∈ C ∧ B ∈ C ∧ C ∈ C) ∧ B ≠ 0) →
((A / B) · C) = ((A / B) · A) by (rule MMI_3eqtr3d)

qed

lemma (in MMIIsar0) MMI_div12t:
  shows ((A ∈ C ∧ B ∈ C ∧ C ∈ C) ∧ C ≠ 0) →
  (A · (B / C)) = (B · (A / C))

proof -
  have S1: (A ∈ C ∧ (B / C) ∈ C) →
  (A · (B / C)) = (B / (C · A)) by (rule MMI_axmulcom)
  have S2: (A ∈ C ∧ B ∈ C ∧ C ∈ C) → A ∈ C by (rule MMI_3simp1)
  from S2 have S3: ((A ∈ C ∧ B ∈ C ∧ C ∈ C) ∧ C ≠ 0) →
  A ∈ C by (rule MMI_adantr)
  have S4: (B ∈ C ∧ C ∈ C ∧ C ≠ 0) →
  (B / C) ∈ C by (rule MMI_divclt)
  from S4 have S5: ((B ∈ C ∧ C ∈ C) ∧ C ≠ 0) →
  (B / C) ∈ C by (rule MMI_3expa)
  from S5 have S6: ((A ∈ C ∧ B ∈ C ∧ C ∈ C) ∧ C ≠ 0) →
  (B / C) ∈ C by (rule MMI_adantl1)
  from S1 S3 S6 have S7: ((A ∈ C ∧ B ∈ C ∧ C ∈ C) ∧ C ≠ 0) →
  (A · (B / C)) = (B / (C · A)) by (rule MMI_sylanc)
  have S8: ((B ∈ C ∧ C ∈ C ∧ A ∈ C) ∧ C ≠ 0) →
  (B / C) · A) = ((A / C) · B) by (rule MMI_div13t)
  from S8 have S9: (B ∈ C ∧ C ∈ C ∧ A ∈ C) →
  (C ≠ 0 →
  ((B / C) · A) =
  ((A / C) · B)) by (rule MMI_ex)
  from S9 have S10: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
  (C ≠ 0 →
  ((B / C) · A) =

1359
\[(A/C) \cdot B\] by (rule MMI_3comr)

from S10 have S11: \((A \in C \land B \in C \land C \in C) \land C \neq 0\) \(\rightarrow\)
\[(B/C) \cdot A = (A/C) \cdot B\] 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_imp)
from S13 have S14: \((A \in C \land B \in C \land C \in C) \land C \neq 0\) \(\rightarrow\)
\[(A/C) \cdot B = (B \cdot (A/C))\] by (rule MMI_axmulcom)

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 S15 have S16: \((A \in C \land B \in C \land C \in C) \land C \neq 0\) \(\rightarrow\)
\[(A/C) \in C\] by (rule MMI_adantr)
from S16 have S17: \((A \in C \land B \in C \land C \in C) \land C \neq 0\) \(\rightarrow\)
\[(A/C) \cdot B = (B \cdot (A/C))\] by (rule MMI_sylanc)

from S17 S11 S18 show \((A \in C \land B \in C \land C \in C) \land C \neq 0\) \(\rightarrow\)
\[(A/C) \cdot B = (B \cdot (A/C))\] by (rule MMI_3expa)

qed

lemma (in MMI_isar0) 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 A1 have S1: \(A \in C\).
from A2 have S2: \(B \in C\).
from A3 have S3: \(C \in C\).
from S1 S2 S3 have S4: \((A \in C \land B \in C \land C \in C)\) by (rule MMI_3pm3_2i)
have S5: \((A \in C \land B \in C \land C \in C) \land C \neq 0\) \(\rightarrow\)
\[(A \cdot B) / C = (A \cdot (B/C))\] by (rule MMI_divasst)
from S4 S5 show \(C \neq 0\) \(\rightarrow\)
\[(A \cdot B) / C = (A \cdot (B/C))\] by (rule MMI_mpan)
qed

lemma (in MMI_isar0) MMI_divass: assumes A1: \(A \in C\) and
A2: \(B \in C\) and
A3: \(C \in C\) and
A4: \(C \neq 0\)
shows \((A \cdot B) / C = (A \cdot (B/C))\)
proof -
from A4 have S1: \(C \neq 0\).
from A1 have S2: \(A \in C\).
from A2 have S3: \(B \in C\).
from A3 have S4: \(C \in C\).
from S2 S3 S4 have S5: \(C \neq 0\) \(\rightarrow\)

1360
( ( A \cdot B ) / C ) = ( A \cdot ( B / C ) ) \text{ by (rule MMI_divassz)}

from S1 S5 show ( ( A \cdot B ) / C ) = ( A \cdot ( B / C ) ) \text{ by (rule MMI_ax_mp)}

qed

lemma (in MMIsar0) MMI_divdir: assumes A1: A \in C and
A2: B \in C and
A3: C \in C and
A4: C \neq 0
shows ( ( A + B ) / C ) = ( ( A / C ) + ( B / C ) )

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: C \neq 0.
from S3 S4 have S5: ( 1 / C ) \in C by (rule MMI_reccl)
from S1 S2 S5 have S6: ( ( A + B ) \cdot ( 1 / C ) ) =
( ( A \cdot ( 1 / C ) ) + ( B \cdot ( 1 / C ) ) ) \text{ by (rule MMI_adddir)}
from A1 have S7: A \in C.
from A2 have S8: B \in C.
from S7 S8 have S9: ( A + B ) \in C by (rule MMI_addcl)
from A3 have S10: C \in C.
from A4 have S11: C \neq 0.
from S9 S10 S11 have S12: ( ( A + B ) / C ) =
( ( A + B ) \cdot ( 1 / C ) ) \text{ by (rule MMI_divrec)}
from A1 have S13: A \in C.
from A3 have S14: C \in C.
from A4 have S15: C \neq 0.
from S13 S14 S15 have S16: ( A / C ) = ( A \cdot ( 1 / C ) ) \text{ by (rule MMI_divrec)}
from A2 have S17: B \in C.
from A3 have S18: C \in C.
from A4 have S19: C \neq 0.
from S17 S18 S19 have S20: ( B / C ) = ( B \cdot ( 1 / C ) ) \text{ by (rule MMI_divrec)}
from S16 S20 have S21: ( ( A / C ) + ( B / C ) ) =
( ( A \cdot ( 1 / C ) ) + ( B \cdot ( 1 / C ) ) ) \text{ by (rule MMI_opreq12i)}
from S6 S12 S21 show ( ( A + B ) / C ) =
( ( A / C ) + ( B / C ) ) \text{ by (rule MMI_3eqtr4)}

qed

lemma (in MMIsar0) MMI_div23: assumes A1: A \in C and
A2: B \in C and
A3: C \in C and
A4: C \neq 0
shows ( ( A \cdot B ) / C ) = ( ( A / C ) \cdot B )

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: \(((A \cdot B) / C) = ((B \cdot A) / C)\)
  by (rule MMI_opreqii)
from A2 have S5: \(B \in C\).
from A1 have S6: \(A \in C\).
from A3 have S7: \(C \in C\).
from A4 have S8: \(C \neq 0\).
from S5 S6 S7 S8 have
  S9: \((B \cdot A) / C = (B \cdot (A / C))\) by (rule MMI_divass)
from A2 have S10: \(B \in C\).
from A1 have S11: \(A \in C\).
from A3 have S12: \(C \in C\).
from A4 have S13: \(C \neq 0\).
from S11 S12 S13 have S14: \((A / C) \in C\) by (rule MMI_divcl)
from S10 S14 have S15: \((B \cdot (A / C)) = ((A / C) \cdot B)\)
  by (rule MMI_mulcom)
from S4 S9 S15 show \(((A \cdot B) / C) = ((A / C) \cdot B)\)
  by (rule MMI_3eqtr)
qed

lemma (in MMIsar0) MMI_divdirz: assumes A1: \(A \in C\) and
         A2: \(B \in C\) and
         A3: \(C \in C\)
  shows \(C \neq 0 \longrightarrow ((A + B) / C) = ((A / C) + (B / C))\)
proof -
  have S1: \(C = \) if \((C \neq 0, C, 1)\) \(\longrightarrow\)
    \(((A + B) / C) = \(((A + B) / (if (C \neq 0, C, 1))\) by (rule MMI_opreq2)
    have S2: \(C = \) if \((C \neq 0, C, 1)\) \(\longrightarrow\)
      \((A / C) = \) (\(A / (if (C \neq 0, C, 1))\) by (rule MMI_opreq2)
      have S3: \(C = \) if \((C \neq 0, C, 1)\) \(\longrightarrow\)
        \((B / C) = \) (\(B / (if (C \neq 0, C, 1))\) by (rule MMI_opreq2)
        from S2 S3 have S4: \(C = \) if \((C \neq 0, C, 1)\) \(\longrightarrow\)
          \(((A / C) + (B / C)) = \) (((\(A / (if (C \neq 0, C, 1))\) + (\(B / (if (C \neq 0, C, 1))\)) by (rule MMI_opreq12d)
        from S1 S4 have S5: \(C = \) if \((C \neq 0, C, 1)\) \(\longrightarrow\)
((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 MMIvar0) 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) =
    (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/C) + (B/C)) =
    ((if(A ∈ C, A, 0)/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)) ↔
    ((if(A ∈ C, A, 0) + B)/C) =
    ((if(A ∈ C, A, 0)/C) + (B/C)) by (rule MMI_eqeq12d)
  from S5 have S6: A =
    if(A ∈ C, A, 0) →
\[
( \frac{A + B}{C} = \frac{A}{C} + \frac{B}{C} ) \quad \text{if} \quad A \in \mathbb{F}, A, 0 \quad \text{by} \quad \text{rule MMI_imbi2d}
\]

\[
\text{have S7: } B =
\]

\[
( \frac{A + B}{C} = \frac{A}{C} + \frac{B}{C} ) \quad \text{if} \quad B \in \mathbb{F}, B, 0 \quad \text{by} \quad \text{rule MMI_opreq2}
\]

\[
\text{have S8: } B =
\]

\[
( \frac{A + B}{C} = \frac{A}{C} + \frac{B}{C} ) \quad \text{if} \quad B \in \mathbb{F}, B, 0 \quad \text{by} \quad \text{rule MMI_opreq1d}
\]

\[
\text{have S9: } B =
\]

\[
( \frac{A + B}{C} = \frac{A}{C} + \frac{B}{C} ) \quad \text{if} \quad C \in \mathbb{F}, C, 0 \quad \text{by} \quad \text{rule MMI_imbi2d}
\]

\[
\text{have S10: } B =
\]

\[
( \frac{A + B}{C} = \frac{A}{C} + \frac{B}{C} ) \quad \text{if} \quad C \in \mathbb{F}, C, 0 \quad \text{by} \quad \text{rule MMI_neeq1}
\]

\[
\text{have S11: } B =
\]

\[
( \frac{A + B}{C} = \frac{A}{C} + \frac{B}{C} ) \quad \text{if} \quad C \in \mathbb{F}, C, 0 \quad \text{by} \quad \text{rule MMI_opreq2}
\]

\[
\text{have S12: } B =
\]

\[
( \frac{A + B}{C} = \frac{A}{C} + \frac{B}{C} ) \quad \text{if} \quad C \in \mathbb{F}, C, 0 \quad \text{by} \quad \text{rule MMI_neeq1}
\]

\[
\text{have S13: } C =
\]

\[
( \frac{A + B}{C} = \frac{A}{C} + \frac{B}{C} ) \quad \text{if} \quad C \in \mathbb{F}, C, 0 \quad \text{by} \quad \text{rule MMI_neeq1}
\]

\[
\text{have S14: } C =
\]

\[
( \frac{A + B}{C} = \frac{A}{C} + \frac{B}{C} ) \quad \text{if} \quad C \in \mathbb{F}, C, 0 \quad \text{by} \quad \text{rule MMI_opreq2}
\]

\[
\text{have S15: } C =
\]

\[
( \frac{A + B}{C} = \frac{A}{C} + \frac{B}{C} ) \quad \text{if} \quad C \in \mathbb{F}, C, 0 \quad \text{by} \quad \text{rule MMI_opreq2}
\]

\[
\text{have S16: } C =
\]

1364
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 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_opreq2)

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))
by (rule MMI_opreq12d)

from S13 S18 have S19: C =
(if (C ∈ C, C, 0) →
((if (A ∈ C, A, 0) / if (C ∈ C, C, 0)) + (if (B ∈ C, B, 0) / if (C ∈ C, C, 0)))
by (rule MMI_eqeq12d)

from S12 S19 have S20: 0 ∈ C by (rule MMI_0cn)

have S21: if (A ∈ C, A, 0) ∈ C by (rule MMI_elimel)

have S22: 0 ∈ C by (rule MMI_0cn)

have S23: if (B ∈ C, B, 0) ∈ C by (rule MMI_elimel)

have S24: 0 ∈ C by (rule MMI_0cn)

have S25: if (C ∈ C, C, 0) ∈ C by (rule MMI_elimel)

have S26: if (C ∈ C, C, 0) = 0 →
((if (A ∈ C, A, 0) + if (B ∈ C, B, 0)) / if (C ∈ C, C, 0))
by (rule MMI_imbi12d)

from S21 S25 S26 have S27: (A ∈ C ∧ B ∈ C ∧ C ∈ C) →
(C = 0) →
((A + B) / C) =
((A / C) + (B / C)) by (rule MMI_dedth3h)

from S27 show ((A ∈ C ∧ B ∈ C ∧ C ∈ C) ∧ C = 0) →
((A + B) / C) =
((A / C) + (B / C)) by (rule MMI_imp)

lemma (in MMIaser0) MMI_divcan3: assumes A1: A ∈ C and
A2: B ∈ C and
A3: \( A \neq 0 \)
shows \( ( A \cdot B ) / A ) = B \)

proof -
from A1 have S1: \( A \in C. \)
from A2 have S2: \( B \in C. \)
from A1 have S3: \( A \in C. \)
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: \( A \in C. \)
from A2 have S7: \( B \in C. \)
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)

qed

lemma (in MMIisar0) MMI_divcan4: assumes A1: \( A \in C \) and
A2: \( B \in C \) and
A3: \( A \neq 0 \)
shows \( ( B \cdot A ) / A ) = B \)

proof -
from A2 have S1: \( B \in C. \)
from A1 have S2: \( A \in C. \)
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: \( A \in C. \)
from A2 have S6: \( B \in C. \)
from A3 have S7: \( A \neq 0. \)
from S5 S6 S7 have S8: \( ( ( A \cdot B ) / A ) ) = B \) by (rule MMI_divcan3)
from S4 S8 show \( ( ( B \cdot A ) / A ) = B \) by (rule MMI_eqtr)

qed

lemma (in MMIisar0) MMI_divcan3z: assumes A1: \( A \in C \) and
A2: \( B \in C \)
shows \( A \neq 0 \longrightarrow ( ( A \cdot B ) / A ) = B \)

proof -
have S1: \( A = \)
if ( \( A \neq 0 \) , \( A , 1 \) ) \( \rightarrow \)
\( ( A \cdot B ) \) = 
( ( if ( \( A \neq 0 \) , \( A , 1 \) ) \cdot B ) ) by (rule MMI_opreq1)

have S2: \( A = \)
if ( \( A \neq 0 \) , \( A , 1 \) ) \( \rightarrow \)
\( A = \) if ( \( A \neq 0 \) , \( A , 1 \) ) by (rule MMI_id)
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 \leftrightarrow
( ( \text{if ( A \neq 0 , A , 1 ) \cdot B ) / \text{if ( A \neq 0 , A , 1 ) } ) =
B ) \text{ by (rule MMI_eqeq1d)}

\text{from A1 have S5: } A \in C.
\text{have S6: } 1 \in C \text{ by (rule MMI_1cn)}
\text{from S5 S6 have S7: } \text{if ( A \neq 0 , A , 1 ) } \in C \text{ by (rule MMI_keepel)}
\text{from A2 have S8: } B \in C.
\text{have S9: } \text{if ( A \neq 0 , A , 1 ) } \neq 0 \text{ by (rule MMI_elimne0)}
\text{from S7 S8 S9 have S10: } ( ( \text{if ( A \neq 0 , A , 1 ) \cdot B ) / \text{if ( A \neq 0 , A , 1 ) } ) =
B ) \text{ by (rule MMI_divcan3)}
\text{from S4 S10 show } A \neq 0 \rightarrow ( ( A \cdot B ) / A ) = B \text{ by (rule MMI_dedth)}
\text{qed}

\text{lemma (in MMIIsar0) MMI_divcan4z: assumes A1: } A \in C \text{ and }
A2: B \in C
\text{shows } A \neq 0 \rightarrow ( ( B \cdot A ) / A ) = B
\text{proof -}
\text{from A1 have S1: } A \in C.
\text{from A2 have S2: } B \in C.
\text{from S1 S2 have S3: } A \neq 0 \rightarrow ( ( A \cdot B ) / A ) = B \text{ by (rule MMI_divcan3z)}
\text{from A2 have S4: } B \in C.
\text{from A1 have S5: } A \in C.
\text{from S4 S5 have S6: } ( B \cdot A ) = ( A \cdot B ) \text{ by (rule MMI_mulcom)}
\text{from S6 have S7: } ( ( B \cdot A ) / A ) = ( ( A \cdot B ) / A ) \text{ by (rule MMI_opreq1i)}
\text{from S3 S7 show } A \neq 0 \rightarrow ( ( B \cdot A ) / A ) = B \text{ by (rule MMI_syl5eq)}
\text{qed}

\text{lemma (in MMIIsar0) MMI_divcan3t:}
\text{shows } ( A \in C \land B \in C \land A \neq 0 ) \rightarrow
( ( A \cdot B ) / A ) = B
\text{proof -}
\text{have S1: } A =
\text{if ( A \in C , A , 0 ) \rightarrow
( A \neq 0 \leftrightarrow \text{if ( A \in C , A , 0 ) } \neq 0 ) \text{ by (rule MMI_neeq1)}
\text{have S2: } A =
\text{if ( A \in C , A , 0 ) \rightarrow
( A \cdot B ) =
( \text{if ( A \in C , A , 0 ) } \cdot B ) \text{ by (rule MMI_opreq1)}
\text{have S3: } A =
\text{if ( A \in C , A , 0 ) \rightarrow
A = \text{if ( A \in C , A , 0 ) by (rule MMI_id)}
\text{from S2 S3 have S4: } A =
\text{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 ) } ) \text{ by (rule MMI_opreq12d)}
\text{from S4 have S5: } A =
\text{if ( A \in C , A , 0 ) \rightarrow}
\text{qed}

1367
\[
((A \cdot B) / A) = B \iff ((if (A \in C, A, 0) \cdot B) / if (A \in C, A, 0)) = B
\]
by (rule MMI_eqeq1d)

from S1 S5 have S6: A = if (A \in C, A, 0) \implies ((A \neq 0 \implies ((A \cdot B) / A) = B) \iff if (A \in C, A, 0) \neq 0 \implies ((if (A \in C, A, 0) \cdot B) / if (A \in C, A, 0)) = B)
by (rule MMI_imbi2d)

have S7: B = if (B \in C, B, 0) \implies (if (A \in C, A, 0) \cdot B) = (if (A \in C, A, 0) \cdot if (B \in C, B, 0)) by (rule MMI_opreq2)

from S7 have S8: B = if (B \in C, B, 0) \implies ((if (A \in C, A, 0) \cdot B) / if (A \in C, A, 0)) = ((if (A \in C, A, 0) \cdot if (B \in C, B, 0)) / if (A \in C, A, 0)) by (rule MMI_opreq1d)

have S9: B = if (B \in C, B, 0) \implies B = if (B \in C, B, 0) by (rule MMI_id)

from S8 S9 have S10: B = if (B \in C, B, 0) \implies ((if (A \in C, A, 0) \cdot B) / if (A \in C, A, 0)) = B \iff ((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_eqeq12d)

from S10 have S11: B = if (B \in C, B, 0) \implies ((if (A \in C, A, 0) \neq 0 \implies ((if (A \in C, A, 0) \cdot B) / if (A \in C, A, 0)) = B) \iff (if (A \in C, A, 0) \neq 0 \implies ((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) \implies 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_elim)

have S14: 0 \in C by (rule MMI_0cn)

from S14 have S15: if (B \in C, B, 0) \in C by (rule MMI_elim)

from S13 S15 have S16: if (A \in C, A, 0) \neq 0 \implies ((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) \implies (A \neq 0 \implies ((A \cdot B) / A) = B) by (rule MMI_dedth2h)

from S17 show (A \in C \land B \in C \land A \neq 0) \implies ((A \cdot B) / A) = B by (rule MMI_3impia)
lemma (in MMIsar0) MMI_divcan4t:
  shows ( A ∈ C ∧ B ∈ C ∧ A ≠ 0 ) →
  ( ( B · A ) / A ) = B

proof -
  have S1: ( A ∈ C ∧ B ∈ C ) →
  ( A · B ) = ( B · A ) by (rule MMI_axmulcom)
  from S1 have S2: ( A ∈ C ∧ B ∈ C ) →
  ( ( A · B ) / A ) = ( ( B · A ) / A ) by (rule MMI_opreq1d)
  from S2 have S3: ( A ∈ C ∧ B ∈ C ∧ A ≠ 0 ) →
  have S4: ( A ∈ C ∧ B ∈ C ∧ A ≠ 0 ) →
  ( ( A · B ) / A ) = B by (rule MMI_3adant3)
  from S3 S4 have
  ( A ∈ C ∧ B ∈ C ∧ A ≠ 0 ) →
  ( ( B · A ) / A ) = B by (rule MMI_divcan3t)
  have S5: ( A ∈ C ∧ B ∈ C ∧ A ≠ 0 ) →
  ( ( B · A ) / A ) = B by (rule MMI_eqtr3d)
qed

lemma (in MMIsar0) MMI_div11: assumes A1: A ∈ C and
  A2: B ∈ C and
  A3: C ∈ C and
  A4: C ≠ 0
  shows ( A / C ) = ( B / C ) ↔ A = B

proof -
  from A3 have S1: C ∈ C.
  from A1 have S2: A ∈ C.
  from A3 have S3: C ∈ C.
  from A4 have S4: C ≠ 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: ( C · ( A / C ) ) =
  ( C · ( B / C ) ) ↔ A = B by (rule MMI_eqtr12i)
  from S11 S20 show ( A / C ) = ( B / C ) ↔ A = B by (rule MMI_bitr3)
qed
lemma (in MMIsar0) 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 ∈ C , 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 ) =
      if ( B ∈ C , B , 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 ) = B ) ) −→
      ( ( if ( A ∈ C , A , 1 ) / C ) =
      ( if ( B ∈ C , B , 1 ) / C ) −→
      if ( A ∈ C , A , 1 ) =
      if ( B ∈ C , B , 1 ) ) ) by (rule MMI_bibi12d)
    have S9: C =

1370
if ( ( C ∈ C ∧ C ≠ 0 ) , C , 1 ) →
( if ( A ∈ C , A , 1 ) / C ) =
( if ( A ∈ C , A , 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 , B , 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 ) / C ) )
( ( if ( A ∈ C , A , 1 ) / if ( ( C ∈ C ∧ C ≠ 0 ) , C , 1 ) ) =
( if ( B ∈ C , B , 1 ) / if ( ( C ∈ C ∧ C ≠ 0 ) , C , 1 ) ) ) by (rule MMI_eqe12d)

from S11 have S12: C =
if ( ( C ∈ C ∧ C ≠ 0 ) , C , 1 ) →
( ( ( if ( A ∈ C , A , 1 ) / C ) =
( if ( B ∈ C , B , 1 ) / C ) ) →
( ( 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 ∈ C 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 ≠ 0 ) , C , 1 ) →
( C ∈ C )
if ( ( C ∈ C ∧ C ≠ 0 ) , C , 1 ) ∈ C ) by (rule MMI_eleq1)

have S18: C =
if ( ( C ∈ C ∧ C ≠ 0 ) , C , 1 ) →
( C ≠ 0 )
if ( ( C ∈ C ∧ C ≠ 0 ) , C , 1 ) ≠ 0 ) by (rule MMI_neeq1)

from S17 S18 have S19: C =
if ( ( C ∈ C ∧ C ≠ 0 ) , C , 1 ) →
( ( C ∈ C ∧ C ≠ 0 ) , C , 1 ) ≠ 0 )
if ( ( C ∈ C ∧ C ≠ 0 ) , C , 1 ) ∈ C ∧ if ( ( C ∈ C ∧ C ≠ 0 ) , C , 1 ) ≠ 0 ) ) by (rule MMI_anbi12d)

have S20: 1 =
if ( ( C ∈ C ∧ C ≠ 0 ) , C , 1 ) →
( 1 ∈ C )
if ( ( C ∈ C ∧ C ≠ 0 ) , C , 1 ) ∈ C ) by (rule MMI_eleq1)

have S21: 1 =
if ( ( C ∈ C ∧ C ≠ 0 ) , C , 1 ) by (rule MMI_neeq1)

( 1 ≠ 0 )
if ( ( C ∈ C ∧ C ≠ 0 ) , C , 1 ) ≠ 0 ) by (rule MMI_neeq1)
from S20 S21 have S22: 1 = 
if ( ( C ∈ C ∧ C ≠ 0 ) , C , 1 ) → 
( ( 1 ∈ C ∧ 1 ≠ 0 ) ←→
( if ( ( C ∈ C ∧ C ≠ 0 ) , C , 1 ) ∈ C ∧ if ( ( C ∈ C ∧ C ≠ 0 ) , C , 1 ) ≠ 0 ) ) by (rule MMI_anbi12d)

have S23: 1 ∈ C by (rule MMI_1cn)
have S24: 1 ≠ 0 by (rule MMI_ax1ne0)
from S23 S24 have S25: 1 ∈ C ∧ 1 ≠ 0 by (rule MMI_pm3_2i)
from S19 S22 S25 have S26: if ( ( C ∈ C ∧ C ≠ 0 ) , C , 1 ) ∈ C
∧ if ( ( C ∈ C ∧ C ≠ 0 ) , C , 1 ) ≠ 0 by (rule MMI_elimhyp)
from S26 have S27: if ( ( C ∈ C ∧ C ≠ 0 ) , C , 1 ) ∈ C
by (rule MMI_pm3_26i)
from S26 have S28: if ( ( C ∈ C ∧ C ≠ 0 ) , C , 1 ) ≠ 0 by (rule MMI_pm3_27i)

from S14 S16 S27 S29 have S30: ( 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 ( ( C ∈ C , A , 1 ) )
if ( ( B ∈ C , B , 1 ) ) by (rule MMI_div11)
from S4 S8 S12 S30 show ( A ∈ C ∧ B ∈ C ∧ ( C ∈ C ∧ C ≠ 0 ) ) →
( ( A / C ) = ( B / C ) ←→ A = B ) by (rule MMI_dedth3h)
qed

end

85 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 MMIar0) 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: \( (A \in \mathcal{F} \land A \neq 0) \rightarrow (A \cdot (1/A)) = 1 \) by (rule MMI_recidt)

from S3 S4 show \( (A \in \mathcal{C} \land A \neq 0) \rightarrow (A / A) = 1 \) by (rule MMI_eqtrd)

qed

lemma (in MMIsar0) MMI_div0t:
shows \( (A \in \mathcal{F} \land A \neq 0) \rightarrow (0 / A) = 0 \)
proof -
  have S1: \( 0 \in \mathcal{C} \) by (rule MMI_0cn)
  have S2: \( 0 \in \mathcal{C} \land A \in \mathcal{C} \land A \neq 0 \) \( \rightarrow \)
    \( (0 / A) = (0 \cdot (1 / A)) \) by (rule MMI_divrect)
  from S1 S2 have S3: \( A \in \mathcal{C} \land A \neq 0 \) \( \rightarrow \)
    \( (0 / A) = (0 \cdot (1 / A)) = 0 \)
    by (rule MMI_mul02t)
  from S4 S5 have S6: \( (A \in \mathcal{C} \land A \neq 0) \rightarrow (0 \cdot (1 / A)) = 0 \)
    by (rule MMI_syl)
  from S3 S6 have S7: \( (A \in \mathcal{C} \land A \neq 0) \rightarrow (0 / A) = 0 \) by (rule MMI_eqtrd)
qed

lemma (in MMIsar0) MMI_diveq0t:
shows \( (A \in \mathcal{F} \land C \in \mathcal{F} \land C \neq 0) \rightarrow ((A / C) = 0 \leftrightarrow A = 0) \)
proof -
  have S1: \( C \in \mathcal{F} \land C \neq 0 \) \( \rightarrow \) \( 0 / C = 0 \) by (rule MMI_div0t)
  from S1 have S2: \( C \in \mathcal{C} \land C \neq 0 \) \( \rightarrow \)
    \((A / C) = 0 \leftrightarrow (A / C) = 0\) by (rule MMI_eqeq2d)
  from S2 have S3: \( A \in \mathcal{C} \land C \in \mathcal{C} \land C \neq 0 \) \( \rightarrow \)
    \((A / C) = (0 / C) \leftrightarrow A = 0\) by (rule MMI_3adant1)
  have S4: \( 0 \in \mathcal{C} \) by (rule MMI_0cn)
  have S5: \( A \in \mathcal{C} \land 0 \in \mathcal{C} \land (C \in \mathcal{C} \land C \neq 0) \) \( \rightarrow \)
    \((A / C) = (0 / C) \leftrightarrow A = 0\) by (rule MMI_div11t)
  from S4 S5 have S6: \( A \in \mathcal{C} \land (C \in \mathcal{C} \land C \neq 0) \) \( \rightarrow \)
    \((A / C) = (0 / C) \leftrightarrow A = 0\) by (rule MMI_mp3an2)
  from S6 have S7: \( A \in \mathcal{C} \land C \in \mathcal{C} \land C \neq 0 \) \( \rightarrow \)
    \((A / C) = (0 / C) \leftrightarrow A = 0\) by (rule MMI_3impb)
  from S3 S7 show \( A \in \mathcal{C} \land C \in \mathcal{C} \land C \neq 0 \) \( \rightarrow \)
    \((A / C) = 0 \leftrightarrow A = 0\) by (rule MMI_bitr3d)
qed

lemma (in MMIsar0) MMI_recrec: assumes A1: \( A \in \mathcal{C} \) and
  A2: \( A \neq 0 \)
shows \( (1 / (1 / A)) = A \)
proof -
  from A1 have S1: \( A \in \mathcal{C} \).
  from A2 have S2: \( A \neq 0 \).
from S1 S2 have S3: ( 1 / A ) ∈ C by (rule MMI_reccl)
have S4: 1 ∈ C by (rule MMI_1cn)
from A1 have S5: A ∈ C.
have S6: 1 ≠ 0 by (rule MMI_ax1ne0)
from A2 have S7: A ≠ 0.
from S4 S5 S6 S7 have S8: ( 1 / A ) ≠ 0 by (rule MMI_divne0)
from S3 S8 have S9: ( ( 1 / A ) · ( 1 / ( 1 / A ) ) ) = 1
   by (rule MMI_reccl)
from S9 have S10: ( 1 / A ) ≠ 0 by (rule MMI_divne0)
from S3 S8 have S11: ( 1 / ( 1 / A ) ) = 1 by (rule MMI_recid)
from S11 have S12: A ∈ C.
from A2 have S13: A ∈ C.
from S14 have S15: A ≠ 0.
from S16 have S17: A ≠ 0.
from S3 S4 have S18: ( 1 / A ) ≠ 0.
from S17 S18 have S19: ( ( 1 / A ) · ( 1 / ( 1 / A ) ) ) ∈ C by (rule MMI_reccl)
from S15 S16 S19 have S20:
   ( ( A · ( 1 / A ) ) · ( 1 / ( 1 / A ) ) ) =
   ( A · ( ( 1 / A ) · ( 1 / ( 1 / A ) ) ) ) by (rule MMI_mulass)
from S19 have S21: ( 1 / ( 1 / A ) ) ∈ C.
from S21 have S22: ( 1 · ( 1 / ( 1 / A ) ) ) =
   ( 1 / ( 1 / A ) ) by (rule MMI_mulid2)
from S14 S20 S22 have S23:
   ( ( A · ( ( 1 / A ) · ( 1 / ( 1 / A ) ) ) ) ) =
   ( 1 / ( 1 / A ) ) by (rule MMI_3eqtr3)
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 MMIIsar0) 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 MMIIsar0) MMI_div0: assumes A1: A ∈ C and
\[ A_2: A \neq 0 \]
shows \((0 / A) = 0\)
\[ \text{proof -} \]
from \(A_1\) have \(S_1: A \in C\).
from \(A_2\) have \(S_2: A \neq 0\).
have \(S_3: (A \in C \land A \neq 0) \rightarrow (0 / A) = 0\) by (rule MMI_div0t)
from \(S_1\) \(S_2\) \(S_3\) show \((0 / A) = 0\) by (rule MMI_mp2an)
\[ \text{qed} \]

lemma (in MMIasar0) MMI_div1: assumes \(A_1: A \in C\)
shows \((A / 1) = A\)
\[ \text{proof -} \]
from \(A_1\) have \(S_1: A \in C\).
from \(S_1\) have \(S_2: (1 \cdot A) = A\) by (rule MMI_mulid2)
from \(A_1\) have \(S_3: A \in C\).
have \(S_4: 1 \in C\) by (rule MMI_1cn)
from \(A_1\) have \(S_5: A \in C\).
have \(S_6: 1 \neq 0\) by (rule MMI_axine0)
from \(S_3\) \(S_4\) \(S_5\) \(S_6\) have \(S_7: (A / 1) = A \leftrightarrow (1 \cdot A) = A\)
by (rule MMI_divmul)
from \(S_2\) \(S_7\) show \((A / 1) = A\) by (rule MMI_mpbir)
\[ \text{qed} \]

lemma (in MMIasar0) MMI_div1t:
shows \(A \in C \rightarrow (A / 1) = A\)
\[ \text{proof -} \]
have \(S_1: A = \)
if \((A \in C, A, 1) \rightarrow (A / 1) =\)
\((A \in C, A, 1) / 1\) by (rule MMI_opreq1)
have \(S_2: A =\)
if \((A \in C, A, 1) \rightarrow A = if (A \in C, A, 1)\) by (rule MMI_id)
from \(S_1\) \(S_2\) have \(S_3: A =\)
if \((A \in C, A, 1) \rightarrow (A / 1) =\)
\((A \in C, A, 1) / 1\) by (rule MMI_opreq1)
have \(S_4: 1 \in C\) by (rule MMI_1cn)
from \(S_4\) have \(S_5: if (A \in C, A, 1) \in C\) by (rule MMI_elimel)
from \(S_5\) have \(S_6: (if (A \in C, A, 1) / 1) =\)
\((A \in C, A, 1) \rightarrow (A / 1) = A\) by (rule MMI_dedth)
\[ \text{qed} \]

lemma (in MMIasar0) MMI_divnegt:
shows \((A \in C \land B \in C \land B \neq 0) \rightarrow (- (A / B)) = (- (A / B))\)
\[ \text{proof -} \]
from \(A_1\) have \(S_1: A \in C\).
from \(A_2\) have \(S_2: A \neq 0\).
have \(S_3: (A \in C \land A \neq 0) \rightarrow (0 / A) = 0\) by (rule MMI_div0t)
from \(S_1\) \(S_2\) \(S_3\) show \((0 / A) = 0\) by (rule MMI_mp2an)
\[ \text{qed} \]
proof - have S1: \((A \in C \land (1 / B) \in C) \rightarrow \)
\(((- A) \cdot (1 / B)) = \)
\((- (A \cdot (1 / B)))\) by (rule MMI_mulneg1t)
  have S2: \((B \in C \land B \neq 0) \rightarrow (1 / B) \in C\) by (rule MMI_reclt)
  from S1 S2 have S3: \((A \in C \land (B \in C \land B \neq 0)) \rightarrow \)
\((- (A) \cdot (1 / B)) = \)
\((- (A \cdot (1 / B)))\) by (rule MMI_sylan2)
  have S4: \((A \in C \land (B \in C \land B \neq 0)) \rightarrow \)
\((- (A) \cdot (1 / B)) = \)
\((- (A \cdot (1 / B)))\) by (rule MMI_3impb)
  have S5: \((- (A) \in C \land (B \in C \land B \neq 0)) \rightarrow \)
\((- (A) / B) = (- (A)) / (B)\) by (rule MMI_divrect)
  have S6: \((A \in C \land (-A) \in C)\) by (rule MMI_negclt)
  from S5 S6 have S7: \((A \in C \land (B \in C \land B \neq 0)) \rightarrow \)
\((- (A) / B) = (- (A)) / (B)\) by (rule MMI_3eqtr4rd)
  have S8: \((A \in C \land (B \in C \land B \neq 0)) \rightarrow \)
\((- (A / B)) = (- (A)) / (B)\) by (rule MMI_3eqtr4rd)
from S8 have S9: \((A \in C \land (B \in C \land B \neq 0)) \rightarrow \)
\((- (A / B)) = (- (A)) / (B)\) by (rule MMI_divsubdirt)

lemma (in MMIar0) MMI_divsubdirt:
  shows \((A \in C \land (B \in C \land C \in C) \land C \neq 0) \rightarrow \)
\(((A - B) / C) = \)
\((A / C) - (B / C))\) proof -
  have S1: \(((A \in C \land (-B) \in C \land C \in C) \land C \neq 0) \rightarrow \)
\((A + (-B)) / C) = \)
\(((A / C) + ((-B) / C))\) by (rule MMI_divdirt)
  have S2: \((B \in C \rightarrow (-B) \in C)\) by (rule MMI_negclt)
  from S1 S2 have S3: \(((A \in C \land B \in C \land C \in C) \land C \neq 0) \rightarrow \)
\((A + (-B)) / C) = \)
\(((A / C) + ((-B) / C))\) by (rule MMI_syl3an12)
  have S4: \((A \in C \land B \in C) \rightarrow \)
\((A + (-B)) = (A - B)\) by (rule MMI_negsubt)
  from S4 have S5: \((A \in C \land B \in C \land C \in C) \rightarrow \)
\((A + (-B)) = (A - B)\) by (rule MMI_3adant3)
  have S6: \((A \in C \land B \in C \land C \in C) \rightarrow \)
\(((A + (-B)) / C) = \)
\(((A - B) / C)\) by (rule MMI_opreq1d)
  from S6 have S7: \(((A \in C \land B \in C \land C \in C) \land C \neq 0) \rightarrow \)
\(((A + (-B)) / C) = \)
\(((A - B) / C)\) by (rule MMI_adantr)
have S8: \( (B \in \mathbb{C} \land C \in \mathbb{C} \land C \neq 0) \rightarrow \)
\( (-(B/C)) = \((-(B/C)) \) \) by (rule MMI_divnegt)
from S8 have S9: \( (B \in \mathbb{C} \land C \in \mathbb{C} \land C \neq 0) \rightarrow \)
\( (-(B/C)) = \((-(B/C)) \) \) by (rule MMI_3expa)

have S12: \( (A \in \mathbb{C} \land B \in \mathbb{C} \land C \in \mathbb{C} \land C \neq 0) \rightarrow \)
\( (A/C) + ((-B)/C) \) \) \) by (rule MMI_opreq2d)

from S12 S15 S18 have S19: \( (A \in \mathbb{C} \land B \in \mathbb{C} \land C \in \mathbb{C} \land C \neq 0) \rightarrow \)
\( (A/C) + ((-B)/C) \) \) \) by (rule MMI_sylanc)

from S3 S7 S20 show \( (A \in \mathbb{C} \land B \in \mathbb{C} \land C \in \mathbb{C} \land C \neq 0) \rightarrow \)
\( (A-B)/C \) \) \) by (rule MMI_3eqtr3d)

qed

end

86 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.
86.1 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 \((\forall a b. a \in \text{real} \land b \in \text{real} \rightarrow (a, b) \in \text{lessrrel} \leftrightarrow \neg (a = b \lor (b, a) \in \text{lessrrel}))\)
    proof -
    have I:
      \((\forall a b. a \in \text{real} \land b \in \text{real} \rightarrow (a <_R b \leftrightarrow \neg (a = b \lor b <_R a)))\)
      using pre_axlttri by blast
      { fix a b assume a \in \text{real} \land b \in \text{real}
        with I have \((a <_R b \leftrightarrow \neg (a = b \lor b <_R a)))\)
        by blast
        hence \((a, b) \in \text{lessrrel} \leftrightarrow \neg (a = b \lor (b, a) \in \text{lessrrel}))\)
        by simp
      } thus \((\forall a b. a \in \text{real} \land b \in \text{real} \rightarrow (a, b) \in \text{lessrrel} \leftrightarrow \neg (a = b \lor (b, a) \in \text{lessrrel})))\)
        by blast
    qed
  moreover
  have \((\forall a b c. a \in \text{real} \land b \in \text{real} \land c \in \text{real} \rightarrow (a, b) \in \text{lessrrel} \land (b, c) \in \text{lessrrel} \rightarrow (a, c) \in \text{lessrrel}))\)
    proof -
    have II: \((\forall a b c. a \in \text{real} \land b \in \text{real} \land c \in \text{real} \rightarrow ((a <_R b \land b <_R c) \rightarrow a <_R c))\)
      using pre_axltttrn by blast
      { fix a b c assume a \in \text{real} \land b \in \text{real} \land c \in \text{real}
        with II have \((a <_R b \land b <_R c) \rightarrow a <_R c)\)
          by blast
        hence \((a, b) \in \text{lessrrel} \land (b, c) \in \text{lessrrel} \rightarrow (a, c) \in \text{lessrrel})\)
          by blast
      } thus \((\forall a b c. a \in \text{real} \land b \in \text{real} \land c \in \text{real} \rightarrow ((a <_R b \land b <_R c) \rightarrow a <_R c))\)
            by blast
          qed
        qed
      qed
    qed
  qed
  thus MMIsar_valid by (simp add: lessrrel_def)
  qed

1378
\begin{align*}
\text{hence} & \quad \langle a, b \rangle \in \text{lessrrel} \land \langle b, c \rangle \in \text{lessrrel} \rightarrow \langle a, c \rangle \in \text{lessrrel} \\
& \quad \text{by simp} \\
\} \quad \text{thus} \quad (\forall a \ b \ c. \ a \in \text{real} \land b \in \text{real} \land c \in \text{real} \rightarrow \langle a, b \rangle \in \text{lessrrel} \land \langle b, c \rangle \in \text{lessrrel} \rightarrow \langle a, c \rangle \in \text{lessrrel}) \\
& \quad \text{by blast} \\
\text{qed}
\end{align*}

moreover have \((\forall A \ B \ C. \ A \in \text{real} \land B \in \text{real} \land C \in \text{real} \rightarrow \langle A, B \rangle \in \text{lessrrel} \rightarrow \langle C, A \rangle \in \text{lessrrel})\)

using \text{pre_axltadd} by simp

moreover have \((\forall A \ B. \ A \in \text{real} \land B \in \text{real} \rightarrow \langle zero, A \rangle \in \text{lessrrel} \land \langle zero, B \rangle \in \text{lessrrel} \rightarrow \langle zero, cmulset \langle A, B \rangle \rangle \in \text{lessrrel})\)

using \text{pre_axmulgt0} by simp

moreover have \((\forall S. S \subseteq \text{real} \land S \neq 0 \land (\exists x \in \text{real}. \ (\forall y \in S. \langle y, x \rangle \in \text{lessrrel}) \rightarrow (\exists z \in S. \ (y, z) \in \text{lessrrel})))))\)

using \text{pre_axsup} by simp

moreover have \text{R \subseteq C} using \text{axresscn} by simp

moreover have \text{1 \neq 0} using \text{axine0} by simp

moreover have \text{C isASet} by simp

moreover have \text{CplxAdd(R,A) : C \times C \rightarrow C}

using \text{axaddopr} by simp

moreover have \text{CplxMul(R,A,M) : C \times C \rightarrow C}

using \text{axmulopr} by simp

moreover have \(\forall a \ b. \ a \in C \land b \in C \rightarrow a \cdot b = b \cdot a\)

using \text{axmulcom} by simp

hence \((\forall a \ b. \ a \in C \land b \in C \rightarrow \text{cmulset} \ (a, b) = \text{cmulset} \ (b, a))\)

by simp

moreover have \(\forall a \ b. \ a \in C \land b \in C \rightarrow a + b \in C\)

using \text{axaddcl} by simp

hence \((\forall a \ b. \ a \in C \land b \in C \rightarrow \text{caddset} \ (a, b) \in C)\)

by simp

moreover have \(\forall a \ b. \ a \in C \land b \in C \rightarrow a \cdot b \in C\)

using \text{axmulcl} by simp

hence \((\forall a \ b. \ a \in C \land b \in C \rightarrow \text{cmulset} \ (a, b) \in C)\)

by simp

moreover have \(\forall a \ b \ C. \ a \in C \land b \in C \land C \in C \rightarrow a \cdot (b + C) = a \cdot b + a \cdot C\)

using \text{axdistr} by simp
hence ∀a b C.
   a ∈ C ∧ b ∈ C ∧ C ∈ C →
   cmulset (a, caddset (b, C)) =
   caddset
   (cmulset (a, b), cmulset (a, C))
by simp
moreover have ∀a b. a ∈ C ∧ b ∈ C →
   a + b = b + a
using axaddcom by simp
hence ∀a b.
   a ∈ C ∧ b ∈ C →
   caddset (a, b) = caddset (b, a) by simp
moreover have ∀a b C. a ∈ C ∧ b ∈ C ∧ C ∈ C →
   a + b + C = a + (b + C)
using axaddass by simp
hence ∀a b C.
   a ∈ C ∧ b ∈ C ∧ C ∈ C →
   cmulset (cmulset (a, b), C) =
   cmulset (a, cmulset (b, C)) by simp
moreover have ∀a b c. a ∈ C ∧ b ∈ C ∧ c ∈ C → a·b·c = a·(b·c)
using axmulass by simp
hence ∀a b C.
   a ∈ C ∧ b ∈ C ∧ C ∈ C →
   cmulset (cmulset (a, b), C) =
   cmulset (a, cmulset (b, C)) by simp
moreover have 1 ∈ R using axi2m1 by simp
moreover have i·i + 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 axrecex by simp
hence ∀a. a ∈ C ∧ a ≠ 0 →
   (∃x∈C. cmulset (a, x) = 1 ) by simp
moreover have ∀a. a ∈ C → a·1 = a
   using axi1d by simp
hence ∀a. a ∈ C →
   cmulset (a, 1) = a by simp
moreover have ∀a b. a ∈ R ∧ b ∈ R → a + b ∈ R
   using axaddrel by simp
hence ∀a b. a ∈ R ∧ b ∈ R →
caddset \( \langle a, b \rangle \in \mathbb{R} \) by simp
moreover have \( \forall a \ b. \ a \in \mathbb{R} \land b \in \mathbb{R} \rightarrow a \cdot b \in \mathbb{R} \)
using axmulrcl by simp
hence \( \forall a \ b. \ a \in \mathbb{R} \land b \in \mathbb{R} \rightarrow \)
cmulset \( \langle a, b \rangle \in \mathbb{R} \) by simp
moreover have \( \forall a. \ a \in \mathbb{R} \rightarrow (\exists x \in \mathbb{R}. \ a + x = 0) \)
using axrnegeq by simp
hence \( \forall a. \ a \in \mathbb{R} \rightarrow \)
( \( \exists x \in \mathbb{R}. \ \text{caddset} \ (a, x) = 0 \) ) by simp
moreover have \( \forall a. \ a \in \mathbb{R} \land a \neq 0 \rightarrow (\exists x \in \mathbb{R}. \ a \cdot x = 1) \)
using axrrecex by simp
hence \( \forall a. \ a \in \mathbb{R} \land a \neq 0 \rightarrow \)
( \( \exists x \in \mathbb{R}. \ \text{cmulset} \ (a, x) = 1 \) ) by simp
ultimately have
( ( ( \( \forall a \ b. \ a \in \mathbb{R} \land b \in \mathbb{R} \rightarrow \)
\( \langle a, b \rangle \in \text{lessrrel} \leftrightarrow \)
\( \neg (a = b \lor (b, a) \in \text{lessrrel}) \) ) \land
( \( \forall a \ b C. \ a \in \mathbb{R} \land b \in \mathbb{R} \land C \in \mathbb{R} \rightarrow \)
\( \langle a, b \rangle \in \text{lessrrel} \land \)
\( \langle b, C \rangle \in \text{lessrrel} \rightarrow \)
\( \langle a, C \rangle \in \text{lessrrel} \) ) \land
( \( \forall a \ b C. \ a \in \mathbb{R} \land b \in \mathbb{R} \land C \in \mathbb{R} \rightarrow \)
\( \langle a, b \rangle \in \text{lessrrel} \rightarrow \)
\( \langle \text{caddset} \ (C, a), \text{caddset} \ (C, b) \rangle \in \text{lessrrel} \) ) \land
( \( \forall S. \ S \subseteq \mathbb{R} \land S \neq 0 \land \)

1381
\[
( \exists x \in \mathbb{R}. \ \forall y \in S. \ \langle y, x \rangle \in \text{lessrel} ) \quad \rightarrow \\
( \exists x \in \mathbb{R}. \\
( \forall y \in S. \ \langle x, y \rangle \notin \text{lessrel} ) \quad \land \\
( \forall y \in \mathbb{R}. \ \langle y, x \rangle \in \text{lessrel} \quad \rightarrow \\
( \exists z \in S. \ \langle y, z \rangle \in \text{lessrel} ) \\
) \\
) \\
) \\
) \quad \land \\
R \subseteq C \quad \land \\
1 \neq 0 \\
) \quad \land \\
(C \text{ isASet} \quad \land \text{ caddset} \in C \times C \rightarrow C \quad \land \\
\text{ cmulset} \in C \times C \rightarrow C \\
) \\
) \\
) \\
( (\forall a. \ b. \\
a \in C \land b \in C \quad \rightarrow \\
\text{ cmulset} \ (a, b) = \text{ cmulset} \ (b, a) \\
) \quad \land \\
(\forall a. \ b. \ a \in C \land b \in C \quad \rightarrow \\
\text{ caddset} \ (a, b) \in C \\
) \\
) \quad \land \\
(\forall a. \ b. \ a \in C \land b \in C \quad \rightarrow \\
\text{ cmulset} \ (a, b) \in C \\
) \quad \land \\
(\forall a. \ b. \ a \in C \land b \in C \land C \in C \quad \rightarrow \\
\text{ cmulset} \ (a, \text{ caddset} \ (b, C)) = \\
\text{ caddset} \\
(\text{ cmulset} \ (a, b), \text{ cmulset} \ (a, C) ) \\
) \\
) \quad \land \\
) \\
) \\
)
\[
\begin{align*}
\forall a, b \in \mathbb{C} & \quad (a \in \mathbb{C} \land b \in \mathbb{C} \rightarrow \\
& \quad \text{caddset } \langle a, b \rangle = \text{caddset } \langle b, a \rangle \\
\) \land \\
\forall a, b, c \in \mathbb{C} & \quad a \in \mathbb{C} \land b \in \mathbb{C} \land c \in \mathbb{C} \rightarrow \\
& \quad \text{caddset } \langle \text{caddset } \langle a, b \rangle, c \rangle = \\
& \quad \text{caddset } \langle a, \text{caddset } \langle b, c \rangle \rangle \\
\) \land \\
\forall a, b, c \in \mathbb{C} & \quad a \in \mathbb{C} \land b \in \mathbb{C} \land c \in \mathbb{C} \rightarrow \\
& \quad \text{cmulset } \langle \text{cmulset } \langle a, b \rangle, c \rangle = \\
& \quad \text{cmulset } \langle a, \text{cmulset } \langle b, c \rangle \rangle \\
\) \land \\
(1 \in \mathbb{R} \land \\
& \quad \text{caddset } \langle \text{cmulset } \langle i, i \rangle, 1 \rangle = 0 \\
\) \land \\
\forall a \in \mathbb{C} & \quad a \in \mathbb{C} \rightarrow \text{caddset } \langle a, 0 \rangle = a \\
\) \land \\
i \in \mathbb{C} \\
\) \land \\
\big( \\
\big( \forall a \in \mathbb{C} & \quad \big( \\
& \quad (\exists x \in \mathbb{C}. \text{caddset } \langle a, x \rangle = 0 \\
\big) \\
\) \land \\
\big( \forall a \in \mathbb{C} & \quad a \neq 0 \rightarrow \\
& \quad (\exists x \in \mathbb{C}. \text{cmulset } \langle a, x \rangle = 1 \\
\big) \\
\) \land \\
\big( \forall a \in \mathbb{C} & \quad \text{cmulset } \langle a, 1 \rangle = a \\
\big) \\
\) \land \\
\big( \\
\big( \forall a, b \in \mathbb{R} & \quad a \in \mathbb{R} \land b \in \mathbb{R} \rightarrow \\
& \quad \text{caddset } \langle a, b \rangle \in \mathbb{R} \\
\big) \\
\) \land \\
\end{align*}
\]
(∀a b. a ∈ R ∧ b ∈ R →
  cmulset ⟨a, b⟩ ∈ R)
)
∧
(∀a. a ∈ R →
  (∃x∈R. caddset ⟨a, x⟩ = 0)
)
∧
(∀a. a ∈ R ∧ a ≠ 0 →
  (∃x∈R. cmulset ⟨a, x⟩ = 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

87 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 theorems 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.
87.1 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** -
let real = \(\mathbb{R}\)
let complex = \(\mathbb{C}\)
let one = 1
let zero = 0
let iunit = i
let caddset = CplxAdd(\(\mathbb{R}, A\))
let cmulset = CplxMul(\(\mathbb{R}, A, M\))
let lessrrel = StrictVersion(CplxROrder(\(\mathbb{R}, A, r\)))

have MMIsar0
  (real, complex, one, zero, iunit, caddset, cmulset, lessrrel)
  using MMIsar_valid by simp

then have real \(\cap\) {complex, {complex}} = 0
  by (rule MMIsar0.MM1_renfdisj)

thus \(\mathbb{R} \cap \{+\infty, -\infty\} = 0\) by simp

**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 \(<\) Orders \(\mathbb{R}^*\)

**Proof** -
let real = \(\mathbb{R}\)
let complex = \(\mathbb{C}\)
let one = 1
let zero = 0
let iunit = i
let caddset = CplxAdd(\(\mathbb{R}, A\))
let cmulset = CplxMul(\(\mathbb{R}, A, M\))
let lessrrel = StrictVersion(CplxROrder(\(\mathbb{R}, A, r\)))

have MMIsar0
  (real, complex, one, zero, iunit, caddset, cmulset, lessrrel)
  using MMIsar_valid by simp

then have
  (lessrrel \(\cap\) real \(\times\) real \(\cup\) \{{complex\}, complex\}) \(\cup\) real \(\times\) {complex} \(\cup\)
  {{complex}} \(\times\) real Orders (real \(\cup\) {complex, {complex}})

1385
by (rule MMIsar0.MMI_xrltso)
moreover have \( \text{lessrrel} \cap \text{real} \times \text{real} = \text{lessrrel} \)
using cplx_strict_ord_on_cplx_reals by auto
ultimately show \( \text{Orders} \ \mathbb{R}^* \) by simp
qed

Metamath defines the usual < and \( \leq \) ordering relations for the extended
real line, including \( +\infty \) and \( -\infty \).

lemma (in complex0) xrrebndt: assumes A1: \( x \in \mathbb{R}^* \)
shows \( x \in \mathbb{R} \longleftrightarrow ( -\infty < x \land x < +\infty ) \)
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))
have MMIsar0
(\( \text{real, complex, one, zero, iunit, caddset, cmulset, lessrrel} \))
using MMIsar_valid by simp
then have \( x \in \mathbb{R} \cup \{\mathbb{C}, \{\mathbb{C}\}\} \longrightarrow \)
\( x \in \mathbb{R} \longleftrightarrow \{\{\mathbb{C}\}, x\} \in \text{lessrrel} \cap \mathbb{R} \times \mathbb{R} \cup \{(\{\mathbb{C}\}, \mathbb{C}\}\} \cup \mathbb{R} \times \{\mathbb{C}\} \cup \{\{\mathbb{C}\}\} \times \mathbb{R} \land \)
\( \{x, \mathbb{C}\} \in \text{lessrrel} \cap \mathbb{R} \times \mathbb{R} \cup \{(\{\mathbb{C}\}, \mathbb{C}\}\} \cup \mathbb{R} \times \{\mathbb{C}\} \cup \{\{\mathbb{C}\}\} \times \mathbb{R} \)
by (rule MMIsar0.MMI_xrrebndt)
then have \( x \in \mathbb{R}^* \longrightarrow ( x \in \mathbb{R} \longleftrightarrow ( -\infty < x \land x < +\infty ) ) \)
by simp
with A1 show thesis by simp
qed

A quite involved inequality.

lemma (in complex0) lt2mul2divt:
assumes A1: \( a \in \mathbb{R} \quad b \in \mathbb{R} \quad c \in \mathbb{R} \quad d \in \mathbb{R} \) and \( A2: \ 0 < b \quad 0 < d \)
shows \( a/b < c/d \longleftrightarrow a/d < c/b \)
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))
have MMIsar0
(\( \text{real, complex, one, zero, iunit, caddset, cmulset, lessrrel} \))
then have
(a ∈ real ∧ b ∈ real) ∧
(c ∈ real ∧ d ∈ real) ∧
⟨zero, b⟩ ∈ lessrrel ∩ real × real ∪
{⟨complex⟩, complex} ∪ real × {complex} ∪ {complex} × real ∧
⟨zero, d⟩ ∈ lessrrel ∩ real × real ∪
{⟨complex⟩, complex} ∪ real × {complex} ∪ {complex} × real −→
⟨cmulset (a, b), cmulset (c, d)⟩ ∈
lessrrel ∩ real × real ∪ {⟨complex⟩, complex} ∪ real × {complex} ∪ {complex} × real
←→
⟨⋃{x ∈ complex . cmulset (d, x) = a}, a⟩ ∈
⋃{x ∈ complex . cmulset (b, x) = c} ∈
lessrrel ∩ real × real ∪ {⟨complex⟩, complex} ∪ real × {complex} ∪ {complex} × real
by (rule MMIsar0.MMI_lt2mul2divt)
with A1 A2 show thesis by simp
qed

A real number is smaller than its half iff it is positive.

lemma (in complex0) halfpos: assumes A1: a ∈ R
shows 0 < a −→ a/2 < 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))

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
\[ 0 \leq a \leq b \]
shows \((a/(b + 1)) \cdot b \leq a\)

**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

\((a \in \text{real} \land (a, zero) \notin \text{lessrrel}) \cap \text{real} \times \text{real} \cup \{\{\text{complex}\}, \text{complex}\} \cup \text{real} \times \{\text{complex}\} \cup\)

\((\text{complex})\) \times \text{real} \rightarrow \)

\((a, \text{cmulset}(\{x \in \text{complex} . \text{cmulset}(\text{caddset}(b, one), x) = a\}, b)) \notin \text{lessrrel} \cap \text{real} \times \text{real} \cup \{\{\text{complex}\}, \text{complex}\} \cup \text{real} \times \{\text{complex}\} \cup \{\text{complex}\} \times \text{real} \rightarrow \)

\(\text{by (rule MMIsar0.MMI_ledivp1t)}\)

with \(A1\) \(A2\) show thesis by simp

qed

### 87.2 Natural real numbers

In standard mathematics natural numbers are treated as a subset of real numbers. From the set theory point of view however those 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

\(n \leq k \leftrightarrow n < k + 1\)

\(n < k \leftrightarrow n + 1 \leq k\)

**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 = StrictVersion(CplxROrder(R,A,r))
then have I: MMIsar0
  (real, complex, one, zero, iunit, caddset, cmulset, lessrrel)
  using MMIsar_valid by simp
then have
  n ∈ ∩{N ∈ Pow(real) . one ∈ N ∧
  (∀n ∈ N → caddset (n, one) ∈ N)} ∧
  k ∈ ∩{N ∈ Pow(real) . one ∈ N ∧
  (∀n ∈ N → caddset (n, one) ∈ N)} →
  (k, n) /∈ lessrrel ∩ real × real ∪ {⟨{complex}, complex⟩} ∪ real × {complex}
  ∪ {{complex}} × real /
  ⟨n, caddset (k, one)⟩ ∈
  lessrrel ∩ real × real ∪ {⟨{complex}, complex⟩} ∪ real × {complex}
  ∪ {{complex}} × real by (rule MMIsar0.MMI_nnltp1let)
then have n ∈ N ∧ k ∈ N → n ≤ k ↔ n < k + 1
  by simp
with A1 show n ≤ k ↔ n < k + 1 by simp
from I have
  n ∈ ∩{N ∈ Pow(real) . one ∈ N ∧
  (∀n ∈ N → caddset (n, one) ∈ N)} ∧
  k ∈ ∩{N ∈ Pow(real) . one ∈ N ∧
  (∀n ∈ N → caddset (n, one) ∈ N)} →
  (k, n) ∈ lessrrel ∩ real × real ∪
  {{complex}} × real /
  ⟨n, caddset (k, one)⟩ ∈ lessrrel ∩ real × real ∪ {⟨{complex}, complex⟩} ∪
  {{complex}} × real /
  ⟨k, caddset (n, one)⟩ /∈ lessrrel ∩ real × real ∪ {⟨{complex}, complex⟩} ∪ real × {complex}
  ∪ {{complex}} × real by (rule MMIsar0.MMI_nnltp1let)
then have n ∈ N ∧ k ∈ N → n < k ↔ n + 1 ≤ k
  by simp
with A1 show n < k ↔ n + 1 ≤ 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: ψ(1) and
  A2: ∀k ∈ N. ψ(k) → ψ(k + 1) and
  A3: n ∈ N
  shows ψ(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
    (real, complex, one, zero, iunit, caddset, cmulset, lessrrel)
    using MMIsar_valid by simp
moreover from A1 A2 A3 have
ψ(one)
∀k∈\{N ∈ Pow(real) . one ∈ N ∧
(∀n. n ∈ N → caddset (n, one) ∈ N}).
ψ(k) → ψ(caddset ⟨k, one⟩)
n ∈ \{N ∈ Pow(real) . one ∈ N ∧
(∀n. n ∈ N → caddset (n, one) ∈ N)}
by auto
ultimately show ψ(n) by (rule MMIsar0.nnind1)
qed

Some simple arithmetics.

lemma (in complex0) arith: shows
2 + 2 = 4
2·2 = 4
3·2 = 6
3·3 = 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 (caddset (one, one), one), one)
by (rule MMIsar0.MMI_2p2e4)
thus 2 + 2 = 4 by simp
from 1 have
cmulset(caddset(one, one), caddset(one, one)) =
cmulset(caddset(caddset(one, one), one), one)
by (rule MMIsar0.MMI_2t2e4)
thus 2·2 = 4 by simp
from 1 have
cmulset(caddset(caddset(one, one), one), caddset(one, one)) =
caddset (caddset(caddset(caddset(caddset
(one, one), one), one), one), one)
by (rule MMIIsar0.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, one, one)
by (rule MMIIsar0.MMI_3t3e9)
thus 3 * 3 = 9 by simp

qed

87.3 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: \( S \subseteq \mathbb{R} \) and
A2: \( \exists x \in S. \forall y \in S. x \leq y \)
shows \( \infim(S, \mathbb{R}, <) \in S \)
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))
have I: MMIIsar0
(real, complex, one, zero, iunit, caddset, cmulset, lessrrel)
using MMIIsar_valid by simp
then have \( S \subseteq \mathbb{R} \land (\exists x \in S. \forall y \in S. (y, x) \not\in \)
lessrrel \( \cap \mathbb{R} \times \mathbb{R} \cup \{\{\text{complex}\}, \text{complex}\} \cup \)
real \( \times \{\text{complex}\} \cup \{\text{complex}\} \times \mathbb{R} \)
\( \rightarrow \)
Sup(S, real, converse(lessrrel \( \cap \mathbb{R} \times \mathbb{R} \cup \)
\{\{\text{complex}\}, \text{complex}\} \cup real \times \{\text{complex}\} \cup \)
\{\{\text{complex}\} \times \mathbb{R} \}) \in S
by (rule MMIIsar0.MMI_lbinfmcl)
then have \( S \subseteq \mathbb{R} \land (\exists x \in S. \forall y \in S. x \leq y) \rightarrow \)
Sup(S, \( \mathbb{R} \), converse(\( < \))) \in 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 $A \subseteq \mathbb{R}$ and $A \neq \emptyset$ and $\exists x \in \mathbb{R} \cdot \forall y \in A. \ y \leq x$
- Shows $\sup(A, \mathbb{R}, <) \in \mathbb{R}$

**Proof:**
1. Let $real = \mathbb{R}$
2. Let $complex = C$
3. Let $one = 1$
4. Let $zero = 0$
5. Let $iunit = i$
6. Let $caddset = \text{CplxAdd}(R, A)$
7. Let $cmulset = \text{CplxMul}(R, A, M)$
8. Let $lessrrel = \text{StrictVersion}(\text{CplxROrder}(R, A, r))$
9. Have $\text{MMIsar0}(\text{real, complex, one, zero, iunit, caddset, cmulset, lessrrel})$
10. Using $\text{MMIsar_valid}$ by simp
11. Then have $A \subseteq real \land A \neq \emptyset \land (\exists x \in \mathbb{R} \cdot \forall y \in A. \ x \leq y) \land lessrrel \cap real \times real \cup \{\{\text{complex}\}, \text{complex}\} \cup real \times \{\text{complex}\} \cup \{\{\text{complex}\} \times real\} \rightarrow Sup(A, real, lessrrel \cap real \times real \cup \{\{\text{complex}\}, \text{complex}\} \cup real \times \{\text{complex}\} \cup \{\{\text{complex}\} \times real\} \in real$
12. By (rule $\text{MMIsar0.MMI_suprcl}$)
13. With assms 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 $A \subseteq \mathbb{R}$ and $A \neq \emptyset$ and $\exists x \in \mathbb{R} \cdot \forall y \in A. \ y \leq x$
- And $B \in \mathbb{R}$ and $B < \sup(A, \mathbb{R}, <)$
- Shows $\exists z \in A \cdot B < z$

**Proof:**
1. Let $real = \mathbb{R}$
2. Let $complex = C$
3. Let $one = 1$
4. Let $zero = 0$
5. Let $iunit = i$
6. Let $caddset = \text{CplxAdd}(R, A)$
7. Let $cmulset = \text{CplxMul}(R, A, M)$
8. Let $lessrrel = \text{StrictVersion}(\text{CplxROrder}(R, A, r))$
9. Have $\text{MMIsar0}(\text{real, complex, one, zero, iunit, caddset, cmulset, lessrrel})$
10. Using $\text{MMIsar_valid}$ by simp
11. Then have $(A \subseteq real \land A \neq \emptyset \land (\exists x \in \mathbb{R} \cdot \forall y \in A. \ x \leq y) \land lessrrel \cap real \times real \cup \{\{\text{complex}\}, \text{complex}\} \cup real \times \{\text{complex}\} \cup \{\{\text{complex}\} \times real\}) \land B \in real \land (B, sup(A, real,
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) infmsup:
  assumes A ⊆ R and A ≠ 0 and ∃x∈R. ∀y∈A. x ≤ y
  shows
  Infim(A,R,<) ∈ R
  Infim(A,R,<) = ( -Sup({z ∈ R. (-z) ∈ A },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 I: 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)) =
  ∪{x ∈ complex . caddset ⟨Sup({z ∈ real . ∪{x ∈ complex . caddset(z, x) = zero} ∈ A}, real, lessrrel ∩ real × real ∪ {{complex}, complex}) ∪ real × {complex} ∪ {{complex}} × real), x⟩ = zero}
  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 \texttt{Infim\_def} by \texttt{simp}

from I have
\begin{align*}
A \subseteq \text{real} & \land A \neq 0 \land (\exists x \in \text{real}. \ \forall y \in A. \ \langle y, x \rangle \not\in \\
\text{lessrrel} \cap \text{real} \times \text{real} & \cup \\{\{\text{complex}\}, \text{complex}\} \cup \\
\text{real} \times \{\text{complex}\} & \cup \\
\{\{\text{complex}\}\} \times \text{real} & \rightarrow \text{Sup}(A, \text{real}, \text{converse} \\
\{\text{lessrrel} \cap \text{real} \times \text{real} \cup \{\{\text{complex}\}, \text{complex}\} \cup \\
\text{real} \times \{\text{complex}\} & \cup \{\{\text{complex}\}\} \times \text{real}) \in \text{real} \\
\text{by (rule MMIsar0.MMInfmrc1)} \nonumber
\end{align*}

with \texttt{assms show Infim(A,R,<)} \in \mathbb{R}

using \texttt{Infim\_def} by \texttt{simp}

qed

end

References


