IsarMathLib

Slawomir Kolodynski

November 29, 2006

Abstract

This is the proof document of the IsarMathLib project version 1.3.0. IsarMathLib is a library of formalized mathematics for Isabelle 2005 (ZF logic).

Contents

1 Fol1.thy

theory `Fol1` **imports** `Trancl`

begin

1.1 Mission statement

Until we come up with something better let's just say that writing formalized proofs protects from Alzheimer's disease better than solving crossword puzzles.

1.2 Release notes

This release continues the process of importing Metamath's [4] set.mm database into IsarMathLib, adding about 440 facts and 200 translated proofs. We also add a construction of a model of complex numbers from a complete ordered field.

1.3 Overview of the project

The theory files `Fo11`, `ZF1`, `Nat_ZF`, `func1`, `func_ZF`, `EquivClass1`, `Finite1`, `Finite_ZF`, `Order_ZF` contain some background material that is needed for the remaining theories.

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

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

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

`Int_ZF` 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`.

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

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` `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 showing that real numbers constructed this way form a complete ordered field.

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.

The 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.thy and MMI_Complex_1 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 known_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 looks like.

1.4 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 results up Isabelle in an infinite loop). We reluctantly use (unfold trans_def) after the proof keyword to workaround this.

lemma Fol1_L2: **assumes**
A1: $\forall$ x y z. $\langle$x, y$\rangle \in$ r $\wedge$ $\langle$y, z$\rangle \in$ r $\longrightarrow$ $\langle$x, z$\rangle \in$ r
shows trans(r)
⟨*proof*⟩

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

lemma Fol1_L3: **assumes** A1: trans(r) **and** A2: <a,b> $\in$ r $\wedge$ <b,c> $\in$ r
shows <a,c> $\in$ r

⟨*proof*⟩

There is a problem with application of the definition of asymetry 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*⟩

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

constdefs
  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*⟩

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*⟩

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*⟩

The next lemma demonstrates an elegant form of the Exactly_1_of_3_holds (p,q,r) predicate. More on that at www.solcon.nl/mklooster/calc/calc-tri.html .

lemma Fol1_L8:
  shows Exactly_1_of_3_holds (p,q,r) ⟷ (p⟷q⟷r) ∧ ¬(p∧q∧r)
⟨*proof*⟩

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*⟩

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.

constdefs
  `Xor (infixl Xor 66)`
  `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)`
  ⟨*proof*⟩

Equivalence relations are symmetric.

lemma `equiv_is_sym: assumes A1: equiv(X,r)` **and** `A2: ⟨x,y⟩ ∈ r`
  shows `⟨y,x⟩ ∈ r`
⟨*proof*⟩

This lemma is needed to be used as a rule in some very complicated cases.

lemma `five_more_conj: assumes Axs Ax1 Ax2 Ax3 Ax4 Ax5`
  shows `Ax1 ∧ Ax2 ∧ Ax3 ∧ Ax4 ∧ Ax5 ∧ Axs` ⟨*proof*⟩

end

2 ZF1.thy

theory ZF1 **imports** pair

begin

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

lemma ZF1_1_L1: **assumes** C≠0 **and** ∀y∈C. b(y) = A
  shows (⋃y∈C. b(y)) = A ⟨*proof*⟩

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

lemma ZF1_1_L2: **assumes** A1:C≠0 **and** A2: ∀x∈C. b(x) ∈ A
  and A3: ∀x y. x∈C ∧ y∈C ⟶ b(x) = b(y)
  shows (⋃x∈C. b(x))∈A
⟨*proof*⟩

A purely technical lemma that shows what it means that something belongs to a subset of cartesian product defined by separation. Seems there is no way to avoid that ugly lambda notation.

lemma ZF1_1_L3: **assumes** A1: x∈X y∈Y **and** A2: z = a(x,y)
  shows z ∈ {a(x,y).⟨x,y⟩ ∈ X×Y}
⟨*proof*⟩

If two meta-functions are the same on a cartesian product, then the subsets defined by them are the same. I am surprised blast can not handle this.

lemma ZF1_1_L4: **assumes** A1: ∀x∈X.∀y∈Y. a(x,y) = b(x,y)
  shows {a(x,y). ⟨x,y⟩ ∈ X×Y} = {b(x,y). ⟨x,y⟩ ∈ X×Y}
⟨*proof*⟩

If two meta-functions are the same on a cartesian product, then the subsets defined by them are the same. I am surprised blast can not handle this. This is similar to ZF1_1_L4, except that the set definition varies over p∈X×Y rather than <x,y>∈X×Y.

lemma ZF1_1_L4A: **assumes** A1: ∀x∈X.∀y∈Y. a(<x,y>) = b(x,y)
  shows {a(p). p ∈ X×Y} = {b(x,y). ⟨x,y⟩ ∈ X×Y}
⟨*proof*⟩

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

lemma ZF1_1_L4B: **assumes** ∀x∈X. a(x) = b(x)
  shows {a(x). x∈X} = {b(x). x∈X}

⟨*proof*⟩

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

lemma ZF1_1_L5: **assumes** X≠0 **and** ∀x∈X. b(x) = c
shows {b(x). x∈X} = {c} ⟨*proof*⟩

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∈X. b(x)}
shows Y ⊆ X
⟨*proof*⟩

We can choose an element from a nonempty set.

lemma nonempty_has_element: **assumes** X≠0 **shows** ∃x. x∈X
⟨*proof*⟩

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

constdefs
ProductCollection(T,S) ≡ ⋃U∈T.{U×V. V∈S}

The untion of the product collection of collections S, T^* is the cartesian product of $\bigcup S$ and $\bigcup T$.

lemma ZF1_1_L6: **shows** ⋃ ProductCollection(S,T) = ⋃S × ⋃T
⟨*proof*⟩

An intersection of subsets is a subset.

lemma ZF1_1_L7: **assumes** A1: I≠0 **and** A2: ∀i∈I. P(i) ⊆ X
shows (⋂i∈I. P(i)) ⊆ X
⟨*proof*⟩

end

3 Nat_ZF.thy

theory Nat_ZF **imports** Nat

begin

This theory contains lemmas that are missing from the standard Isabelle's Nat.thy file.

3.1 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 $\Longrightarrow$ 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 induction step for the first order induction.

lemma Nat_ZF_1_L1: **assumes** x∈nat P(x)
  and ∀k∈nat. P(k)⟶P(succ(k))
  shows P(succ(x)) ⟨*proof*⟩

The actual first order induction on natural numbers.

lemma Nat_ZF_1_L2:
  assumes A1: n∈nat **and** A2: P(0) **and** A3: ∀k∈nat. P(k)⟶P(succ(k))
  shows P(n)
⟨*proof*⟩

A nonzero natural number has a predecessor.

lemma Nat_ZF_1_L3: **assumes** A1: n∈nat **and** A2: n≠0
  shows ∃k∈nat. n = succ(k)
⟨*proof*⟩

end

4 func1.thy

theory func1 **imports** func Fol1 ZF1

begin

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.

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

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

If a function maps A into another set, then A is the domain of the function.

lemma func1_1_L1: **assumes** f:A→C **shows** domain(f) = A
  ⟨*proof*⟩

A first-order version of Pi_type.

lemma func1_1_L1A: **assumes** A1: f:X→Y **and** A2: ∀x∈X. f(x) ∈ Z
  shows f:X→Z
⟨*proof*⟩

There is a value for each argument.

lemma func1_1_L2: **assumes** A1: f:X→Y x∈X
  shows ∃y∈Y. <x,y> ∈ f
⟨*proof*⟩

Inverse image of any set is contained in the domain.

lemma func1_1_L3: **assumes** A1: f:X→Y **shows** f-(D) ⊆ X
⟨*proof*⟩

The inverse image of the range is the domain.

lemma func1_1_L4: **assumes** f:X→Y **shows** f-(Y) = X
  ⟨*proof*⟩

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

lemma func1_1_L5:
  assumes A1: <x,y> ∈ f **and** A2: f:X→Y
  shows x∈X ∧ y∈Y
⟨*proof*⟩

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

lemma `func1_1_L5A:`
  assumes `A1: f:X→Y x∈X y = f(x)`
  shows `<x,y> ∈ f y ∈ range(f)`
⟨*proof*⟩

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

lemma `func1_1_L5B:`
  assumes `A1:f:X→Y` **shows** `range(f) ⊆ Y`
⟨*proof*⟩

The image of any set is contained in the range.

lemma `func1_1_L6:` **assumes** `A1: f:X→Y`
  shows `f(B) ⊆ range(f)  f(B) ⊆ Y`
⟨*proof*⟩

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

lemma `func1_1_L6A:` **assumes** `A1: f:X→Y` **shows** `f-(A)⊆X`
⟨*proof*⟩

Inverse image of a greater set is greater.

lemma `func1_1_L7:` **assumes** `A⊆B` **and** `function(f)`
  shows `f-(A)⊆ f-(B)` ⟨*proof*⟩

Image of a greater set is greater.

lemma `func1_1_L8:` **assumes** `A1: A⊆B` **shows** `f(A)⊆ f(B)`
  ⟨*proof*⟩

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→Y` **and** `A2: A⊆X`
  shows `A ⊆ f-(f(A))`
⟨*proof*⟩

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

lemma `func1_1_L10:`
  assumes `A1: f ⊆ X×Y` **and** `A2: ∃!y. (y∈Y & <x,y> ∈ f)`
  shows `∃!y. <x,y> ∈ f`
⟨*proof*⟩

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 ⊆ X×Y` **and** `∀x∈X. ∃!y. y∈Y & <x,y> ∈ f`
  shows `f: X→Y` ⟨*proof*⟩

A set defined by a lambda-type expression is a fuction. There is a similar lemma in func.thy, but I had problems with lamda expressions syntax so I could not apply it. This lemma is a workaround this. Besides, lambda expressions are not readable.

lemma func1_1_L11A: **assumes** A1: ∀x∈X. b(x)∈Y
shows {<x,y> ∈ X×Y. b(x) = y} : X→Y
⟨*proof*⟩

The next lemma will replace func1_1_L11A one day.

lemma ZF_fun_from_total: **assumes** A1: ∀x∈X. b(x)∈Y
shows {⟨x,b(x)⟩. x∈X} : X→Y
⟨*proof*⟩

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

lemma func1_1_L11B:
assumes A1: f:X→Y x∈X
and A2: f = {<x,y> ∈ X×Y. b(x) = y}
shows f(x) = b(x)
⟨*proof*⟩

The next lemma will replace func1_1_L11B one day.

lemma ZF_fun_from_tot_val:
assumes A1: f:X→Y x∈X
and A2: f = {⟨x,b(x)⟩. x∈X}
shows f(x) = b(x)
⟨*proof*⟩

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

lemma func1_1_L11C: **assumes** A1: f:X→Y **and** A2: ∀x∈A. b(x)∈B
and A3: X∩A = 0 **and** Dg : g = f ∪ {⟨x,b(x)⟩. x∈A}
shows
g : X∪A → Y∪B
∀x∈X. g(x) = f(x)
∀x∈A. g(x) = b(x)
⟨*proof*⟩

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*⟩

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→Y **and**
  A2: ∀x∈A. b(x)∈B **and**
  A3: X∩A = 0 **and** A4: a∉ X∪A
  and Dg: g = f ∪ {⟨x,b(x)⟩. x∈A} ∪ {⟨a,c⟩}
  shows
  g : X∪A∪{a} → Y∪B∪{c}
  ∀x∈X. g(x) = f(x)
  ∀x∈A. g(x) = b(x)
  g(a) = c
⟨*proof*⟩

The inverse image of an intersection of a nonempty collection of sets is the intersection of the inverse images. This generalizes function_vimage_Int 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*⟩

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
⟨*proof*⟩

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
⟨*proof*⟩

What is the inverse image of a singleton?

lemma func1_1_L14: **assumes** f∈X→Y
  shows f-({y}) = {x∈X. f(x) = y}
  ⟨*proof*⟩

A more familiar definition of inverse image.

lemma func1_1_L15: **assumes** A1: f:X→Y
  shows f-(A) = {x∈X. f(x) ∈ A}
⟨*proof*⟩

A more familiar definition of image.

lemma func_imagedef: **assumes** A1: f:X→Y **and** A2: A⊆X
  shows f(A) = {f(x). x ∈ A}

⟨*proof*⟩

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*⟩

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*⟩

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→Y **and** A⊆X **and** ∀y∈f(A). P(y)
shows ∀x∈A. P(f(x))
⟨*proof*⟩

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

lemma func1_1_L15C: **assumes** A1: f:X→Y **and** A2: g:Y→Z
and A3: A⊆X
shows
g(f(A)) = {g(f(x)). x∈A}
g(f(A)) = (g O f)(A)
⟨*proof*⟩

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

lemma func1_1_L15D: **assumes** f:X→Y x∈A A⊆X
shows f(x) ∈ f(A)
⟨*proof*⟩

What is the image of a set defined by a meta-fuction?

lemma func1_1_L17:
assumes A1: f ∈ X→Y **and** A2: ∀x∈A. b(x) ∈ X
shows f({b(x). x∈A}) = {f(b(x)). x∈A}
⟨*proof*⟩

What are the values of composition of three functions?

lemma func1_1_L18: **assumes** A1: f:A→B g:B→C h:C→D
and A2: x∈A
shows
(h O g O f)(x) ∈ D
(h O g O f)(x) = h(g(f(x)))
⟨*proof*⟩

4.2 Functions restricted to a set

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

lemma func1_2_L1: **assumes** A1: f:X→Y **and** A2: B⊆X
shows restrict(f,B)-(A) = f-(A) ∩ B
⟨*proof*⟩

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→Y **and** A2: g ∈ A→Z
and A3: A⊆X **and** A4: f ∩ A×Z = g
shows ∀x∈A. g(x) = f(x)
⟨*proof*⟩

Here is the actual criterion.

lemma func1_2_L3:
assumes A1: f:X→Y **and** A2: g:A→Z
and A3: A⊆X **and** A4: f ∩ A×Z = g
shows g = restrict(f,A)
⟨*proof*⟩

Which function space a restricted function belongs to?

lemma func1_2_L4:
assumes A1: f:X→Y **and** A2: A⊆X **and** A3: ∀x∈A. f(x) ∈ Z
shows restrict(f,A) : A→Z
⟨*proof*⟩

4.3 Constant functions

We define constant($= c$) functions on a set X in a natural way as ConstantFunction(X, c).

constdefs
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*⟩

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*⟩

4.4 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`.

The domain of a bijection between X and Y is X.

lemma `domain_of_bij:`
  assumes `A1: f ∈ bij(X,Y)` **shows** `domain(f) = X`
⟨*proof*⟩

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 ∈ inj(A,B)` **and** `A2: C⊆A` **and** `A3: y ∈ f(C)`
  shows
  `converse(f)(y) ∈ C`
  `f(converse(f)(y)) = y`
⟨*proof*⟩

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 ∈ inj(A,B)` **and** `A2: C⊆A` **and**
  `A3: x∈A` **and** `A4: f(x) ∈ f(C)`
  shows `x ∈ C`
⟨*proof*⟩

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

lemma `inj_image_of_Inter:` **assumes** `A1: f ∈ inj(A,B)` **and**
  `A2: I≠0` **and** `A3: ∀i∈I. P(i) ⊆ A`
  shows `f(⋂i∈I. P(i)) = ( ⋂i∈I. f(P(i)) )`
⟨*proof*⟩

This concludes func1.thy.

end

5 Order_ZF.thy

theory `Order_ZF` **imports** `Fol1`

begin

This theory file considers various notion 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 that finite sets are bounded in Finite_ZF.thy.

5.1 Definitions

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

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. The sets that are bounded below and above are also defined, as are bounded sets. Empty sets are defined as bounded. The notation for the definition of an interval may be mysterious for some readers, see `Order_ZF_2_L1` for more intuitive notation. We aslo define the maximum (the greater of) two elemnts and the minmum (the smaller of) two elements. We say that a set has a maximum (minimum) if it has an element that is not smaller (not greater, resp.) that any other one. We show that under some conditions this element of the set is unique (if exists). The element with this property is called the maximum (minimum) of the set. The supremum of a set A is defined as the minimum of the set of upper bounds, i.e. the set $\{u.\forall_{a\in A}\langle a,u\rangle \in r\} = \bigcap_{a\in A} r\{a\}$. Infimum is defined analogously. Recall that `r-(A)`$=\{x : \langle x,y\rangle \in r$ for some $y \in A$ is the inverse image of the set A by relation r. We define a (order) relation to be complete if every nonempty bounded above set has a supremum. This terminolgy may conflict with the one for complete metric space. We will worry about that when we actually define a complete metric space.

constdefs

```
IsTotal (infixl {is total on} 65)
r {is total on} X ≡ (∀a∈X.∀b∈X. <a,b> ∈ r ∨ <b,a> ∈ r)

IsLinOrder(X,r) ≡ ( antisym(r) ∧ trans(r) ∧ (r {is total on} X))

IsPartOrder(X,r) ≡ (refl(X,r) ∧ antisym(r) ∧ trans(r))

IsBoundedAbove(A,r) ≡ ( A=0 ∨ (∃u. ∀x∈A. <x,u> ∈ r))
```

IsBoundedBelow(A,r) ≡ (A=0 ∨ (∃l. ∀x∈A. <l,x> ∈ r))

IsBounded(A,r) ≡ (IsBoundedAbove(A,r) ∧ IsBoundedBelow(A,r))

Interval(r,a,b) ≡ r{a} ∩ r-{b}

GreaterOf(r,a,b) ≡ (if <a,b> ∈ r then b else a)

SmallerOf(r,a,b) ≡ (if <a,b> ∈ r then a else b)

HasAmaximum(r,A) ≡ ∃M∈A.∀x∈A. <x,M> ∈ r

HasAminimum(r,A) ≡ ∃m∈A.∀x∈A. <m,x> ∈ r

Maximum(r,A) ≡ THE M. M∈A ∧ (∀x∈A. <x,M> ∈ r)

Minimum(r,A) ≡ THE m. m∈A ∧ (∀x∈A. <m,x> ∈ r)

Supremum(r,A) ≡ Minimum(r,⋂a∈A. r{a})

Infimum(r,A) ≡ Maximum(r,⋂a∈A. r-{a})

IsComplete (_ {is complete})
r {is complete} ≡
∀A. IsBoundedAbove(A,r) ∧ A≠0 ⟶ HasAminimum(r,⋂a∈A. r{a})

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

lemma Order_ZF_1_L1: **assumes** r {is total on} X **and** a∈X
shows <a,a> ∈ r ⟨*proof*⟩

A total relation is reflexive.

lemma total_is_refl:
assumes r {is total on} X
shows refl(X,r) ⟨*proof*⟩

A linear order is partial order.

lemma Order_ZF_1_L2: **assumes** IsLinOrder(X,r)
shows IsPartOrder(X,r)
⟨*proof*⟩

Partial order that is total is linear.

lemma Order_ZF_1_L3:
assumes IsPartOrder(X,r) **and** r {is total on} X
shows IsLinOrder(X,r)
⟨*proof*⟩

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

lemma Order_ZF_1_L4: **assumes** r {is total on} X **and** A⊆X
shows r {is total on} A
⟨*proof*⟩

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

lemma Order_ZF_1_L5:
assumes r {is total on} X **and** A⊆X **and** a∈X
shows A = {x∈A. ⟨x,a⟩ ∈ r} ∪ {x∈A. ⟨a,x⟩ ∈ r}
⟨*proof*⟩

5.2 Intervals

In this section we discuss intervals.

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

lemma Order_ZF_2_L1:
shows x ∈ Interval(r,a,b) ⟷ <a,x> ∈ r ∧ <x,b> ∈ r
⟨*proof*⟩

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

lemma Order_ZF_2_L1A: **assumes** x ∈ Interval(r,a,b)
shows <a,x> ∈ r <x,b> ∈ r
⟨*proof*⟩

Order_ZF_2_L1, implication from right to left.

lemma Order_ZF_2_L1B: **assumes** <a,x> ∈ r <x,b> ∈ r
shows x ∈ Interval(r,a,b)
⟨*proof*⟩

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

lemma Order_ZF_2_L2: **assumes** refl(X,r)
and a∈X b∈X **and** <a,b> ∈ r
shows
a ∈ Interval(r,a,b)
b ∈ Interval(r,a,b)
⟨*proof*⟩

Under the assumptions of Order_ZF_2_L2, the interval is nonempty.

lemma Order_ZF_2_L2A: **assumes** refl(X,r)
and a∈X b∈X **and** <a,b> ∈ r
shows Interval(r,a,b) ≠ 0
⟨*proof*⟩

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

lemma Order_ZF_2_L3:
  assumes A1: trans(r) **and** A2:<a,b>∈r <b,c>∈r <c,d>∈r
shows Interval(r,b,c) ⊆ Interval(r,a,d)
⟨*proof*⟩

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

lemma Order_ZF_2_L4:
  assumes A1: refl(X,r) **and** A2: antisym(r) **and** A3: a∈X
  shows Interval(r,a,a) = {a}
⟨*proof*⟩

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

lemma Order_ZF_2_L5: **assumes** A1: trans(r) **and** A2: <a,b> ∉ r
  shows Interval(r,a,b) = 0
⟨*proof*⟩

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

lemma Order_ZF_2_L6: **assumes** A1: r ⊆ X×X
  shows Interval(r,a,b) ⊆ X
  ⟨*proof*⟩

5.3 Bounded sets

In this section we consider properties of bounded sets.

For reflexive relations singletons are bounded.

lemma Order_ZF_3_L1: **assumes** refl(X,r) **and** a∈X
  shows IsBounded({a},r)
  ⟨*proof*⟩

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

lemma Order_ZF_3_L1A: **assumes** r ⊆ X×X
  and IsBoundedAbove(A,r)
  shows A⊆X ⟨*proof*⟩

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 ⟨*proof*⟩

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
  ⟨*proof*⟩

If A is bounded above by u, B is bounded above by w, then $A \cup B$ 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*⟩

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

lemma Order_ZF_3_L3:
  assumes A1: r {is total on} X **and** A2: trans(r)
  and A3: IsBoundedAbove(A,r) IsBoundedAbove(B,r)
  and A4: r ⊆ X×X
  shows IsBoundedAbove(A∪B,r)
⟨*proof*⟩

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

lemma Order_ZF_3_L4:
  assumes A1: r {is total on} X **and** A2: trans(r)
  and A3: IsBoundedAbove(A,r) **and** A4: a∈X **and** A5: r ⊆ X×X
  shows IsBoundedAbove(A∪{a},r)
⟨*proof*⟩

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.

lemma Order_ZF_3_L5B:
  assumes A1: r {is total on} X **and** A2: trans(r)
  and A3: l∈X m∈X
  and A4: ∀x∈A. <l,x> ∈ r ∀x∈B. <m,x> ∈ r
  shows ∀x∈A∪B. ⟨SmallerOf(r,l,m),x⟩ ∈ r
⟨*proof*⟩

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*⟩

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

lemma Order_ZF_3_L7:
assumes A1: r {is total on} X **and** A2: trans(r)
and A3: IsBoundedBelow(A,r) **and** A4: a∈X **and** A5: r ⊆ X×X
shows IsBoundedBelow(A∪{a},r)
⟨*proof*⟩

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)
⟨*proof*⟩

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

lemma Order_ZF_3_L8:
assumes r {is total on} X **and** trans(r)
and IsBounded(A,r) **and** a∈X **and** r ⊆ X×X
shows IsBounded(A∪{a},r)
⟨*proof*⟩

A sufficient condition for a set to be bounded below.

lemma Order_ZF_3_L9: **assumes** A1: ∀a∈A. ⟨l,a⟩ ∈ r
shows IsBoundedBelow(A,r)
⟨*proof*⟩

A sufficient condition for a set to be bounded above.

lemma Order_ZF_3_L10: **assumes** A1: ∀a∈A. ⟨a,u⟩ ∈ r
shows IsBoundedAbove(A,r)
⟨*proof*⟩

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*⟩

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

lemma Order_ZF_3_L12: **assumes** IsBoundedBelow(A,r) **and** B⊆A
shows IsBoundedBelow(B,r)
⟨*proof*⟩

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

lemma Order_ZF_3_L13: **assumes** IsBoundedAbove(A,r) **and** B⊆A
shows IsBoundedAbove(B,r)
⟨*proof*⟩

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.

lemma Order_ZF_3_L14:
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 ¬IsBoundedAbove(A,r)
⟨*proof*⟩

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

lemma Order_ZF_3_L15: **shows** IsBoundedAbove({x∈A. ⟨x,a⟩ ∈ r},r)
⟨*proof*⟩

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

lemma Order_ZF_3_L16: **assumes** A1: IsBoundedBelow(A,r)
shows IsBounded({x∈A. ⟨x,a⟩ ∈ r},r)
⟨*proof*⟩

5.4 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.thy) that every finite set has well-defined maximum and minimum.

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

lemma Order_ZF_4_L1: **assumes** A1: antisym(r) **and** A2: HasAmaximum(r,A)
shows ∃!M. M∈A ∧ (∀x∈A. <x,M> ∈ r)
⟨*proof*⟩

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 ∃!m. m∈A ∧ (∀x∈A. <m,x> ∈ r)

⟨*proof*⟩

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) ∈ A ∀x∈A. ⟨x,Maximum(r,A)⟩ ∈ r
⟨*proof*⟩

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) ∈ A ∀x∈A. ⟨Minimum(r,A),x⟩ ∈ r
⟨*proof*⟩

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*⟩

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*⟩

Set that has a maximum is bounded above.

lemma Order_ZF_4_L7:
  assumes HasAmaximum(r,A)
  shows IsBoundedAbove(A,r)
  ⟨*proof*⟩

Set that has a minimum is bounded below.

lemma Order_ZF_4_L8A:
  assumes HasAminimum(r,A)
  shows IsBoundedBelow(A,r)
  ⟨*proof*⟩

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})
  ⟨*proof*⟩

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*⟩

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*⟩

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 ⊆ X×X **and**
  A4: ∀A. IsBounded(A,r) ∧ A≠0 ⟶ HasAminimum(r,A) **and**
  A5: B≠0 **and** A6: IsBoundedBelow(B,r)
  shows HasAminimum(r,B)
⟨*proof*⟩

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 A1: r {is total on} X **and**
  A2: trans(r) **and**
  A3: r ⊆ X×X **and**
  A4: ∀A. IsBounded(A,r) ∧ A≠0 ⟶ HasAmaximum(r,A) **and**
  A5: B≠0 **and** A6: IsBoundedAbove(B,r)
  shows HasAmaximum(r,B)
⟨*proof*⟩

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** ∀a∈A. ⟨L,a⟩ ∈ r
  shows ⟨L,Minimum(r,A)⟩ ∈ r
  ⟨*proof*⟩

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** ∀a∈A. ⟨a,M⟩ ∈ r
  shows ⟨Maximum(r,A),M⟩ ∈ r
  ⟨*proof*⟩

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

lemma Order_ZF_4_L14:
  assumes A1: antisym(r) **and** A2: M ∈ A **and**
  A3: ∀a∈A. ⟨a,M⟩ ∈ r
  shows Maximum(r,A) = M
⟨*proof*⟩

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

lemma Order_ZF_4_L15:
  assumes A1: antisym(r) **and** A2: m ∈ A **and**
  A3: ∀a∈A. ⟨m,a⟩ ∈ r
  shows Minimum(r,A) = m
⟨*proof*⟩

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⊆X **and**
  A4: ¬HasAmaximum(r,A) **and**
  A5: x∈A
  shows ∃y∈A. ⟨x,y⟩ ∈ r ∧ y≠x
⟨*proof*⟩

5.5 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 ∈ (⋂a∈A. r{a}) **and** a∈A
  shows ⟨a,u⟩ ∈ r
  ⟨*proof*⟩

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

lemma Order_ZF_5_L2: **assumes** l ∈ (⋂a∈A. r-{a}) **and** a∈A
  shows ⟨l,a⟩ ∈ r
  ⟨*proof*⟩

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.

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

Infimum is greater or equal than any lower bound.

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

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

lemma Order_ZF_5_L5: **assumes** A1: antisym(r) **and** A2: A≠0 **and**
A3: ∀x∈A. ⟨x,z⟩ ∈ r **and**
A4: ∀y. (∀x∈A. ⟨x,y⟩ ∈ r) ⟶ ⟨z,y⟩ ∈ r
shows
HasAminimum(r,⋂a∈A. r{a})
z = Supremum(r,A)
⟨*proof*⟩

If a set has a maximum, then the maximum is the supremum.

lemma Order_ZF_5_L6:
assumes A1: antisym(r) **and** A2: A≠0 **and**
A3: HasAmaximum(r,A)
shows
HasAminimum(r,⋂a∈A. r{a})
Maximum(r,A) = Supremum(r,A)
⟨*proof*⟩

Properties of supremum of a set for complete relations.

lemma Order_ZF_5_L7:
assumes A1: r ⊆ X×X **and** A2: antisym(r) **and**
A3: r {is complete} **and**
A4: A⊆X A≠0 **and** A5: ∃x∈X. ∀y∈A. ⟨y,x⟩ ∈ r
shows
Supremum(r,A) ∈ X
∀x∈A. ⟨x,Supremum(r,A)⟩ ∈ r
⟨*proof*⟩

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 ⊆ X×X **and** A2: IsLinOrder(X,r) **and**
  A3: r {is complete} **and**
  A4: A⊆X A≠0 **and** A5: ∃x∈X. ∀y∈A. ⟨y,x⟩ ∈ r **and**
  A6: ⟨y,Supremum(r,A)⟩ ∈ r y ≠ Supremum(r,A)
  shows ∃z∈A. ⟨y,z⟩ ∈ r ∧ y ≠ z
⟨*proof*⟩

5.6 Strict versions of order relations

One of the problems with translating formalized mathematics from Metamath 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.

constdefs
  StrictVersion(r) ≡ r - {⟨x,x⟩. x ∈ domain(r)}

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

lemma def_of_strict_ver: **shows**
  ⟨x,y⟩ ∈ StrictVersion(r) ⟷ ⟨x,y⟩ ∈ r ∧ x≠y
  ⟨*proof*⟩

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

lemma strict_of_antisym:
  assumes A1: antisym(r) **and** A2: ⟨a,b⟩ ∈ StrictVersion(r)
  shows ⟨b,a⟩ ∉ StrictVersion(r)
⟨*proof*⟩

The strict version of totality.

lemma strict_of_tot:
  assumes r {is total on} X **and** a∈X b∈X a≠b
  shows ⟨a,b⟩ ∈ StrictVersion(r) ∨ ⟨b,a⟩ ∈ StrictVersion(r)
  ⟨*proof*⟩

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∈X b∈X
  and A4: s = StrictVersion(r)
  shows Exactly_1_of_3_holds(⟨a,b⟩ ∈ s, a=b,⟨b,a⟩ ∈ s)

⟨*proof*⟩

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(⟨a,b⟩ ∈ s, a=b,⟨b,a⟩ ∈ s)`
⟨*proof*⟩

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: ⟨a,b⟩ ∈ r`
shows `⟨b,a⟩ ∉ StrictVersion(r)`
⟨*proof*⟩

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: ⟨a,b⟩ ∈ s  ⟨b,c⟩ ∈ s`
shows `⟨a,c⟩ ∈ s`
⟨*proof*⟩

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

lemma `strict_of_transB`:
assumes `A1: trans(r)` **and** `A2: antisym(r)`
shows `trans(StrictVersion(r))`
⟨*proof*⟩

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

lemma `strict_of_compl`:
assumes `A1: r ⊆ X×X` **and** `A2: IsLinOrder(X,r)` **and**
`A3: r {is complete}` **and**
`A4: A⊆X  A≠0` **and** `A5: s = StrictVersion(r)` **and**
`A6: ∃u∈X. ∀y∈A. ⟨y,u⟩ ∈ s`
shows
`∃x∈X. ( ∀y∈A. ⟨x,y⟩ ∉ s ) ∧ (∀y∈X. ⟨y,x⟩ ∈ s ⟶ (∃z∈A. ⟨y,z⟩ ∈ s))`
⟨*proof*⟩

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

lemma `strict_ver_rel`: **assumes** `A1: r ⊆ A×A`
shows `StrictVersion(r) ⊆ A×A`
⟨*proof*⟩

end

6 func_ZF.thy

theory func_ZF **imports** Order func1 Order_ZF

begin

In this theory we consider properties of functions that are binary operations, that is they map $X \times X$ into X. We also consider some properties of functions related to order.

6.1 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 $f, g : X \to \mathbf{R}$ 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.

constdefs
```
  Lift2FcnSpce (infix {lifted to function space over} 65)
  f {lifted to function space over} X ≡
  {<p,g> ∈ ((X→range(f))×(X→range(f)))×(X→range(f)).
  {<x,y> ∈ X×range(f). f<fst(p)(x),snd(p)(x)> = y} = g}
```

The result of the lift belongs to the function space.

lemma func_ZF_1_L1:
assumes A1: f : Y×Y→Y
and A2: p ∈(X→range(f))×(X→range(f))
shows
{<x,y> ∈ X×range(f). f<fst(p)(x),snd(p)(x)> = y} : X→range(f)
⟨*proof*⟩

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

lemma func_ZF_1_L2:
assumes f : Y×Y→Y
and p∈(X→range(f))×(X→range(f)) **and** x∈X
and P = {<x,y> ∈ X×range(f). f<fst(p)(x),snd(p)(x)> = y}
shows P(x) = f⟨fst(p)(x),snd(p)(x)⟩
⟨*proof*⟩

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

lemma func_ZF_1_L3:
assumes f ∈ Y×Y→Y
and F = f {lifted to function space over} X
shows F : (X→range(f))×(X→range(f))→(X→range(f))

⟨*proof*⟩

The values of the lift are defined by the values of the liftee in the natural way. For some reason we need to be extremely detailed and explicit to be able to apply func1_3_L2. simp and auto fail miserably here.

lemma func_ZF_1_L4:
assumes A1: f : Y×Y→Y
and A2: F = f {lifted to function space over} X
and A3: s:X→range(f) r:X→range(f)
and A4: x∈X
shows (F<s,r>)(x) = f<s(x),r(x)>
⟨*proof*⟩

6.2 Associative and commutative operations

In this section we define associative and commutative oparations and prove that they remain such when we lift them to a function space.

constdefs

IsAssociative (**infix** {is associative on} 65)
f {is associative on} G ≡ f ∈ G×G→G ∧
(∀ x ∈ G. ∀ y ∈ G. ∀ z ∈ G.
(f(<f(<x,y>),z>) = f(< x,f(<y,z>)>)))

IsCommutative (**infix** {is commutative on} 65)
f {is commutative on} G ≡ ∀x∈G. ∀y∈G. f<x,y> = f<y,x>

The lift of a commutative function is commutative.

lemma func_ZF_2_L1:
assumes A1: f : G×G→G
and A2: F = f {lifted to function space over} X
and A3: s : X→range(f) r : X→range(f)
and A4: f {is commutative on} G
shows F<s,r> = F<r,s>
⟨*proof*⟩

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))
⟨*proof*⟩

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
shows F⟨F<s,r>,q⟩ = F⟨s,F<r,q>⟩
⟨*proof*⟩

The lift of an associative function is associative on the function space.

lemma func_ZF_2_L4:
assumes A1: f {is associative on} G
and A2: F = f {lifted to function space over} X
shows F {is associative on} (X→range(f))
⟨*proof*⟩

6.3 Restricting operations

In this section we consider when 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×X→Y **and** A2: A⊆X
and A3: f {is commutative on} X
shows restrict(f,A×A) {is commutative on} A
⟨*proof*⟩

Next we define sets closed with respect to an operation.

constdefs
IsOpClosed (**infix** {is closed under} 65)
A {is closed under} f ≡ ∀x∈A. ∀y∈A. f<x,y> ∈ 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⊆X **and** A3: A {is closed under} f
and A4: x∈A y∈A z∈A
and A5: g = restrict(f,A×A)
shows g⟨g<x,y>,z⟩ = g⟨x,g<y,z>⟩
⟨*proof*⟩

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

lemma func_ZF_4_L3: **assumes** A1: f {is associative on} X
and A2: A⊆X **and** A3: A {is closed under} f
shows restrict(f,A×A) {is associative on} A
⟨*proof*⟩

The essential condition to show that if a set A is closed with respect to an operation, then it is closed under this operation restricted to any superset of A.

lemma func_ZF_4_L4: **assumes** A {is closed under} f
and A⊆B **and** x∈A y∈A **and** g = restrict(f,B×B)
shows g<x,y> ∈ A
⟨*proof*⟩

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

lemma func_ZF_4_L5:
assumes A1: A {is closed under} f
and A2: A⊆B
shows A {is closed under} restrict(f,B×B)
⟨*proof*⟩

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 ⟨*proof*⟩

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
⟨*proof*⟩

6.4 Composition

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. In this section we consider the corresponding two-argument ZF-function (binary operation), that is a subset of $((X \to X) \times (X \to X)) \times (X \to X)$.

constdefs
Composition(X) ≡
{<p,f> ∈ ((X→X)×(X→X))×(X→X). fst(p) O snd(p) = f}

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)
⟨*proof*⟩

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

lemma func_ZF_5_L2: **assumes** f:X→X g:X→X

shows Composition(X)<f,g> = f O g
⟨*proof*⟩

What is the falue of a composition on an argument?

lemma func_ZF_5_L3: **assumes** f:X→X **and** g:X→X **and** x∈X
shows (Composition(X)<f,g>)(x) = f(g(x))
⟨*proof*⟩

The essential condition to show that composition is associative.

lemma func_ZF_5_L4: **assumes** A1: f:X→X g:X→X h:X→X
and A2: C = Composition(X)
shows C⟨C<f,g>,h⟩ = C⟨ f,C<g,h>⟩
⟨*proof*⟩

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

lemma func_ZF_5_L5: **shows** Composition(X) {is associative on} (X→X)
⟨*proof*⟩

6.5 Identity function

In this section we show some additional facts about the identity function defined in the standard Isabelle's Perm.thy file.

Composing a function with identity does not change the function.

lemma func_ZF_6_L1A: **assumes** A1: f : X→X
shows Composition(X)<f,id(X)> = f
Composition(X)<id(X),f> = f
⟨*proof*⟩

6.6 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.thy 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.

constdefs
IsDistributive(X,A,M) ≡ (∀a∈X.∀b∈X.∀c∈X.
M⟨a,A<b,c>⟩ = A⟨M<a,b>,M<a,c>⟩ ∧
M⟨A<b,c>,a⟩ = A⟨M<b,a>,M<c,a> ⟩)

The essential condition to show that distributivity is preserved by 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⊆X
  and A3: Y {is closed under} A Y {is closed under} M
  and A4: A_r = restrict(A,Y×Y) M_r = restrict(M,Y×Y)
  and A5: a∈Y b∈Y c∈Y
  shows M_r⟨ a,A_r<b,c> ⟩ = A_r⟨ M_r<a,b>,M_r<a,c> ⟩ ∧
  M_r⟨ A_r<b,c>,a ⟩ = A_r⟨ M_r<b,a>,M_r<c,a> ⟩
⟨*proof*⟩

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*⟩

6.7 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*⟩

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→Y **and** A⊆X **and** ∀x∈A. ⟨f(x),U⟩ ∈ r
  shows IsBoundedAbove(f(A),r)
⟨*proof*⟩

6.8 Projections in cartesian products

In this section we consider maps arising naturally in cartesian products.

There is a natural bijection etween $X = Y \times \{y\}$ (a "slice") and Y. We will call this the SliceProjection(Y×{y}). This is really the ZF equivalent of the meta-function fst(x).

constdefs
```
  SliceProjection(X) ≡ {⟨p,fst(p)⟩. p ∈ X }
```

A slice projection is a bijection between $X \times \{y\}$ 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*⟩

6.9 Induced relations and order isomorphisms

When we have two sets X, Y, function $f : X \to Y$ 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.thy` theory defines the notion of a space of order isomorphisms between two sets relative to a relation. We expand that material proving that order isomrphisms preserve interesting properties of the relation.

We call the relation created by a relation on Y and a mapping $f : X \to Y$ the `InducedRelation(f,R)`.

constdefs
```
  InducedRelation(f,R) ≡
  {p ∈ domain(f)×domain(f). ⟨f(fst(p)),f(snd(p))⟩ ∈ R}
```

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

lemma `def_of_ind_relA:`
```
  assumes ⟨x,y⟩ ∈ InducedRelation(f,R)
  shows ⟨f(x),f(y)⟩ ∈ R
```
⟨*proof*⟩

A reformulation of the definition of the relation induced by a function, kind of converse of `def_of_ind_relA`.

lemma `def_of_ind_relB:` **assumes** `f:A→B` **and**
```
  x∈A  y∈A and ⟨f(x),f(y)⟩ ∈ R
  shows ⟨x,y⟩ ∈ InducedRelation(f,R)
```
⟨*proof*⟩

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

lemma `ord_iso_apply_conv:`
```
  assumes f ∈ ord_iso(A,r,B,R) and
  ⟨f(x),f(y)⟩ ∈ R and x∈A  y∈A
  shows ⟨x,y⟩ ∈ r
```

⟨*proof*⟩

The next lemma tells us where the induced relation is defined

lemma ind_rel_domain:
  assumes R ⊆ B×B **and** f:A→B
  shows InducedRelation(f,R) ⊆ A×A
  ⟨*proof*⟩

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

lemma bij_is_ord_iso: **assumes** A1: f ∈ bij(A,B)
  shows f ∈ ord_iso(A,InducedRelation(f,R),B,R)
⟨*proof*⟩

An order isomoprhism 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*⟩

Order isomoprhisms 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*⟩

Order isomorphisms preserve totality.

lemma ord_iso_pres_tot: **assumes** A1: f ∈ ord_iso(A,r,B,R) **and**
  A2: r ⊆ A×A **and** A3: R {is total on} B
  shows r {is total on} A
⟨*proof*⟩

Order isomorphisms preserve linearity.

lemma ord_iso_pres_lin: **assumes** f ∈ ord_iso(A,r,B,R) **and**
  r ⊆ A×A **and** IsLinOrder(B,R)
  shows IsLinOrder(A,r)
  ⟨*proof*⟩

If a relation is a linear order, then the relation induced on another set is by a bijection is also a linear order.

lemma ind_rel_pres_lin:
  assumes A1: f ∈ bij(A,B) **and** A2: IsLinOrder(B,R)
  shows IsLinOrder(A,InducedRelation(f,R))
⟨*proof*⟩

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 ∈ ord_iso(A,r,B,R) **and** A2: r ⊆ A×A **and**
  A3: IsBoundedAbove(C,r) C≠0
  shows IsBoundedAbove(f(C),R) f(C) ≠ 0
⟨*proof*⟩

Order isomorphisms preserve the property of having a minimum.

lemma ord_iso_pres_has_min:
  assumes A1: f ∈ ord_iso(A,r,B,R) **and** A2: r ⊆ A×A **and**
  A3: C⊆A **and** A4: HasAminimum(R,f(C))
  shows HasAminimum(r,C)
⟨*proof*⟩

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_rel_image:
  assumes A1: f ∈ ord_iso(A,r,B,R) **and**
  A2: r ⊆ A×A R ⊆ B×B **and**
  A3: a∈A
  shows f(r{a}) = R{f(a)}
⟨*proof*⟩

Order isomorphisms preserve collections of upper bounds.

lemma ord_iso_pres_up_bounds:
  assumes A1: f ∈ ord_iso(A,r,B,R) **and**
  A2: r ⊆ A×A R ⊆ B×B **and**
  A3: C⊆A
  shows {f(r{a}). a∈C} = {R{b}. b ∈ f(C)}
⟨*proof*⟩

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 ∈ ord_iso(A,r,B,R) **and** A2: r ⊆ A×A R ⊆ B×B **and**
  A3: C⊆A **and** A4: C≠0
  shows f(⋂a∈C. r{a}) = (⋂b∈f(C). R{b})
⟨*proof*⟩

Order isomorphisms preserve completeness.

lemma ord_iso_pres_compl:
  assumes A1: f ∈ ord_iso(A,r,B,R) **and**
  A2: r ⊆ A×A R ⊆ B×B **and** A3: R {is complete}
  shows r {is complete}
⟨*proof*⟩

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

lemma ind_rel_pres_compl: **assumes** A1: f ∈ bij(A,B)

and A2: R $\subseteq$ B$\times$B **and** A3: R {is complete}
shows InducedRelation(f,R) {is complete}
⟨*proof*⟩

end

7 EquivClass1.thy

theory EquivClass1 **imports** EquivClass func_ZF ZF1

begin

In this theory file we extend the work on equivalence relations done in the standard Isabelle's EquivClass.thy file. The problem that we have with the EquivClass.thy is that the notions congruent and congruent2 are defined for meta-functions rather then ZF - functions (subsets of Cartesian products). This causes inflexibility (that is typical for typed set theories) in making the notions depend on additional parameters For example the congruent2 there takes $[i, [i, i] => i]$ as parameters, that is the second parameter is a meta-function that takes two sets and results in a set. So, when our function depends on additional parameters, (for example the function we want to be congruent depends on a group and we want to show that for all groups the function is congruent) there is no easy way to use that notion. The ZF functions are sets and there is no problem if in actual application this set depends on some parameters.

7.1 Congruent functions and projections on the quotient

First we define the notion of function that maps equivalent elements to equivalent values. We use similar names as in the original EquivClass.thy file to indicate the conceptual correspondence of the notions. Then we define the projection of a function onto the quotient space. We will show that if the function is congruent the projection is a mapping from the quotient space into itself. In standard math the condion that the function is congruent allows to show that the value of the projection does not depend on the choice of elements that represent the equivalence classes. We set up things a little differently to avoid making choices.

constdefs
```
  Congruent(r,f) ≡
  (∀x y. <x,y> ∈ r  ⟶ <f(x),f(y)> ∈ r)

  ProjFun(A,r,f) ≡
  {<c,d> ∈ (A//r)×(A//r). (⋃x∈c. r{f(x)}) = d}
```

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*⟩

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
⟨*proof*⟩

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*⟩

Elements that belong to the same equivalence class are equivalent.

lemma EquivClass_1_L2A:
assumes equiv(A,r) C ∈ A//r x∈C y∈C
shows <x,y> ∈ r
⟨*proof*⟩

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

lemma EquivClass_1_L2B:
assumes A1: equiv(A,r) **and** A2: y∈A **and** A3: x ∈ r{y}
shows <x,y> ∈ r
⟨*proof*⟩

If a function is congruent then the equivalence classes of the values that come from the arguments from the same class are the same.

lemma EquivClass_1_L3:
assumes A1: equiv(A,r) **and** A2: Congruent(r,f)
and A3: C ∈ A//r x∈C y∈C
shows r{f(x)} = r{f(y)}
⟨*proof*⟩

The values of congruent functions are in the space.

lemma EquivClass_1_L4:
assumes A1: equiv(A,r) **and** A2: C ∈ A//r x∈C
and A3: Congruent(r,f)
shows f(x) ∈ A
⟨*proof*⟩

Equivalence classes are not empty.

lemma EquivClass_1_L5:
assumes A1: refl(A,r) **and** A2: C ∈ A//r
shows C≠0
⟨*proof*⟩

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 (⋃x∈C. r{f(x)}) ∈ A//r
⟨*proof*⟩

Congruent functions can be projected.

lemma EquivClass_1_T1:
assumes equiv(A,r) Congruent(r,f)
shows ProjFun(A,r,f) ∈ A//r → A//r
⟨*proof*⟩

We now define congruent functions of two variables. Congruent2 corresponds to congruent2 in EquivClass.thy, but uses ZF-functions rather than meta-functions.

constdefs
Congruent2(r,f) ≡
(∀x1 x2 y1 y2. <x1,x2> ∈ r ∧ <y1,y2> ∈ r ⟶
<f<x1,y1>,f<x2,y2> > ∈ r)

ProjFun2(A,r,f) ≡
{<p,d> ∈ ((A//r)×(A//r))×(A//r) .
(⋃ z ∈ fst(p)×snd(p). r{f(z)}) = d}

The following lemma is a two-variables equivalent of EquivClass_1_L3.

lemma EquivClass_1_L7:
assumes A1: equiv(A,r) **and** A2: Congruent2(r,f)
and A3: C1 ∈ A//r C2 ∈ A//r
and A4: z1 ∈ C1×C2 z2 ∈ C1×C2
shows r{f(z1)} = r{f(z2)}
⟨*proof*⟩

The values of congruent functions of two variables are in the space.

lemma EquivClass_1_L8:
assumes A1: equiv(A,r) **and** A2: C1 ∈ A//r **and** A3: C2 ∈ A//r
and A4: z ∈ C1×C2 **and** A5: Congruent2(r,f)
shows f(z) ∈ A
⟨*proof*⟩

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 we 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> ∈ A
⟨*proof*⟩

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 ∈ (A//r)×(A//r)
  shows (⋃ z ∈ fst(p)×snd(p). r{f(z)}) ∈ A//r
⟨*proof*⟩

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)×(A//r) → A//r
  ⟨*proof*⟩

We define the projection on the quotient space as a function that takes an element of A and assigns its equivalence class in A/r.

constdefs
  Proj(A,r) ≡ {<x,c> ∈ A×(A//r). r{x} = c}

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∈A y∈A
  shows ProjFun2(A,r,f)<r{x},r{y}> = r{f<x,y>}
⟨*proof*⟩

7.2 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: equiv(A,r) **and** A2: Congruent2(r,f)
  and A3: f {is commutative on} A
  and A4: c1 ∈ A//r c2 ∈ A//r
  shows ProjFun2(A,r,f) <c1,c2> = ProjFun2(A,r,f)<c2,c1>
⟨*proof*⟩

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
  ⟨*proof*⟩

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⟨g<c1,c2>,c3⟩ = g⟨c1,g<c2,c3>⟩
⟨*proof*⟩

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*⟩

The essential condition to show that distributivity is preserved by projections to quotient spaces, provided both operations are congruent with respect to the equivalence relation.

lemma EquivClass_2_L3:
  assumes A1: IsDistributive(X,A,M)
  and A2: equiv(X,r)
  and A3: Congruent2(r,A) Congruent2(r,M)
  and A4: a ∈ X//r b ∈ X//r c ∈ X//r
  and A5: Ap = ProjFun2(X,r,A) Mp = ProjFun2(X,r,M)
  shows Mp⟨a,Ap<b,c>⟩ = Ap⟨ Mp<a,b>,Mp<a,c>⟩ ∧
  Mp⟨ Ap<b,c>,a ⟩ = Ap⟨ Mp<b,a>,Mp<c,a>⟩
⟨*proof*⟩

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

lemma EquivClass_2_L4: **assumes** A1: IsDistributive(X,A,M)
  and A2: equiv(X,r)
  and A3: Congruent2(r,A) Congruent2(r,M)
  shows IsDistributive(X//r,ProjFun2(X,r,A),ProjFun2(X,r,M))
⟨*proof*⟩

7.3 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 equivalemce classes. This makes them useful as a tool to define subsets of the quoutient 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 saturated set. Recall that in Isabelle r-(A) is the inverse image of A with respect to relation r. This definition is not specific to equivalence relations.

constdefs
  IsSaturated(r,A) ≡ A = r-(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*⟩

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*⟩

The next lemma shows that if "$\sim$" is an equivalence relation and a set A is such that $a \in A$ and $a \sim b$ implies $b \in A$, then A is saturated with respect to the relation.

lemma EquivClass_3_L3: **assumes** A1: equiv(X,r)
  and A2: r ⊆ X×X **and** A3: A⊆X
  and A4: ∀x∈A. ∀y∈X. ⟨x,y⟩ ∈ r ⟶ y∈A
  shows IsSaturated(r,A)
⟨*proof*⟩

If $A \subseteq X$ and A is saturated and $x \sim y$, then $x \in A$ iff $y \in A$. Here we we show only one direction.

lemma EquivClass_3_L4: **assumes** A1: equiv(X,r)
  and A2: IsSaturated(r,A) **and** A3: A⊆X
  and A4: ⟨x,y⟩ ∈ r
  and A5: x∈X y∈A
  shows x∈A
⟨*proof*⟩

If $A \subseteq X$ and A is saturated and $x \sim y$, then $x \in A$ iff $y \in A$.

lemma EquivClass_3_L5: **assumes** A1: equiv(X,r)
  and A2: IsSaturated(r,A) **and** A3: A⊆X
  and A4: x∈X y∈X
  and A5: ⟨x,y⟩ ∈ r
  shows x∈A ⟷ y∈A
⟨*proof*⟩

If A is saturated then $x \in A$ iff its class is in the projection of A.

lemma EquivClass_3_L6: **assumes** A1: equiv(X,r)
  and A2: IsSaturated(r,A) **and** A3: A⊆X **and** A4: x∈X
  and A5: B = {r{x}. x∈A}
  shows x∈A ⟷ r{x} ∈ B

⟨*proof*⟩

A technical lemma involving a projection of a saturated set and a logical epression with exclusive or.

lemma EquivClass_3_L7: **assumes** A1: equiv(X,r)
  and A2: IsSaturated(r,A) **and** A3: A⊆X
  and A4: x∈X y∈X
  and A5: B = {r{x}. x∈A}
  and A6: (x∈A) Xor (y∈A)
  shows (r{x} ∈ B) Xor (r{y} ∈ B)
  ⟨*proof*⟩

end

8 Finite1.thy

theory Finite1 **imports** Finite func1 ZF1

begin

8.1 Finite powerset

Intersection of a collection is contained in every element of the collection.

lemma ZF11: **assumes** A: A ∈ M **shows** ⋂M ⊆ A
⟨*proof*⟩

Intersection of a nonempty collection M of subsets of X is a subset of X.

lemma ZF12: **assumes** A1: ∀A∈ M. A⊆X **and** A2:M≠0
shows (⋂ M) ⊆ X
⟨*proof*⟩

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.

constdefs
RestrictedTo (**infixl** {restricted to} 70)
M {restricted to} X ≡ {X ∩ A . A ∈ M}

In Topology_ZFTopology_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.

constdefs
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 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*⟩

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*⟩

This is a different form of the above theorem:

theorem ZF1FinRestr:
assumes A1:M∈ Fin(TA) **and** A2: M≠0
and A3: ∀ V∈ TA. ∃ U∈ T. V=U∩A
shows ∃N∈ Fin(T). (∀V∈ M. ∃ U∈ N. (V = U∩A)) ∧ N≠0
⟨*proof*⟩

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 A:∃U V. (U∈T ∧ V∈T ∧ x∈U ∧ y∈V ∧ U∩V=0)
shows ∃U∈T. (x∈U ∧ y∉U)
⟨*proof*⟩

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.

The induction step:

lemma Finite1_L3_IndStep:
assumes A1:∀A B. ((A∈C ∧ B∈C) ⟶ A∪B∈C)
and A2: A∈C **and** A3:N∈Fin(C) **and** A4:A∉N **and** A5:⋃N ∈ C
shows ⋃cons(A,N) ∈ C
⟨*proof*⟩

The lemma:

lemma Finite1_L3:
assumes A1:0 ∈ C **and** A2:∀A B. ((A∈C ∧ B∈C) ⟶ A∪B∈C) **and**
A3:N∈ Fin(C)
shows ⋃N∈C
⟨*proof*⟩

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 sligltly 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 constdef.

constdefs

```
IntPr(T,N) ≡ (N = 0 | ⋂N ∈ T)
```

The induction step.

lemma `Finite1_L4_IndStep:`
assumes `A1:∀A B. ((A∈T ∧ B∈T) ⟶ A∩B∈T)`
and `A2: A∈T` **and** `A3:N∈Fin(T)` **and** `A4:A∉N` **and** `A5:IntPr(T,N)`
shows `IntPr(T,cons(A,N))`
⟨*proof*⟩

The lemma.

lemma `Finite1_L4:`
assumes `A1:∀A B. A∈T ∧ B∈T ⟶ A∩B ∈ T`
and `A2:N∈Fin(T)`
shows `IntPr(T,N)`
⟨*proof*⟩

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*⟩

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 induction step:

lemma `Finite1_L6_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)`
⟨*proof*⟩

The lemma:

lemma `Finite1_L6:` **assumes** `A1:∀V∈B. K(V)∈C` **and** `A2:N∈Fin(B)`
shows `{K(V). V∈N} ∈ Fin(C)`
⟨*proof*⟩

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*⟩

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∈X. a(x) ∈ Y **and** A2: {b(y).y∈Y} ∈ Fin(Z)
  shows {b(a(x)).x∈X} ∈ Fin(Z)
⟨*proof*⟩

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∈Y. b(y) ∈ Z **and** A2: {a(x). x∈X} ∈ Fin(Y)
  shows {b(a(x)).x∈X} ∈ Fin(Z)
⟨*proof*⟩

Next we show an identity that is used to prove sufficiency of some condition for a collection of sets to be a base for a topology. Should be in ZF1.thy.

lemma Finite1_L8: **assumes** A1:$\forall$U∈C. $\exists$A∈B. U = $\bigcup$A
  shows $\bigcup\bigcup$ {$\bigcup${A∈B. U = $\bigcup$A}. U∈C} = $\bigcup$C
⟨*proof*⟩

If an intersection of a collection is not empty, then the collection is not empty. We are (ab)using the fact the the intesection of empty collection is defined to be empty and prove by contradiction. Should be in ZF1.thy

lemma Finite1_L9: **assumes** A1:$\bigcap$A ≠ 0 **shows** A≠0
⟨*proof*⟩

Cartesian product of finite sets is finite.

lemma Finite1_L12: **assumes** A1:A ∈ Fin(A) **and** A2:B ∈ Fin(B)
  shows A×B ∈ Fin(A×B)
⟨*proof*⟩

We define the characterisic meta-function that is the identity on a set and assigns a default value everywhere else.

constdefs
  Characteristic(A,default,x) ≡ (if x∈A then x else default)

A finite subset is a finite subset of itself.

lemma Finite1_L13:
  assumes A1:A ∈ Fin(X) **shows** A ∈ Fin(A)
⟨*proof*⟩

Cartesian product of finite subsets is a finite subset of cartesian product.

lemma Finite1_L14: **assumes** A1:A ∈ Fin(X) B ∈ Fin(Y)
  shows A×B ∈ Fin(X×Y)
⟨*proof*⟩

The next lemma is needed in the Group_ZF_3 theory in a couple of places.

lemma Finite1_L15:

assumes A1: {b(x). x∈A} ∈ Fin(B) {c(x). x∈A} ∈ Fin(C)
and A2: f : B×C→E
shows {f<b(x),c(x)>. x∈A} ∈ Fin(E)
⟨*proof*⟩

Singletons are in the finite powerset.

lemma Finite1_L16: **assumes** x∈X **shows** {x} ∈ Fin(X)
⟨*proof*⟩

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

lemma Finite1_L16AA: **assumes** A1: {b(x). x∈A} ∈ Fin(B)
and A2: c∈C **and** A3: f : B×C→E
shows {f<b(x),c>. x∈A} ∈ Fin(E)
⟨*proof*⟩

In the IsarMathLib coding convention it is rather difficult to use results that take $\Longrightarrow$ (that is, another lemma) as one of the assumptions. It is easier to use a condition written with the first order implication ($\longrightarrow$). The next lemma is the induction step of the lemma about the first order induction.

lemma Finite1_L16A:
assumes ∀A∈Fin(X).∀x∈X. x∉A ∧ P(A)⟶P(A∪{x})
and x∈X **and** A∈Fin(X) **and** x∉A **and** P(A)
shows P(cons(x,A))
⟨*proof*⟩

First order version of the induction for the finite powerset.

lemma Finite1_L16B: **assumes** A1: P(0) **and** A2: B∈Fin(X)
and A3: ∀A∈Fin(X).∀x∈X. x∉A ∧ P(A)⟶P(A∪{x})
shows P(B)
⟨*proof*⟩

8.2 Finite range functions

In this section we define functions $f : X \to Y$, 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 Real_ZF_x.thy series.

constdefs
FinRangeFunctions(X,Y) ≡ {f:X→Y. f(X) ∈ Fin(Y)}

Constant functions have finite range.

lemma Finite1_L17: **assumes** c∈Y **and** X≠0
shows ConstantFunction(X,c) ∈ FinRangeFunctions(X,Y)
⟨*proof*⟩

Finite range functions have finite range.

lemma Finite1_L18: **assumes** f ∈ FinRangeFunctions(X,Y)
shows {f(x). x∈X} ∈ Fin(Y)
⟨*proof*⟩

An alternative form of the definition of finite range functions.

lemma Finite1_L19: **assumes** f:X→Y
and {f(x). x∈X} ∈ Fin(Y)
shows f ∈ FinRangeFunctions(X,Y)
⟨*proof*⟩

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

lemma Finite1_L20: **assumes** A1:f ∈ FinRangeFunctions(X,Y)
and A2:g : Y→Z
shows g O f ∈ FinRangeFunctions(X,Z)
⟨*proof*⟩

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

lemma Finite1_L21:
assumes A1: f ∈ FinRangeFunctions(X,Y) **and** A2: A⊆X
shows f(A) ∈ Fin(Y)
⟨*proof*⟩

end

9 Finite_ZF.thy

theory Finite_ZF_1 **imports** Finite1 Order_ZF

begin

This theory file contains properties of finite sets related to order relations.

9.1 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*⟩

For total and transitive relations finite set has a maximum.

theorem Finite_ZF_1_1_T1A:
  assumes A1: r {is total on} X **and** A2: trans(r)
  and A3: B∈Fin(X)
  shows B=0 ∨ HasAmaximum(r,B)
⟨*proof*⟩

Finite set has a minimum - induction step.

lemma Finite_ZF_1_1_L2:
  assumes A1: r {is total on} X **and** A2: trans(r)
  and A3: A∈Fin(X) **and** A4: x∈X **and** A5: A=0 ∨ HasAminimum(r,A)
  shows A∪{x} = 0 ∨ HasAminimum(r,A∪{x})
⟨*proof*⟩

For total and transitive relations finite set has a minimum.

theorem Finite_ZF_1_1_T1B:
  assumes A1: r {is total on} X **and** A2: trans(r)
  and A3: B ∈ Fin(X)
  shows B=0 ∨ HasAminimum(r,B)
⟨*proof*⟩

For transitive and total relations finite sets are bounded.

theorem Finite_ZF_1_T1:
  assumes A1: r {is total on} X **and** A2: trans(r)
  and A3: B∈Fin(X)
  shows IsBounded(B,r)
⟨*proof*⟩

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*⟩

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*⟩

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 antisymmetric.

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*⟩

end

10 Topology_ZF.thy

theory Topology_ZF **imports** Finite1 Fol1

begin

This theory file provides basic definitions and properties of topology, open and closed sets, closure and boundary.

10.1 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. Hence, we decided to define a topology as a collection of sets that contains the empty set and is closed under arbitrary unions and intersections of two sets, without any mention of the set on which the topology is defined. Recall that `Pow(T)` is the powerset of T, so that if $M \in$`Pow(T)` then M is a subset of T. We define interior of a set A as the union of all open sets contained in A. We use `Interior(A,T)` to denote the interior of A. Closed set is one such that it is contained in the carrier of the topology (i.e. $\bigcup T$) and its complement is open (i.e. belongs to the topology). The closure of a set is the intersection of all closed sets that contain it. To prove varius properties of closure we will often use the collection of closed sets that contain a given set A. Such collection does not have a name in romantic math. We will call it `ClosedCovers(A,T)`. The closure of a set A is defined as the intersection of the collection of the closed sets D such that $A \subseteq D$. We also define boundary of a set as the intersection of its closure with the closure of the complement (with respect to the carrier). 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 Fin(M) is the collection of finite subsets of M (finite powerset of M), defined in the `Finite` theory of Isabelle/ZF.

constdefs

```
  IsATopology (_ {is a topology} [90] 91)
  T {is a topology} ≡ ( 0 ∈ T) ∧ ( ∀M∈Pow(T). ⋃M ∈ T ) ∧
  ( ∀U∈T. ∀ V∈T. U∩V ∈ T)

  Interior(A,T) ≡ ⋃ {U∈T. U⊆A}

  IsClosed (infixl {is closed in} 90)
  D {is closed in} T ≡ (D ⊆ ⋃T ∧ ⋃T - D ∈ T)

  ClosedCovers(A,T) ≡ {D ∈ Pow(⋃T). D {is closed in} T ∧ A⊆D}
```

```
Closure(A,T) ≡ ⋂ ClosedCovers(A,T)

Boundary(A,T) ≡ Closure(A,T) ∩ Closure(⋃T - A,T)


IsCompact (infixl {is compact in} 90)
K {is compact in} T ≡ (K ⊆ ⋃T ∧
(∀ M∈Pow(T). K ⊆ ⋃M ⟶ (∃ N∈Fin(M). K ⊆ ⋃N)))
```

A basic example of a topology: the powerset of any set is a topology.

lemma `Top_1_L1: shows Pow(X) {is a topology}`
⟨*proof*⟩

The intersection of any nonempty collection of topologies on a set X is a topology.

lemma `Top_1_L2: assumes A1: 𝓜 ≠ 0 and A2: ∀T∈𝓜. T {is a topology}`
shows `(⋂𝓜) {is a topology}`
⟨*proof*⟩

We will now introduce some notation. In Isar, this is done by definining a "locale". Locale is kind of a context that holds some assumptions and notation used in all theorems proven in it. In the locale (context) below called `topology0` we assume that T is a topolgy. The interior of the set A (with respect to the topology in the context) is denoted `int(A)`. The closure of a set $A \subseteq \bigcup T$ is denoted `cl(A)` and the boundary is ∂A.

```
locale topology0 =
  fixes T
  assumes topSpaceAssum: T {is a topology}

  fixes int
  defines int_def [simp]: int(A) ≡ Interior(A,T)

  fixes cl
  defines cl_def [simp]: cl(A) ≡ Closure(A,T)

  fixes boundary (∂_ [91] 92)
  defines boundary_def [simp]: ∂A ≡ Boundary(A,T)
```

Intersection of a finite nonempty collection of open sets is open.

```
lemma (in topology0) Top_1_L3: assumes N≠0 N ∈ Fin(T)
  shows ⋂N ∈ T
  ⟨proof⟩
```

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 Finite1.thy.

```
lemma (in topology0) Top_1_L4:
```

shows (T {restricted to} X) {is a topology}
⟨*proof*⟩

10.2 Interior of a set

In 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** int(A) ⊆ A
⟨*proof*⟩

Interior is open.

lemma (in topology0) Top_2_L2: **shows** int(A) ∈ T
⟨*proof*⟩

A set is open iff it is equal to its interior.

lemma (in topology0) Top_2_L3: U∈T ⟷ int(U) = U
⟨*proof*⟩

Interior of the interior is the interior.

lemma (in topology0) Top_2_L4: **shows** int(int(A)) = int(A)
⟨*proof*⟩

Interior of a bigger set is bigger.

lemma (in topology0) interior_mono:
assumes A1: A⊆B **shows** int(A) ⊆ int(B)
⟨*proof*⟩

An open subset of any set is a subset of the interior of that set.

lemma (in topology0) Top_2_L5: **assumes** U⊆A **and** U∈T
shows U ⊆ int(A)
⟨*proof*⟩

If a point of a set has an open neighboorhood contained in the set, then the point belongs to the interior of the set.

lemma (in topology0) Top_2_L6: **assumes** ∃U∈T. (x∈U ∧ U⊆A)
shows x ∈ int(A)
⟨*proof*⟩

A set is open iff its every point has a an open neighbourhood contained in the set. We will formulate this statement as two lemmas (implication one way and the other way). The lemma below shows that if a set is open then every point has a an open neighbourhood contained in the set.

lemma (in topology0) Top_2_L7:
assumes A1: V∈T
shows ∀x∈V. ∃U∈T. (x∈U ∧ U⊆V)

⟨*proof*⟩

If every point of a set has a an open neighbourhood contained in the set then the set is open.

lemma (in topology0**)** Top_2_L8:
  assumes A1: ∀x∈V. ∃U∈T. (x∈U ∧ U⊆V)
  shows V∈T
⟨*proof*⟩

10.3 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** (⋃T) {is closed in} T
⟨*proof*⟩

Empty set is closed.

lemma (in topology0**)** Top_3_L2: **shows** 0 {is closed in} T
  ⟨*proof*⟩

The collection of closed covers of a subset of the carrier of topology is never empty. This is good to know, as we want to intersect this collection to get the closure.

lemma (in topology0**)** Top_3_L3:
  assumes A1: A ⊆ ⋃T **shows** ClosedCovers(A,T) ≠ 0
⟨*proof*⟩

Intersection of a nonempty family of closed sets is closed.

lemma (in topology0**)** Top_3_L4: **assumes** A1: K≠0 **and**
  A2: ∀D∈K. D {is closed in} T
  shows (⋂K) {is closed in} T
⟨*proof*⟩

The union and intersection of two closed sets are closed.

lemma (in topology0**)** Top_3_L5:
  assumes A1: D_1 {is closed in} T D_2 {is closed in} T
  shows
  (D_1∩D_2) {is closed in} T
  (D_1∪D_2) {is closed in} T
⟨*proof*⟩

Finite union of closed sets is closed. To understand the proof recall that D ∈Pow(⋃T) means that D is as subset of the carrier of the topology.

lemma (in topology0**)** Top_3_L6:
  assumes A1: N ∈ Fin({D∈Pow(⋃T). D {is closed in} T})

shows (⋃N) {is closed in} T
⟨*proof*⟩

Closure of a set is closed.

lemma (in topology0) Top_3_L7: **assumes** A ⊆ ⋃T
shows cl(A) {is closed in} T
⟨*proof*⟩

Closure of a bigger sets is bigger.

lemma (in topology0) top_closure_mono:
assumes A1: A ⊆ ⋃T B ⊆ ⋃T **and** A2:A⊆B
shows cl(A) ⊆ cl(B)
⟨*proof*⟩

Boundary of a set is closed.

lemma (in topology0) boundary_closed:
assumes A1: A ⊆ ⋃T **shows** ∂A {is closed in} T
⟨*proof*⟩

A set is closed iff it is equal to its closure.

lemma (in topology0) Top_3_L8: **assumes** A1: A ⊆ ⋃T
shows A {is closed in} T ⟷ cl(A) = A
⟨*proof*⟩

Complement of an open set is closed.

lemma (in topology0) Top_3_L9:
assumes A1: A∈T
shows (⋃T - A) {is closed in} T
⟨*proof*⟩

A set is contained in its closure.

lemma (in topology0) Top_3_L10: **assumes** A ⊆ ⋃T **shows** A ⊆ cl(A)
⟨*proof*⟩

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 ⊆ ⋃T
shows
cl(A) ⊆ ⋃T
cl(⋃T - A) = ⋃T - int(A)
⟨*proof*⟩

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*⟩

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*⟩

If two open sets are disjoint, then we can close one of them and they will still be disjoint.

lemma (in topology0) Top_3_L14:
  assumes A1: U∈T V∈T **and** A2: U∩V = 0
  shows cl(U) ∩ V = 0
⟨*proof*⟩

end

11 Topology_ZF_1.thy

theory Topology_ZF_1 **imports** Topology_ZF Fol1

begin

11.1 Separation axioms.

Topological spaces cas be classified according to certain properties called "separation axioms". This section defines what it means that a topological space is T_0, T_1 or T_2.

A topology on X is T_0 if for every pair of distinct points of X there is an open set that contains only one of them. A topology is T_1 if for every such pair there exist an open set that contains the first point but not the second. 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.

constdefs

```
isT0 (_ {is T₀} [90] 91)
T {is T₀} ≡ ∀ x y. ((x ∈ ⋃T ∧ y ∈ ⋃T ∧  x≠y) ⟶
(∃U∈T. (x∈U ∧ y∉U) ∨ (y∈U ∧ x∉U)))

isT1 (_ {is T₁} [90] 91)
T {is T₁} ≡ ∀ x y. ((x ∈ ⋃T ∧ y ∈ ⋃T ∧  x≠y) ⟶
(∃U∈T. (x∈U ∧ y∉U)))

isT2 (_ {is T₂} [90] 91)
T {is T₂} ≡ ∀ x y. ((x ∈ ⋃T ∧ y ∈ ⋃T ∧  x≠y) ⟶
(∃U∈T. ∃V∈T. x∈U ∧ y∈V ∧ U∩V=0))
```

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 isT0 condition and isT1.

lemma T1_is_T0: **assumes** A1: T {is T₁} **shows** T {is T₀}
⟨*proof*⟩

If a topology is T_2 then it is T_1.

lemma T2_is_T1: **assumes** A1: T {is T₂} **shows** T {is T₁}
⟨*proof*⟩

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 {is T₀} **and** A2: x ∈ ⋃T y ∈ ⋃T
and A3: ∀U∈T. (x∈U ⟷ y∈U)
shows x=y
⟨*proof*⟩

In a T_2 space two points can be separated by an open set with its boundary.

lemma (in topology0**)** Top_1_1_L2:
  assumes A1: T {is T_2} **and** A2: x ∈ ⋃T y ∈ ⋃T x≠y
  shows ∃U∈T. (x∈U ∧ y ∉ cl(U))
⟨*proof*⟩

In a T_2 space compact sets are closed. Doing a formal proof of this theorem gave me an interesting insight into the role of the Axiom of Choice in romantic proofs.

A typical romantic 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 have never realized that such reasoning requires (an) 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 Top_1_1_L2 above). This only states that the set of such open sets U is not empty. To get the collection $\{U_x\}_{x\in K}$ in the above proof we have to select one such set among many for every $x \in K$ and this is where we use (an) Axiom of Choice. Probably in 99/100 cases when a romatic calculus proof states something like $\forall\varepsilon\exists\delta_\varepsilon \cdots$ the proof uses Axiom of Choice. In the proof below we avoid using Axiom of Choice (read it to find out how). It is an interesting question which such calculus proofs can be reformulated so that the usage of AC is avoided. I remember Sierpiński published a paper in 1919 (or was it 1914? my memory is not that good any more) where he showed that one needs an Axiom of Choice to show the equivalence of the Heine and Cauchy definitions of limits.

theorem (in topology0**)** in_t2_compact_is_cl:
  assumes A1: T {is T_2} **and** A2: K {is compact in} T
  shows K {is closed in} T
⟨*proof*⟩

11.2 Bases and subbases.

A base of topology is a collection of open sets such that every open set is a union of the sets from the base. A subbase is a collection of open sets such that finite intersection of those sets form a base. 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 neighboorhod from B contained in $U \cap V$.

constdefs

  IsAbaseFor (**infixl** {is a base for} 65)

```
B {is a base for} T ≡ B⊆T ∧ T = {⋃A. A∈Pow(B)}

IsAsubBaseFor (infixl {is a subbase for} 65)
B {is a subbase for} T ≡
B ⊆ T ∧ {⋂A. A∈Fin(B)} {is a base for} T

SatisfiesBaseCondition (_ {satisfies the base condition} [50] 50)
B {satisfies the base condition} ≡
∀U V. ((U∈B ∧ V∈B) ⟶ (∀x ∈ U∩V. ∃W∈B. x∈W ∧ W ⊆ U∩V))
```

Each open set is a union of some sets from the base.

lemma `Top_1_2_L1:` **assumes** `B {is a base for} T` **and** `U∈T`
shows `∃A∈Pow(B). U = ⋃A`
⟨*proof*⟩

A necessary conditionfor a collection of sets to be a base for some topology : every point in the intersection of two sets in the base has a neighboorhood from the base contained in the intersection.

lemma `Top_1_2_L2:`
assumes `A1:∃T. T {is a topology} ∧ B {is a base for} T`
and `A2: V∈B  W∈B`
shows `∀ x ∈ V∩W. ∃U∈B. x∈U ∧ U ⊆ V ∩ W`
⟨*proof*⟩

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: ∀x∈ V∩W . ∃U∈B. x∈U ∧ U ⊆ V∩W`
shows `V∩W ∈ {⋃A. A∈Pow(B)}`
⟨*proof*⟩

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:  U₁ ∈ {⋃A. A∈Pow(B)}  U₂ ∈ {⋃A. A∈Pow(B)}`
and `A2: B {satisfies the base condition}`
shows `∃C. C ⊆ {⋃A. A∈Pow(B)} ∧ U₁∩U₂ = ⋃C`
⟨*proof*⟩

If B satisfies the base condition, then the collection of unions of sets from B is a topology and B is a base for this topology.

theorem `Top_1_2_T1:`

assumes A1: B {satisfies the base condition}
and A2: T = {⋃A. A∈Pow(B)}
shows T {is a topology} **and** B {is a base for} T
⟨*proof*⟩

The carrier of the base and topology are the same.

lemma Top_1_2_L5: **assumes** B {is a base for} T
shows ⋃T = ⋃B
⟨*proof*⟩

11.3 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 $A \times B$, where $A \in S, B \in T$.

constdefs

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*⟩

The product topology is indeed a topology on the product.

theorem Top_1_4_T1: **assumes** A1: T {is a topology} S {is a topology}
shows
ProductTopology(T,S) {is a topology}
ProductCollection(T,S) {is a base for} ProductTopology(T,S)
⋃ ProductTopology(T,S) = ⋃T × ⋃S
⟨*proof*⟩

end

12 Topology_ZF_2.thy

theory Topology_ZF_2 **imports** Topology_ZF_1 func1 Fol1

begin

12.1 Continuous functions.

In standard math we say that a function is contiuous with respect to two topologies τ_1, τ_2 if the inverse image of sets from topology τ_2 are in τ_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 τ_1, τ_2 are topologies. This means for example that when we define measurable functions, the definition will be the same.

Recall that in Isabelle/ZF `f-(A)` denotes the inverse image of (set) A with respect to (function) f.

constdefs
  IsContinuous(τ_1,τ_2,f) $\equiv$ ($\forall$U$\in\tau_2$. f-(U) $\in$ τ_1)

We will work with a pair of topological spaces. The following locale sets up our context that consists of two topologies τ_1, τ_2 and a function $f : X_1 \to X_2$, where X_i is defined as $\bigcup \tau_i$ for $i = 1, 2$. We also define notation $\mathtt{cl}_1\mathtt{(A)}$ and $\mathtt{cl}_2\mathtt{(A)}$ for closure of a set A in topologies τ_1 and τ_2, respectively.

locale two_top_spaces0 =

  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 $\mathtt{X}_1$
  defines $\mathtt{X}_1$_def [simp]: $\mathtt{X}_1 \equiv \bigcup \tau_1$

  fixes $\mathtt{X}_2$
  defines $\mathtt{X}_2$_def [simp]: $\mathtt{X}_2 \equiv \bigcup \tau_2$

  fixes f
  assumes fmapAssum: f: $\mathtt{X}_1 \to \mathtt{X}_2$

  fixes isContinuous (_ {is continuous} [50] 50)
  defines isContinuous_def [simp]: g {is continuous} $\equiv$ IsContinuous(τ_1,τ_2,g)

  fixes $\mathtt{cl}_1$
  defines $\mathtt{cl}_1$_def [simp]: $\mathtt{cl}_1$(A) $\equiv$ Closure(A,τ_1)

fixes cl$_2$
defines cl$_2$_def [simp]: cl$_2$(A) $\equiv$ Closure(A,τ_2)

First we show that theorems proven in locale topology0 are valid when applied to topologies τ_1 and τ_2.

lemma (in two_top_spaces0**)** topol_cntxs_valid:
shows topology0(τ_1) **and** topology0(τ_2)
⟨*proof*⟩

For continuous functions the inverse image of a closed set is closed.

lemma (in two_top_spaces0**)** TopZF_2_1_L1:
assumes A1: f {is continuous} **and** A2: D {is closed in} τ_2
shows f-(D) {is closed in} τ_1
⟨*proof*⟩

If the inverse image of every closed set is closed, then the image of a closure is contained in the closure of the image.

lemma (in two_top_spaces0**)** Top_ZF_2_1_L2:
assumes A1: $\forall$D. ((D {is closed in} τ_2) $\longrightarrow$ f-(D) {is closed in} τ_1)
and A2: A $\subseteq$ X$_1$
shows f(cl$_1$(A)) $\subseteq$ cl$_2$(f(A))
⟨*proof*⟩

If $f\left(\overline{A}\right) \subseteq \overline{f(A)}$ (the image of the closure is contained in the closure of the image), then $\overline{f^{-1}(B)} \subseteq f^{-1}\left(\overline{B}\right)$ (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$_1$ $\longrightarrow$ f(cl$_1$(A)) $\subseteq$ cl$_2$(f(A)))
shows $\forall$B. (B $\subseteq$ X$_2$ $\longrightarrow$ cl$_1$(f-(B)) $\subseteq$ f-(cl$_2$(B)))
⟨*proof*⟩

If $\overline{f^{-1}(B)} \subseteq f^{-1}\left(\overline{B}\right)$ (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 showing equavalence of four definitions of continuity.

lemma (in two_top_spaces0**)** Top_ZF_2_1_L4:
assumes A1: $\forall$B. (B $\subseteq$ X$_2$ $\longrightarrow$ cl$_1$(f-(B)) $\subseteq$ f-(cl$_2$(B)))
shows f {is continuous}
⟨*proof*⟩

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**)** Top_ZF_2_1_L5:
assumes A1: B {is a base for} τ_2 **and** A2: $\forall$U$\in$B. f-(U) $\in$ τ_1
shows f {is continuous}
⟨*proof*⟩

We can strenghten the previous lemma: it is sufficient to check if the inverse image of every set in a subbase is open. The proof is rather awkward, as

usual when we deal with general intersections. We have to keep track of the case when the collection is empty.

lemma (in two_top_spaces0) Top_ZF_2_1_L6:
  assumes A1: B {is a subbase for} τ_2 **and** A2: $\forall$U$\in$B. f-(U) $\in \tau_1$
  shows f {is continuous}
⟨*proof*⟩

end

13 Group_ZF.thy

theory Group_ZF **imports** func_ZF

begin

This theory file will cover basics of group theory.

13.1 Monoids.

Monoid is a set with an associative operation and a neutral element. The operation is of course a function on $G \times G$ with values in G, and therefore it is a subset of $(G \times G) \times G$. Those who don't like that can go to HOL. Monoid is like a group except that we don't require existence of the inverse.

constdefs
```
  IsAmonoid(G,f) ≡
  f {is associative on} G ∧
  (∃e∈G. (∀ g∈G. ( (f(<e,g>) = g) ∧ (f(<g,e>) = g))))
```

We use locales to define notation. This allows to separate notation and notion definitions. We would like to use additive notation for monoid, but unfortunately + is already taken.

locale monoid0 =
```
  fixes G and 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
```
⟨*proof*⟩

There is only one neutral element in monoid.

lemma (in monoid0) group0_1_L2:
```
  ∃!e. e∈G ∧ (∀ g∈G. ( (e⊕g = g) ∧ g⊕e = g))
```
⟨*proof*⟩

We could put the definition of neutral element anywhere, but it is only usable in conjuction with the above lemma.

constdefs
```
 TheNeutralElement(G,f) ≡
  ( THE e. e∈G ∧ (∀ g∈G. f<e,g> = g ∧ f<g,e> = g))
```

The neutral element is neutral.

lemma (in monoid0) group0_1_L3:

assumes A1: e = TheNeutralElement(G,f)
shows e ∈ G ∧ (∀g∈G. e ⊕ g = g ∧ g ⊕ e = g)
⟨*proof*⟩

The monoid carrier is not empty.

lemma (in monoid0**)** group0_1_L3A: G≠0
⟨*proof*⟩

The range of the monoid operation is the whole monoid carrier.

lemma (in monoid0**)** group0_1_L3B: range(f) = G
⟨*proof*⟩

In a monoid a 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*⟩

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. This is proven separately because it is used more than once.

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*⟩

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 will be useful when we study subgroups.

theorem (in monoid0**)** group0_1_T1:
assumes A1: H {is closed under} f
and A2: H⊆G
and A3: TheNeutralElement(G,f) ∈ H
shows IsAmonoid(H,restrict(f,H×H))
⟨*proof*⟩

Under the assumptions of group0_1_T1 the neutral element of a submonoid is the same as that of the monoid.

lemma group0_1_L6:
assumes A1: IsAmonoid(G,f)
and A2: H {is closed under} f

and A3: H⊆G
and A4: TheNeutralElement(G,f) ∈ H
shows TheNeutralElement(H,restrict(f,H×H)) = TheNeutralElement(G,f)
⟨*proof*⟩

13.2 Basic definitions and results for 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. We also define the group inverse as a relation on the group carrier. Later we will show that this relation is a function. The GroupInv below is really the right inverse, understood as a function, that is a subset of $G \times G$.

constdefs
IsAgroup(G,f) ≡
(IsAmonoid(G,f) ∧ (∀g∈G. ∃b∈G. f<g,b> = TheNeutralElement(G,f)))

GroupInv(G,f) ≡ {<x,y> ∈ G×G. f<x,y> = TheNeutralElement(G,f)}

We will use the miltiplicative notation for groups.

locale group0 =
fixes G **and** f
assumes groupAssum: IsAgroup(G,f)

fixes neut (1)
defines neut_def[simp]: **1** ≡ TheNeutralElement(G,f)

fixes groper (**infixl** · 70)
defines groper_def [simp]: a · b ≡ f<a,b>

fixes inv (_$^{-1}$ [90] 91)
defines inv_def[simp]: x^{-1} ≡ GroupInv(G,f)(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: monoid0(G,f)
⟨*proof*⟩

In some strange cases Isabelle has difficulties with applying the definition of a group. The next lemma defines a rule to be applied in such cases.

lemma definition_of_group: **assumes** IsAmonoid(G,f)
and ∀g∈G. ∃b∈G. f⟨g,b⟩ = TheNeutralElement(G,f)
shows IsAgroup(G,f)
⟨*proof*⟩

Technical lemma that allows to use 1 as the neutral element of the group without referencing a list of lemmas and definitions.

lemma (**in** group0) group0_2_L2:

shows 1∈G ∧ (∀g∈G.(1·g = g ∧ g·1 = g))
⟨*proof*⟩

The group is closed under the group operation. Used all the time, useful to have handy.

lemma (in group0**)** group_op_closed: **assumes** a∈G b∈G
shows a·b ∈ G ⟨*proof*⟩

The group operation is associative. This is another technical lemma that allows to shorten the list of referenced lemmas in some proofs.

lemma (in group0**)** group_oper_assoc:
assumes a∈G b∈G c∈G **shows** a·(b·c) = a·b·c
⟨*proof*⟩

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_assocA: **shows** f : G×G→G
⟨*proof*⟩

The definition of group requires the existence of the right inverse. We show that this is also the left inverse.

theorem (in group0**)** group0_2_T1:
assumes A1: g∈G **and** A2: b∈G **and** A3: g·b = 1
shows b·g = 1
⟨*proof*⟩

For every element of a group there is only one inverse.

lemma (in group0**)** group0_2_L4:
assumes A1:x∈G **shows** ∃!y. y∈G ∧ x·y = 1
⟨*proof*⟩

The group inverse is a function that maps G into G.

theorem group0_2_T2:
assumes A1: IsAgroup(G,f) **shows** GroupInv(G,f) : G→G
⟨*proof*⟩

We can think about the group inverse (the function) as the inverse image of the neutral element.

theorem (in group0**)** group0_2_T3: **shows** f-{1} = GroupInv(G,f)
⟨*proof*⟩

The inverse is in the group.

lemma (in group0**)** inverse_in_group: **assumes** A1: x∈G **shows** x^{-1}∈G
⟨*proof*⟩

The notation for the inverse means what it is supposed to mean.

lemma (in group0) group0_2_L6:
assumes A1: x∈G shows $x \cdot x^{-1} = 1 \wedge x^{-1} \cdot x = 1$
⟨*proof*⟩

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*⟩

lemma (in group0) group0_2_L8:
assumes A1: a∈G and A2: b∈G and A3:a·b = b
shows a=1
⟨*proof*⟩

The inverse of the neutral element is the neutral element.

lemma (in group0) group_inv_of_one: shows 1^{-1}**=1**
⟨*proof*⟩

if $a^{-1} = 1$, then $a = 1$.

lemma (in group0) group0_2_L8A:
assumes A1: a∈G and A2: $a^{-1} = 1$
shows a = 1
⟨*proof*⟩

If a is not a unit, then its inverse is not either.

lemma (in group0) group0_2_L8B:
assumes a∈G and a $\neq$ **1**
shows $a^{-1} \neq 1$ ⟨*proof*⟩

If a^{-1} is not a unit, then a is not either.

lemma (in group0) group0_2_L8C:
assumes a∈G and $a^{-1} \neq 1$
shows a≠1
⟨*proof*⟩

If a product of two elements of a group is equal to the neutral element then they are inverses of each other.

lemma (in group0) group0_2_L9:
assumes A1: a∈G and A2: b∈G and A3: a·b = 1
shows $a = b^{-1}$ $b = a^{-1}$
⟨*proof*⟩

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: $\forall g \in G.\ b(g) \in G \wedge g \cdot b(g) = 1$
shows $\forall g \in G.\ b(g) = g^{-1}$
⟨*proof*⟩

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 $b^{-1} \cdot a^{-1} = (a \cdot b)^{-1}$
⟨*proof*⟩

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
$(a \cdot b \cdot c)^{-1} = c^{-1} \cdot (a \cdot b)^{-1}$
$(a \cdot b \cdot c)^{-1} = c^{-1} \cdot (b^{-1} \cdot a^{-1})$
$(a \cdot b \cdot c)^{-1} = c^{-1} \cdot b^{-1} \cdot a^{-1}$
⟨*proof*⟩

The inverse of the inverse is the element.

lemma (in group0) group_inv_of_inv:
assumes $a \in G$ **shows** $a = (a^{-1})^{-1}$
⟨*proof*⟩

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*⟩

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*⟩

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*⟩

What is the inverse of ab^{-1} ?

lemma (in group0) group0_2_L12:
assumes A1: $a \in G$ $b \in G$

shows
$(a \cdot b^{-1})^{-1} = b \cdot a^{-1}$
$(a^{-1} \cdot b)^{-1} = b^{-1} \cdot a$
⟨*proof*⟩

A couple useful rearrangements with three elements: we can insert a $b \cdot b^{-1}$ between two group elements (another version) and one about a product of an element and inverse of a product, and two others.

lemma (in group0**)** group0_2_L14A:
assumes A1: $a \in G$ $b \in G$ $c \in G$
shows
$a \cdot c^{-1} = (a \cdot b^{-1}) \cdot (b \cdot c^{-1})$
$a^{-1} \cdot c = (a^{-1} \cdot b) \cdot (b^{-1} \cdot c)$
$a \cdot (b \cdot c)^{-1} = a \cdot c^{-1} \cdot b^{-1}$
$a \cdot (b \cdot c^{-1}) = a \cdot b \cdot c^{-1}$
$(a \cdot b^{-1} \cdot c^{-1})^{-1} = c \cdot b \cdot a^{-1}$
$a \cdot b \cdot c^{-1} \cdot (c \cdot b^{-1}) = a$
$a \cdot (b \cdot c) \cdot c^{-1} = a \cdot b$
⟨*proof*⟩

Another lemma about rearranging a product.

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*⟩

We can cancel an element with its inverse that is written next to it.

lemma (in group0**)** group0_2_L16:
assumes A1: $a \in G$ $b \in G$
shows
$a \cdot b^{-1} \cdot b = a$
$a \cdot b \cdot b^{-1} = a$
$a^{-1} \cdot (a \cdot b) = b$
$a \cdot (a^{-1} \cdot b) = b$
⟨*proof*⟩

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*⟩

A hard to clasify fact: 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 A1: $H \subseteq G$
and A2: H {is closed under} f
shows $(H \cup \{1\})$ {is closed under} f

⟨*proof*⟩

We can put an element on the other side of an equation.

lemma (in group0**)** group0_2_L18:
  assumes A1: a∈G b∈G c∈G
  and A2: c = a·b
  shows c·b^{-1} = a a^{-1}·c = b
⟨*proof*⟩

Multiplying different group elements by the same factor results in different group elements.

lemma (in group0**)** group0_2_L19:
  assumes A1: a∈G b∈G c∈G **and** A2: a≠b
  shows
  a·c ≠ b·c
  c·a ≠ c·b
⟨*proof*⟩

13.3 Subgroups

There are two common ways to define subgroups. One requires that the group operations are 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. We do not require H to be a subset of G as this can be inferred from our definition. The rest of this section is devoted to proving the equivalence of these two definitions of the notion of a subgroup.

constdefs
  IsAsubgroup(H,f) ≡ IsAgroup(H, restrict(f,H×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 neutral element of the 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*⟩

Subgroup is contained in the group.

lemma (in group0**)** group0_3_L2:
assumes A1:IsAsubgroup(H,f)
shows H⊆G
⟨*proof*⟩

The group neutral element (denoted 1 in the group0 context) is a neutral element for the subgroup with respect to the froup action.

lemma (in group0**)** group0_3_L3:
assumes A1:IsAsubgroup(H,f)
shows ∀h∈H. **1**·h = h ∧ h·**1** = h
⟨*proof*⟩

The neutral element of a subgroup is the same as that of the group.

lemma (in group0**)** group0_3_L4: **assumes** A1:IsAsubgroup(H,f)
shows TheNeutralElement(H,restrict(f,H×H)) = **1**
⟨*proof*⟩

The neutral element of the group (denoted 1 in the group0 context) belongs to every subgroup.

lemma (in group0**)** group0_3_L5: **assumes** A1: IsAsubgroup(H,f)
shows **1**∈H
⟨*proof*⟩

Subgroups are closed with respect to the group operation.

lemma (in group0**)** group0_3_L6: **assumes** A1:IsAsubgroup(H,f)
and A2:a∈H b∈H
shows a·b ∈ H
⟨*proof*⟩

A preliminary lemma that we need to show that taking the inverse in the subgroup is the same as taking the inverse in the group.

lemma group0_3_L7A:
assumes A1:IsAgroup(G,f)
and A2:IsAsubgroup(H,f) **and** A3:g=restrict(f,H×H)
shows GroupInv(G,f) ∩ H×H = GroupInv(H,g)
⟨*proof*⟩

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,f)
and A2:g=restrict(f,H×H)
shows GroupInv(H,g) = restrict(GroupInv(G,f),H)
⟨*proof*⟩

A sligtly weaker, but more convenient in applications, reformulation of the above theorem.

theorem (in group0**)** group0_3_T2:
assumes IsAsubgroup(H,f)
and g=restrict(f,H×H)
shows ∀h∈H. GroupInv(H,g)(h) = h^{-1}
⟨*proof*⟩

Subgroups are closed with respect to taking the group inverse. Again, I was unable to apply inverse_in_group directly to the group H. This problem is worked around by repeating the (short) proof of inverse_in_group in the proof below.

theorem (in group0**)** group0_3_T3A:
assumes A1:IsAsubgroup(H,f) **and** A2:h∈H
shows h^{-1}∈ H
⟨*proof*⟩

The next theorem states that a nonempty subset of 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≠0
and A2: H⊆G
and A3: H {is closed under} f
and A4: ∀x∈H. x^{-1} ∈ H
shows IsAsubgroup(H,f)
⟨*proof*⟩

Intersection of subgroups is a subgroup of each factor.

lemma group0_3_L7:
assumes A1:IsAgroup(G,f)
and A2:IsAsubgroup(H_1,f)
and A3:IsAsubgroup(H_2,f)
shows IsAsubgroup(H_1∩H_2,restrict(f,H_1×H_1))
⟨*proof*⟩

13.4 Abelian groups

Here we will prove some facts specific to abelian groups.

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. For the parantheses simp does it very well.

lemma (in group0**)** group0_4_L2:
assumes A1:f {is commutative on} G
and A2:a∈G b∈G c∈G d∈G E∈G F∈G
shows (a·b)·(c·d)·(E·F) = (a·(d·F))·(b·(c·E))
⟨*proof*⟩

Another useful rearrangement.

lemma (in group0**)** group0_4_L3:
assumes A1:f {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*⟩

Some useful rearrangements for two elements of a group.

lemma (in group0**)** group0_4_L4:
assumes A1:f {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}
(a·b^{-1})$^{-1}$ = a^{-1}·b
⟨*proof*⟩

Another bunch of useful rearrangements with three elements.

lemma (in group0**)** group0_4_L4A:
assumes A1:f {is commutative on} G
and A2: a∈G b∈G c∈G
shows
a·b·c = c·a·b
a^{-1}·(b^{-1}·c^{-1})$^{-1}$ = (a·(b·c)$^{-1}$)$^{-1}$
a·(b·c)$^{-1}$ = a·b^{-1}·c^{-1}
a·(b·c^{-1})$^{-1}$ = a·b^{-1}·c
a·b^{-1}·c^{-1} = a·c^{-1}·b^{-1}
⟨*proof*⟩

Another useful rearrangement.

lemma (in group0**)** group0_4_L4B:
assumes f {is commutative on} G
and a∈G b∈G c∈G
shows a·b^{-1}·(b·c^{-1}) = a·c^{-1}
⟨*proof*⟩

A couple of permutations of order for three alements.

lemma (in group0**)** group0_4_L4C:
assumes A1: f {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*⟩

Some rearrangement with three elements and inverse.

lemma (in group0**)** group0_4_L4D:
assumes A1: f {is commutative on} G
and A2: a∈G b∈G c∈G
shows
$a^{-1}\cdot b^{-1}\cdot c = c\cdot a^{-1}\cdot b^{-1}$
$b^{-1}\cdot a^{-1}\cdot c = c\cdot a^{-1}\cdot b^{-1}$
$(a^{-1}\cdot b\cdot c)^{-1} = a\cdot b^{-1}\cdot c^{-1}$
⟨*proof*⟩

Another rearrangement lemma with three elements and equation.

lemma (in group0**)** group0_4_L5: **assumes** A1:f {is commutative on} G
and A2: a∈G b∈G c∈G
and A3: $c = a\cdot b^{-1}$
shows a = b·c
⟨*proof*⟩

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: f {is commutative on} G
and A2: a∈G b∈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*⟩

Another lemma about cancelling with two elements.

lemma (in group0**)** group0_4_L6AA:
assumes A1: f {is commutative on} G **and** A2: a∈G b∈G
shows
$a\cdot b^{-1}\cdot a^{-1} = b^{-1}$
⟨*proof*⟩

Another lemma about cancelling with two elements.

lemma (in group0**)** group0_4_L6AB:
assumes A1: f {is commutative on} G **and** A2: a∈G b∈G
shows
$a\cdot(a\cdot b)^{-1} = b^{-1}$

a·(b·a^{-1}) = b
⟨*proof*⟩

Another lemma about cancelling with two elements.

lemma (in group0**)** group0_4_L6AC:
assumes f {is commutative on} G **and** a∈G b∈G
shows a·(a·b^{-1})$^{-1}$ = b
⟨*proof*⟩

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: f {is commutative on} G
and A2: a∈G b∈G c∈G
shows
a·b·c·a^{-1} = b·c
a^{-1}·b·c·a = b·c
⟨*proof*⟩

In abelian groups we can cancel an element with its inverse even if separated by three other elements.

lemma (in group0**)** group0_4_L6C: **assumes** A1: f {is commutative on} G
and A2: a∈G b∈G c∈G d∈G
shows a·b·c·d·a^{-1} = b·c·d
⟨*proof*⟩

Another couple of useful rearrangements of three elements and cancelling.

lemma (in group0**)** group0_4_L6D:
assumes A1: f {is commutative on} G
and A2: a∈G b∈G c∈G
shows
a·b^{-1}·(a·c^{-1})$^{-1}$ = c·b^{-1}
(a·c)$^{-1}$·(b·c) = a^{-1}·b
a·(b·(c·a^{-1}·b^{-1})) = c
a·b·c^{-1}·(c·a^{-1}) = b
⟨*proof*⟩

Another useful rearrangement of three elements and cancelling.

lemma (in group0**)** group0_4_L6E:
assumes A1: f {is commutative on} G
and A2: a∈G b∈G c∈G
shows
a·b·(a·c)$^{-1}$ = b·c^{-1}
⟨*proof*⟩

A rearrangement with two elements and canceelling, special case of group0_4_L6D when $c = b^{-1}$.

lemma (in group0**)** group0_4_L6F:

assumes A1: f {is commutative on} G
and A2: a∈G b∈G
shows $a{\cdot}b^{-1}{\cdot}(a{\cdot}b)^{-1} = b^{-1}{\cdot}b^{-1}$
⟨*proof*⟩

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: f {is commutative on} G
and A2: a∈G b∈G c∈G d∈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*⟩

Some rearrangements with four elements and inverse that are applications of rearr_ab_gr_4_elem

lemma (in group0**)** rearr_ab_gr_4_elemB:
assumes A1: f {is commutative on} G
and A2: a∈G b∈G c∈G d∈G
shows
$a{\cdot}b^{-1}{\cdot}c^{-1}{\cdot}d^{-1} = a{\cdot}d^{-1}{\cdot}b^{-1}{\cdot}c^{-1}$
$a{\cdot}b{\cdot}c{\cdot}d^{-1} = a{\cdot}d^{-1}{\cdot}b{\cdot}c$
$a{\cdot}b{\cdot}c^{-1}{\cdot}d^{-1} = a{\cdot}c^{-1}{\cdot}(b{\cdot}d^{-1})$
⟨*proof*⟩

Some rearrangement lemmas with four elements.

lemma (in group0**)** group0_4_L7:
assumes A1: f {is commutative on} G
and A2: a∈G b∈G c∈G d∈G
shows
$a{\cdot}b{\cdot}c{\cdot}d^{-1} = a{\cdot}d^{-1}{\cdot}\ b{\cdot}c$
$a{\cdot}d{\cdot}(b{\cdot}d{\cdot}(c{\cdot}d))^{-1} = a{\cdot}(b{\cdot}c)^{-1}{\cdot}d^{-1}$
$a{\cdot}(b{\cdot}c){\cdot}d = a{\cdot}b{\cdot}d{\cdot}c$
⟨*proof*⟩

Some other rearrangements with four elements.

lemma (in group0**)** group0_4_L8:
assumes A1: f {is commutative on} G
and A2: a∈G b∈G c∈G d∈G
shows
$a{\cdot}(b{\cdot}c)^{-1} = (a{\cdot}d^{-1}{\cdot}c^{-1}){\cdot}(d{\cdot}b^{-1})$
$a{\cdot}b{\cdot}(c{\cdot}d) = c{\cdot}a{\cdot}(b{\cdot}d)$
$a{\cdot}b{\cdot}(c{\cdot}d) = a{\cdot}c{\cdot}(b{\cdot}d)$
$a{\cdot}(b{\cdot}c^{-1}){\cdot}d = a{\cdot}b{\cdot}d{\cdot}c^{-1}$
$(a{\cdot}b){\cdot}(c{\cdot}d)^{-1}{\cdot}(b{\cdot}d^{-1})^{-1} = a{\cdot}c^{-1}$
⟨*proof*⟩

Some other rearrangements with four elements.

lemma (in group0) group0_4_L8A:
assumes A1: f {is commutative on} G
and A2: a∈G b∈G c∈G d∈G
shows
$a \cdot b^{-1} \cdot (c \cdot d^{-1}) = a \cdot c \cdot (b^{-1} \cdot d^{-1})$
$a \cdot b^{-1} \cdot (c \cdot d^{-1}) = a \cdot c \cdot b^{-1} \cdot d^{-1}$
⟨*proof*⟩

Another rearrangement about equation.

lemma (in group0) group0_4_L9:
assumes A1: f {is commutative on} G
and A2: a∈G b∈G c∈G d∈G
and A3: $a = b \cdot c^{-1} \cdot d^{-1}$
shows
$d = b \cdot a^{-1} \cdot c^{-1}$
$d = a^{-1} \cdot b \cdot c^{-1}$
$b = a \cdot d \cdot c$
⟨*proof*⟩

13.5 Translations

In this section we consider translations. Translations are maps $T : G \to G$ of the form $T_g(a) = g \cdot a$ or $T_g(a) = a \cdot g$. We also consider two-dimensional translations $T_g : G \times G \to G \times G$, where $T_g(a, b) = (a \cdot g, b \cdot g)$ or$T_g(a, b) = (g \cdot a, g \cdot b)$.

constdefs
RightTranslation(G,P,g) ≡ {<a,b> ∈ G×G. P<a,g> = b}

LeftTranslation(G,P,g) ≡ {<a,b> ∈ G×G. P<g,a> = b}

RightTranslation2(G,P,g) ≡
{<x,y> ∈ (G×G)×(G×G). ⟨P<fst(x),g>, P<snd(x),g>⟩ = y}

LeftTranslation2(G,P,g) ≡
{<x,y> ∈ (G×G)×(G×G). ⟨P<g,fst(x)>, P<g,snd(x)>⟩ = y}

Translations map G into G. Two dimensional translations map $G \times G$ into itself.

lemma (in group0) group0_5_L1: assumes A1: g∈G
shows RightTranslation(G,f,g) : G→G
LeftTranslation(G,f,g) : G→G
RightTranslation2(G,f,g) : (G×G)→(G×G)
LeftTranslation2(G,f,g) : (G×G)→(G×G)
⟨*proof*⟩

The values of the translations are what we expect.

lemma (in group0) group0_5_L2: assumes A1: g∈G a∈G

shows
RightTranslation(G,f,g)(a) = a·g
LeftTranslation(G,f,g)(a) = g·a
⟨*proof*⟩

The values of the two-dimensional translations are what we expect.

lemma (in group0**)** group0_5_L3: **assumes** A1: g∈G a∈G b∈G
shows RightTranslation2(G,f,g)<a,b> = <a·g,b·g>
LeftTranslation2(G,f,g)<a,b> = <g·a,g·b>
⟨*proof*⟩

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,f,g) T_h = LeftTranslation(G,f,h)
shows $T_g(T_h$(a)) = g·h·a
$T_g(T_h$(a)) = LeftTranslation(G,f,g·h)(a)
⟨*proof*⟩

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 = RightTranslation(G,f,g) T_h = RightTranslation(G,f,h)
shows $T_g(T_h$(a)) = a·h·g
$T_g(T_h$(a)) = RightTranslation(G,f,h·g)(a)
⟨*proof*⟩

The image of a set under a composition of translations is the same as the image under translation by a product.

lemma (in group0**)** group0_5_L6: **assumes** A1: g∈G h∈G **and** A2: A⊆G
and A3: T_g = RightTranslation(G,f,g) T_h = RightTranslation(G,f,h)
shows $T_g(T_h$(A)) = {a·h·g. a∈A}
⟨*proof*⟩

13.6 Odd functions

This section is about odd functions.

Odd functions are those that commute with the group inverse: $f(a^{-1}) = (f(a))^{-1}$.

constdefs

IsOdd(G,P,f) ≡ (∀a∈G. f(GroupInv(G,P)(a)) = 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,f,p) ⟷ (∀a∈G. p(a^{-1}) = (p(a))$^{-1}$)
⟨*proof*⟩

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*⟩

end

14 Group_ZF_1.thy

theory Group_ZF_1 **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 $e \cdot a = a$ and $a \cdot e = a$. 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 neutral elements e is not empty. One way around this is to first use condition A to define the notion of monoid, then prove the uniqueness of e and then use the condition B to define groups. However, there 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 assocative operation "$\cdot$" such that

C: for every $a, b \in G$ the equations $a \cdot x = b$ and $y \cdot a = b$ can be solved in G.

This theory file aims at proving the equivalence of this alternative definition with the usual definition of the group, as formulated in Group_ZF.thy. The romantic proofs come from an Aug. 14, 2005, 2006 post by buli on the matematyka.org forum.

14.1 An alternative definition of group

We will use the multiplicative notation for the group. To do this, we define a context (locale) similar to group0, 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 · 70)
  defines dot_def [simp]: a · b ≡ P<a,b>
```

A set G with an associative operation that satisfies condition C is a group, as defined in Group_ZF theory file.

```
theorem (in group2) Group_ZF_1_T1:
  assumes A1: G≠0 and A2: P {is associative on} G
  and A3: ∀a∈G.∀b∈G. ∃x∈G. a·x = b
  and A4: ∀a∈G.∀b∈G. ∃y∈G. y·a = b
```

shows `IsAgroup(G,P)`
⟨*proof*⟩

end

15 Group_ZF_2.thy

theory Group_ZF_2 **imports** Group_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 qroup.

15.1 Lifting groups to function spaces

If we have a monoid (group) G than we get a monoid (group) structure on a space of functions valued in in G by defining $(f \cdot g)(x) := f(x) \cdot g(x)$. We call this process "lifting the monoid (group) to function space. This section formalizes this "lifting".

The lifted operation is an operation on the function space.

lemma (in monoid0) Group_ZF_2_1_L0A:
  assumes A1: F = f {lifted to function space over} X
  shows F : (X→G)×(X→G)→(X→G)
⟨*proof*⟩

The result of the lifted operation is in the function space.

lemma (in monoid0) Group_ZF_2_1_L0:
  assumes A1:F = f {lifted to function space over} X
  and A2:s:X→G r:X→G
  shows F<s,r> : X→G
⟨*proof*⟩

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→G ∧ (∀s∈X→G. F<E,s> = s ∧ F<s,E> = s)
⟨*proof*⟩

Monoids can be lifted to a function space.

lemma (in monoid0) Group_ZF_2_1_T1:
  assumes A1:F = f {lifted to function space over} X
  shows IsAmonoid(X→G,F)
⟨*proof*⟩

The constant function with the neutral element as the value is the neutral element of the lifted monoid.

lemma Group_ZF_2_1_L2:

assumes A1:IsAmonoid(G,f)
and A2:F = f {lifted to function space over} X
and A3:E = ConstantFunction(X,TheNeutralElement(G,f))
shows E = TheNeutralElement(X→G,F)
⟨*proof*⟩

The lifted operation acts on the functions in a natural way defined by the group operation.

lemma (in group0) Group_ZF_2_1_L3:
assumes A1:F = f {lifted to function space over} X
and A2:s:X→G r:X→G
and A3:x∈X
shows (F<s,r>)(x) = s(x)·r(x)
⟨*proof*⟩

In the group0 context we can apply theorems proven in monoid0 context to the lifted monoid.

lemma (in group0) Group_ZF_2_1_L4:
assumes A1:F = f {lifted to function space over} X
shows monoid0(X→G,F)
⟨*proof*⟩

The compostion of a function $f : X \to G$ with the group inverse is a right inverse for the lifted group. Recall that in the group0 context e is the neutral element of the group.

lemma (in group0) Group_ZF_2_1_L5:
assumes A1: F = f {lifted to function space over} X
and A2: s : X→G
and A3: i = GroupInv(G,f) O s
shows i: X→G F<s,i> = TheNeutralElement(X→G,F)
⟨*proof*⟩

Groups can be lifted to the function space.

theorem (in group0) Group_ZF_2_1_T2:
assumes A1: F = f {lifted to function space over} X
shows IsAgroup(X→G,F)
⟨*proof*⟩

What is the group inverse for the lifted group?

lemma (in group0) Group_ZF_2_1_L6:
assumes A1: F = f {lifted to function space over} X
shows ∀s∈(X→G). GroupInv(X→G,F)(s) = GroupInv(G,f) O s
⟨*proof*⟩

What is the group inverse in a subgroup of the lifted group?

lemma (in group0) Group_ZF_2_1_L6A:
assumes A1: F = f {lifted to function space over} X

and A2: IsAsubgroup(H,F)
and A3: g = restrict(F,H×H)
and A4: s∈H
shows GroupInv(H,g)(s) = GroupInv(G,f) ` s
⟨*proof*⟩

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 = f {lifted to function space over} X
and A2: f {is commutative on} G
shows F {is commutative on} (X→G)
⟨*proof*⟩

15.2 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} ∈ G//r ∧
(∀c ∈ G//r. F<r{e},c> = c ∧ F<c,r{e}> = c)
⟨*proof*⟩

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)
shows IsAmonoid(G//r,F)
⟨*proof*⟩

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*⟩

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,f)
and A3: F = ProjFun2(G,r,f)
and A4: a∈G b∈G
shows F<r{a},r{b}> = r{a·b}
⟨*proof*⟩

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,f)
and A3: F = ProjFun2(G,r,f)
and A4: a∈G
shows F⟨r{a},r{a^{-1}}⟩ = TheNeutralElement(G//r,F)
⟨*proof*⟩

The group structure can be projected to the quotient space.

theorem (in group0) Group_ZF_3_T2:
assumes A1: equiv(G,r) **and** A2: Congruent2(r,f)
shows IsAgroup(G//r,ProjFun2(G,r,f))
⟨*proof*⟩

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: equiv(G,r) **and**
A2: Congruent2(r,f) **and**
A3: F = ProjFun2(G,r,f) **and**
A4: a∈G
shows r{a^{-1}} = GroupInv(G//r,F)(r{a})
⟨*proof*⟩

15.3 Normal subgroups and quotient groups

A normal subgrup N of a group G is such that aba^{-1} belongs to N if $a \in G, b \in N$. Having a group and a normal subgroup N we can create another group consisting of eqivalence 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.

constdefs
IsAnormalSubgroup(G,f,N) ≡ IsAsubgroup(N,f) ∧
(∀n∈N.∀g∈G. f< f< g,n >,GroupInv(G,f)(g) > ∈ N)

QuotientGroupRel(G,f,H) ≡
{<a,b> ∈ G×G. f<a, GroupInv(G,f)(b)> ∈ H}

QuotientGroupOp(G,f,H) ≡ ProjFun2(G,QuotientGroupRel(G,f,H),f)

Definition of a normal subgroup in a more readable notation.

lemma (in group0) Group_ZF_2_4_L0:

assumes IsAnormalSubgroup(G,f,H)
and g∈G n∈H
shows g·n·g^{-1} ∈ H
⟨*proof*⟩

The quotient group relation is reflexive.

lemma (in group0**)** Group_ZF_2_4_L1:
assumes IsAsubgroup(H,f)
shows refl(G,QuotientGroupRel(G,f,H))
⟨*proof*⟩

The quotient group relation is symmetric.

lemma (in group0**)** Group_ZF_2_4_L2:
assumes A1:IsAsubgroup(H,f)
shows sym(QuotientGroupRel(G,f,H))
⟨*proof*⟩

The quotient group relation is transistive.

lemma (in group0**)** Group_ZF_2_4_L3A:
assumes A1: IsAsubgroup(H,f) **and**
A2: <a,b> ∈ QuotientGroupRel(G,f,H) **and**
A3: <b,c> ∈ QuotientGroupRel(G,f,H)
shows <a,c> ∈ QuotientGroupRel(G,f,H)
⟨*proof*⟩

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,f)
shows equiv(G,QuotientGroupRel(G,f,H))
⟨*proof*⟩

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,f,H)
and A2:⟨a1,a2⟩ ∈ QuotientGroupRel(G,f,H)
and A3:⟨b1,b2⟩ ∈ QuotientGroupRel(G,f,H)
shows ⟨a1·b1, a2·b2⟩ ∈ QuotientGroupRel(G,f,H)
⟨*proof*⟩

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,f)
and IsAnormalSubgroup(G,f,H)
shows Congruent2(QuotientGroupRel(G,f,H),f)
⟨*proof*⟩

The quotient group is indeed a group.

theorem Group_ZF_2_4_T1:
assumes IsAgroup(G,f) **and** IsAnormalSubgroup(G,f,H)
shows
IsAgroup(G//QuotientGroupRel(G,f,H),QuotientGroupOp(G,f,H))
⟨*proof*⟩

The class (coset)of the neutral element is the neutral element of the quotient group.

lemma Group_ZF_2_4_L5B:
assumes IsAgroup(G,f) **and** IsAnormalSubgroup(G,f,H)
and r = QuotientGroupRel(G,f,H)
and e = TheNeutralElement(G,f)
shows r{e} = TheNeutralElement(G//r,QuotientGroupOp(G,f,H))
⟨*proof*⟩

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,f,H) ⟷ a∈H
⟨*proof*⟩

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,f,H) **and**
A2: a∈G **and**
A3: r = QuotientGroupRel(G,f,H) **and**
A4: TheNeutralElement(G//r,QuotientGroupOp(G,f,H)) = e
shows r{a} = e ⟷ ⟨a,**1**⟩ ∈ r
⟨*proof*⟩

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,f,H) **and**
a∈G **and** r = QuotientGroupRel(G,f,H) **and**
TheNeutralElement(G//r,QuotientGroupOp(G,f,H)) = e
shows r{a} = e ⟷ a∈H
⟨*proof*⟩

Essential condition to show that every subgroup of an abelian group is normal.

lemma (in group0) Group_ZF_2_4_L5:
assumes A1:f {is commutative on} G
and A2:IsAsubgroup(H,f)
and A3:g∈G h∈H
shows g·h·g^{-1} ∈ H
⟨*proof*⟩

Every subgroup of an abelian group is normal. Moreover, the quotient group is also abelian.

lemma Group_ZF_2_4_L6:
  assumes A1: IsAgroup(G,f)
  and A2: f {is commutative on} G
  and A3: IsAsubgroup(H,f)
  shows IsAnormalSubgroup(G,f,H)
  QuotientGroupOp(G,f,H) {is commutative on} (G//QuotientGroupRel(G,f,H))
⟨*proof*⟩

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,f,H)
  and a∈G **and** r = QuotientGroupRel(G,f,H)
  and F = QuotientGroupOp(G,f,H)
  shows r{a^{-1}} = GroupInv(G//r,F)(r{a})
  ⟨*proof*⟩

15.4 Function spaces as monoids

On every space of functions $\{f : X \to X\}$ 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*⟩

The space of functions that map a set X into itsef 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*⟩

This concludes Group_ZF_2 theory.

end

16 Group_ZF_3.thy

theory Group_ZF_3 **imports** Group_ZF_2 Finite1

begin

In this theory we consider notions in group theory that are useful for the construction of real numbers in the Real_ZF_x series of theories.

16.1 Group valued finite range functions

In this section show that the group valued functions $f : X \to G$, 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_x.thy 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 = f {lifted to function space over} X
  and
  A2:s ∈ FinRangeFunctions(X,G) r ∈ FinRangeFunctions(X,G)
  shows F<s,r> ∈ FinRangeFunctions(X,G)
⟨*proof*⟩

The set of group valued finite range functions is closed with respect to the lifted group operation.

lemma (in group0) Group_ZF_3_1_L2:
  assumes A1: F = f {lifted to function space over} X
  shows FinRangeFunctions(X,G) {is closed under} F
⟨*proof*⟩

A composition of a finite range function with the group inverse is a finite range function.

lemma (in group0) Group_ZF_3_1_L3:
  assumes A1: s ∈ FinRangeFunctions(X,G)
  shows GroupInv(G,f) O s ∈ FinRangeFunctions(X,G)
  ⟨*proof*⟩

The set of finite range functions is s subgroup of the lifted group.

theorem Group_ZF_3_1_T1:
  assumes A1:IsAgroup(G,f)
  and A2:F = f {lifted to function space over} X
  and A3:X≠0
  shows IsAsubgroup(FinRangeFunctions(X,G),F)
⟨*proof*⟩

16.2 Almost homomorphisms

An almost homomorphism is a group valued function defined on a monoid M with the property that the set $\{f(m+n)-f(m)-f(n)\}_{m,n\in M}$ is finite. This term is used by R. D. Arthan in "The Eudoxus Real Numbers". We use this term in the general group context and use the A'Campo's term "slopes" (see his "A natural construction for the real numbers") to mean an almost homomorphism mapping interegers 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. Almost homomorphisms are defined as those maps $s : G \to G$ such that the homomorphism difference takes only finite number of values on $G \times G$. Although almost homomorphisms can be in principle defined on a monoid with values in a group, we limit ourselves to the situation where the monoid and the group are the same. The set of slopes related to a specific group is called AlmostHoms(G, f). AlHomOp1(G, f) is the group operation on almost homomorphisms defined in a natural way by $(s \cdot r)(n) = s(n) \cdot r(n)$. In the terminology defined in func1.thy this is the group operation f (on G) lifted to the function space $G \to G$ and restricted to the set 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_x.thy series.

constdefs

```
  HomDiff(G,f,s,x) ≡
  f⟨s(f<fst(x),snd(x)>) ,
  (GroupInv(G,f)(f<s(fst(x)),s(snd(x))>))⟩

  AlmostHoms(G,f) ≡
  {s ∈ G→G.{HomDiff(G,f,s,x). x ∈ G×G } ∈ Fin(G)}

  AlHomOp1(G,f) ≡
  restrict(f {lifted to function space over} G,
  AlmostHoms(G,f)×AlmostHoms(G,f))

  AlHomOp2(G,f) ≡
  restrict(Composition(G),AlmostHoms(G,f)×AlmostHoms(G,f))
```

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`**)** `Group_ZF_3_2_L1`**:**

shows HomDiff(G,f,s,<m,n>) = s(m·n)·(s(m)·s(n))$^{-1}$
⟨*proof*⟩

The next lemma shows the set from the definition of almost homomorphism in a different form.

lemma (in group0) Group_ZF_3_2_L1A:
{HomDiff(G,f,s,x). x ∈ G×G } = {s(m·n)·(s(m)·s(n))$^{-1}$. <m,n> ∈ G×G}
⟨*proof*⟩

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, $(\sim p)(n) = p(n)^{-1}$. δ 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 $\{s(n) \cdot r(n)^{-1} : n \in G\}$ 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: f {is commutative on} G

fixes AH
defines AH_def [simp]: AH ≡ AlmostHoms(G,f)

fixes Op1
defines Op1_def [simp]: Op1 ≡ AlHomOp1(G,f)

fixes Op2
defines Op2_def [simp]: Op2 ≡ AlHomOp2(G,f)

fixes FR
defines FR_def [simp]: FR ≡ FinRangeFunctions(G,G)

fixes neg :: i⇒i (∼_ [90] 91)
defines neg_def [simp]: ∼s ≡ GroupInv(G,f) O s

fixes δ
defines δ_def [simp]: δ(s,x) ≡ HomDiff(G,f,s,x)

fixes AHprod (**infix** · 69)
defines AHprod_def [simp]: s · r ≡ AlHomOp1(G,f)<s,r>

fixes AHcomp (**infix** ∘ 70)
defines AHcomp_def [simp]: s ∘ r ≡ AlHomOp2(G,f)<s,r>

fixes AlEq (**infix** ≈ 68)

defines AlEq_def [simp]:
s ≈ r ≡ <s,r> ∈ QuotientGroupRel(AH,Op1,FR)

HomDiff is a homomorphism on the lifted group structure.

lemma (in group1) Group_ZF_3_2_L1:
assumes A1: s:G→G r:G→G
and A2: x ∈ G×G
and A3: F = f {lifted to function space over} G
shows δ(F<s,r>,x) = δ(s,x)·δ(r,x)
⟨*proof*⟩

The group operation lifted to the function space over G preserves almost homomorphisms.

lemma (in group1) Group_ZF_3_2_L2: **assumes** A1: s ∈ AH r ∈ AH
and A2: F = f {lifted to function space over} G
shows F<s,r> ∈ AH
⟨*proof*⟩

The set of almost homomorphisms is closed under the lifted group operation.

lemma (in group1) Group_ZF_3_2_L3:
assumes F = f {lifted to function space over} G
shows AH {is closed under} F
⟨*proof*⟩

The terms in the homomorphism difference for a function are in the group.

lemma (in group1) Group_ZF_3_2_L4:
assumes s:G→G **and** m∈G n∈G
shows
m·n ∈ G
s(m·n) ∈ G
s(m) ∈ G s(n) ∈ G
δ(s,<m,n>) ∈ G
s(m)·s(n) ∈ G
⟨*proof*⟩

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 ∈ AH **and** m∈G n∈G
shows m·n ∈ G
s(m·n) ∈ G
s(m) ∈ G s(n) ∈ G
δ(s,<m,n>) ∈ G
s(m)·s(n) ∈ G
⟨*proof*⟩

The terms in the homomorphism difference are in the group, a different form.

lemma (in group1**)** Group_ZF_3_2_L4B:
assumes A1:s ∈ AH **and** A2:x∈G×G
shows fst(x)·snd(x) ∈ G
s(fst(x)·snd(x)) ∈ G
s(fst(x)) ∈ G s(snd(x)) ∈ G
δ(s,x) ∈ G
s(fst(x))·s(snd(x)) ∈ G
⟨*proof*⟩

What are the values of the inverse of an almost homomorphism?

lemma (in group1**)** Group_ZF_3_2_L5:
assumes s ∈ AH **and** n∈G
shows (∼s)(n) = (s(n))$^{-1}$
⟨*proof*⟩

Homomorphism difference commutes with the inverse for almost homomorphisms.

lemma (in group1**)** Group_ZF_3_2_L6:
assumes A1:s ∈ AH **and** A2:x∈G×G
shows δ(∼s,x) = (δ(s,x))$^{-1}$
⟨*proof*⟩

The inverse of an almost homomorphism maps the group into itself.

lemma (in group1**)** Group_ZF_3_2_L7:
assumes s ∈ AH
shows ∼s : G→G
⟨*proof*⟩

The inverse of an almost homomorphism is an almost homomorphism.

lemma (in group1**)** Group_ZF_3_2_L8:
assumes A1: F = f {lifted to function space over} G
and A2: s ∈ AH
shows GroupInv(G→G,F)(s) ∈ AH
⟨*proof*⟩

The function that assigns the neutral element everywhere is an almost homomorphism.

lemma (in group1**)** Group_ZF_3_2_L9:
ConstantFunction(G,**1**) ∈ AH
AH≠0
⟨*proof*⟩

If the group is abelian, then almost homomorphisms form a subgroup of the lifted group.

lemma Group_ZF_3_2_L10:
assumes A1: IsAgroup(G,f)
and A2: f {is commutative on} G

and A3: F = f {lifted to function space over} G
shows IsAsubgroup(AlmostHoms(G,f),F)
⟨*proof*⟩

If the group is abelian, then almost homomorphisms form a group with the first operation, hence we can use theorems proven in group0 context aplied to this group.

lemma (in group1) Group_ZF_3_2_L10A:
shows IsAgroup(AH,Op1) group0(AH,Op1)
⟨*proof*⟩

The group of almost homomorphisms is abelian

lemma Group_ZF_3_2_L11: **assumes** A1: IsAgroup(G,f)
and A2: f {is commutative on} G
shows
IsAgroup(AlmostHoms(G,f),AlHomOp1(G,f))
AlHomOp1(G,f) {is commutative on} AlmostHoms(G,f)
⟨*proof*⟩

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)
⟨*proof*⟩

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,f) O s
GroupInv(AH,Op1)(s) ∈ AH
GroupInv(G,f) O s ∈ AH
⟨*proof*⟩

The group inverse in the group of almost homomorphisms acts in a natural way on its operand.

lemma (in group1) Group_ZF_3_2_L14:
assumes s∈AH **and** n∈G
shows (GroupInv(AH,Op1)(s))(n) = $(s(n))^{-1}$
⟨*proof*⟩

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** IsAgroup(G,f)
and f {is commutative on} G
and AH = AlmostHoms(G,f) Op1 = AlHomOp1(G,f)
and s ∈ AH r ∈ AH

shows
Op1<s,r> ∈ AH
GroupInv(AH,Op1)(r) ∈ AH
Op1<s,GroupInv(AH,Op1)(r)> ∈ AH
⟨*proof*⟩

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 (pointwise) inverse of an almost homomorphism is an almost homomorphism.

corollary (in group1) Group_ZF_3_2_L16: assumes s ∈ AH r ∈ AH
shows s·r ∈ AH s·(∼r) ∈ AH
⟨*proof*⟩

16.3 The classes of almost homomorphisms

In the Real_ZF_x 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: FR ⊆ AH
⟨*proof*⟩

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*⟩

The group of almost homomorphisms divided by the subgroup of finite range functions is an abelian group.

theorem (in group1) Group_ZF_3_3_T1:
shows
IsAgroup(AH//QuotientGroupRel(AH,Op1,FR),QuotientGroupOp(AH,Op1,FR))
and
QuotientGroupOp(AH,Op1,FR) {is commutative on}
(AH//QuotientGroupRel(AH,Op1,FR))
⟨*proof*⟩

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:

```
QuotientGroupRel(AH,Op1,FR) ⊆ AH × AH
equiv(AH,QuotientGroupRel(AH,Op1,FR))
```
⟨*proof*⟩

The "almost equal" relation is symmetric.

lemma (in group1**)** Group_ZF_3_3_L3A: **assumes** A1: s≈r
shows r≈s
⟨*proof*⟩

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)
⟨*proof*⟩

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 ∈ AH **and**
r = QuotientGroupRel(AH,Op1,FR) **and**
TheNeutralElement(AH//r,QuotientGroupOp(AH,Op1,FR)) = e
shows r{s} = e ⟷ s ∈ FR
⟨*proof*⟩

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 ∈ AH **and**
r = QuotientGroupRel(AH,Op1,FR) **and**
F = ProjFun2(AH,r,Op1)
shows r{∼s} = GroupInv(AH//r,F)(r{s})
⟨*proof*⟩

16.4 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.thy serias. 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∈AH **and** m∈G n∈G
shows s(m·n) = s(m)·s(n)·δ(s,<m,n>)
⟨*proof*⟩

What is the value of a composition of almost homomorhisms?

lemma (in group1**)** Group_ZF_3_4_L2:
assumes s∈AH r∈AH **and** m∈G
shows (s∘r)(m) = s(r(m)) s(r(m)) ∈ G
⟨*proof*⟩

What is the homomorphism difference of a composition?

lemma (in group1**)** Group_ZF_3_4_L3:
assumes A1: s∈AH r∈AH **and** A2: m∈G n∈G
shows δ(s∘r,<m,n>) =
δ(s,<r(m),r(n)>)·s(δ(r,<m,n>))·δ(s,<r(m)·r(n),δ(r,<m,n>)>)
⟨*proof*⟩

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∈AH r∈AH **and** A2: x ∈ G×G
and A3:
A = δ(s,<r(fst(x)),r(snd(x))>)
B = s(δ(r,x))
C = δ(s,<(r(fst(x))·r(snd(x))),δ(r,x)>)
shows δ(s∘r,x) = A·B·C
⟨*proof*⟩

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∈AH r∈AH
shows {δ(Composition(G)<s,r>,x). x ∈ G×G} ∈ Fin(G)
⟨*proof*⟩

Composition of almost homomorphisms is an almost homomorphism.

theorem (in group1**)** Group_ZF_3_4_T1:
assumes A1: s∈AH r∈AH
shows Composition(G)<s,r> ∈ AH s∘r ∈ AH
⟨*proof*⟩

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,f) {is associative on} AH
⟨*proof*⟩

Type information related to the situation of two almost homomorphisms.

lemma (in group1**)** Group_ZF_3_4_L7:
assumes A1: s∈AH r∈AH **and** A2: n∈G
shows
s(n) ∈ G $(r(n))^{-1}$ ∈ G
$s(n)\cdot(r(n))^{-1}$ ∈ G s(r(n)) ∈ G
⟨*proof*⟩

Type information related to the situation of three almost homomorphisms.

lemma (in group1**)** Group_ZF_3_4_L8:
assumes A1: s∈AH r∈AH q∈AH **and** A2: n∈G
shows
q(n)∈G
s(r(n)) ∈ G
$r(n)\cdot(q(n))^{-1}$ ∈ G
$s(r(n)\cdot(q(n))^{-1})$ ∈ G
δ(s,<q(n),$r(n)\cdot(q(n))^{-1}$>) ∈ G
⟨*proof*⟩

A formula useful in showing that the composition of almost homomorphisms is congruent with respect to the quotient group relation.

lemma (in group1**)** Group_ZF_3_4_L9:
assumes A1: s1 ∈ AH r1 ∈ AH s2 ∈ AH r2 ∈ AH
and A2: n∈G
shows $(s1\circ r1)(n)\cdot((s2\circ r2)(n))^{-1}$ =
$s1(r2(n))\cdot (s2(r2(n)))^{-1}\cdot s1(r1(n)\cdot(r2(n))^{-1})\cdot$
δ(s1,<r2(n),$r1(n)\cdot(r2(n))^{-1}$>)
⟨*proof*⟩

The next lemma shows a formula that translates an expression in terms of the first group operation on almost homomorphisms and the group inverse in the group of almost homomorphisms to an expression using only the underlying group operations.

lemma (in group1**)** Group_ZF_3_4_L10: **assumes** A1: s ∈ AH r ∈ AH
and A2: n ∈ G
shows $(s\cdot(\text{GroupInv}(AH,Op1)(r)))(n) = s(n)\cdot(r(n))^{-1}$
⟨*proof*⟩

A neccessary condition for two a. h. to be almost equal.

lemma (in group1**)** Group_ZF_3_4_L11:
assumes A1: s≈r
shows $\{s(n)\cdot(r(n))^{-1}.\ n\in G\}$ ∈ Fin(G)
⟨*proof*⟩

A sufficient condition for two a. h. to be almost equal.

lemma (in group1**)** Group_ZF_3_4_L12: **assumes** A1: s∈AH r∈AH
  and A2: {s(n)·(r(n))$^{-1}$. n∈G} ∈ Fin(G)
  shows s≈r
⟨*proof*⟩

Another sufficient consdition for two a.h. to be almost equal. It is actually just an expansion of the definition of the quotient group relation.

lemma (in group1**)** Group_ZF_3_4_L12A: **assumes** s∈AH r∈AH
  and s·(GroupInv(AH,Op1)(r)) ∈ FR
  shows s≈r r≈s
⟨*proof*⟩

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≈r
  shows s·(GroupInv(AH,Op1)(r)) ∈ FR
  ⟨*proof*⟩

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≈s2 r1≈r2
  shows (s1∘r1) ≈ (s2∘r2)
⟨*proof*⟩

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 "∘" 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:
  Congruent2(QuotientGroupRel(AH,Op1,FR),Op2)
⟨*proof*⟩

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 ∈ G×G
  shows δ(id(G),x) = **1**
⟨*proof*⟩

The identity function ($I(x) = x$) on G is an almost homomorphism.

lemma (in group1**)** Group_ZF_3_4_L15: id(G) ∈ AH
⟨*proof*⟩

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
IsAmonoid(AH,Op2)
monoid0(AH,Op2)
id(G) = TheNeutralElement(AH,Op2)
⟨*proof*⟩

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 = QuotientGroupRel(AH,Op1,FR)
shows
IsAmonoid(AH//R,ProjFun2(AH,R,Op2))
R{id(G)} = TheNeutralElement(AH//R,ProjFun2(AH,R,Op2))
⟨*proof*⟩

16.5 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 ∈ AH **and** A2: c∈G **and**
A3: r = {⟨x,s(x)·c⟩. x∈G}
shows
∀x∈G. r(x) = s(x)·c
r ∈ AH
s ≈ r
⟨*proof*⟩

end

17 OrderedGroup_ZF.thy

theory OrderedGroup_ZF **imports** Group_ZF Order_ZF Finite_ZF_1

begin

This theory file defines and shows the basic properties of (partially or linearly) ordered groups. We define the set of nonnegative elements and the absolute value function. 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.thy that subsets of integers are bounded iff they are finite.

17.1 Ordered groups

This section defines ordered groups.

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 $g \cdot a \leq g \cdot b$. We define the set of nonnegative elements in the obvious way as $G^+ = \{x \in G : 1 \leq x\}$. G_+ is a similar set, but without the unit. 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. We also define the maximum absolute value of a set, that is the maximum of the set $\{|x|.x \in A\}$. The odd functions are defined as those having property $f(a^{-1}) = (f(a))^{-1}$. Looks a bit strange in the multiplicative notation. For linearly oredered 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.

constdefs

```
IsAnOrdGroup(G,P,r) ≡
(IsAgroup(G,P) ∧ r⊆G×G ∧ IsPartOrder(G,r) ∧ (∀g∈G. ∀a b.
<a,b> ∈ r ⟶ <P<a,g>,P<b,g> > ∈ r ∧ <P<g,a>,P<g,b> > ∈ r ) )

Nonnegative(G,P,r) ≡ {x∈G. <TheNeutralElement(G,P),x> ∈ r}

PositiveSet(G,P,r) ≡
{x∈G. <TheNeutralElement(G,P),x> ∈ r ∧ TheNeutralElement(G,P)≠ x}

AbsoluteValue(G,P,r) ≡ id(Nonnegative(G,P,r)) ∪
restrict(GroupInv(G,P),G - Nonnegative(G,P,r))

OddExtension(G,P,r,f) ≡
(f ∪ {⟨a, GroupInv(G,P)(f(GroupInv(G,P)(a)))⟩.
a ∈ GroupInv(G,P)(PositiveSet(G,P,r))} ∪
{⟨TheNeutralElement(G,P),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$^\circ$ 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** ≡ 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)

fixes lesseq (**infix** ≤ 68)
defines lesseq_def [simp]: a ≤ b ≡ <a,b> ∈ r

fixes sless (**infix** < 68)
defines sless_def [simp]: a < b ≡ a≤b ∧ a≠b

fixes nonnegative (G$^+$)
defines nonnegative_def [simp]: G$^+$ ≡ Nonnegative(G,P,r)

fixes positive (G$_+$)
defines nonnegative_def [simp]: G$_+$ ≡ PositiveSet(G,P,r)

fixes setinv :: i⇒i (- _ 72)
defines setninv_def [simp]: -A ≡ GroupInv(G,P)(A)

fixes abs (| _ |)
defines abs_def [simp]: |a| ≡ AbsoluteValue(G,P,r)(a)

fixes oddext (_ $^\circ$)
defines oddext_def [simp]: f$^\circ$ ≡ 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)
⟨*proof*⟩

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
⟨*proof*⟩

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
⟨*proof*⟩

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)**
⟨*proof*⟩

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*⟩

The group order is reflexive.

lemma (in group3) OrderedGroup_ZF_1_L3: assumes g∈G
shows g≤g
⟨*proof*⟩

1 is nonnegative.

lemma (in group3) OrderedGroup_ZF_1_L3A: shows 1∈G^+
⟨*proof*⟩

In this context $a \leq b$ implies that both a and b belong to G.

lemma (in group3) OrderedGroup_ZF_1_L4:
assumes a≤b **shows** a∈G b∈G
⟨*proof*⟩

It is good to have transitivity handy.

lemma (in group3) Group_order_transitive:
assumes A1: a≤b b≤c **shows** a≤c
⟨*proof*⟩

The order in an ordered group is antisymmetric.

lemma (in group3) group_order_antisym:
assumes A1: a≤b b≤a **shows** a=b
⟨*proof*⟩

Transitivity for the strict order: if $a < b$ and $b \leq c$, then $a < c$.

lemma (in group3) OrderedGroup_ZF_1_L4A:
assumes A1: a<b **and** A2: b≤c

shows a<c
⟨*proof*⟩

Another version of transitivity for the strict order: if $a \leq b$ and $b < c$, then $a < c$.

lemma (in group3**)** group_strict_ord_transit:
assumes A1: a≤b **and** A2: b<c
shows a<c
⟨*proof*⟩

Strict order is preserved by translations.

lemma (in group3**)** group_strict_ord_transl_inv:
assumes a<b **and** c∈G
shows
a·c < b·c
c·a < c·b
⟨*proof*⟩

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)
⟨*proof*⟩

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∈G^+ **and** b ∈ G-G^+
shows b≤a
⟨*proof*⟩

If $a \leq 1$ and $a \neq 1$, then $a \in G \setminus G^+$.

lemma (in group3**)** OrderedGroup_ZF_1_L4C:
assumes A1: a≤**1** **and** A2: a≠**1**
shows a ∈ G-G^+
⟨*proof*⟩

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∈G-G^+ **and** A2: b≤a
shows b∈G-G^+
⟨*proof*⟩

The nonnegative set is contained in the group.

lemma (in group3**)** OrderedGroup_ZF_1_L4E: **shows** $G^+ \subseteq$ G
⟨*proof*⟩

Taking the inverse on both sides reverses the inequality.

lemma (in group3**)** OrderedGroup_ZF_1_L5:
assumes A1: a≤b **shows** $b^{-1} \leq a^{-1}$
⟨*proof*⟩

If an element is smaller that the unit, then its inverse is greater.

lemma (in group3**)** OrderedGroup_ZF_1_L5A:
assumes A1: a≤**1** **shows** **1**$\leq a^{-1}$
⟨*proof*⟩

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∈G **and** A2: **1**$\leq a^{-1}$
shows a≤**1**
⟨*proof*⟩

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**≤a **shows** $a^{-1} \leq$ **1** **and** ¬(a≤**1** ∧ a≠**1**)
⟨*proof*⟩

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**≤a **1**≤b
shows $a^{-1} \leq b$
⟨*proof*⟩

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 $b \leq a^{-1}$
⟨*proof*⟩

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*⟩

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: $a \cdot b^{-1} \leq a \cdot c^{-1}$
shows c≤b
⟨*proof*⟩

Taking negative on both sides reverses the inequality, another case with an inverse on one side.

lemma (in group3**)** OrderedGroup_ZF_1_L5AG:
assumes A1: a ∈ G **and** A2: $a^{-1} \leq b$
shows $b^{-1} \leq a$
⟨*proof*⟩

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*⟩

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≤b_1
shows a≤b_1·c
⟨*proof*⟩

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≤b_1
shows a ≤ b·b_1
⟨*proof*⟩

We can replace factors on one side of an inequality with greater ones.

lemma (in group3**)** OrderedGroup_ZF_1_L5E:
assumes A1: a ≤ b·c **and** A2: b≤b_1 c≤c_1
shows a ≤ b_1·c_1
⟨*proof*⟩

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**≤a **and** A2: b∈G
shows b≤a·b b≤b·a
⟨*proof*⟩

We can multiply the right hand side of an inequality by a nonnegative element.

lemma (in group3**)** OrderedGroup_ZF_1_L5G: **assumes** A1: a≤b

and A2: $1 \leq c$ **shows** $a \leq b \cdot c$ $a \leq c \cdot b$
⟨*proof*⟩

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*⟩

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 $a \cdot d^{-1} \leq b \cdot c^{-1}$
⟨*proof*⟩

We can put an element on the other side of an inequality changing its sign, version with the inverse.

lemma (in group3**)** OrderedGroup_ZF_1_L5J:
assumes A1: $a \in G$ $b \in G$ **and** A2: $c \leq a \cdot b^{-1}$
shows $c \cdot b \leq a$
⟨*proof*⟩

We can put an element on the other side of an inequality changing its sign, version with the inverse.

lemma (in group3**)** OrderedGroup_ZF_1_L5JA:
assumes A1: $a \in G$ $b \in G$ **and** A2: $c \leq a^{-1} \cdot b$
shows $a \cdot c \leq b$
⟨*proof*⟩

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: $1 \leq a \cdot b^{-1}$
shows $b \leq a$
⟨*proof*⟩

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*⟩

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∈G-G^+
shows a≤**1** $a^{-1} \in G^+$ restrict(GroupInv(G,P),G-G^+)(a) ∈ G^+
⟨*proof*⟩

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: ∀a∈G^+.∀b∈G^+. Q(a,b)
and A3: ∀a∈G.∀b∈G. Q(a,b)⟶Q(a^{-1},b)
and A4: ∀a∈G.∀b∈G. Q(a,b)⟶Q(a,b^{-1})
and A5: a∈G b∈G
shows Q(a,b)
⟨*proof*⟩

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∈G b∈G
shows
1≤a ∧ **1**≤b ∨ a≤**1** ∧ b≤**1** ∨
a≤**1** ∧ **1**≤b ∧ **1** ≤ a·b ∨ a≤**1** ∧ **1**≤b ∧ a·b ≤ **1** ∨
1≤a ∧ b≤**1** ∧ **1** ≤ a·b ∨ **1**≤a ∧ b≤**1** ∧ a·b ≤ **1**
⟨*proof*⟩

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 $a^{-1} \leq b^{-1}$ a≠b b<a

⟨*proof*⟩

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*⟩

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*⟩

A negative element can not be nonnegative.

lemma (in group3**)** OrderedGroup_ZF_1_L8B:
assumes A1: $a \leq \mathbf{1}$ **and** A2: $a \neq \mathbf{1}$ **shows** $\neg(\mathbf{1} \leq a)$
⟨*proof*⟩

An element is greater or equal than another iff the difference is nonpositive.

lemma (in group3**)** OrderedGroup_ZF_1_L9:
assumes A1: $a \in G$ $b \in G$
shows $a \leq b \longleftrightarrow a \cdot b^{-1} \leq \mathbf{1}$
⟨*proof*⟩

We can move an element to the other side of an inequality.

lemma (in group3**)** OrderedGroup_ZF_1_L9A:
assumes A1: $a \in G$ $b \in G$ $c \in G$
shows $a \cdot b \leq c \longleftrightarrow a \leq c \cdot b^{-1}$
⟨*proof*⟩

A one side version of the previous lemma with weaker assuptions.

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*⟩

We can put en 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*⟩

If an element is greater or equal than another then the difference is nonnegative.

lemma (in group3**)** OrderedGroup_ZF_1_L9D: **assumes** A1: $a \leq b$
shows $\mathbf{1} \leq b \cdot a^{-1}$
⟨*proof*⟩

If an element is greater than another then the difference is positive.

lemma (in group3**)** OrderedGroup_ZF_1_L9E:
assumes A1: $a \leq b$ $a \neq b$
shows $\mathbf{1} \leq b \cdot a^{-1}$ $\mathbf{1} \neq b \cdot a^{-1}$ $b \cdot a^{-1} \in G_+$
⟨*proof*⟩

If the difference is nonnegative, then $a \leq b$.

lemma (in group3**)** OrderedGroup_ZF_1_L9F:
assumes A1: $a \in G$ $b \in G$ **and** A2: $\mathbf{1} \leq b \cdot a^{-1}$
shows $a \leq b$

⟨*proof*⟩

If we increase the middle term in a product, the whole product increases.

lemma (in group3**)** OrderedGroup_ZF_1_L10**:**
assumes a∈G b∈G **and** c≤d
shows a·c·b ≤ a·d·b
⟨*proof*⟩

A product of (strictly) positive elements is not the unit.

lemma (in group3**)** OrderedGroup_ZF_1_L11**:**
assumes A1**:** **1**≤a **1**≤b
and A2**:** **1** ≠ a **1** ≠ b
shows **1** ≠ a·b
⟨*proof*⟩

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*⟩

If a is not greater than b, then 1 is not greater than $b \cdot a^{-1}$.

lemma (in group3**)** OrderedGroup_ZF_1_L12A**:**
assumes A1**:** a≤b **shows** **1** ≤ b·a^{-1}
⟨*proof*⟩

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*⟩

We can multiply the sides of two inequalities, first of them strict and we get a strict inequality.

lemma (in group3**)** OrderedGroup_ZF_1_L12C**:**
assumes A1**:** a<b **and** A2**:** c≤d
shows a·c < b·d
⟨*proof*⟩

We can multiply the sides of two inequalities, second of them strict and we get a strict inequality.

lemma (in group3**)** OrderedGroup_ZF_1_L12D**:**
assumes A1**:** a≤b **and** A2**:** c<d
shows a·c < b·d
⟨*proof*⟩

17.2 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: G_+ **{is closed under}** P
⟨*proof*⟩

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 $a = \mathbf{1} \vee a \in G_+ \vee a^{-1} \in G_+$
⟨*proof*⟩

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** $a \neq \mathbf{1}$ $a^{-1} \notin G_+$
⟨*proof*⟩

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** $a \neq \mathbf{1}$ $a \notin G_+$
⟨*proof*⟩

For linearly ordered groups each element is either the unit, positive or its inverse is positive.

lemma (in group3) OrdGroup_decomp:
assumes A1: r **{is total on}** G **and A2:** $a \in G$
shows Exactly_1_of_3_holds $(a = \mathbf{1}, a \in G_+, a^{-1} \in G_+)$
⟨*proof*⟩

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 **{is total on}** G **and A2:** $a \in G$
and A3: $a \neq \mathbf{1}$ $a \notin G_+$
shows $a^{-1} \in G_+$
⟨*proof*⟩

Elements from $G \setminus G_+$ are not greater that the unit.

lemma (in group3) OrderedGroup_ZF_1_L17:
assumes A1: r {is total on} G **and** A2: a ∈ G-G_+
shows a≤**1**
⟨*proof*⟩

The next lemma allows to split proofs that something holds for all $a \in G$ into cases $a = 1$, $a \in G_+$, $-a \in G_+$.

lemma (in group3) OrderedGroup_ZF_1_L18:
assumes A1: r {is total on} G **and** A2: b∈G
and A3: Q(**1**) **and** A4: ∀a∈G_+. Q(a) **and** A5: ∀a∈G_+. Q(a^{-1})
shows Q(b)
⟨*proof*⟩

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*⟩

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*⟩

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*⟩

If $b \in G_+$, then $a < a \cdot b$. Multiplying a by a positive elemnt 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*⟩

If G is a nontrivial linearly ordered hroup, 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
shows ∃b∈G_+. a≤b
⟨*proof*⟩

The G^+ is G_+ plus the unit.

lemma (in group3) OrderedGroup_ZF_1_L24: shows $G^+ = G_+ \cup \{1\}$
⟨*proof*⟩

What is $-G_+$, really?

lemma (in group3) OrderedGroup_ZF_1_L25: shows
$(-G_+) = \{a^{-1}.\ a \in G_+\}$
$(-G_+) \subseteq G$
⟨*proof*⟩

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*⟩

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 $a^{-1} \in G_+$
⟨*proof*⟩

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*⟩

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 prescibing 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*⟩

A special case of OrderedGroup_ZF_1_L28 when $a \cdot b^{-1}$ is positive.

corollary (in group3) OrderedGroup_ZF_1_L29:
assumes A1: $a \in G$ $b \in G$ **and** A2: $a \cdot b^{-1} \in G_+$
shows $b \leq a$ $b \neq a$
⟨*proof*⟩

A bit stronger that OrderedGroup_ZF_1_L29, adds case when two elements are equal.

lemma (in group3) OrderedGroup_ZF_1_L30:

assumes a∈G b∈G **and** a=b ∨ b·a^{-1} ∈ G$_+$
shows a≤b
⟨*proof*⟩

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∈G b∈G
shows a=b ∨ (a≤b ∧ a≠b) ∨ (b≤a ∧ b≠a)
⟨*proof*⟩

17.3 Intervals and bounded sets

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: ∀g∈A. L≤g ∧ g≤M
and A2: S = RightTranslation(G,P,L^{-1})
and A3: a ∈ S(A)
shows a ≤ M·L^{-1} **1**≤a
⟨*proof*⟩

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: IsBounded(A,r)
shows ∃B.∃g∈G$^+$.∃T∈G→G. A = T(B) ∧ B ⊆ Interval(r,**1**,g)
⟨*proof*⟩

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: ∀g∈G$^+$. Interval(r,**1**,g) ∈ Fin(G)
and A2: IsBounded(A,r)
shows A ∈ Fin(G)
⟨*proof*⟩

In linearly ordered groups finite sets are bounded.

theorem (in group3**)** ord_group_fin_bounded:
assumes r {is total on} G **and** B∈Fin(G)
shows IsBounded(B,r)
⟨*proof*⟩

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: $\forall$a∈G. $\exists$b∈A. a≤b
shows
$\forall$a∈G. $\exists$b∈A. a≠b $\wedge$ a≤b
¬IsBoundedAbove(A,r)
A $\notin$ Fin(G)
⟨*proof*⟩

Nontrivial linearly ordered groups are infinite. Recall that Fin(A) is the collection of finite subsets of A. In this lemma we show that $G \notin$ 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 $\neq$ {1}
shows
$G_+ \notin$ Fin(G)
G $\notin$ Fin(G)
⟨*proof*⟩

A property of nonempty subsets of linearly ordered groups that don't have a maximum: for any element in such subset we can find one that is strictly greater.

lemma (in group3) OrderedGroup_ZF_2_L2B:
assumes A1: r {is total on} G **and** A2: A⊆G **and**
A3: ¬HasAmaximum(r,A) **and** A4: x∈A
shows $\exists$y∈A. x<y
⟨*proof*⟩

In linearly ordered groups $G \setminus G_+$ is bounded above.

lemma (in group3) OrderedGroup_ZF_2_L3:
assumes A1: r {is total on} G **shows** IsBoundedAbove(G-G_+,r)
⟨*proof*⟩

In linearly ordered groups if $A \cap G_+$ is finite, then A is bounded above.

lemma (in group3) OrderedGroup_ZF_2_L4:
assumes A1: r {is total on} G **and** A2: A⊆G
and A3: A $\cap$ G_+ $\in$ Fin(G)
shows IsBoundedAbove(A,r)
⟨*proof*⟩

If a set $-A \subseteq G$ is bounded above, then A is bounded below.

lemma (in group3) OrderedGroup_ZF_2_L5:
assumes A1: A⊆G **and** A2: IsBoundedAbove(-A,r)
shows IsBoundedBelow(A,r)
⟨*proof*⟩

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≤b **and** f:G→G
shows f(Interval(r,a,b)) ≠ 0
⟨*proof*⟩

17.4 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:
AbsoluteValue(G,P,r) : G→G
⟨*proof*⟩

If $a \in G^+$, then $|a| = a$.

lemma (in group3) OrderedGroup_ZF_3_L2:
assumes A1: a∈G^+ **shows** |a| = a
⟨*proof*⟩

lemma (in group3) OrderedGroup_ZF_3_L2A:
shows |1| = 1 ⟨*proof*⟩

If a is positive, then $|a| = a$.

lemma (in group3) OrderedGroup_ZF_3_L2B:
assumes a∈G_+ **shows** |a| = a
⟨*proof*⟩

If $a \in G \setminus G^+$, then $|a| = a^{-1}$.

lemma (in group3) OrderedGroup_ZF_3_L3:
assumes A1: a ∈ G-G^+ **shows** |a| = a^{-1}
⟨*proof*⟩

For elements that not greater than the unit, the absolute value is the inverse.

lemma (in group3) OrderedGroup_ZF_3_L3A:
assumes A1: a≤**1**
shows |a| = a^{-1}
⟨*proof*⟩

In linearly ordered groups the absolute value of any element is in G^+.

lemma (in group3) OrderedGroup_ZF_3_L3B:
assumes A1: r {is total on} G **and** A2: a∈G
shows |a| ∈ G^+
⟨*proof*⟩

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 {is total on} G
shows AbsoluteValue(G,P,r) : G→G^+
⟨*proof*⟩

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*⟩

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|
⟨*proof*⟩

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*⟩

In linearly ordered groups $a \leq |a|$.

lemma (in group3) OrderedGroup_ZF_3_L5:
assumes A1: r {is total on} G **and** A2: a∈G
shows a ≤ |a|
⟨*proof*⟩

$a^{-1} \leq |a|$ (in additive notation it would be $-a \leq |a|$.

lemma (in group3) OrderedGroup_ZF_3_L6:
assumes A1: a∈G **shows** a^{-1} ≤ |a|
⟨*proof*⟩

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 {is total on} G **and** a∈G b∈G
shows
a·b ≤|a|·|b|
a·b^{-1} ≤|a|·|b|
a^{-1}·b ≤|a|·|b|
a^{-1}·b^{-1} ≤|a|·|b|
⟨*proof*⟩

$|a^{-1}| \leq |a|$.

lemma (in group3) OrderedGroup_ZF_3_L7:

assumes r {is total on} G **and** a∈G
shows |a^{-1}|≤|a|
⟨*proof*⟩

$|a^{-1}| = |a|$.

lemma (in group3) OrderedGroup_ZF_3_L7A:
assumes A1: r {is total on} G **and** A2: a∈G
shows |a^{-1}| = |a|
⟨*proof*⟩

$|a \cdot b^{-1}| = |b \cdot a^{-1}|$. It doesn't look so strange in the additive notation: $|a - b| = |b - a|$.

lemma (in group3) OrderedGroup_ZF_3_L7B:
assumes A1: r {is total on} G **and** A2: a∈G b∈G
shows |a·b^{-1}| = |b·a^{-1}|
⟨*proof*⟩

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∈G b∈G
shows |a·b| ≤ |a|·|b|
⟨*proof*⟩

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*⟩

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^{-1}| ≤ c·d
⟨*proof*⟩

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*⟩

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
and A4: $|a \cdot b^{-1}| \leq c$
shows
$|a| \leq c \cdot |b|$
$|a| \leq |b| \cdot c$
$c^{-1} \cdot a \leq b$
$a \cdot c^{-1} \leq b$
$a \leq b \cdot c$
⟨*proof*⟩

Some more variants of the triangle inequality.

lemma (in group3**)** OrderedGroup_ZF_3_L7E:
assumes A1: P {is commutative on} G
and A2: r {is total on} G **and** A3: a∈G b∈G
and A4: $|a \cdot b^{-1}| \leq c$
shows $b \cdot c^{-1} \leq a$
⟨*proof*⟩

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∈G b∈G c∈G d∈G
shows $|a \cdot c^{-1}| \leq |a \cdot b| \cdot |c \cdot d| \cdot |b \cdot d^{-1}|$
⟨*proof*⟩

$|a| \leq L$ 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∈G **and** A2: $|a| \leq L$
shows
$L^{-1} \leq a$
⟨*proof*⟩

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∈G **and** A3: $|a| \leq L$
shows
$a \leq L$
$\mathbf{1} \leq L$
⟨*proof*⟩

A somewhat generalized version of the above lemma.

lemma (in group3**)** OrderedGroup_ZF_3_L8B:

assumes A1: a∈G **and** A2: |a|≤L **and** A3: **1**≤c
shows $(L \cdot c)^{-1} \leq a$
⟨*proof*⟩

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≤b **and** A2: c∈G **and** A3: b≤c·a
shows $|b \cdot a^{-1}| \leq c$
⟨*proof*⟩

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*⟩

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 ∧ |b(x)|≤L
shows IsBounded({b(x). x∈X},r)
⟨*proof*⟩

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*⟩

For linearly ordered groups if $l \leq a \leq u$ then $|a|$ is smaller than the greater of $|l|, |u|$.

lemma (in group3**)** OrderedGroup_ZF_3_L10:
assumes A1: r {is total on} G
and A2: l≤a a≤u
shows
|a| ≤ GreaterOf(r,|l|,|u|)
⟨*proof*⟩

For linearly ordered groups if a set is bounded then the absolute values are bounded.

lemma (in group3**)** OrderedGroup_ZF_3_L10A:

assumes A1: r {is total on} G
and A2: IsBounded(A,r)
shows ∃L. ∀a∈A. |a| ≤ L
⟨*proof*⟩

A slightly more general version of the previous lemma, stating the same fact for a set defined by separation.

lemma (in group3**)** OrderedGroup_ZF_3_L11:
assumes A1: r {is total on} G
and A2: IsBounded({b(x).x∈X},r)
shows ∃L. ∀x∈X. |b(x)| ≤ L
⟨*proof*⟩

Absolute values of elements of a finite image of a nonempty set are bounded by an element of the group.

lemma (in group3**)** OrderedGroup_ZF_3_L11A:
assumes A1: r {is total on} G
and A2: X≠0 **and** A3: {b(x). x∈X} ∈ Fin(G)
shows ∃L∈G. ∀x∈X. |b(x)| ≤ L
⟨*proof*⟩

In totally oredered groups the absolute value of a nonunit element is in G_+.

lemma (in group3**)** OrderedGroup_ZF_3_L12:
assumes A1: r {is total on} G
and A2: a∈G **and** A3: a≠**1**
shows |a| ∈ G_+
⟨*proof*⟩

17.5 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 ⊆ G
and HasAmaximum(r,A) HasAminimum(r,A)
and M = GreaterOf(r,|Minimum(r,A)|,|Maximum(r,A)|)
shows M ∈ AbsoluteValue(G,P,r)(A)
⟨*proof*⟩

If a set has a maximum and minimum, then the greater of the absolute value of the maximum and minimum bounds absolute values of all elements of the set.

lemma (in group3) OrderedGroup_ZF_4_L2:
assumes A1: r {is total on} G
and A2: HasAmaximum(r,A) HasAminimum(r,A)
and A3: a∈A
shows |a|≤ GreaterOf(r,|Minimum(r,A)|,|Maximum(r,A)|)
⟨*proof*⟩

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 ekements of a set are represented as the elements of the image of the set by the absolute value function.

lemma (in group3) OrderedGroup_ZF_4_L3:
assumes r {is total on} G **and** A ⊆ G
and HasAmaximum(r,A) HasAminimum(r,A)
and b ∈ AbsoluteValue(G,P,r)(A)
shows b≤ GreaterOf(r,|Minimum(r,A)|,|Maximum(r,A)|)
⟨*proof*⟩

If a set has a maximum and minimum, then the set of absolute values also has a maximum.

lemma (in group3) OrderedGroup_ZF_4_L4:
assumes A1: r {is total on} G **and** A2: A ⊆ G
and A3: HasAmaximum(r,A) HasAminimum(r,A)
shows HasAmaximum(r,AbsoluteValue(G,P,r)(A))
⟨*proof*⟩

If a set has a maximum and a minimum, then all absolute values are bounded by the maximum of the set of absolute values.

lemma (in group3) OrderedGroup_ZF_4_L5:
assumes A1: r {is total on} G **and** A2: A ⊆ G
and A3: HasAmaximum(r,A) HasAminimum(r,A)
and A4: a∈A
shows |a| ≤ Maximum(r,AbsoluteValue(G,P,r)(A))
⟨*proof*⟩

17.6 Alternative definitions

Sometimes it is usful to define the order by prescibing the set of positive or nonnegative elements. This section deals with two such definitions. One takes a subset H of G that is closed under the group operation, $1 \notin H$ and for every $a \in H$ we have either $a \in H$ or $a^{-1} \in H$. Then the order is defined as $a \leq b$ iff $a = b$ or $a^{-1}b \in H$. For abelian groups this makes a linearly ordered group. We will refer to order defined this way in the comments as the order defined by a positive set. The context used in this section is the group0 context defined in Group_ZF theory. Recall that f in that context denotes the group operation (unlike in the previous sections where the group operation was denoted P.

The order defined by a positive set is the same as the order defined by a nonnegative set.

lemma (in group0**)** OrderedGroup_ZF_5_L1:
assumes A1: r = {p ∈ G×G. fst(p) = snd(p) ∨ fst(p)$^{-1}$·snd(p) ∈ H}
shows ⟨a,b⟩ ∈ r ⟷ a∈G ∧ b∈G ∧ a^{-1}·b ∈ H ∪ {**1**}
⟨*proof*⟩

The relation defined by a positive set is antisymmetric.

lemma (in group0**)** OrderedGroup_ZF_5_L2:
assumes A1: r = {p ∈ G×G. fst(p) = snd(p) ∨ fst(p)$^{-1}$·snd(p) ∈ H}
and A2: ∀a∈G. a≠**1** ⟶ (a∈H) Xor (a^{-1}∈H)
shows antisym(r)
⟨*proof*⟩

The relation defined by a positive set is transitive.

lemma (in group0**)** OrderedGroup_ZF_5_L3:
assumes A1: r = {p ∈ G×G. fst(p) = snd(p) ∨ fst(p)$^{-1}$·snd(p) ∈ H}
and A2: H⊆G H {is closed under} f
shows trans(r)
⟨*proof*⟩

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 ∈ G×G. fst(p) = snd(p) ∨ fst(p)$^{-1}$·snd(p) ∈ H}
and A2: f {is commutative on} G
and A3: ⟨a,b⟩ ∈ r **and** A4: c∈G
shows ⟨a·c,b·c⟩ ∈ r ∧ ⟨c·a,c·b⟩ ∈ r
⟨*proof*⟩

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 $a \leq b \Leftrightarrow a^{-1}b \in H$ 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: f {is commutative on} G
and A2: H⊆G H {is closed under} f
and A3: ∀a∈G. a≠**1** ⟶ (a∈H) Xor (a^{-1}∈H)
and A4: r = {p ∈ G×G. fst(p) = snd(p) ∨ fst(p)$^{-1}$·snd(p) ∈ H}
shows
IsAnOrdGroup(G,f,r)
r {is total on} G
Nonnegative(G,f,r) = PositiveSet(G,f,r) ∪ {**1**}
⟨*proof*⟩

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 f {is commutative on} G
and H⊆G **and** **1** ∉ H
and r = {p ∈ G×G. fst(p) = snd(p) ∨ fst(p)$^{-1}$·snd(p) ∈ H}
shows PositiveSet(G,f,r) = H
⟨*proof*⟩

The next definition describes how we construct an order relation from the prescribed set of positive elements.

constdefs
OrderFromPosSet(G,P,H) ≡
{p ∈ G×G. fst(p) = snd(p) ∨ P⟨GroupInv(G,P)(fst(p)),snd(p)⟩ ∈ 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 simmarize, this is what it says: Suppose that $H \subseteq G$ is a set closed under that group operation such that $1 \notin H$ and for every nonunit group element a either $a \in H$ or $a^{-1} \in H$. Define the order as $a \leq b$ iff $a = b$ or $a^{-1} \cdot b \in H$. Then this order makes G into a linearly ordered group such H is the set of positive elements (and then of course $H \cup \{1\}$ is the set of nonnegative elements).

theorem (in group0**)** Group_ord_by_positive_set:
assumes f {is commutative on} G
and H⊆G H {is closed under} f **1** ∉ H
and ∀a∈G. a≠**1** ⟶ (a∈H) Xor (a^{-1}∈H)
shows
IsAnOrdGroup(G,f,OrderFromPosSet(G,f,H))
OrderFromPosSet(G,f,H) {is total on} G
PositiveSet(G,f,OrderFromPosSet(G,f,H)) = H
Nonnegative(G,f,OrderFromPosSet(G,f,H)) = H ∪ {**1**}
⟨*proof*⟩

17.7 Odd Extensions

In this section we verify properties of odd extensions of functions defined on G_+. An odd extension of a function $f : G_+ \to G$ is a function $f^\circ : G \to G$ defined by $f^\circ(x) = f(x)$ if $x \in G_+$, $f(1) = 1$ and $f^\circ(x) = (f(x^{-1}))^{-1}$ for $x < 1$. Such function is the unique odd function that is equal to f when restricted to G_+.

The next lemma is just to see the definition of the odd extension in the notation used in the group1 context.

lemma (in group3**)** OrderedGroup_ZF_6_L1:
shows f$^\circ$ = f ∪ {⟨a, (f(a^{-1}))$^{-1}$⟩. a ∈ -G$_+$} ∪ {⟨**1**,**1**⟩}
⟨*proof*⟩

A technical lemma that states that from a function defined on G$_+$ with values in G we have $(f(a^{-1}))^{-1} \in G$.

lemma (in group3) OrderedGroup_ZF_6_L2:
assumes f: $G_+ \to G$ **and** a∈-G_+
shows
f(a^{-1}) ∈ G
(f(a^{-1}))$^{-1}$ ∈ G
⟨*proof*⟩

The main theorem about odd extensions. It basically says that the odd extension of a function is what we want to to be.

lemma (in group3) odd_ext_props:
assumes A1: r {is total on} G **and** A2: f: $G_+ \to G$
shows
f° : G → G
∀a∈G_+. (f°)(a) = f(a)
∀a∈(-G_+). (f°)(a) = (f(a^{-1}))$^{-1}$
(f°)(**1**) = **1**
⟨*proof*⟩

Odd extensions are odd, of course.

lemma (in group3) oddext_is_odd:
assumes A1: r {is total on} G **and** A2: f: $G_+ \to G$
and A3: a∈G
shows $(f^\circ)(a^{-1}) = ((f^\circ)(a))^{-1}$
⟨*proof*⟩

Another way of saying that odd extensions are odd.

lemma (in group3) oddext_is_odd_alt:
assumes A1: r {is total on} G **and** A2: f: $G_+ \to G$
and A3: a∈G
shows $((f^\circ)(a^{-1}))^{-1} = (f^\circ)(a)$
⟨*proof*⟩

17.8 Functions with infinite limits

In this section we consider functions $f : G \to G$ with the property that for $f(x)$ is arbitrarily large for large enough x. More precisely, for every $a \in G$ there exist $b \in G_+$ such that for every $x \geq b$ we have $f(x) \geq a$. In a sense this means that $\lim_{x\to\infty} f(x) = \infty$, hence the title of this section. We also prove dual statements for functions such that $\lim_{x\to-\infty} f(x) = -\infty$.

If an image of a set by a function with infinite positive limit is bounded above, then the set itself is bounded above.

lemma (in group3) OrderedGroup_ZF_7_L1:
assumes A1: r {is total on} G **and** A2: G ≠ {1} **and**
A3: f:G→G **and**
A4: ∀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*⟩

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*⟩

If the image of a set defined by separation by a function with infinite negative limit is bounded below, then the set itself is bounded above. This is dual to `OrderedGroup_ZF_7_L2`.

```
lemma (in group3) OrderedGroup_ZF_7_L3:
  assumes A1: r {is total on} G and A2: G ≠ {1} and
  A3: X≠0 and A4: f:G→G and
  A5: ∀a∈G.∃b∈G_+.∀y. b≤y ⟶ f(y^-1) ≤ a and
  A6: ∀x∈X. b(x) ∈ G ∧ L ≤ f(b(x))
  shows ∃l.∀x∈X. l ≤ b(x)
```
⟨*proof*⟩

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 i bounded.

```
lemma (in group3) OrderedGroup_ZF_7_L4:
  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: ∀a∈G.∃b∈G_+.∀y. b≤y ⟶ f(y^-1) ≤ a and
  A7: ∀x∈X. b(x) ∈ G ∧ L ≤ f(b(x)) ∧ f(b(x)) ≤ U
shows ∃M.∀x∈X. |b(x)| ≤ M
```
⟨*proof*⟩

end

18 Ring_ZF.thy

theory Ring_ZF **imports** Group_ZF

begin

This theory file covers basic facts about rings.

18.1 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. We also define the notion of having no zero divisors.

constdefs
```
  IsAring(R,A,M) ≡ IsAgroup(R,A) ∧ (A {is commutative on} R) ∧
  IsAmonoid(R,M) ∧ IsDistributive(R,A,M)

  HasNoZeroDivs(R,A,M) ≡ (∀a∈R. ∀b∈R.
  M<a,b> = TheNeutralElement(R,A) ⟶
  a = TheNeutralElement(R,A) ∨ b = TheNeutralElement(R,A))
```

Next we define a locale that will be used when considering rings.

locale ring0 =

```
  fixes R and A and M

  assumes ringAssum: IsAring(R,A,M)

  fixes ringa (infixl + 90)
  defines ringa_def [simp]: a+b ≡ A<a,b>

  fixes ringminus (- _ 89)
  defines ringminus_def [simp]: (-a) ≡ GroupInv(R,A)(a)

  fixes ringsub (infixl - 90)
  defines ringsub_def [simp]: a-b ≡ 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 (_^{2} [96] 97)
defines ringsq_def [simp]: a^2 ≡ 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
⟨*proof*⟩

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
⟨*proof*⟩

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**
⟨*proof*⟩

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·a = a+a
(-a)+a = **0**
⟨*proof*⟩

Properties that require two elements of a ring.

lemma (in ring0**)** Ring_ZF_1_L4: **assumes** A1: a∈R b∈R
shows
a+b ∈ R
a-b ∈ R
a·b ∈ R
a+b = b+a
⟨*proof*⟩

Any element of a ring multiplied by zero is zero.

lemma (in ring0**)** Ring_ZF_1_L6:
assumes A1: x∈R **shows** **0**·x = **0** x·**0** = **0**
⟨*proof*⟩

Negative can be pulled out of a product.

lemma (in ring0**)** Ring_ZF_1_L7:
assumes A1: a∈R b∈R
shows
(-a)·b = -(a·b)
a·(-b) = -(a·b)
(-a)·b = a·(-b)
⟨*proof*⟩

Minus times minus is plus.

lemma (in ring0**)** Ring_ZF_1_L7A: **assumes** a∈R b∈R
shows (-a)·(-b) = a·b
⟨*proof*⟩

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
⟨*proof*⟩

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
⟨*proof*⟩

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*⟩

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
⟨*proof*⟩

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
⟨*proof*⟩

Associativity of addition and multiplication.

lemma (in ring0**)** Ring_ZF_1_L11:
assumes a∈R b∈R c∈R
shows
a+b+c = a+(b+c)
a·b·c = a·(b·c)
⟨*proof*⟩

An interpretation of what it means that a ring has no zero divisors.

lemma (in ring0**)** Ring_ZF_1_L12:
assumes HasNoZeroDivs(R,A,M)
and a∈R a≠**0** b∈R b≠**0**
shows a·b≠**0**
⟨*proof*⟩

In rings with no zero divisors we can cancel nonzero factors.

lemma (in ring0**)** Ring_ZF_1_L12A:
assumes A1: HasNoZeroDivs(R,A,M) **and** A2: a∈R b∈R c∈R
and A3: a·c = b·c **and** A4: c≠**0**
shows a=b
⟨*proof*⟩

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:
assumes A1: HasNoZeroDivs(R,A,M)
a∈R b∈R c∈R a≠b c≠**0**
shows a·c ≠ b·c
⟨*proof*⟩

In rings with no zero divisors multiplying a nonzero element by a nonone element changes the value.

lemma (in ring0**)** Ring_ZF_1_L12C:
assumes A1: HasNoZeroDivs(R,A,M) **and**
A2: a∈R b∈R **and** A3: **0**≠a **1**≠b
shows a ≠ a·b
⟨*proof*⟩

If a square is nonzero, then the element is nonzero.

lemma (in ring0**)** Ring_ZF_1_L13:
assumes a∈R **and** $a^2 \neq \mathbf{0}$
shows a≠**0**
⟨*proof*⟩

Square of an element and its opposite are the same.

lemma (in ring0**)** Ring_ZF_1_L14:
assumes a∈R **shows** $(-a)^2 = ((a)^2)$
⟨*proof*⟩

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
⟨*proof*⟩

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
⟨*proof*⟩

The ring is trivial iff $0 = 1$.

lemma (in ring0**)** Ring_ZF_1_L17: **shows** R = {**0**} ⟷ **0=1**
⟨*proof*⟩

The sets $\{m \cdot x . x \in R\}$ and $\{-m \cdot x . x \in R\}$ are the same.

lemma (in ring0**)** Ring_ZF_1_L18: **assumes** A1: m∈R
shows {m·x. x∈R} = {(-m)·x. x∈R}
⟨*proof*⟩

18.2 Rearrangement lemmas

In happens quite often that we want to show a fact like $(a + b)c + d = (ac + d - e) + (bc + e)$in rings. This is trivial in romantic math and probably there is a way to make it trivial in formalized math. However, I don't know any other way than to tediously prove each such rearrangement when it is needed. This section collects facts of this type.

Rearrangements with two elements of a ring.

lemma (in ring0**)** Ring_ZF_2_L1: **assumes** a∈R b∈R
shows a+b·a = (b+**1**)·a
⟨*proof*⟩

Raearrangements with two elements and cancelling.

lemma (in ring0**)** Ring_ZF_2_L1A: **assumes** a∈R b∈R
shows
a-b+b = a
a+b-a = b
(-a)+b+a = b
(-a)+(b+a) = b
a+(b-a) = b
⟨*proof*⟩

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∈R b∈R c∈R d∈R
shows a-(b+**1**)·c = (a-d-c)+(d-b·c)
⟨*proof*⟩

Rerrangement about adding linear functions.

lemma (in ring0**)** Ring_ZF_2_L3:
assumes A1: a∈R b∈R c∈R d∈R x∈R
shows (a·x + b) + (c·x + d) = (a+c)·x + (b+d)
⟨*proof*⟩

Rearrangement with three elements

lemma (in ring0**)** Ring_ZF_2_L4:
assumes M {is commutative on} R
and a∈R b∈R c∈R
shows a·(b·c) = a·c·b
⟨*proof*⟩

Some other rearrangements with three elements.

lemma (in ring0**)** ring_rearr_3_elemA:
assumes A1: M {is commutative on} R **and**
A2: a∈R b∈R c∈R
shows
a·(a·c) - b·(-b·c) = (a·a + b·b)·c
a·(-b·c) + b·(a·c) = **0**
⟨*proof*⟩

Some rearrangements with four elements. Properties of abelian groups.

lemma (in ring0**)** Ring_ZF_2_L5:
assumes a∈R b∈R c∈R d∈R

shows
a - b - c - d = a - d - b - c
a + b + c - d = a - d + b + c
a + b - c - d = a - c + (b - d)
a + b + c + d = a + c + (b + d)
⟨*proof*⟩

Two big rearranegements with six elements, useful for proving properties of complex addition and multiplication.

lemma (in ring0) Ring_ZF_2_L6:
assumes A1: a∈R b∈R c∈R d∈R e∈R f∈R
shows
a·(c·e - d·f) - b·(c·f + d·e) =
(a·c - b·d)·e - (a·d + b·c)·f
a·(c·f + d·e) + b·(c·e - d·f) =
(a·c - b·d)·f + (a·d + b·c)·e
a·(c+e) - b·(d+f) = a·c - b·d + (a·e - b·f)
a·(d+f) + b·(c+e) = a·d + b·c + (a·f + b·e)
⟨*proof*⟩

end

19 Ring_ZF_1.thy

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.

19.1 The ring of classes of almost homomorphisms

Almost homomorphisms do not form a ring as the regular homomorphisms do because the lifted group operation is not distributive with respect to composition – we have $s \circ (r \cdot q) \neq s \circ r \cdot s \circ q$ in general. However, we do have $s \circ (r \cdot q) \approx s \circ r \cdot s \circ q$ in the sense of the equivalence relation defined by the group of finite range functions (that is a normal subgroup of almost homomorphisms, if the group is abelian). This allows to define a natural ring structure on the classes of almost homomorphisms.

The next lemma provides a formula useful for proving that two sides of the distributive law equation for almost homomorphisms are almost equal.

lemma (in group1**)** Ring_ZF_1_1_L1:
  assumes A1: s∈AH r∈AH q∈AH **and** A2: n∈G
  shows
  ((s∘(r·q))(n))·(((s∘r)·(s∘q))(n))$^{-1}$= δ(s,<r(n),q(n)>)
  ((r·q)∘s)(n) = ((r∘s)·(q∘s))(n)
⟨*proof*⟩

The sides of the distributive law equations for almost homomorphisms are almost equal.

lemma (in group1**)** Ring_ZF_1_1_L2:
  assumes A1: s∈AH r∈AH q∈AH
  shows
  s∘(r·q) ≈ (s∘r)·(s∘q)
  (r·q)∘s = (r∘s)·(q∘s)
⟨*proof*⟩

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
  and A3: A = ProjFun2(AH,R,Op1) M = ProjFun2(AH,R,Op2)
  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*⟩

The projection of the first group operation on almost homomorphisms is distributive with respect to the second group operation.

lemma (in group1**)** Ring_ZF_1_1_L4:
  assumes A1: R = QuotientGroupRel(AH,Op1,FR)
  and A2: A = ProjFun2(AH,R,Op1) M = ProjFun2(AH,R,Op2)
  shows IsDistributive(AH//R,A,M)
⟨*proof*⟩

The classes of almost homomorphisms form a ring.

theorem (in group1**)** Ring_ZF_1_1_T1:
  assumes R = QuotientGroupRel(AH,Op1,FR)
  and A = ProjFun2(AH,R,Op1) M = ProjFun2(AH,R,Op2)
  shows IsAring(AH//R,A,M)
  ⟨*proof*⟩

end

20 OrderedRing_ZF.thy

theory OrderedRing_ZF **imports** Ring_ZF OrderedGroup_ZF

begin

In this theory file we consider ordered rings.

20.1 Definition and notation

This section defines ordered rings and sets up appriopriate 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.

constdefs

```
IsAnOrdRing(R,A,M,r) ≡
( IsAring(R,A,M) ∧ (M {is commutative on} R) ∧
r⊆R×R ∧ IsLinOrder(R,r) ∧
(∀a b. ∀ c∈R. <a,b> ∈ r ⟶ ⟨A<a,c>,A<b,c>⟩ ∈ r) ∧
(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 ring0 locale and adding some assumptions to make sure we are talking about ordered rings in this context.

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** ≤ 68)
defines lesseq_def [simp]: a ≤ b ≡ <a,b> ∈ r

fixes sless (**infix** < 68)
defines sless_def [simp]: a < b ≡ a≤b ∧ a≠b

assumes ordgroup: ∀a b. ∀ c∈R. a≤b ⟶ a+c ≤ b+c

assumes pos_mult_closed: Nonnegative(R,A,r) {is closed under} M

fixes abs (| _ |)

defines abs_def [simp]: |a| ≡ AbsoluteValue(R,A,r)(a)

fixes positiveset (R_+)
defines positiveset_def [simp]: R_+ ≡ PositiveSet(R,A,r)

The next lemma assures us that we are talking about ordered rings in the ring1 context.

lemma (in ring1) OrdRing_ZF_1_L1: **shows** IsAnOrdRing(R,A,M,r)
⟨*proof*⟩

We can use theorems proven in the ring1 context whenever we talk about an ordered ring.

lemma OrdRing_ZF_1_L2: **assumes** IsAnOrdRing(R,A,M,r)
shows ring1(R,A,M,r)
⟨*proof*⟩

In the ring1 context $a \leq b$ implies that a, b are elements of the ring.

lemma (in ring1) OrdRing_ZF_1_L3: **assumes** a≤b
shows a∈R b∈R
⟨*proof*⟩

Ordered ring is an ordered group, hence we can use theorems proven in the group3 context.

lemma (in ring1) OrdRing_ZF_1_L4: **shows**
IsAnOrdGroup(R,A,r)
r {is total on} R
A {is commutative on} R
group3(R,A,r)
⟨*proof*⟩

The order relation in rings is transitive.

lemma (in ring1) ring_ord_transitive: **assumes** A1: a≤b b≤c
shows a≤c
⟨*proof*⟩

Transitivity for the strict order: if $a < b$ and $b \leq c$, then $a < c$. Property of ordered groups.

lemma (in ring1) ring_strict_ord_trans:
assumes A1: a<b **and** A2: b≤c
shows a<c
⟨*proof*⟩

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≤b **and** A2: b<c
shows a<c

⟨*proof*⟩

The next lemma shows what happens when one element of an ordered ring is not greater or equal than another.

lemma (in ring1**)** OrdRing_ZF_1_L4A**: assumes** A1**:** a∈R b∈R
  and A2**:** ¬(a≤b)
  shows b ≤ a (-a) ≤ (-b) a≠b
⟨*proof*⟩

A special case of OrdRing_ZF_1_L4A when one of the constants is 0. This is useful for many proofs by cases.

corollary (in ring1**)** ord_ring_split2**: assumes** A1**:** a∈R
  shows a≤**0** ∨ (**0**≤a ∧ a≠**0**)
⟨*proof*⟩

Taking minus on both sides reverses an inequality.

lemma (in ring1**)** OrdRing_ZF_1_L4B**: assumes** a≤b
  shows (-b) ≤ (-a)
  ⟨*proof*⟩

The next lemma just expands the condition that requires the set of non-negative elements to be closed with respect to multiplication. These are properties of totally ordered groups.

lemma (in ring1**)** OrdRing_ZF_1_L5:
  assumes **0**≤a **0**≤b
  shows **0** ≤ a·b
  ⟨*proof*⟩

Double nonnegative is nonnegative.

lemma (in ring1**)** OrdRing_ZF_1_L5A**: assumes** A1**:** **0**≤a
  shows **0**≤**2**·a
  ⟨*proof*⟩

A sufficient (somewhat redundant) condition for a structure to be an ordered ring. It says that a commutative ring that is a totally ordered group with respect to the additive operation such that set of nonnegative elements is closed under multiplication, is an ordered ring.

lemma OrdRing_ZF_1_L6:
  assumes
  IsAring(R,A,M)
  M {is commutative on} R
  Nonnegative(R,A,r) {is closed under} M
  IsAnOrdGroup(R,A,r)
  r {is total on} R
  shows IsAnOrdRing(R,A,M,r)
  ⟨*proof*⟩

$a \leq b$ 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∈R b∈R
shows a≤b ⟷ a-b ≤ **0**
⟨*proof*⟩

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*⟩

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*⟩

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*⟩

A square is nonnegative.

lemma (in ring1) OrdRing_ZF_1_L10:
assumes A1: a∈R **shows** **0**≤(a^2)
⟨*proof*⟩

1 is nonnegative.

corollary (in ring1) ordring_one_is_nonneg: shows 0 ≤ 1
⟨*proof*⟩

In nontrivial rings one is positive.

lemma (in ring1) ordring_one_is_pos: assumes 0≠1
shows 1 ∈ R_+
⟨*proof*⟩

Nonnegative is not negative. Property of ordered groups.

lemma (in ring1) OrdRing_ZF_1_L11: assumes 0≤a

shows ¬(a≤**0** ∧ a≠**0**)
⟨*proof*⟩

A negative element cannot be a square.

lemma (in ring1**)** OrdRing_ZF_1_L12:
assumes A1: a≤**0** a≠**0**
shows ¬(∃b∈R. a = (b^2))
⟨*proof*⟩

If $a \leq b$, then $0 \leq b - a$.

lemma (in ring1**)** OrdRing_ZF_1_L13: **assumes** a≤b
shows **0** ≤ b-a
⟨*proof*⟩

If $a < b$, then $0 < b - a$.

lemma (in ring1**)** OrdRing_ZF_1_L14: **assumes** a≤b a≠b
shows
0 ≤ b-a **0** ≠ b-a
b-a ∈ R_+
⟨*proof*⟩

If the difference is nonnegative, then $a \leq b$.

lemma (in ring1**)** OrdRing_ZF_1_L15:
assumes a∈R b∈R **and** **0** ≤ b-a
shows a≤b
⟨*proof*⟩

A nonnegative number is does not decrease when multiplied by a number greater or equal 1.

lemma (in ring1**)** OrdRing_ZF_1_L16:
assumes A1: **0**≤a **and** A2: **1**≤b
shows a≤a·b
⟨*proof*⟩

We can multiply the right hand side of an inequality between nonnegative ring elements by an element greater or equal 1.

lemma (in ring1**)** OrdRing_ZF_1_L17:
assumes A1: **0**≤a **and** A2: a≤b **and** A3: **1**≤c
shows a≤b·c
⟨*proof*⟩

Strict order is preserved by translations.

lemma (in ring1**)** ring_strict_ord_trans_inv:
assumes a<b **and** c∈R
shows
a+c < b+c
c+a < c+b

⟨*proof*⟩

We can put an element on the other side of a strict inequality, changing its sign.

lemma (in ring1**)** OrdRing_ZF_1_L18:
assumes a∈R b∈R **and** a-b < c
shows a < c+b
⟨*proof*⟩

We can add the sides of two inequalities, the first of them strict, and we get a strict inequality. Property of ordered groups.

lemma (in ring1**)** OrdRing_ZF_1_L19:
assumes a<b **and** c≤d
shows a+c < b+d
⟨*proof*⟩

We can add the sides of two inequalities, the second of them strict and we get a strict inequality. Property of ordered groups.

lemma (in ring1**)** OrdRing_ZF_1_L20:
assumes a≤b **and** c<d
shows a+c < b+d
⟨*proof*⟩

20.2 Absolute value for ordered rings

Absolute value is defined for ordered groups as a function that is the identity on the nonnegative set and the negative of the element (the inverse in the multiplicative notation) on the rest. In this section we consider properties of absolute value related to multiplication in ordered rings.

Absolute value of a product is the product of absolute values: the case when both elements of the ring are nonnegative.

lemma (in ring1**)** OrdRing_ZF_2_L1:
assumes 0≤a 0≤b
shows |a·b| = |a|·|b|
⟨*proof*⟩

The absolue value of an element and its negative are the same.

lemma (in ring1**)** OrdRing_ZF_2_L2: **assumes** a∈R
shows |-a| = |a|
⟨*proof*⟩

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∈R b∈R
shows
|(-a)·b| = |a·b|

```
|a·(-b)| = |a·b|
|(-a)·(-b)| = |a·b|
```
⟨*proof*⟩

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∈R **and** ¬(**0**≤a)
shows 0 ≤ (-a) **0**≠a
⟨*proof*⟩

Absolute value of a product is the product of absolute values.

lemma (in ring1**)** OrdRing_ZF_2_L5:
assumes A1: a∈R b∈R
shows |a·b| = |a|·|b|
⟨*proof*⟩

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|
⟨*proof*⟩

If $a \leq c$ and $b \leq c$, then $a + b \leq 2 \cdot c$.

lemma (in ring1**)** OrdRing_ZF_2_L6:
assumes a≤c b≤c **shows** a+b ≤ **2**·c
⟨*proof*⟩

20.3 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
⟨*proof*⟩

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_+)
⟨*proof*⟩

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_+
⟨*proof*⟩

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$) \longleftrightarrow$ HasNoZeroDivs(R,A,M)
⟨*proof*⟩

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$\subseteq$R P {is closed under} A **0** $\notin$ P **and**
A3: $\forall$a$\in$R. a$\neq$**0** $\longrightarrow$ (a$\in$P) Xor ((-a) $\in$ 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 $\cup$ {**0**}
HasNoZeroDivs(R,A,M)
⟨*proof*⟩

Nontrivial ordered rings are infinite. More precisely we assume that the neutral element of the additive operation is not equal to the multiplicative neutral element and show that the the set of positive elements of the ring is not a finite subset of the ring and the ring is not a finite subset of itself.

theorem (in ring1) ord_ring_infinite: assumes 0$\neq$1
shows
$R_+ \notin$ Fin(R)
R $\notin$ Fin(R)
⟨*proof*⟩

lemma (in ring1) OrdRing_ZF_3_L4:
assumes 0$\neq$1 and $\forall$a$\in$R. $\exists$b$\in$B. a$\leq$b
shows
$\neg$IsBoundedAbove(B,r)
B $\notin$ Fin(R)
⟨*proof*⟩

If m is greater or equal the multiplicative unit, then the set $\{m \cdot n : n \in R\}$ is infinite (unless the ring is trivial).

lemma (in ring1) OrdRing_ZF_3_L5: assumes A1: 0$\neq$1 and A2: 1$\leq$m
shows
{m$\cdot$x. x$\in R_+$} $\notin$ Fin(R)
{m$\cdot$x. x$\in$R} $\notin$ Fin(R)
{(-m)$\cdot$x. x$\in$R} $\notin$ Fin(R)
⟨*proof*⟩

If m is less or equal than the negative of multiplicative unit, then the set $\{m \cdot n : n \in R\}$ is infinite (unless the ring is trivial).

lemma (in ring1) OrdRing_ZF_3_L6: assumes A1: 0≠1 and A2: m ≤ -1
shows {m·x. x∈R} ∉ Fin(R)
⟨*proof*⟩

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₊
⟨*proof*⟩

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*⟩

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
⟨*proof*⟩

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
⟨*proof*⟩

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*⟩

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
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*⟩

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*⟩

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_+
⟨*proof*⟩

If a ring has no zero divisors, the square of a nonzero element is positive.

lemma (in ring1**)** OrdRing_ZF_3_L15:
assumes HasNoZeroDivs(R,A,M) **and** a∈R a≠**0**
shows **0** ≤ a^2 a^2 ≠ **0** a^2 ∈ R_+
⟨*proof*⟩

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 HasNoZeroDivs(R,A,M) **and** a ∈ R_+ **and** **1**≤b **1**≠b
shows a≤a·b a ≠ a·b
⟨*proof*⟩

If the right hand side of an inequality is positive we can multiply it by a number that is greater than one.

lemma (in ring1**)** OrdRing_ZF_3_L17:
assumes A1: HasNoZeroDivs(R,A,M) **and** A2: b∈R_+ **and**
A3: a≤b **and** A4: **1**<c
shows a<b·c
⟨*proof*⟩

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 A1: HasNoZeroDivs(R,A,M) **and** A2: a ∈ R_+ **and**
A3: a≤b **and** A4: **1**<c
shows a<b·c
⟨*proof*⟩

In ordered rings with no zero divisors if at least one of a, b is not zero, then $a^2 + b^2 > 0$, in particular $a^2 + b^2 \neq 0$.

lemma (in ring1**)** OrdRing_ZF_3_L19:
assumes A1: HasNoZeroDivs(R,A,M) **and** A2: a∈R b∈R **and**
A3: a ≠ **0** ∨ b ≠ **0**

shows **0** < a^2 + b^2
⟨*proof*⟩

end

21 Field_ZF.thy

theory Field_ZF **imports** Ring_ZF

begin

This theory covers basic facts about fields.

21.1 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.

constdefs
```
  IsAfield(K,A,M) ≡
  (IsAring(K,A,M) ∧ (M {is commutative on} K) ∧
  TheNeutralElement(K,A) ≠ TheNeutralElement(K,M) ∧
  (∀a∈K. a≠TheNeutralElement(K,A) ⟶
  (∃b∈K. M⟨a,b⟩ = TheNeutralElement(K,M))))
```

The field0 context extends the ring0 context adding field-related assumptions and notation related to the multiplicative inverse.

locale field0 = ring0 K +
```
  assumes mult_commute: M {is commutative on} K

  assumes not_triv: 0 ≠ 1

  assumes 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  [96] 97)
  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 field0 context.

lemma (in field0) Field_ZF_1_L1: **shows** IsAfield(K,A,M)
⟨*proof*⟩

We can use theorems proven in the field0 context whenever we talk about a field.

lemma Field_ZF_1_L2: **assumes** IsAfield(K,A,M)
shows field0(K,A,M)

⟨*proof*⟩

Let's have an explicit statement that the multiplication in fields is commutative.

lemma (in field0**)** field_mult_comm: **assumes** a∈K b∈K
shows a·b = b·a
⟨*proof*⟩

Fields do not have zero divisors.

lemma (in field0**)** field_has_no_zero_divs: **shows** HasNoZeroDivs(K,A,M)
⟨*proof*⟩

K_0 (the set of nonzero field elements is closed with respect to multiplication.

lemma (in field0**)** Field_ZF_1_L2: K_0 {is closed under} M
⟨*proof*⟩

Any nonzero element has a right inverse that is nonzero.

lemma (in field0**)** Field_ZF_1_L3: **assumes** A1: a∈K_0
shows ∃b∈K_0. a·b = **1**
⟨*proof*⟩

If we remove zero, the field with multiplication becomes a group and we can use all theorems proven in group0 context.

theorem (in field0**)** Field_ZF_1_L4: **shows**
IsAgroup(K_0,restrict(M,K_0×K_0))
group0(K_0,restrict(M,K_0×K_0))
1 = TheNeutralElement(K_0,restrict(M,K_0×K_0))
⟨*proof*⟩

The inverse of a nonzero field element is nonzero.

lemma (in field0**)** Field_ZF_1_L5: **assumes** A1: a∈K a≠**0**
shows $a^{-1} \in K_0$ $(a^{-1})^2 \in K_0$ $a^{-1} \in K$ $a^{-1} \neq \mathbf{0}$
⟨*proof*⟩

The inverse is really the inverse.

lemma (in field0**)** Field_ZF_1_L6: **assumes** A1: a∈K a≠**0**
shows $a \cdot a^{-1} = \mathbf{1}$ $a^{-1} \cdot a = \mathbf{1}$
⟨*proof*⟩

A lemma with two field elements and cancelling.

lemma (in field0**)** Field_ZF_1_L7: **assumes** a∈K b∈K b≠**0**
shows
$a \cdot b \cdot b^{-1} = a$
$a \cdot b^{-1} \cdot b = a$
⟨*proof*⟩

21.2 Equations and identities

This section deals with more specialized identities that are true in fields.

$a/(a^2) = a$.

lemma (in field0**)** Field_ZF_2_L1: **assumes** A1: a∈K a≠**0**
  shows a·(a^{-1})^2 = a^{-1}
⟨*proof*⟩

If we multiply two different numbers by a nonzero number, the results will be different.

lemma (in field0**)** Field_ZF_2_L2:
  assumes a∈K b∈K c∈K a≠b c≠**0**
  shows a·c^{-1} ≠ b·c^{-1}
  ⟨*proof*⟩

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∈K b∈K b≠**0** c∈K **and** A2: a·b ≠ c
  shows a ≠ c·b^{-1}
⟨*proof*⟩

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∈K a≠**0** **and** b^{-1} ≠ a
  shows a^{-1} ≠ b
  ⟨*proof*⟩

An identity with two field elements, one and an inverse.

lemma (in field0**)** Field_ZF_2_L5:
  assumes a∈K b∈K b≠**0**
  shows (**1** + a·b)·b^{-1} = a + b^{-1}
  ⟨*proof*⟩

An identity with three field elements, inverse and cancelling.

lemma (in field0**)** Field_ZF_2_L6: **assumes** A1: a∈K b∈K b≠**0** c∈K
  shows a·b·(c·b^{-1}) = a·c
⟨*proof*⟩

end

22 OrderedField_ZF.thy

theory OrderedField_ZF **imports** OrderedRing_ZF Field_ZF

begin

This theory covers basic facts about ordered fiels.

22.1 Definition and basic properties

Ordered field is a notrivial ordered ring such that all non-zero elements have an inverse. We define the notion of being a ordered field as a statement about four sets. The first set, denoted K is the carrier of the field. The second set, denoted A represents the additive operation on K (recall that in ZF set theory functions are sets). The third set M represents the multiplicative operation on K. The fourth set r is the order relation on K.

constdefs
IsAnOrdField(K,A,M,r) ≡ (IsAnOrdRing(K,A,M,r) ∧
(M {is commutative on} K) ∧
TheNeutralElement(K,A) ≠ TheNeutralElement(K,M) ∧
(∀a∈K. a≠TheNeutralElement(K,A) ⟶
(∃b∈K. M⟨a,b⟩ = TheNeutralElement(K,M))))

The next context (locale) defines notation used for ordered fields. We do that by extending the notation defined in the ring1 context that is used for oredered 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 appriopriate for fields. Theoretically the Isar locale facility supports such renaming, but we experienced diffculties using some lemmas from ring1 locale after renaming.

locale field1 = ring1 +

assumes mult_commute: M {is commutative on} R

assumes not_triv: **0** ≠ **1**

assumes inv_exists: ∀a∈R. a≠**0** ⟶ (∃b∈R. a·b = **1**)

fixes non_zero (R_0)
defines non_zero_def[simp]: R_0 ≡ R-{**0**}

fixes inv ($\_^{-1}$ [96] 97)
defines inv_def[simp]: a^{-1} ≡ GroupInv(R_0,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)
⟨*proof*⟩

Ordered field is a field, of course.

lemma OrdField_ZF_1_L1A: **assumes** IsAnOrdField(K,A,M,r)
shows IsAfield(K,A,M)
⟨*proof*⟩

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)
⟨*proof*⟩

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)
⟨*proof*⟩

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: $\forall a \in R_+.\ \exists b \in R.\ a \cdot b = \mathbf{1}$ **and** A2: $c \in R\ \ c \neq \mathbf{0}$
shows $\exists b \in R.\ c \cdot b = \mathbf{1}$
⟨*proof*⟩

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 $\mathbf{0} \neq \mathbf{1}$ **and** M {is commutative on} R
and $\forall a \in R_+.\ \exists b \in R.\ a \cdot b = \mathbf{1}$
shows IsAnOrdField(R,A,M,r)
⟨*proof*⟩

The set of positive field elements is closed under multiplication.

lemma (in field1**)** OrdField_ZF_1_L5: **shows** R_+ {is closed under} M
⟨*proof*⟩

The set of positive field elements is closed under multiplication: the explicit version.

lemma (in field1**)** pos_mul_closed:
assumes A1: $\mathbf{0} < a\ \ \mathbf{0} < b$
shows $\mathbf{0} < a \cdot b$
⟨*proof*⟩

In fields square of a nonzero element is positive.

lemma (in field1**)** OrdField_ZF_1_L6: **assumes** $a \in R\ \ a \neq \mathbf{0}$
shows $a^2 \in R_+$
⟨*proof*⟩

The next lemma restates the fact Field_ZF that out notation for the field inverse means what it is supposed to mean.

lemma (in field1) OrdField_ZF_1_L7: assumes a∈R a≠**0**
  shows a·(a^{-1}) = **1** (a^{-1})·a = **1**
  ⟨*proof*⟩

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*⟩

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*⟩

If $a < b$, then $(b-a)^{-1}$ is positive.

lemma (in field1) OrdField_ZF_1_L9: assumes a<b
  shows (b-a)$^{-1}$ ∈ R$_+$
  ⟨*proof*⟩

In ordered fields if at least one of a, b is not zero, then $a^2 + b^2 > 0$, in particular $a^2 + b^2 \neq 0$ and exists the (multiplicative) inverse of $a^2 + b^2$.

lemma (in field1) OrdField_ZF_1_L10:
  assumes A1: a∈R b∈R **and** A2: a ≠ **0** ∨ b ≠ **0**
  shows 0 < a^2 + b^2 **and** ∃c∈R. (a^2 + b^2)·c = **1**
⟨*proof*⟩

22.2 Inequalities

In this section we develop tools to deal inequalities in fields.

We can multiply strict inequality by a positive element.

lemma (in field1) OrdField_ZF_2_L1:
  assumes a<b **and** c∈R$_+$
  shows a·c < b·c
  ⟨*proof*⟩

A special case of OrdField_ZF_2_L1 when we multiply an inverse by an element.

lemma (in field1) OrdField_ZF_2_L2:
  assumes A1: a∈R$_+$ **and** A2: a^{-1} < b
  shows 1 < b·a

⟨*proof*⟩

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*⟩

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
$a \cdot c < b \cdot c$
$c \cdot a < c \cdot b$
$a \cdot c^{-1} < b \cdot c^{-1}$
⟨*proof*⟩

We can put a positive factor on the other side of an inequality, changing it to its inverse.

lemma (in field1**)** OrdField_ZF_2_L5:
assumes A1: $a \in R$ $b \in R_+$ **and** A2: $a \cdot b \leq c$
shows $a \leq c \cdot b^{-1}$
⟨*proof*⟩

We can put a positive factor on the other side of an inequality, changing it to its inverse, version with a product initially on the right hand side.

lemma (in field1**)** OrdField_ZF_2_L5A:
assumes A1: $b \in R$ $c \in R_+$ **and** A2: $a \leq b \cdot c$
shows $a \cdot c^{-1} \leq b$
⟨*proof*⟩

We can put a positive factor on the other side of a strict inequality, changing it to its inverse, version with a product initially on the left hand side.

lemma (in field1**)** OrdField_ZF_2_L6:
assumes A1: $a \in R$ $b \in R_+$ **and** A2: $a \cdot b < c$
shows $a < c \cdot b^{-1}$
⟨*proof*⟩

We can put a positive factor on the other side of a strict inequality, changing it to its inverse, version with a product initially on the right hand side.

lemma (in field1**)** OrdField_ZF_2_L6A:
assumes A1: $b \in R$ $c \in R_+$ **and** A2: $a < b \cdot c$
shows $a \cdot c^{-1} < b$
⟨*proof*⟩

Sometimes we can reverse an inequality by taking inverse on both sides.

lemma (in field1**)** OrdField_ZF_2_L7:
assumes A1: $a \in R_+$ **and** A2: $a^{-1} \leq b$

shows $\texttt{b}^{-1} \leq \texttt{a}$
⟨*proof*⟩

Sometimes we can reverse a strict inequality by taking inverse on both sides.

lemma (in field1**)** OrdField_ZF_2_L8:
assumes A1: $\texttt{a} \in \texttt{R}_+$ **and** A2: $\texttt{a}^{-1} < \texttt{b}$
shows $\texttt{b}^{-1} < \texttt{a}$
⟨*proof*⟩

A technical lemma about solving a strict inequality with three field elements and inverse of a difference.

lemma (in field1**)** OrdField_ZF_2_L9:
assumes A1: a<b **and** A2: $(\texttt{b}-\texttt{a})^{-1} < \texttt{c}$
shows $\texttt{1} + \texttt{a} \cdot \texttt{c} < \texttt{b} \cdot \texttt{c}$
⟨*proof*⟩

22.3 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.

constdefs
IsAmodelOfReals(K,A,M,r) $\equiv$ IsAnOrdField(K,A,M,r) $\wedge$ (r {is complete})

end

23 Int_ZF.thy

theory Int_ZF **imports** OrderedGroup_ZF Finite_ZF_1 Int Nat_ZF

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 $Z \times Z$. We show that a subset of intergers is bounded iff it is finite.

23.1 Addition and multiplication as ZF-functions.

In this section we provide definitions of addition and multiplication as subsets of $(Z \times Z) \times Z$. We use the \$ $\leq$ (higher order) relation defined in the standard Int theory to define a subset of $Z \times Z$ that constitutes the ZF order relation corresponding to it. We define positive integers using the notion of positive set from the OrderedGroup theory.

constdefs

```
IntegerAddition ≡ { <x,c> ∈ (int×int)×int. fst(x) $+ snd(x) = c}

IntegerMultiplication ≡
  { <x,c> ∈ (int×int)×int. fst(x) $× snd(x) = c}

IntegerOrder ≡ {p ∈ int×int. fst(p) $≤ snd(p)}

PositiveIntegers ≡ PositiveSet(int,IntegerAddition,IntegerOrder)
```

IntegerAddition and IntegerMultiplication are functions on int×int.

lemma Int_ZF_1_L1:
```
IntegerAddition : int×int → int
IntegerMultiplication : int×int → int
```
⟨*proof*⟩

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≤n for integers and write m..n to denote the integer interval with endpoints in m and n. 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 $x \leq 0$. Unforunately we 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 `maxf(f,M)` denotes tha maximum of function f over the set A. We also introduce a similar notation for the minimum.

locale `int0 =`

fixes `ints (ℤ)`
defines `ints_def [simp]: ℤ ≡ int`

fixes `ia (infixl + 69)`
defines `ia_def [simp]: a+b ≡ IntegerAddition<a,b>`

fixes `iminus :: i⇒i (- _ 72)`
defines `rminus_def [simp]: -a ≡ GroupInv(ℤ,IntegerAddition)(a)`

fixes `isub (infixl - 69)`
defines `isub_def [simp]: a-b ≡ a+ (- b)`

fixes `imult (infixl · 70)`
defines `imult_def [simp]: a·b ≡ IntegerMultiplication<a,b>`

fixes `setneg :: i⇒i (- _ 72)`
defines `setneg_def [simp]: -A ≡ GroupInv(ℤ,IntegerAddition)(A)`

fixes `izero (0)`
defines `izero_def [simp]: 0 ≡ TheNeutralElement(ℤ,IntegerAddition)`

fixes `ione (1)`
defines `ione_def [simp]: 1 ≡ TheNeutralElement(ℤ,IntegerMultiplication)`

fixes `itwo (2)`
defines `itwo_def [simp]: 2 ≡ 1+1`

fixes `ithree (3)`
defines `itwo_def [simp]: 3 ≡ 2+1`

fixes `nonnegative (ℤ⁺)`
defines `nonnegative_def [simp]:`
`ℤ⁺ ≡ Nonnegative(ℤ,IntegerAddition,IntegerOrder)`

fixes `positive (ℤ₊)`
defines `positive_def [simp]:`
`ℤ₊ ≡ PositiveSet(ℤ,IntegerAddition,IntegerOrder)`

fixes `abs`
defines `abs_def [simp]:`
`abs(m) ≡ AbsoluteValue(ℤ,IntegerAddition,IntegerOrder)(m)`

fixes `lesseq (infix ≤ 60)`
defines `lesseq_def [simp]: m ≤ n ≡ ⟨m,n⟩ ∈ IntegerOrder`

fixes interval (**infix** .. 70)
defines interval_def [simp]: m..n ≡ Interval(IntegerOrder,m,n)

fixes maxf
defines maxf_def [simp]: maxf(f,A) ≡ Maximum(IntegerOrder,f(A))

fixes minf
defines minf_def [simp]: minf(f,A) ≡ Minimum(IntegerOrder,f(A))

IntegerAddition adds integers and IntegerMultiplication multiplies integers. This states that the ZF functions IntegerAddition and IntegerMultiplication give the same results as the higher-order $+ and $× defined in the standard Int theory.

lemma (in int0) Int_ZF_1_L2: **assumes** A1: a ∈ ℤ b ∈ ℤ
shows
a+b = a $+ b
a·b = a $× b
⟨*proof*⟩

Integer addition and multiplication are associative.

lemma (in int0) Int_ZF_1_L3:
assumes x∈ℤ y∈ℤ z∈ℤ
shows x+y+z = x+(y+z) x·y·z = x·(y·z)
⟨*proof*⟩

Integer addition and multiplication are commutative.

lemma (in int0) Int_ZF_1_L4:
assumes x∈ℤ y∈ℤ
shows x+y = y+x x·y = y·x
⟨*proof*⟩

Zero is neutral for addition and one for multiplication.

lemma (in int0) Int_ZF_1_L5: **assumes** A1:x∈ℤ
shows ($# 0) + x = x ∧ x + ($# 0) = x
($# 1)·x = x ∧ x·($# 1) = x
⟨*proof*⟩

Zero is neutral for addition and one for multiplication.

lemma (in int0) Int_ZF_1_L6: **shows** ($# 0)∈ℤ ∧
(∀x∈ℤ. ($# 0)+x = x ∧ x+($# 0) = x)
($# 1)∈ℤ ∧
(∀x∈ℤ. ($# 1)·x = x ∧ x·($# 1) = x)
⟨*proof*⟩

Integers with addition and integers with multiplication form monoids.

theorem (in int0) Int_ZF_1_T1: **shows**

```
  IsAmonoid(ℤ,IntegerAddition)
  IsAmonoid(ℤ,IntegerMultiplication)
```
⟨*proof*⟩

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:` `($# 0) = 0  ($# 1) = 1`
⟨*proof*⟩

0 and 1, as defined in `int0` context, are integers.

lemma (in `int0`**)** `Int_ZF_1_L8A:` **shows** $0 \in \mathbb{Z}$ $1 \in \mathbb{Z}$
⟨*proof*⟩

Zero is not one.

lemma (in `int0`**)** `int_zero_not_one:` **shows** $0 \neq 1$
⟨*proof*⟩

The set of integers is not empty, of course.

lemma (in `int0`**)** `int_not_empty:` **shows** $\mathbb{Z} \neq 0$
⟨*proof*⟩

The set of integers has more than just zero in it.

lemma (in `int0`**)** `int_not_trivial:` **shows** $\mathbb{Z} \neq \{0\}$
⟨*proof*⟩

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*⟩

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 assumpion that some pair of sets form a group like locale `group0`.

theorem `Int_ZF_1_T2:` **shows**
```
  IsAgroup(int,IntegerAddition)
  IntegerAddition {is commutative on} int
  group0(int,IntegerAddition)
```
⟨*proof*⟩

What is the additive group inverse in the group of integers?

lemma (in `int0`**)** `Int_ZF_1_L9A:` **assumes** `A1: m∈ℤ`
shows `$-m = -m`
⟨*proof*⟩

Subtracting integers corresponds to adding the negative.

lemma (in `int0`**)** `Int_ZF_1_L10:` **assumes** `A1: m∈ℤ  n∈ℤ`

shows m-n = m $+ $-n
⟨*proof*⟩

Negative of zero is zero.

lemma (in int0) Int_ZF_1_L11: **shows** (-**0**) = **0**
⟨*proof*⟩

A trivial calculation lemma that allows to subtract and add one.

lemma Int_ZF_1_L12:
assumes m∈int **shows** m $- $#1 $+ $#1 = m
⟨*proof*⟩

A trivial calculation lemma that allows to subtract and add one, version with ZF-operation.

lemma (in int0) Int_ZF_1_L13: **assumes** m∈ℤ
shows (m $- $#1) + **1** = m
⟨*proof*⟩

Adding or subtracing one changes integers.

lemma (in int0) Int_ZF_1_L14: **assumes** A1: m∈ℤ
shows
m+**1** ≠ m
m-**1** ≠ m
⟨*proof*⟩

If the difference is zero, the integers are equal.

lemma (in int0) Int_ZF_1_L15:
assumes A1: m∈ℤ n∈ℤ **and** A2: m-n = **0**
shows m=n
⟨*proof*⟩

23.2 Integers as an ordered group

In this section we define order on integers as a relation, that is a subset of $Z \times Z$ and show that integers form an ordered group.

The next lemma interprets the order definition one way.

lemma (in int0) Int_ZF_2_L1:
assumes A1: m∈ℤ n∈ℤ **and** A2: m $≤ n
shows m ≤ n
⟨*proof*⟩

The next lemma interprets the definition the other way.

lemma (in int0) Int_ZF_2_L1A: **assumes** A1: m ≤ n
shows m $≤ n m∈ℤ n∈ℤ
⟨*proof*⟩

Integer order is a relation on integers.

lemma Int_ZF_2_L1B: IntegerOrder $\subseteq$ int$\times$int
⟨*proof*⟩

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*⟩

The order on integers is reflexive.

lemma (in int0**)** int_ord_is_refl: **shows** refl($\mathbb{Z}$,IntegerOrder)
  ⟨*proof*⟩

The essential condition to show antisymmetry of the order on integers.

lemma (in int0**)** Int_ZF_2_L3:
  assumes A1: m $\leq$ n n $\leq$ m
  shows m=n
⟨*proof*⟩

The order on integers is antisymmetric.

lemma (in int0**)** Int_ZF_2_L4: antisym(IntegerOrder)
⟨*proof*⟩

The essential condition to show that the order on integers is transitive.

lemma Int_ZF_2_L5:
  assumes A1: ⟨m,n⟩ $\in$ IntegerOrder ⟨n,k⟩ $\in$ IntegerOrder
  shows ⟨m,k⟩ $\in$ IntegerOrder
⟨*proof*⟩

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*⟩

The order on integers is transitive.

lemma Int_ZF_2_L6: trans(IntegerOrder)
⟨*proof*⟩

The order on integers is a partial order.

lemma Int_ZF_2_L7: **shows** IsPartOrder(int,IntegerOrder)
  ⟨*proof*⟩

The essential condition to show that the order on integers is preserved by translations.

lemma (in int0**)** int_ord_transl_inv:
assumes A1: k ∈ ℤ **and** A2: m ≤ n
shows m+k ≤ n+k k+m≤ k+n
⟨*proof*⟩

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(ℤ,IntegerAddition,IntegerOrder)
IntegerOrder {is total on} ℤ
group3(ℤ,IntegerAddition,IntegerOrder)
IsLinOrder(ℤ,IntegerOrder)
⟨*proof*⟩

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 ≤ m **and** A2: i≠m
shows i $< m
⟨*proof*⟩

This shows how Isabelle's $< operator translates to IsarMathLib notation.

lemma (in int0**)** Int_ZF_2_L9AA: **assumes** A1: m∈ℤ n∈ℤ
and A2: m $< n
shows m≤n m ≠ n
⟨*proof*⟩

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*⟩

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 ⟷ i ≤ k-m
⟨*proof*⟩

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*⟩

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*⟩

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*⟩

We can cancel the same element on 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*⟩

If an integer is nonpositive, then its opposite is nonnegative.

lemma (in int0**)** Int_ZF_2_L10A**: assumes** k ≤ **0**
shows 0≤(-k)
⟨*proof*⟩

If the opposite of an integers is nonnegative, then the integer is nonpositive.

lemma (in int0**)** Int_ZF_2_L10B:
assumes k∈ℤ **and 0**≤(-k)
shows k≤**0**
⟨*proof*⟩

Adding one to an integer corresponds to taking a successor for a natural number.

lemma (in int0**)** Int_ZF_2_L11**:** i $+ $# n $+ ($# 1) = i $+ $# succ(n)
⟨*proof*⟩

Adding a natural number increases integers.

lemma (in int0**)** Int_ZF_2_L12**: assumes** A1**:** i∈ℤ **and** A2**:** n∈nat
shows i ≤ i $+ $#n
⟨*proof*⟩

Adding one increases integers.

lemma (in int0**)** Int_ZF_2_L12A**: assumes** A1**:** j≤k
shows j ≤ k $+ $#1 j ≤ k**+1**
⟨*proof*⟩

Adding one increases integers, yet one more version.

lemma (in int0**)** Int_ZF_2_L12B**: assumes** A1**:** m∈ℤ **shows** m ≤ m**+1**
⟨*proof*⟩

If $k + 1 = m + n$, where n is a non-zero natural number, then $m \leq k$.

lemma (in int0**)** Int_ZF_2_L13:
assumes A1: k∈ℤ m∈ℤ **and** A2: n∈nat
and A3: k $+ ($# 1) = m $+ $# succ(n)
shows m ≤ k
⟨*proof*⟩

The absolute value of an integer is an integer.

lemma (in int0**)** Int_ZF_2_L14: **assumes** A1: m∈ℤ
shows abs(m) ∈ ℤ
⟨*proof*⟩

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**≤m **0**≤n
shows
(-m) ≤ n
0 ≤ m + n
⟨*proof*⟩

We can increase components in an estimate.

lemma (in int0**)** Int_ZF_2_L15:
assumes b≤b_1 c≤c_1 **and** a≤b+c
shows a≤b_1+c_1
⟨*proof*⟩

We can add or subtract the sides of two inequalities.

lemma (in int0**)** int_ineq_add_sides:
assumes a≤b **and** c≤d
shows
a+c ≤ b+d
a-d ≤ b-c
⟨*proof*⟩

We can increase the second component in an estimate.

lemma (in int0**)** Int_ZF_2_L15A:
assumes b∈ℤ **and** a≤b+c **and** A3: c≤c_1
shows a≤b+c_1
⟨*proof*⟩

If we increase the second component in a sum of three integers, the whole sum inceases.

lemma (in int0**)** Int_ZF_2_L15C:
assumes A1: m∈ℤ n∈ℤ **and** A2: k ≤ L
shows m+k+n ≤ m+L+n
⟨*proof*⟩

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 $m \leq n+m$
⟨*proof*⟩

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
$m+n \leq |m|+|n|$
$m-n \leq |m|+|n|$
$(-m)+n \leq |m|+|n|$
$(-m)-n \leq |m|+|n|$
⟨*proof*⟩

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 $m \leq k+n$ $m \leq n+k$
⟨*proof*⟩

Triangle inequality for integers.

lemma (in int0) Int_triangle_ineq:
assumes $m \in \mathbb{Z}$ $n \in \mathbb{Z}$
shows $|m+n| \leq |m|+|n|$
⟨*proof*⟩

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** $|m| = m$
⟨*proof*⟩

$0 \leq 1$, so $|1| = 1$.

lemma (in int0) Int_ZF_2_L16A: shows $0 \leq 1$ **and** $|1| = 1$
⟨*proof*⟩

$1 \leq 2$.

lemma (in int0) Int_ZF_2_L16B: shows $1 \leq 2$
⟨*proof*⟩

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*⟩

Absolute value is the same for an integer and its opposite.

lemma (in int0**)** Int_ZF_2_L17:
  assumes m∈ℤ **shows** abs(-m) = abs(m)
  ⟨*proof*⟩

The absolute value of zero is zero.

lemma (in int0**)** Int_ZF_2_L18: **shows** abs(**0**) = **0**
  ⟨*proof*⟩

A different version of the triangle inequality.

lemma (in int0**)** Int_triangle_ineq1:
  assumes A1: m∈ℤ n∈ℤ
  shows
  abs(m-n) ≤ abs(n)+abs(m)
  abs(m-n) ≤ abs(m)+abs(n)
⟨*proof*⟩

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
  ⟨*proof*⟩

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*⟩

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*⟩

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*⟩

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∈ℤ **and** A2: ¬(**0**≤m) **shows** m≤**0** **0** ≤ (-m) m≠**0**
⟨*proof*⟩

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∈ℤ **and** Q(**0**) **and** ∀n∈$\mathbb{Z}_+$. Q(n) **and** ∀n∈$\mathbb{Z}_+$. Q(-n) **shows** Q(m)
⟨*proof*⟩

An integer is not greater than its absolute value.

lemma (in int0**)** Int_ZF_2_L19C: **assumes** A1: m∈ℤ **shows** m ≤ abs(m) (-m) ≤ abs(m)
⟨*proof*⟩

$|m - n| = |n - m|$.

lemma (in int0**)** Int_ZF_2_L20: **assumes** m∈ℤ n∈ℤ **shows** abs(m-n) = abs(n-m)
⟨*proof*⟩

We can add the sides of inequalities with absolute values.

lemma (in int0**)** Int_ZF_2_L21: **assumes** A1: m∈ℤ n∈ℤ **and** A2: abs(m) ≤ k abs(n) ≤ l **shows** abs(m+n) ≤ k + l abs(m-n) ≤ k + l
⟨*proof*⟩

Absolute value is nonnegative.

lemma (in int0**)** int_abs_nonneg: **assumes** A1: m∈ℤ **shows** abs(m) ∈ $\mathbb{Z}^+$ **0** ≤ abs(m)
⟨*proof*⟩

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**≤m m≤k **shows** abs(m) ≤ k
⟨*proof*⟩

23.3 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 ≤ m
  shows ∃n∈nat. m = i $+ $# n
⟨*proof*⟩

Induction for integers, the induction step.

lemma (in int0**)** Int_ZF_3_L6: **assumes** A1: i∈ℤ
  and A2: ∀m. i≤m ∧ Q(m) ⟶ Q(m $+ ($# 1))
  shows ∀k∈nat. Q(i $+ ($# k)) ⟶ Q(i $+ ($# succ(k)))
⟨*proof*⟩

Induction on integers, version with higher-order increment function.

lemma (in int0**)** Int_ZF_3_L7:
  assumes A1: i≤k **and** A2: Q(i)
  and A3: ∀m. i≤m ∧ Q(m) ⟶ Q(m $+ ($# 1))
  shows Q(k)
⟨*proof*⟩

Induction on integer, implication between two forms of the induction step.

lemma (in int0**)** Int_ZF_3_L7A: **assumes**
  A1: ∀m. i≤m ∧ Q(m) ⟶ Q(m+**1**)
  shows ∀m. i≤m ∧ Q(m) ⟶ Q(m $+ ($# 1))
⟨*proof*⟩

Induction on integers, version with ZF increment function.

theorem (in int0**)** Induction_on_int:
  assumes A1: i≤k **and** A2: Q(i)
  and A3: ∀m. i≤m ∧ Q(m) ⟶ Q(m+**1**)
  shows Q(k)
⟨*proof*⟩

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≤k **and** A2: P($-i)
  and A3: ∀m. i≤m ∧ P($-m) ⟶ P($-(m $+ ($# 1)))
  shows P($-k)
⟨*proof*⟩

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≤i **and** A2: P(i)

and A3: $\forall$m. $-i$\leq$m $\wedge$ P($-m) $\longrightarrow$ P($-(m $+ ($# 1)))
shows P(k)
⟨*proof*⟩

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 $\wedge$ P(n) $\longrightarrow$ P(n $+ $-($#1))
shows $\forall$m. $-i$\leq$m $\wedge$ P($-m) $\longrightarrow$ P($-(m $+ ($# 1)))
⟨*proof*⟩

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 $\wedge$ P(n) $\longrightarrow$P(n $+ $-($#1))
shows P(k)
⟨*proof*⟩

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 $\wedge$ P(n) $\longrightarrow$ P(n**-1**)
shows $\forall$n. n$\leq$i $\wedge$ P(n) $\longrightarrow$ P(n $+ $-($#1))
⟨*proof*⟩

Backwards induction on integers.

theorem (in int0**)** Back_induct_on_int:
assumes A1: k$\leq$i **and** A2: P(i)
and A3: $\forall$n. n$\leq$i $\wedge$ P(n) $\longrightarrow$ P(n**-1**)
shows P(k)
⟨*proof*⟩

23.4 Bounded vs. finite subsets of integers

The goal of this section is to establish that a subset of integers is bounded is and only is it is finite. The fact that all finite sets are bounded is already shown for all linearly ordered groups in OrderedGroups_ZF.thy. To show the other implication we show that all intervals starting at 0 are finite and then use a result from OrderedGroups_ZF.thy.

There are no integers between k and $k+1$.

lemma (in int0**)** Int_ZF_4_L1:
assumes A1: k$\in\mathbb{Z}$ m$\in\mathbb{Z}$ n$\in$nat **and** A2: k $+ $#1 = m $+ $#n
shows m = k $+ $#1 $\vee$ m $\leq$ k
⟨*proof*⟩

A trivial calculation lemma that allows to subtract and add one.

lemma Int_ZF_4_L1A:
assumes m$\in$int **shows** m $- $#1 $+ $#1 = m

⟨*proof*⟩

There are no integers between k and $k+1$, another formulation.

lemma (in int0**)** Int_ZF_4_L1B: **assumes** A1: m ≤ L
shows
m = L ∨ m+**1** ≤ L
m = L ∨ m ≤ L-**1**
⟨*proof*⟩

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∈ℤ
and A2: j ∈ m..(k $+ $#1)
shows j ∈ m..k ∨ j ∈ {k $+ $#1}
⟨*proof*⟩

Extending an integer interval by one is the same as adding the new endpoint.

lemma (in int0**)** Int_ZF_4_L3: **assumes** A1: m≤ k
shows m..(k $+ $#1) = m..k ∪ {k $+ $#1}
⟨*proof*⟩

Integer intervals are finite - induction step.

lemma (in int0**)** Int_ZF_4_L4:
assumes A1: i≤m **and** A2: i..m ∈ Fin(ℤ)
shows i..(m $+ $#1) ∈ Fin(ℤ)
⟨*proof*⟩

Integer intervals are finite.

lemma (in int0**)** Int_ZF_4_L5: **assumes** A1: i∈ℤ k∈ℤ
shows i..k ∈ Fin(ℤ)
⟨*proof*⟩

Bounded integer sets are finite.

lemma (in int0**)** Int_ZF_4_L6: **assumes** A1: IsBounded(A,IntegerOrder)
shows A ∈ Fin(ℤ)
⟨*proof*⟩

A subset of integers is bounded iff it is finite.

theorem (in int0**)** Int_bounded_iff_fin:
shows IsBounded(A,IntegerOrder)⟷ A∈Fin(ℤ)
⟨*proof*⟩

The image of an interval by any integer function is finite, hence bounded.

lemma (in int0**)** Int_ZF_4_L8:
assumes A1: i∈ℤ k∈ℤ **and** A2: f:ℤ→ℤ
shows
f(i..k) ∈ Fin(ℤ)
IsBounded(f(i..k),IntegerOrder)

⟨*proof*⟩

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
¬IsBoundedAbove(A,IntegerOrder)
A $\notin$ Fin($\mathbb{Z}$)
⟨*proof*⟩

end

24 Int_ZF_1.thy

theory Int_ZF_1 **imports** Int_ZF OrderedRing_ZF

begin

This theory file considers the set of integers as an ordered ring.

24.1 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∈ℤ b∈ℤ c∈ℤ
shows
a·(b+c) = a·b + a·c
(b+c)·a = b·a + c·a
⟨*proof*⟩

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(ℤ,IntegerAddition,IntegerMultiplication)
IntegerMultiplication {is commutative on} ℤ
ring0(ℤ,IntegerAddition,IntegerMultiplication)
⟨*proof*⟩

Zero and one are integers.

lemma (in int0**)** int_zero_one_are_int: **shows** **0**∈ℤ **1**∈ℤ
⟨*proof*⟩

Negative of zero is zero.

lemma (in int0**)** int_zero_one_are_intA: **shows** (-**0**) = **0**
⟨*proof*⟩

Properties with one integer.

lemma (in int0**)** Int_ZF_1_1_L4: **assumes** A1: a ∈ ℤ
shows
a+**0** = a
0+a = a
a·**1** = a **1**·a = a
0·a = **0** a·**0** = **0**
(-a) ∈ ℤ (-(-a)) = a
a-a = **0** a-**0** = a **2**·a = a+a
⟨*proof*⟩

Properties that require two integers.

lemma (in int0**)** Int_ZF_1_1_L5: **assumes** A1: a∈ℤ b∈ℤ
  shows
  a+b ∈ ℤ
  a-b ∈ ℤ
  a·b ∈ ℤ
  a+b = b+a
  a·b = b·a
  (-b)-a = (-a)-b
  (-(a+b)) = (-a)-b
  (-(a-b)) = ((-a)+b)
  (-a)·b = -(a·b)
  a·(-b) = -(a·b)
  (-a)·(-b) = a·b
  ⟨*proof*⟩

2 and 3 are integers.

lemma (in int0**)** int_two_three_are_int: **shows** **2** ∈ ℤ **3** ∈ ℤ
  ⟨*proof*⟩

Another property with two integers.

lemma (in int0**)** Int_ZF_1_1_L5B:
  assumes A1: a∈ℤ b∈ℤ
  shows a-(-b) = a+b
  ⟨*proof*⟩

Properties that require three integers.

lemma (in int0**)** Int_ZF_1_1_L6: **assumes** A1: a∈ℤ b∈ℤ c∈ℤ
  shows
  a-(b+c) = a-b-c
  a-(b-c) = a-b+c
  a·(b-c) = a·b - a·c
  (b-c)·a = b·a - c·a
  ⟨*proof*⟩

One more property with three integers.

lemma (in int0**)** Int_ZF_1_1_L6A: **assumes** A1: a∈ℤ b∈ℤ c∈ℤ
  shows a+(b-c) = a+b-c
  ⟨*proof*⟩

Associativity of addition and multiplication.

lemma (in int0**)** Int_ZF_1_1_L7: **assumes** A1: a∈ℤ b∈ℤ c∈ℤ
  shows
  a+b+c = a+(b+c)
  a·b·c = a·(b·c)
  ⟨*proof*⟩

24.2 Rearrangement lemmas

In this section we collect lemmas about identities related to rearranging the terms in expresssions

A formula with a positive integer.

lemma (in int0**)** Int_ZF_1_2_L1**: assumes** 0≤a
shows abs(a)+1 = abs(a+1)
⟨*proof*⟩

A formula with two integers, one positive.

lemma (in int0**)** Int_ZF_1_2_L2**: assumes** A1**:** a∈ℤ **and** A2**:** 0≤b
shows a+(abs(b)+1)·a = (abs(b+1)+1)·a
⟨*proof*⟩

A couple of formulae about canceling opposite integers.

lemma (in int0**)** Int_ZF_1_2_L3**: assumes** A1**:** a∈ℤ b∈ℤ
shows
a+b-a = b
a+(b-a) = b
a+b-b = a
a-b+b = a
(-a)+(a+b) = b
a+(b-a) = b
(-b)+(a+b) = a
a-(b+a) = -b
a-(a+b) = -b
a-(a-b) = b
a-b-a = -b
a-b - (a+b) = (-b)-b
⟨*proof*⟩

Subtracting one does not increase integers. This may be moved to a theory about ordered rings one day.

lemma (in int0**)** Int_ZF_1_2_L3A**: assumes** A1**:** a≤b
shows a-1 ≤ b
⟨*proof*⟩

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*⟩

A formula with a nonpositive integer.

lemma (in int0**)** Int_ZF_1_2_L4: **assumes** a≤**0**
shows abs(a)+**1** = abs(a-**1**)
⟨*proof*⟩

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*⟩

A rearrangement with four integers.

lemma (in int0**)** Int_ZF_1_2_L6:
assumes A1: a∈ℤ b∈ℤ c∈ℤ d∈ℤ
shows
a-(b-**1**)·c = (d-b·c)-(d-a-c)
⟨*proof*⟩

Some other rearrangements with two integers.

lemma (in int0**)** Int_ZF_1_2_L7: **assumes** a∈ℤ b∈ℤ
shows
a·b = (a-**1**)·b+b
a·(b+**1**) = a·b+a
(b+**1**)·a = b·a+a
(b+**1**)·a = a+b·a
⟨*proof*⟩

Another rearrangement with two integers.

lemma (in int0**)** Int_ZF_1_2_L8:
assumes A1: a∈ℤ b∈ℤ
shows a+**1**+(b+**1**) = b+a+**2**
⟨*proof*⟩

A couple of rearrangement with three integers.

lemma (in int0**)** Int_ZF_1_2_L9:
assumes a∈ℤ b∈ℤ c∈ℤ
shows
(a-b)+(b-c) = a-c
(a-b)-(a-c) = c-b
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
⟨*proof*⟩

Another couple of rearrangements with three integers.

lemma (in int0**)** Int_ZF_1_2_L9A:
assumes A1: a∈ℤ b∈ℤ c∈ℤ

shows (-(a-b-c)) = c+b-a
⟨*proof*⟩

Another rearrangement with three integers.

lemma (in int0**)** Int_ZF_1_2_L10:
assumes A1: a∈ℤ b∈ℤ c∈ℤ
shows (a+**1**)·b + (c+**1**)·b = (c+a+**2**)·b
⟨*proof*⟩

A technical rearrangement involing inequalities with absolute value.

lemma (in int0**)** Int_ZF_1_2_L10A:
assumes A1: a∈ℤ b∈ℤ c∈ℤ e∈ℤ
and A2: abs(a·b-c) ≤ d abs(b·a-e) ≤ f
shows abs(c-e) ≤ f+d
⟨*proof*⟩

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*⟩

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
⟨*proof*⟩

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*⟩

Rerrangement about adding linear functions.

lemma (in int0**)** Int_ZF_1_2_L14:
assumes a∈ℤ b∈ℤ c∈ℤ d∈ℤ x∈ℤ
shows (a·x + b) + (c·x + d) = (a+c)·x + (b+d)
⟨*proof*⟩

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∈ℤ b∈ℤ c∈ℤ d∈ℤ
and A2: a = b-c-d
shows

```
  d = b-a-c
  d = (-a)+b-c
  b = a+d+c
```
⟨*proof*⟩

A rearrangement with four integers. Property of groups.

lemma (in int0**)** Int_ZF_1_2_L16:
  assumes a∈ℤ b∈ℤ c∈ℤ d∈ℤ
  shows a+(b-c)+d = a+b+d-c
  ⟨*proof*⟩

Some rearrangements with three integers. Properties of groups.

lemma (in int0**)** Int_ZF_1_2_L17:
  assumes A1: a∈ℤ b∈ℤ c∈ℤ
  shows
  a+b-c+(c-b) = a
  a+(b+c)-c = a+b
⟨*proof*⟩

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*⟩

24.3 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. Since we don't have the theory of oredered rings we temporarily put some facts about integers as an ordered ring in this section.

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*⟩

Product of nonnegative integers is nonnegative.

lemma (in int0**)** Int_ZF_1_3_L2: **assumes** A1: **0**≤a **0**≤b
  shows 0≤a·b
⟨*proof*⟩

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*⟩

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(ℤ,IntegerAddition,IntegerMultiplication,IntegerOrder)
ring1(ℤ,IntegerAddition,IntegerMultiplication,IntegerOrder)
⟨*proof*⟩

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**≤a **1**≤b
and A2: **1** ≤ a·b
shows 1 ≤ a·(b+**1**)
⟨*proof*⟩

Product of integers that are greater that one is greater than one.

lemma (in int0**)** Int_ZF_1_3_L3:
assumes A1: **1**≤a **1**≤b
shows 1 ≤ a·b
⟨*proof*⟩

$|a \cdot (-b)| = |(-a) \cdot b| = |(-a) \cdot (-b)| = |a \cdot b|$ This is a property of ordered rings..

lemma (in int0**)** Int_ZF_1_3_L4**: assumes** a∈ℤ b∈ℤ
shows
abs((-a)·b) = abs(a·b)
abs(a·(-b)) = abs(a·b)
abs((-a)·(-b)) = abs(a·b)
⟨*proof*⟩

Absolute value of a product is the product of absolute values. Property of ordered rings.

lemma (in int0**)** Int_ZF_1_3_L5:
assumes A1: a∈ℤ b∈ℤ
shows abs(a·b) = abs(a)·abs(b)
⟨*proof*⟩

Double nonnegative is nonnegative. Property of ordered rings.

lemma (in int0**)** Int_ZF_1_3_L5A**: assumes 0**≤a
shows 0≤**2**·a
⟨*proof*⟩

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∈ℤ b∈ℤ
shows ¬(b≤a) ⟷ a**+1** ≤ b
⟨*proof*⟩

Another form of stating that there are no integers between integers m and $m+1$.

corollary (in int0**)** no_int_between**: assumes** A1**:** a∈ℤ b∈ℤ
shows b≤a ∨ a**+1** ≤ b
⟨*proof*⟩

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∈ℤ b∈ℤ **and** A2**:** ¬(b≤a)
shows a ≤ b**-1**
⟨*proof*⟩

Yet another form of stating that there are nointegers between m and $m+1$.

lemma (in int0**)** no_int_between1**:**
assumes A1**:** a≤b **and** A2**:** a≠b
shows
a**+1** ≤ b
a ≤ b**-1**
⟨*proof*⟩

We can decompose proofs into three cases: $a = b$, $a \leq b - 1b$ or $a \geq b + 1b$.

lemma (in int0**)** Int_ZF_1_3_L6B**: assumes** A1**:** a∈ℤ b∈ℤ
shows a=b ∨ (a ≤ b**-1**) ∨ (b**+1** ≤a)
⟨*proof*⟩

A special case of Int_ZF_1_3_L6B when $b = 0$. This allows to split the proofs in cases $a \leq -1$, $a = 0$ and $a \geq 1$.

corollary (in int0**)** Int_ZF_1_3_L6C**: assumes** A1**:** a∈ℤ
shows a**=0** ∨ (a ≤ **-1**) ∨ (**1**≤a)
⟨*proof*⟩

An integer is not less or equal zero iff it is greater or equal one.

lemma (in int0**)** Int_ZF_1_3_L7**: assumes** a∈ℤ
shows ¬(a≤**0**) ⟷ **1** ≤ a
⟨*proof*⟩

Product of positive integers is positive.

lemma (in int0**)** Int_ZF_1_3_L8**:**
assumes a∈ℤ b∈ℤ
and ¬(a≤**0**) ¬(b≤**0**)
shows ¬((a·b) ≤ **0**)

⟨*proof*⟩

If $a \cdot b$ is nonnegative and b is positive, then a is nonnegative. Proof by contradiction.

lemma (in int0**)** Int_ZF_1_3_L9:
assumes A1: a∈ℤ b∈ℤ
and A2: ¬(b≤**0**) **and** A3: a·b ≤ **0**
shows a≤**0**
⟨*proof*⟩

One integer is less or equal another iff the difference is nonpositive.

lemma (in int0**)** Int_ZF_1_3_L10:
assumes a∈ℤ b∈ℤ
shows a≤b ⟷ a-b ≤ **0**
⟨*proof*⟩

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
⟨*proof*⟩

We can simplify out a positive element on both sides of an inequality.

lemma (in int0**)** Int_ineq_simpl_positive:
assumes A1: a∈ℤ b∈ℤ c∈ℤ
and A2: a·c ≤ b·c **and** A4: ¬(c≤**0**)
shows a ≤ b
⟨*proof*⟩

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∈ℤ b∈ℤ
and A2: ¬(abs(a) ≤ abs(b))
shows ¬(abs(a) ≤ **0**)
⟨*proof*⟩

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**
⟨*proof*⟩

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
⟨*proof*⟩

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*⟩

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*⟩

A rearrangement with four integers and absolute value.

lemma (in int0**)** Int_ZF_1_3_L14:
assumes A1: a∈$\mathbb{Z}$ b∈$\mathbb{Z}$ c∈$\mathbb{Z}$ d∈$\mathbb{Z}$
shows abs(a·b)+(abs(a)+c)·d = (d+abs(b))·abs(a)+c·d
⟨*proof*⟩

A technical lemma about what happens when one absolute value is not greater or equal than another.

lemma (in int0**)** Int_ZF_1_3_L15: **assumes** A1: m∈$\mathbb{Z}$ n∈$\mathbb{Z}$
and A2: ¬(abs(m) ≤ abs(n))
shows n ≤ abs(m) m≠**0**
⟨*proof*⟩

Negative of a nonnegative is nonpositive.

lemma (in int0**)** Int_ZF_1_3_L16: **assumes** A1: **0** ≤ m
shows (-m) ≤ **0**
⟨*proof*⟩

Some statements about intervals centered at 0.

lemma (in int0**)** Int_ZF_1_3_L17: **assumes** A1: m∈$\mathbb{Z}$
shows
(-abs(m)) ≤ abs(m)
(-abs(m))..abs(m) ≠ 0
⟨*proof*⟩

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∈$\mathbb{Z}$ n∈$\mathbb{Z}$

shows
m ≤ GreaterOf(IntegerOrder,m,n)
n ≤ GreaterOf(IntegerOrder,m,n)
SmallerOf(IntegerOrder,m,n) ≤ m
SmallerOf(IntegerOrder,m,n) ≤ n
⟨*proof*⟩

If $|m| \leq n$, then $m \in -n..n$.

lemma (in int0**)** Int_ZF_1_3_L19:
assumes A1: m∈ℤ **and** A2: abs(m) ≤ n
shows
(-n) ≤ m m ≤ n
m ∈ (-n)..n
0 ≤ n
⟨*proof*⟩

A slight generalization of the above lemma.

lemma (in int0**)** Int_ZF_1_3_L19A:
assumes A1: m∈ℤ **and** A2: abs(m) ≤ n **and** A3: **0**≤k
shows (-(n+k)) ≤ m
⟨*proof*⟩

Sets of integers that have absolute value bounded are bounded.

lemma (in int0**)** Int_ZF_1_3_L20:
assumes A1: ∀x∈X. b(x) ∈ ℤ ∧ abs(b(x)) ≤ L
shows IsBounded({b(x). x∈X},IntegerOrder)
⟨*proof*⟩

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
⟨*proof*⟩

Absolute vaues 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
⟨*proof*⟩

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*⟩

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*⟩

A triangle inequality with three integers. Property of linearly ordered abelian groups.

lemma (in int0) int_triangle_ineq3:
assumes A1: a∈ℤ b∈ℤ c∈ℤ
shows abs(a-b-c) ≤ abs(a) + abs(b) + abs(c)
⟨*proof*⟩

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≤c b≤c **shows** a+b ≤ **2**·c
⟨*proof*⟩

If an integer a is between b and $b + c$, then $|b - a| \leq c$. Property of ordered groups.

lemma (in int0) Int_ZF_1_3_L22:
assumes a≤b **and** c∈ℤ **and** b≤ c+a
shows abs(b-a) ≤ c
⟨*proof*⟩

An application of the triangle inequality with four integers. Property of linearly ordered abelian groups.

lemma (in int0) Int_ZF_1_3_L22A:
assumes a∈ℤ b∈ℤ c∈ℤ d∈ℤ
shows abs(a-c) ≤ abs(a+b) + abs(c+d) + abs(b-d)
⟨*proof*⟩

If an integer a is between b and $b + c$, then $|b - a| \leq c$. Property of ordered groups. A version of Int_ZF_1_3_L22 with sligtly different assumptions.

lemma (in int0) Int_ZF_1_3_L23:
assumes A1: a≤b **and** A2: c∈ℤ **and** A3: b≤ a+c
shows abs(b-a) ≤ c
⟨*proof*⟩

24.4 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(ℤ) **and** A2: A≠0
shows
HasAmaximum(IntegerOrder,A)
HasAminimum(IntegerOrder,A)
Maximum(IntegerOrder,A) ∈ A
Minimum(IntegerOrder,A) ∈ A
∀x∈A. x ≤ Maximum(IntegerOrder,A)
∀x∈A. Minimum(IntegerOrder,A) ≤ x
Maximum(IntegerOrder,A) ∈ ℤ
Minimum(IntegerOrder,A) ∈ ℤ
⟨*proof*⟩

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)
HasAminimum(IntegerOrder,A)
Maximum(IntegerOrder,A) ∈ A
Minimum(IntegerOrder,A) ∈ A
∀x∈A. x ≤ Maximum(IntegerOrder,A)
∀x∈A. Minimum(IntegerOrder,A) ≤ x
Maximum(IntegerOrder,A) ∈ ℤ
Minimum(IntegerOrder,A) ∈ ℤ
⟨*proof*⟩

Nonempty set of integers that is bounded below attains its minimum.

theorem (in int0**)** int_bounded_below_has_min:
assumes A1: IsBoundedBelow(A,IntegerOrder) **and** A2: A≠0
shows
HasAminimum(IntegerOrder,A)
Minimum(IntegerOrder,A) ∈ A

∀x∈A. Minimum(IntegerOrder,A) ≤ x
⟨*proof*⟩

Nonempty set of integers that is bounded above attains its maximum.

theorem (in int0**)** int_bounded_above_has_max:
assumes A1: IsBoundedAbove(A,IntegerOrder) **and** A2: A≠0
shows
HasAmaximum(IntegerOrder,A)
Maximum(IntegerOrder,A) ∈ A
Maximum(IntegerOrder,A) ∈ ℤ
∀x∈A. x ≤ Maximum(IntegerOrder,A)
⟨*proof*⟩

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*⟩

A three element set has a maximume and minimum.

lemma (in int0) Int_ZF_1_4_L1A: **assumes** A1: a∈ℤ b∈ℤ c∈ℤ
shows
Maximum(IntegerOrder,{a,b,c}) ∈ ℤ
a ≤ Maximum(IntegerOrder,{a,b,c})
b ≤ Maximum(IntegerOrder,{a,b,c})
c ≤ Maximum(IntegerOrder,{a,b,c})
⟨*proof*⟩

Integer functions attain maxima and minima over intervals.

lemma (in int0) Int_ZF_1_4_L2:
assumes A1: f:ℤ→ℤ **and** A2: a≤b
shows
maxf(f,a..b) ∈ ℤ
∀c ∈ a..b. f(c) ≤ maxf(f,a..b)
∃c ∈ a..b. f(c) = maxf(f,a..b)
minf(f,a..b) ∈ ℤ
∀c ∈ a..b. minf(f,a..b) ≤ f(c)
∃c ∈ a..b. f(c) = minf(f,a..b)
⟨*proof*⟩

24.5 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 rephrasse some lemmas about the set of positive integers known from the theory of oredered grups.

The set of positive integers is closed under addition.

lemma (in int0) pos_int_closed_add:
shows $\mathbb{Z}_+$ {is closed under} IntegerAddition
⟨*proof*⟩

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∈$\mathbb{Z}_+$ b∈$\mathbb{Z}_+$ **shows** a+b ∈ $\mathbb{Z}_+$
⟨*proof*⟩

$\mathbb{Z}^+$ is bounded below.

lemma (in int0**)** Int_ZF_1_5_L1: **shows**
IsBoundedBelow($\mathbb{Z}^+$,IntegerOrder)
IsBoundedBelow($\mathbb{Z}_+$,IntegerOrder)
⟨*proof*⟩

Subsets of $\mathbb{Z}^+$ are bounded below.

lemma (in int0**)** Int_ZF_1_5_L1A: **assumes** A1: A ⊆ $\mathbb{Z}^+$
shows IsBoundedBelow(A,IntegerOrder)
⟨*proof*⟩

Subsets of $\mathbb{Z}_+$ are bounded below.

lemma (in int0**)** Int_ZF_1_5_L1B: **assumes** A1: A ⊆ $\mathbb{Z}_+$
shows IsBoundedBelow(A,IntegerOrder)
⟨*proof*⟩

Every nonempty subset of positive integers has a mimimum.

lemma (in int0**)** Int_ZF_1_5_L1C: **assumes** A ⊆ $\mathbb{Z}_+$ **and** A ≠ 0
shows
HasAminimum(IntegerOrder,A)
Minimum(IntegerOrder,A) ∈ A
∀x∈A. Minimum(IntegerOrder,A) ≤ x
⟨*proof*⟩

Infinite subsets of Z^+ do not have a maximum - If $A \subseteq Z^+$ 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 ⊆ $\mathbb{Z}^+$ **and** A2: A ∉ Fin($\mathbb{Z}$) **and** A3: D∈$\mathbb{Z}$
shows ∃n∈A. D≤n
⟨*proof*⟩

Infinite subsets of Z_+ do not have a maximum - If $A \subseteq Z_+$ then for every integer we can find one in the set that is not smaller. This is very similar to Int_ZF_1_5_L2, except we have $\mathbb{Z}_+$ instead of $\mathbb{Z}^+$ here.

lemma (in int0**)** Int_ZF_1_5_L2A:
assumes A1: A ⊆ $\mathbb{Z}_+$ **and** A2: A ∉ Fin($\mathbb{Z}$) **and** A3: D∈$\mathbb{Z}$
shows ∃n∈A. D≤n
⟨*proof*⟩

An integer is either positive, zero, or its opposite is postitive.

lemma (in int0**)** Int_decomp: **assumes** m∈$\mathbb{Z}$
shows Exactly_1_of_3_holds (m=**0**,m∈$\mathbb{Z}_+$,(-m)∈$\mathbb{Z}_+$)
⟨*proof*⟩

An integer is zero, positive, or it's inverse is positive.

lemma (in int0) int_decomp_cases: assumes m∈ℤ
shows m=**0** ∨ m∈$\mathbb{Z}_+$ ∨ (-m) ∈ $\mathbb{Z}_+$
⟨*proof*⟩

An integer is in the positive set iff it is greater or equal one.

lemma (in int0) Int_ZF_1_5_L3: shows m∈$\mathbb{Z}_+$ ⟷ **1**≤m
⟨*proof*⟩

The set of positive integers is closed under multiplication. The unfolded form.

lemma (in int0) pos_int_closed_mul_unfold:
assumes a∈$\mathbb{Z}_+$ b∈$\mathbb{Z}_+$
shows a·b ∈ $\mathbb{Z}_+$
⟨*proof*⟩

The set of positive integers is closed under multiplication.

lemma (in int0) pos_int_closed_mul: shows
$\mathbb{Z}_+$ {is closed under} IntegerMultiplication
⟨*proof*⟩

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(ℤ,IntegerAddition,IntegerMultiplication)
⟨*proof*⟩

Nonnegative integers are positive ones plus zero.

lemma (in int0) Int_ZF_1_5_L3A: shows $\mathbb{Z}^+$ = $\mathbb{Z}_+$ ∪ {**0**}
⟨*proof*⟩

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:ℤ→ℤ **and** A2: K∈ℤ N∈ℤ
shows ∃C∈ℤ. ∀n∈$\mathbb{Z}_+$. K ≤ f(n) + C ⟶ N≤n
⟨*proof*⟩

Absolute value is identity on positive integers.

lemma (in int0) Int_ZF_1_5_L4A:
assumes a∈$\mathbb{Z}_+$ **shows** abs(a) = a
⟨*proof*⟩

One and two are in $\mathbb{Z}_+$.

lemma (in int0) int_one_two_are_pos: shows 1 ∈ $\mathbb{Z}_+$ **2** ∈ $\mathbb{Z}_+$
⟨*proof*⟩

The image of $\mathbb{Z}_+$ by a function defined on integers is not empty.

lemma (in int0**)** Int_ZF_1_5_L5: **assumes** A1: f : $\mathbb{Z}\to$X
  shows f($\mathbb{Z}_+$) $\neq$ 0
⟨*proof*⟩

If n is positive, then $n-1$ is nonnegative.

lemma (in int0**)** Int_ZF_1_5_L6: **assumes** A1: n $\in \mathbb{Z}_+$
  shows
  0 $\leq$ n-**1**
  0 $\in$ **0**..(n-**1**)
  0..(n-**1**) $\subseteq \mathbb{Z}$
⟨*proof*⟩

Intgers greater than one in $\mathbb{Z}_+$ belong to $\mathbb{Z}_+$. This is a property of ordered groups and follows from OrderedGroup_ZF_1_L19, but Isabelle's simplifier has problems using that result directly, so we reprove it specifically for integers.

lemma (in int0**)** Int_ZF_1_5_L7: **assumes** a $\in \mathbb{Z}_+$ **and** a$\leq$b
  shows b $\in \mathbb{Z}_+$
⟨*proof*⟩

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 $\in \mathbb{Z}$
  ⟨*proof*⟩

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 $\leq$ GreaterOf(IntegerOrder,**1**,a)
  GreaterOf(IntegerOrder,**1**,a) $\in \mathbb{Z}_+$
  GreaterOf(IntegerOrder,**1**,a) + **1** $\in \mathbb{Z}_+$
  a $\leq$ GreaterOf(IntegerOrder,**1**,a) + **1**
  a $\neq$ GreaterOf(IntegerOrder,**1**,a) + **1**
  ⟨*proof*⟩

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) $\notin \mathbb{Z}_+$
  ⟨*proof*⟩

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
  ⟨*proof*⟩

A theorem about odd extensions. Recall from OrdereGroup_ZF.thy that the odd extension of an integer function f defined on $\mathbb{Z}_+$ is the odd function on $\mathbb{Z}$ equal to f on $\mathbb{Z}_+$. First we show that the odd extension is defined on $\mathbb{Z}$.

lemma (in int0**)** Int_ZF_1_5_L10: **assumes** f : $\mathbb{Z}_+ \to \mathbb{Z}$
shows OddExtension($\mathbb{Z}$,IntegerAddition,IntegerOrder,f) : $\mathbb{Z} \to \mathbb{Z}$
⟨*proof*⟩

On $\mathbb{Z}_+$, the odd extension of f is the same as f.

lemma (in int0**)** Int_ZF_1_5_L11: **assumes** f : $\mathbb{Z}_+ \to \mathbb{Z}$ **and** a ∈ $\mathbb{Z}_+$ **and**
g = OddExtension($\mathbb{Z}$,IntegerAddition,IntegerOrder,f)
shows g(a) = f(a)
⟨*proof*⟩

On -$\mathbb{Z}_+$, the value of the odd extension of f is the negative of $f(-a)$.

lemma (in int0**)** Int_ZF_1_5_L12:
assumes f : $\mathbb{Z}_+ \to \mathbb{Z}$ **and** a ∈ (-$\mathbb{Z}_+$) **and**
g = OddExtension($\mathbb{Z}$,IntegerAddition,IntegerOrder,f)
shows g(a) = -(f(-a))
⟨*proof*⟩

Odd extensions are odd on $\mathbb{Z}$.

lemma (in int0**)** int_oddext_is_odd:
assumes f : $\mathbb{Z}_+ \to \mathbb{Z}$ **and** a∈$\mathbb{Z}$ **and**
g = OddExtension($\mathbb{Z}$,IntegerAddition,IntegerOrder,f)
shows g(-a) = -(g(a))
⟨*proof*⟩

Alternative definition of an odd function.

lemma (in int0**)** Int_ZF_1_5_L13: **assumes** A1: f: $\mathbb{Z} \to \mathbb{Z}$ **shows**
(∀a∈$\mathbb{Z}$. f(-a) = (-f(a))) ⟷ (∀a∈$\mathbb{Z}$. (-(f(-a))) = f(a))
⟨*proof*⟩

Another way of expressing the fact that odd extensions are odd.

lemma (in int0**)** int_oddext_is_odd_alt:
assumes f : $\mathbb{Z}_+ \to \mathbb{Z}$ **and** a∈$\mathbb{Z}$ **and**
g = OddExtension($\mathbb{Z}$,IntegerAddition,IntegerOrder,f)
shows (-g(-a)) = g(a)
⟨*proof*⟩

24.6 Functions with infinite limits

In this section we consider functions (integer sequences) that have infinite limits. An integer function has infinite positive limit if it is arbitrarily large for large enough arguments. Similarly, a function has infinite negative limit if it is arbitrarily small for small enough arguments. The material in this come mostly from the section in OrderedGroup_ZF.thy with he same title.

Here we rewrite the theorems from that section in the notation we use for integers and add some results specific for the ordered group of integers.

If an image of a set by a function with infinite positive limit is bounded above, then the set itself is bounded above.

lemma (in int0**)** Int_ZF_1_6_L1: **assumes** f: $\mathbb{Z}\to\mathbb{Z}$ **and**
$\forall a\in\mathbb{Z}.\exists b\in\mathbb{Z}_+.\forall x.\ b\leq x \longrightarrow a \leq f(x)$ **and** $A \subseteq \mathbb{Z}$ **and**
IsBoundedAbove(f(A),IntegerOrder)
shows IsBoundedAbove(A,IntegerOrder)
⟨*proof*⟩

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: $\mathbb{Z}\to\mathbb{Z}$ **and**

A3: $\forall a\in\mathbb{Z}.\exists b\in\mathbb{Z}_+.\forall x.\ b\leq x \longrightarrow a \leq f(x)$ **and**
A4: $\forall x\in X.\ b(x) \in \mathbb{Z} \ \wedge f(b(x)) \leq U$
shows $\exists u.\forall x\in X.\ b(x) \leq u$
⟨*proof*⟩

If an image of a set defined by separation by a integer 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: $\mathbb{Z}\to\mathbb{Z}$ **and**

A3: $\forall a\in\mathbb{Z}.\exists b\in\mathbb{Z}_+.\forall y.\ b\leq y \longrightarrow f(-y) \leq a$ **and**
A4: $\forall x\in X.\ b(x) \in \mathbb{Z} \ \wedge L \leq f(b(x))$
shows $\exists l.\forall x\in X.\ l \leq b(x)$
⟨*proof*⟩

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.thy.

lemma (in int0**)** Int_ZF_1_6_L4:
assumes A1: $X\neq 0$ **and** A2: f: $\mathbb{Z}\to\mathbb{Z}$ **and**
A3: $\forall a\in\mathbb{Z}.\exists b\in\mathbb{Z}_+.\forall x.\ b\leq x \longrightarrow a \leq f(x)$ **and**
A4: $\forall a\in\mathbb{Z}.\exists b\in\mathbb{Z}_+.\forall y.\ b\leq y \longrightarrow f(-y) \leq a$ **and**
A5: $\forall x\in X.\ b(x) \in \mathbb{Z} \wedge f(b(x)) \leq U \wedge L \leq f(b(x))$
shows $\exists M.\forall x\in X.$ abs(b(x)) $\leq M$
⟨*proof*⟩

If a function is larger than some constant for arguments large enough, then the image of a set that is bounded below is bounded below. This is not true for ordered groups in general, but only for those for which bounded sets are finite. This does not require the function to have infinite limit, but such functions do have this property.

lemma (in int0**)** Int_ZF_1_6_L5:
assumes A1: f: $\mathbb{Z}\to\mathbb{Z}$ **and** A2: N$\in\mathbb{Z}$ **and**
A3: $\forall$m. N$\leq$m $\longrightarrow$ L $\leq$ f(m) **and**
A4: IsBoundedBelow(A,IntegerOrder)
shows IsBoundedBelow(f(A),IntegerOrder)
⟨*proof*⟩

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$\in\mathbb{Z}$ **and**
A2: $\forall$m. N$\leq$m $\longrightarrow$ L $\leq$ f(m) **and**
A3: f: $\mathbb{Z}\to\mathbb{Z}$ **and** A4: K$\in\mathbb{Z}$
shows $\exists$c$\in\mathbb{Z}$. $\forall$n$\in\mathbb{Z}_+$. K $\leq$ f(n)+c
⟨*proof*⟩

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: $\mathbb{Z}\to\mathbb{Z}$ **and** A2: K$\in\mathbb{Z}$ N$\in\mathbb{Z}$ **and**
A3: $\forall$a$\in\mathbb{Z}$.$\exists$b$\in\mathbb{Z}_+$.$\forall$x. b$\leq$x $\longrightarrow$ a $\leq$ f(x)
shows $\exists$C$\in\mathbb{Z}$. N $\leq$ Minimum(IntegerOrder,{n$\in\mathbb{Z}_+$. K $\leq$ f(n)+C})
⟨*proof*⟩

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 $\{a \cdot x : x \in Z\}$ can finite only if $a = 0$.

lemma (in int0**)** Int_ZF_1_6_L8:
assumes A1: a$\in\mathbb{Z}$ **and** A2: {a·x. x$\in\mathbb{Z}$} $\in$ Fin($\mathbb{Z}$)
shows a = **0**
⟨*proof*⟩

24.7 Miscelaneous

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 $F(p)|p| \leq A|p| + B$. 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: $\forall q \in \mathbb{Z}.\ F(q) \cdot abs(q) \leq A \cdot abs(q) + B$ **and**
A3: $A \in \mathbb{Z}$ $B \in \mathbb{Z}$
shows $\exists L.\ \forall p \in \mathbb{Z}.\ F(p) \leq L$
⟨*proof*⟩

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 qaudrant, 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
$\mathbf{0} \leq a \wedge \mathbf{0} \leq b \ \vee\ a \leq \mathbf{0} \wedge b \leq \mathbf{0} \ \vee$
$a \leq \mathbf{0} \wedge \mathbf{0} \leq b \wedge \mathbf{0} \leq a+b \ \vee\ a \leq \mathbf{0} \wedge \mathbf{0} \leq b \wedge a+b \leq \mathbf{0} \ \vee$
$\mathbf{0} \leq a \wedge b \leq \mathbf{0} \wedge \mathbf{0} \leq a+b \ \vee\ \mathbf{0} \leq a \wedge b \leq \mathbf{0} \wedge a+b \leq \mathbf{0}$
⟨*proof*⟩

end

25 IntDiv_ZF.thy

theory IntDiv_ZF **imports** Int_ZF_1 IntDiv

begin

This theory translates some results form the Isabelle's IntDiv.thy theory to the notation used by IsarMathLib.

25.1 Quotient and reminder

For any integers m, n , $n > 0$ there are unique integers q, p such that $0 \leq p < n$ and $m = n \cdot q + p$. Number p in this decompsition is usually called m mod n. Standard Isabelle denotes numbers q, p as m zdiv n and m zmod n, resp., and we will use the same notation.

The next lemma is sometimes called the "quotient-reminder theorem".

lemma (in int0) IntDiv_ZF_1_L1: **assumes** m∈ℤ n∈ℤ
shows m = n·(m zdiv n) + (m zmod n)
⟨*proof*⟩

If $n > 0$ then m zmod n is between 0 and $n - 1$.

lemma (in int0) IntDiv_ZF_1_L2:
assumes A1: m∈ℤ **and** A2: **0**≤n n≠**0**
shows
0 ≤ m zmod n
m zmod n ≤ n m zmod n ≠ n
m zmod n ≤ n-**1**
⟨*proof*⟩

$(m \cdot k)$ 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
⟨*proof*⟩

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*⟩

A quotient-reminder theorem about integers greater than a given product.

lemma (in int0) IntDiv_ZF_1_L5:

assumes A1: n ∈ $\mathbb{Z}_+$ **and** A2: n ≤ k **and** A3: k·n ≤ m
shows
m = n·(m zdiv n) + (m zmod n)
m = (m zdiv n)·n + (m zmod n)
(m zmod n) ∈ **0**..(n-**1**)
k ≤ (m zdiv n)
m zdiv n ∈ $\mathbb{Z}_+$
⟨*proof*⟩

end

26 Int_ZF_2.thy

theory `Int_ZF_2` **imports** `Int_ZF_1 IntDiv_ZF 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_x.thy` series.

26.1 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 $\mathcal{S}$. We also define "positive" slopes as those that take infinite number of positive values on positive integers. We write δ`(s,m,n)` to denote the homomorphism difference of s at m, n (i.e the expression $s(m+n) - s(m) - s(n)$). We denote `max`δ`(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.thy` we define the equivalence relation on almost homomorphisms using the notion of a quotient group relation and use "$\approx$" 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 "`+`" denotes the first operation on almost homomorphisms. For slopes this is addition of functions defined in the natural way. The "$\circ$" symbol denotes the second operation on almost homomorphisms (see `Group_ZF_3.thy` for definition), defined for the group of integers. In short "$\circ$" is the composition of slopes. The "$^{-1}$" symbol acts as an infix operator that assigns the value $\min\{n \in Z_+ : p \leq f(n)\}$ to a pair (of sets) f and p. In application f represents a function defined on 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 ε defined as $\varepsilon(f, \langle m, n\rangle) = f^{-1}(m+n) - f^{-1}(m) - f^{-1}(n)$. Of course the intention is to use the fact that $\varepsilon(f, \langle m, n\rangle)$ is the homomorphism difference of the function g defined as $g(m) = f^{-1}(m)$. We also define $\gamma(s, m, n)$ as the expression $\delta(f, m, -n) + s(0) - \delta(f, n, -n)$. This is useful because of the

identity $f(m-n) = \gamma(m,n) + f(m) - f(n)$ that allows to obtain bounds on the value of a slope at the difference of of two integers. For every integer m we introduce notation m^S defined by $m^E(n) = m \cdot n$. The mapping $q \mapsto q^S$ embeds integers into $\mathcal{S}$ preserving the order, (that is, maps positive integers into $\mathcal{S}_+$).

locale int1 = int0 +

fixes slopes ($\mathcal{S}$)
defines slopes_def [simp]: $\mathcal{S}$ ≡ AlmostHoms($\mathbb{Z}$,IntegerAddition)

fixes posslopes ($\mathcal{S}_+$)
defines posslopes_def [simp]: $\mathcal{S}_+$ ≡ {s∈$\mathcal{S}$. s($\mathbb{Z}_+$) ∩ $\mathbb{Z}_+$ ∉ Fin($\mathbb{Z}$)}

fixes δ
defines δ_def [simp] : δ(s,m,n) ≡ s(m+n)-s(m)-s(n)

fixes maxhomdiff (maxδ)
defines maxhomdiff_def [simp]:
maxδ(s) ≡ Maximum(IntegerOrder,{abs(δ(s,m,n)). <m,n> ∈ $\mathbb{Z}$×$\mathbb{Z}$})

fixes AlEqRel
defines AlEqRel_def [simp]:
AlEqRel ≡ QuotientGroupRel($\mathcal{S}$,AlHomOp1($\mathbb{Z}$,IntegerAddition),FinRangeFunctions($\mathbb{Z}$,$\mathbb{Z}$))

fixes AlEq :: [i,i]⇒o (**infix** ∼ 68)
defines AlEq_def [simp]: s ∼ r ≡ <s,r> ∈ AlEqRel

fixes slope_add (**infix** + 70)
defines slope_add_def [simp]: s + r ≡ AlHomOp1($\mathbb{Z}$,IntegerAddition)<s,r>

fixes slope_comp (**infix** ∘ 70)
defines slope_comp_def [simp]: s ∘ r ≡ AlHomOp2($\mathbb{Z}$,IntegerAddition)<s,r>

fixes neg :: i⇒i (-_ [90] 91)
defines neg_def [simp]: -s ≡ GroupInv($\mathbb{Z}$,IntegerAddition) ` s

fixes slope_inv (**infix** $^{-1}$ 71)
defines slope_inv_def [simp]:
f^{-1}(p) ≡ Minimum(IntegerOrder,{n∈$\mathbb{Z}_+$. 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]: $m^S \equiv \{\langle n,m\cdot n\rangle.\ n\in\mathbb{Z}\}$

We can use theorems proven in the group1 context.

lemma (in int1) Int_ZF_2_1_L1: **shows** group1($\mathbb{Z}$,IntegerAddition)
⟨*proof*⟩

Type information related to the homomorphism difference expression.

lemma (in int1) Int_ZF_2_1_L2: **assumes** $f\in\mathcal{S}$ **and** $n\in\mathbb{Z}$ $m\in\mathbb{Z}$
shows
$m+n \in \mathbb{Z}$
$f(m+n) \in \mathbb{Z}$
$f(m) \in \mathbb{Z}$ $f(n) \in \mathbb{Z}$
$f(m) + f(n) \in \mathbb{Z}$
HomDiff($\mathbb{Z}$,IntegerAddition,f,<m,n>) $\in \mathbb{Z}$
⟨*proof*⟩

Type information related to the homomorphism difference expression.

lemma (in int1) Int_ZF_2_1_L2A:
assumes $f:\mathbb{Z}\to\mathbb{Z}$ **and** $n\in\mathbb{Z}$ $m\in\mathbb{Z}$
shows
$m+n \in \mathbb{Z}$
$f(m+n) \in \mathbb{Z}$ $f(m) \in \mathbb{Z}$ $f(n) \in \mathbb{Z}$
$f(m) + f(n) \in \mathbb{Z}$
HomDiff($\mathbb{Z}$,IntegerAddition,f,<m,n>) $\in \mathbb{Z}$
⟨*proof*⟩

Slopes map integers into integers.

lemma (in int1) Int_ZF_2_1_L2B:
assumes A1: $f\in\mathcal{S}$ **and** A2: $m\in\mathbb{Z}$
shows $f(m) \in \mathbb{Z}$
⟨*proof*⟩

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 δ(f,m,n).

lemma (in int1) Int_ZF_2_1_L3:
assumes $f:\mathbb{Z}\to\mathbb{Z}$ **and** $m\in\mathbb{Z}$ $n\in\mathbb{Z}$
shows HomDiff($\mathbb{Z}$,IntegerAddition,f,<m,n>) = δ(f,m,n)
⟨*proof*⟩

The next formula restates the definition of the homomorphism difference to express the value an almost homomorphism on a sum.

lemma (in int1) Int_ZF_2_1_L3A:
assumes A1: $f\in\mathcal{S}$ **and** A2: $m\in\mathbb{Z}$ $n\in\mathbb{Z}$
shows
$f(m+n) = f(m)+(f(n)+\delta(f,m,n))$

⟨*proof*⟩

The homomorphism difference of any integer function is integer.

lemma (in int1**)** Int_ZF_2_1_L3B:
assumes f:ℤ→ℤ **and** m∈ℤ n∈ℤ
shows δ(f,m,n) ∈ ℤ
⟨*proof*⟩

The value of an integer function at a sum expressed in terms of δ.

lemma (in int1**)** Int_ZF_2_1_L3C: **assumes** A1: f:ℤ→ℤ **and** A2: m∈ℤ n∈ℤ
shows f(m+n) = δ(f,m,n) + f(n) + f(m)
⟨*proof*⟩

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:ℤ→ℤ
shows {abs(HomDiff(ℤ,IntegerAddition,f,x)). x ∈ ℤ×ℤ} =
{abs(δ(f,m,n)). <m,n> ∈ ℤ×ℤ}
⟨*proof*⟩

If f maps integers into integers and for all $m, n \in Z$ 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:ℤ→ℤ
and A2: ∀m∈ℤ.∀n∈ℤ. abs(δ(f,m,n)) ≤ L
shows f∈$\mathcal{S}$
⟨*proof*⟩

The absolute value of homomorphism difference of a slope s does not exceed maxδ(s).

lemma (in int1**)** Int_ZF_2_1_L7:
assumes A1: s∈$\mathcal{S}$ **and** A2: n∈ℤ m∈ℤ
shows
abs(δ(s,m,n)) ≤ maxδ(s)
δ(s,m,n) ∈ ℤ maxδ(s) ∈ ℤ
(-maxδ(s)) ≤ δ(s,m,n)
⟨*proof*⟩

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∈$\mathcal{S}$
shows
abs(s(**0**)) ≤ maxδ(s)
0 ≤ maxδ(s)
abs(s(**0**)) ∈ ℤ maxδ(s) ∈ ℤ
abs(s(**0**)) + maxδ(s) ∈ ℤ
⟨*proof*⟩

Int Group_ZF_3.thy we show that finite range functions valued in an abelian group form a normal subgroup of almost homomorphisms. This allows to define the equivalence relation between almost homomorphisms as the relation resulting from dividing by that normal subgroup. Then we show in Group_ZF_3_4_L12 that if the difference of f and g has finite range (actually $f(n) \cdot g(n)^{-1}$ as we use multiplicative notation in Group_ZF_3.thy), then f and g are equivalent. The next lemma translates that fact into the notation used in int1 context.

lemma (in int1) Int_ZF_2_1_L9: assumes A1: s∈$\mathcal{S}$ r∈$\mathcal{S}$
and A2: ∀m∈ℤ. abs(s(m)-r(m)) ≤ L
shows s ∼ r
⟨*proof*⟩

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 imples that $|f(m) - g(m)|$ is bounded (by some integer) as m varies over integers. We also mention here that in this context s ∼ r implies that both s and r are slopes.

lemma (in int1) Int_ZF_2_1_L9A: assumes s ∼ r
shows
∃L∈ℤ. ∀m∈ℤ. abs(s(m)-r(m)) ≤ L
s∈$\mathcal{S}$ r∈$\mathcal{S}$
⟨*proof*⟩

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 ⊆ $\mathcal{S}$×$\mathcal{S}$
equiv($\mathcal{S}$,AlEqRel)
⟨*proof*⟩

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∈$\mathcal{S}$ r∈$\mathcal{S}$ and
s + (-r) ∈ FinRangeFunctions(ℤ,ℤ)
shows
s ∼ r
r ∼ s
⟨*proof*⟩

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 ∼ r
shows s + (-r) ∈ FinRangeFunctions(ℤ,ℤ)
⟨*proof*⟩

What is the value of a composition of slopes?

lemma (in int1**)** Int_ZF_2_1_L10:
assumes s∈$\mathcal{S}$ r∈$\mathcal{S}$ **and** m∈$\mathbb{Z}$
shows (s∘r)(m) = s(r(m)) s(r(m)) ∈ $\mathbb{Z}$
⟨*proof*⟩

Composition of slopes is a slope.

lemma (in int1**)** Int_ZF_2_1_L11:
assumes s∈$\mathcal{S}$ r∈$\mathcal{S}$
shows s∘r ∈ $\mathcal{S}$
⟨*proof*⟩

Negative of a slope is a slope.

lemma (in int1**)** Int_ZF_2_1_L12: **assumes** s∈$\mathcal{S}$ **shows** -s ∈ $\mathcal{S}$
⟨*proof*⟩

What is the value of a negative of a slope?

lemma (in int1**)** Int_ZF_2_1_L12A:
assumes s∈$\mathcal{S}$ **and** m∈$\mathbb{Z}$ **shows** (-s)(m) = -(s(m))
⟨*proof*⟩

What are the values of a sum of slopes?

lemma (in int1**)** Int_ZF_2_1_L12B: **assumes** s∈$\mathcal{S}$ r∈$\mathcal{S}$ **and** m∈$\mathbb{Z}$
shows (s+r)(m) = s(m) + r(m)
⟨*proof*⟩

Sum of slopes is a slope.

lemma (in int1**)** Int_ZF_2_1_L12C: **assumes** s∈$\mathcal{S}$ r∈$\mathcal{S}$
shows s+r ∈ $\mathcal{S}$
⟨*proof*⟩

A simple but useful identity.

lemma (in int1**)** Int_ZF_2_1_L13:
assumes s∈$\mathcal{S}$ **and** n∈$\mathbb{Z}$ m∈$\mathbb{Z}$
shows s(n·m) + (s(m) + δ(s,n·m,m)) = s((n+**1**)·m)
⟨*proof*⟩

Some estimates for the absolute value of a slope at the opposite integer.

lemma (in int1**)** Int_ZF_2_1_L14: **assumes** A1: s∈$\mathcal{S}$ **and** A2: m∈$\mathbb{Z}$
shows
s(-m) = s(**0**) - δ(s,m,-m) - s(m)
abs(s(m)+s(-m)) ≤ **2**·maxδ(s)
abs(s(-m)) ≤ **2**·maxδ(s) + abs(s(m))
s(-m) ≤ abs(s(**0**)) + maxδ(s) - s(m)
⟨*proof*⟩

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:$\mathbb{Z}\to\mathbb{Z}$ **and** A2: m$\in\mathbb{Z}$
shows f(-m) = (-δ(f,m,-m)) + f(**0**) - f(m)
⟨*proof*⟩

The next lemma allows to use the expression maxf(f,**0**..M-**1**). Recall that maxf(f,A) is the maximum of (function) f on (the set) A.

lemma (in int1**)** Int_ZF_2_1_L15:
assumes s$\in\mathcal{S}$ **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)
⟨*proof*⟩

A lower estimate for the value of a slope at $nM + k$.

lemma (in int1**)** Int_ZF_2_1_L16:
assumes A1: s$\in\mathcal{S}$ **and** A2: m$\in\mathbb{Z}$ **and** A3: M $\in \mathbb{Z}_+$ **and** A4: k $\in$ **0**..(M-**1**)
shows s(m·M) + (minf(s,**0**..(M-**1**))- maxδ(s)) $\leq$ s(m·M+k)
⟨*proof*⟩

Identity is a slope.

lemma (in int1**)** Int_ZF_2_1_L17: **shows** id($\mathbb{Z}$) $\in \mathcal{S}$
⟨*proof*⟩

Simple identities about (absolute value of) homomorphism differences.

lemma (in int1**)** Int_ZF_2_1_L18:
assumes A1: f:$\mathbb{Z}\to\mathbb{Z}$ **and** A2: m$\in\mathbb{Z}$ n$\in\mathbb{Z}$
shows
abs(f(n) + f(m) - f(m+n)) = abs(δ(f,m,n))
abs(f(m) + f(n) - f(m+n)) = abs(δ(f,m,n))
(-(f(m))) - f(n) + f(m+n) = δ(f,m,n)
(-(f(n))) - f(m) + f(m+n) = δ(f,m,n)
abs((-f(m+n)) + f(m) + f(n)) = abs(δ(f,m,n))
⟨*proof*⟩

Some identities about the homomorphism difference of odd functions.

lemma (in int1**)** Int_ZF_2_1_L19:
assumes A1: f:$\mathbb{Z}\to\mathbb{Z}$ **and** A2: $\forall$x$\in\mathbb{Z}$. (-f(-x)) = f(x)
and A3: m$\in\mathbb{Z}$ n$\in\mathbb{Z}$
shows
abs(δ(f,-m,m+n)) = abs(δ(f,m,n))
abs(δ(f,-n,m+n)) = abs(δ(f,m,n))
δ(f,n,-(m+n)) = δ(f,m,n)
δ(f,m,-(m+n)) = δ(f,m,n)

```
  abs(δ(f,-m,-n)) = abs(δ(f,m,n))
```
⟨*proof*⟩

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:$\mathbb{Z}\to\mathbb{Z}$ **and**
  A2: $\forall$a$\in\mathbb{Z}_+$. $\forall$b$\in\mathbb{Z}_+$. abs(δ(f,a,b)) $\leq$ L **and**
  A3: m$\in\mathbb{Z}^+$ n$\in\mathbb{Z}_+$
  shows
  0 $\leq$ L
  abs(δ(f,m,n)) $\leq$ L + abs(f(**0**))
⟨*proof*⟩

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:$\mathbb{Z}\to\mathbb{Z}$ **and**
  A2: $\forall$a$\in\mathbb{Z}^+$. $\forall$b$\in\mathbb{Z}_+$. abs(δ(f,a,b)) $\leq$ L **and**
  A3: n$\in\mathbb{Z}^+$ m$\in\mathbb{Z}^+$
  shows abs(δ(f,m,n)) $\leq$ L + abs(f(**0**))
⟨*proof*⟩

If the homomorphism difference is bounded on $\mathbb{Z}_+\times\mathbb{Z}_+$, then it is bounded on $\mathbb{Z}^+\times\mathbb{Z}^+$.

lemma (in int1**)** Int_ZF_2_1_L22**: assumes** A1: f:$\mathbb{Z}\to\mathbb{Z}$ **and**
  A2: $\forall$a$\in\mathbb{Z}_+$. $\forall$b$\in\mathbb{Z}_+$. abs(δ(f,a,b)) $\leq$ L
  shows $\exists$M. $\forall$m$\in\mathbb{Z}^+$. $\forall$n$\in\mathbb{Z}^+$. abs(δ(f,m,n)) $\leq$ M
⟨*proof*⟩

For odd functions we can do better than in Int_ZF_2_1_L22: if the homomorphism difference of f is bounded on $\mathbb{Z}^+\times\mathbb{Z}^+$, then it is bounded on $\mathbb{Z}\times\mathbb{Z}$, hence f is a slope. Loong prof by splitting the $\mathbb{Z}\times\mathbb{Z}$ into six subsets.

lemma (in int1**)** Int_ZF_2_1_L23**: assumes** A1: f:$\mathbb{Z}\to\mathbb{Z}$ **and**
  A2: $\forall$a$\in\mathbb{Z}_+$. $\forall$b$\in\mathbb{Z}_+$. abs(δ(f,a,b)) $\leq$ L
  and A3: $\forall$x$\in\mathbb{Z}$. (-f(-x)) = f(x)
  shows f$\in\mathcal{S}$
⟨*proof*⟩

If the homomorphism difference of a function defined on positive integers is bounded, then the odd extension of this function is a slope.

lemma (in int1**)** Int_ZF_2_1_L24:
  assumes A1: f:$\mathbb{Z}_+\to\mathbb{Z}$ **and** A2: $\forall$a$\in\mathbb{Z}_+$. $\forall$b$\in\mathbb{Z}_+$. abs(δ(f,a,b)) $\leq$ L
  shows OddExtension($\mathbb{Z}$,IntegerAddition,IntegerOrder,f) $\in\mathcal{S}$
⟨*proof*⟩

Type information related to γ.

lemma (in int1**)** Int_ZF_2_1_L25:
assumes A1: f:$\mathbb{Z}\to\mathbb{Z}$ **and** A2: m$\in\mathbb{Z}$ n$\in\mathbb{Z}$
shows
δ(f,m,-n) $\in \mathbb{Z}$
δ(f,n,-n) $\in \mathbb{Z}$
(-δ(f,n,-n)) $\in \mathbb{Z}$
f(**0**) $\in \mathbb{Z}$
γ(f,m,n) $\in \mathbb{Z}$
⟨*proof*⟩

A couple of formulae involving $f(m-n)$ and $\gamma(f,m,n)$.

lemma (in int1**)** Int_ZF_2_1_L26:
assumes A1: f:$\mathbb{Z}\to\mathbb{Z}$ **and** A2: m$\in\mathbb{Z}$ n$\in\mathbb{Z}$
shows
f(m-n) = γ(f,m,n) + f(m) - f(n)
f(m-n) = γ(f,m,n) + (f(m) - f(n))
f(m-n) + (f(n) - γ(f,m,n)) = f(m)
⟨*proof*⟩

A formula expressing the difference between $f(m-n-k)$ and $f(m)-f(n)-f(k)$ in terms of γ.

lemma (in int1**)** Int_ZF_2_1_L26A:
assumes A1: f:$\mathbb{Z}\to\mathbb{Z}$ **and** A2: m$\in\mathbb{Z}$ n$\in\mathbb{Z}$ k$\in\mathbb{Z}$
shows
f(m-n-k) - (f(m)- f(n) - f(k)) = γ(f,m-n,k) + γ(f,m,n)
⟨*proof*⟩

If s is a slope, then $\gamma(s,m,n)$ is uniformly bounded.

lemma (in int1**)** Int_ZF_2_1_L27: **assumes** A1: s$\in\mathcal{S}$
shows $\exists$L$\in\mathbb{Z}$. $\forall$m$\in\mathbb{Z}$.$\forall$n$\in\mathbb{Z}$. abs(γ(s,m,n)) $\leq$ L
⟨*proof*⟩

If s is a slope, then $s(m) \leq s(m-1) + M$, where L does not depend on m.

lemma (in int1**)** Int_ZF_2_1_L28: **assumes** A1: s$\in\mathcal{S}$
shows $\exists$M$\in\mathbb{Z}$. $\forall$m$\in\mathbb{Z}$. s(m) $\leq$ s(m-**1**) + M
⟨*proof*⟩

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$\in\mathcal{S}$
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*⟩

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∈$\mathcal{S}$
shows
∃M∈ℤ. ∀m∈ℤ.∀n∈ℤ.∀k∈ℤ. s(m-n-k) ≤ s(m)-s(n)-s(k)+M
∃K∈ℤ. ∀m∈ℤ.∀n∈ℤ.∀k∈ℤ. s(m)-s(n)-s(k)+K ≤ s(m-n-k)
⟨*proof*⟩

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∈$\mathcal{S}$ r∈$\mathcal{S}$
and A2: ∀m∈$\mathbb{Z}_+$. abs(s(m)-r(m)) ≤ L
shows s ∼ r
⟨*proof*⟩

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 plus some constant independent of m, then the slope is almost identity.

lemma (in int1**)** Int_ZF_2_1_L32: **assumes** A1: s∈$\mathcal{S}$ M∈ℤ
and A2: ∀m∈$\mathbb{Z}_+$. m ≤ s(m) ∧ s(m) ≤ m+M
shows s ∼ id(ℤ)
⟨*proof*⟩

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∈$\mathcal{S}$ **and** A2: c∈ℤ **and**
A3: r = {⟨m,s(m)+c⟩. m∈ℤ}
shows
∀m∈ℤ. r(m) = s(m)+c
r∈$\mathcal{S}$
s ∼ r
⟨*proof*⟩

26.2 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:ℤ→ℤ **and** A2: p∈ℤ q∈ℤ
shows
abs(f((p+**1**)·q)-(p+**1**)·f(q)) ≤ abs(δ(f,p·q,q))+abs(f(p·q)-p·f(q))
abs(f((p-**1**)·q)-(p-**1**)·f(q)) ≤ abs(δ(f,(p-**1**)·q,q))+abs(f(p·q)-p·f(q))

⟨*proof*⟩

If f is a slope, then $|f(p \cdot q) - p \cdot f(q)| \leq (|p| + 1) \cdot \mathtt{max}\delta\mathtt{(f)}$. 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∈$\mathcal{S}$ **and** A2: **0**≤p q∈$\mathbb{Z}$
  and A3: abs(f(p·q)-p·f(q)) ≤ (abs(p)+**1**)·maxδ(f)
  shows
  abs(f((p+**1**)·q)-(p+**1**)·f(q)) ≤ (abs(p+**1**)+ **1**)·maxδ(f)
⟨*proof*⟩

If f is a slope, then $|f(p \cdot q) - p \cdot f(q)| \leq (|p| + 1) \cdot \mathtt{max}\delta$. The proof is by induction on p and the next lemma is the induction step for the case when $p \leq 0$.

lemma (in int1**)** Int_ZF_2_2_L3:
  assumes A1: f∈$\mathcal{S}$ **and** A2: p≤**0** q∈$\mathbb{Z}$
  and A3: abs(f(p·q)-p·f(q)) ≤ (abs(p)+**1**)·maxδ(f)
  shows abs(f((p-**1**)·q)-(p-**1**)·f(q)) ≤ (abs(p-**1**)+ **1**)·maxδ(f)
⟨*proof*⟩

If f is a slope, then $|f(p \cdot q) - p \cdot f(q)| \leq (|p| + 1) \cdot \mathtt{max}\delta(f)$.

lemma (in int1**)** Int_ZF_2_2_L4:
  assumes A1: f∈$\mathcal{S}$ **and** A2: p∈$\mathbb{Z}$ q∈$\mathbb{Z}$
  shows abs(f(p·q)-p·f(q)) ≤ (abs(p)+**1**)·maxδ(f)
⟨*proof*⟩

The next elegant result is Lemma 7 in the Arthan's paper [2] .

lemma (in int1**)** Arthan_Lem_7:
 assumes A1: f∈$\mathcal{S}$ **and** A2: p∈$\mathbb{Z}$ q∈$\mathbb{Z}$
  shows abs(q·f(p)-p·f(q)) ≤ (abs(p)+abs(q)+**2**)·maxδ(f)
⟨*proof*⟩

This is Lemma 8 in the Arthan's paper.

lemma (in int1**)** Arthan_Lem_8: **assumes** A1: f∈$\mathcal{S}$
  shows ∃A B. A∈$\mathbb{Z}$ ∧ B∈$\mathbb{Z}$ ∧ (∀p∈$\mathbb{Z}$. abs(f(p)) ≤ A·abs(p)+B)
⟨*proof*⟩

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∈$\mathcal{S}$ g∈$\mathcal{S}$
  shows f∘g ∼ g∘f
⟨*proof*⟩

26.3 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: $f \in \mathcal{S}_+$ **shows** $f: \mathbb{Z} \to \mathbb{Z}$
⟨*proof*⟩

A small technical lemma to simplify the proof of the next theorem.

lemma (in int1**)** Int_ZF_2_3_L1A:
assumes A1: $f \in \mathcal{S}_+$ **and** A2: $\exists n \in f(\mathbb{Z}_+) \cap \mathbb{Z}_+ .\ a \leq n$
shows $\exists M \in \mathbb{Z}_+ .\ a \leq f(M)$
⟨*proof*⟩

The next lemma is Lemma 3 in the Arthan's paper.

lemma (in int1**)** Arthan_Lem_3:
assumes A1: $f \in \mathcal{S}_+$ **and** A2: $D \in \mathbb{Z}_+$
shows $\exists M \in \mathbb{Z}_+ .\ \forall m \in \mathbb{Z}_+ .\ (m+1) \cdot D \leq f(m \cdot M)$
⟨*proof*⟩

A special case of Arthan_Lem_3 when $D = 1$.

corollary (in int1**)** Arthan_L_3_spec: **assumes** A1: $f \in \mathcal{S}_+$
shows $\exists M \in \mathbb{Z}_+ . \forall n \in \mathbb{Z}_+ .\ n+1 \leq f(n \cdot M)$
⟨*proof*⟩

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 $\mathcal{S}_+$. 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 $\mathcal{S}_+$ 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: $f \in$ FinRangeFunctions$(\mathbb{Z}, \mathbb{Z})$
shows $f \in \mathcal{S}$ $\quad f \notin \mathcal{S}_+$
⟨*proof*⟩

We want to show that if f is a slope and neither f nor $-f$ are in $\mathcal{S}_+$, then f is bounded. The next lemma is the first step towards that goal and shows that if slope is not in $\mathcal{S}_+$ then $f(\mathbb{Z}_+)$ is bounded above.

lemma (in int1**)** Int_ZF_2_3_L2: **assumes** A1: $f \in \mathcal{S}$ **and** A2: $f \notin \mathcal{S}_+$
shows IsBoundedAbove$(f(\mathbb{Z}_+),$ IntegerOrder$)$
⟨*proof*⟩

If f is a slope and $-f \notin \mathcal{S}_+$, then $f(\mathbb{Z}_+)$ is bounded below.

lemma (in int1**)** Int_ZF_2_3_L3: **assumes** A1: $f \in \mathcal{S}$ **and** A2: $-f \notin \mathcal{S}_+$
shows IsBoundedBelow$(f(\mathbb{Z}_+),$ IntegerOrder$)$
⟨*proof*⟩

A slope that is bounded on $\mathbb{Z}_+$ is bounded everywhere.

lemma (in int1**)** Int_ZF_2_3_L4:
  assumes A1: f∈$\mathcal{S}$ **and** A2: m∈$\mathbb{Z}$
  and A3: ∀n∈$\mathbb{Z}_+$. abs(f(n)) ≤ L
  shows abs(f(m)) ≤ **2**·maxδ(f) + L
⟨*proof*⟩

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∈$\mathcal{S}$ **and** A2: IsBounded(f($\mathbb{Z}_+$), IntegerOrder)
  shows f ∈ FinRangeFunctions($\mathbb{Z}$,$\mathbb{Z}$)
⟨*proof*⟩

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∈$\mathcal{S}$ **and** IsBoundedBelow(f($\mathbb{Z}_+$), IntegerOrder)
  shows f ∈ FinRangeFunctions($\mathbb{Z}$,$\mathbb{Z}$) ∨ f∈$\mathcal{S}_+$
  ⟨*proof*⟩

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∈$\mathcal{S}$ g∈$\mathcal{S}$ **and**
  A2: ∀n∈$\mathbb{Z}_+$. f(n) ≤ g(n)
  shows f∼g ∨ g + (-f) ∈ $\mathcal{S}_+$
⟨*proof*⟩

Positive slopes are arbitrarily large for large enough arguments.

lemma (in int1**)** Int_ZF_2_3_L5:
  assumes A1: f∈$\mathcal{S}_+$ **and** A2: K∈$\mathbb{Z}$
  shows ∃N∈$\mathbb{Z}_+$. ∀m. N≤m ⟶ K ≤ f(m)
⟨*proof*⟩

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∈$\mathcal{S}_+$ **and** A2: K∈$\mathbb{Z}$
  shows ∃N∈$\mathbb{Z}_+$. ∀m. N≤m ⟶ f(-m) ≤ K
⟨*proof*⟩

A special case of Int_ZF_2_3_L5 where $K = 1$.

corollary (in int1**)** Int_ZF_2_3_L6: **assumes** f∈$\mathcal{S}_+$
  shows ∃N∈$\mathbb{Z}_+$. ∀m. N≤m ⟶ f(m) ∈ $\mathbb{Z}_+$
  ⟨*proof*⟩

A special case of Int_ZF_2_3_L5 where $m = N$.

corollary (in int1**)** Int_ZF_2_3_L6A: **assumes** f∈$\mathcal{S}_+$ **and** K∈$\mathbb{Z}$
  shows ∃N∈$\mathbb{Z}_+$. K ≤ f(N)

⟨*proof*⟩

If values of a slope are not bounded above, then the slope is positive.

lemma (in int1**)** Int_ZF_2_3_L7: **assumes** A1: f∈$\mathcal{S}$
and A2: ∀K∈$\mathbb{Z}$. ∃n∈$\mathbb{Z}_+$. K ≤ f(n)
shows f ∈ $\mathcal{S}_+$
⟨*proof*⟩

For unbounded slope f either $f \in \mathcal{S}_+$ of $-f \in \mathcal{S}_+$.

theorem (in int1**)** Int_ZF_2_3_L8:
assumes A1: f∈$\mathcal{S}$ **and** A2: f ∉ FinRangeFunctions($\mathbb{Z}$,$\mathbb{Z}$)
shows (f ∈ $\mathcal{S}_+$) Xor ((-f) ∈ $\mathcal{S}_+$)
⟨*proof*⟩

The sum of positive slopes is a positive slope.

theorem (in int1**)** sum_of_pos_sls_is_pos_sl:
assumes A1: f ∈ $\mathcal{S}_+$ g ∈ $\mathcal{S}_+$
shows f+g ∈ $\mathcal{S}_+$
⟨*proof*⟩

The composition of positive slopes is a positive slope.

theorem (in int1**)** comp_of_pos_sls_is_pos_sl:
assumes A1: f ∈ $\mathcal{S}_+$ g ∈ $\mathcal{S}_+$
shows f∘g ∈ $\mathcal{S}_+$
⟨*proof*⟩

A slope equivalent to a positive one is positive.

lemma (in int1**)** Int_ZF_2_3_L9:
assumes A1: f ∈ $\mathcal{S}_+$ **and** A2: ⟨f,g⟩ ∈ AlEqRel **shows** g ∈ $\mathcal{S}_+$
⟨*proof*⟩

The set of positive slopes is saturated with respect to the relation of equivalence of slopes.

lemma (in int1**)** pos_slopes_saturated: **shows** IsSaturated(AlEqRel,$\mathcal{S}_+$)
⟨*proof*⟩

A technical lemma involving a projection of the set of positive slopes and a logical epression with exclusive or.

lemma (in int1**)** Int_ZF_2_3_L10:
assumes A1: f∈$\mathcal{S}$ g∈$\mathcal{S}$
and A2: R = {AlEqRel{s}. s∈$\mathcal{S}_+$}
and A3: (f∈$\mathcal{S}_+$) Xor (g∈$\mathcal{S}_+$)
shows (AlEqRel{f} ∈ R) Xor (AlEqRel{g} ∈ R)
⟨*proof*⟩

Identity function is a positive slope.

lemma (in int1**)** Int_ZF_2_3_L11: **shows** id($\mathbb{Z}$) ∈ $\mathcal{S}_+$

⟨*proof*⟩

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*⟩

26.4 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 $f \circ g$ 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 $\{n \in Z_+ : p \leq f(n)\}$ 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 ∈ $\mathcal{S}_+$ **and** A2: p∈$\mathbb{Z}_+$ **and** A3: A = {n∈$\mathbb{Z}_+$. p ≤ f(n)}
shows
A ⊆ $\mathbb{Z}_+$
A ≠ 0
f^{-1}(p) ∈ A
∀m∈A. f^{-1}(p) ≤ m
⟨*proof*⟩

If f is a positive slope and p is a positive integer p, then $f^{-1}(p)$ (defined as the minimum of the set $\{n \in Z_+ : p \leq f(n)\}$) is a (well defined) positive integer.

lemma (in int1**)** Int_ZF_2_4_L2:
assumes f ∈ $\mathcal{S}_+$ **and** p∈$\mathbb{Z}_+$
shows
f^{-1}(p) ∈ $\mathbb{Z}_+$
p ≤ f(f^{-1}(p))
⟨*proof*⟩

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 ∈ $\mathcal{S}_+$ **and** m∈$\mathbb{Z}_+$ p∈$\mathbb{Z}_+$ **and** m ≤ f(p)
shows f^{-1}(m) ≤ p
⟨*proof*⟩

An upper bound $f(f^{-1}(m) - 1)$ for positive slopes.

lemma (in int1**)** Int_ZF_2_4_L4:
assumes A1: f ∈ $\mathcal{S}_+$ **and** A2: m∈$\mathbb{Z}_+$ **and** A3: f^{-1}(m)-1 ∈ $\mathbb{Z}_+$
shows f(f^{-1}(m)-1) ≤ m f(f^{-1}(m)-1) ≠ m

⟨*proof*⟩

The (candidate for) the inverse of a positive slope is nondecreasing.

lemma (in int1**)** Int_ZF_2_4_L5:
assumes A1: f ∈ $\mathcal{S}_+$ **and** A2: m∈$\mathbb{Z}_+$ **and** A3: m≤n
shows f^{-1}(m) ≤ f^{-1}(n)
⟨*proof*⟩

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 ∈ $\mathcal{S}_+$ **and** A2: m∈$\mathbb{Z}_+$ n∈$\mathbb{Z}_+$ **and**
A3: f^{-1}(m)-1 ∈ $\mathbb{Z}_+$
shows f^{-1}(m+n)-1 ∈ $\mathbb{Z}_+$
⟨*proof*⟩

If f is a slope, then $f(f^{-1}(m+n)-f^{-1}(m)-f^{-1}(n))$ is uniformly bounded above and below. Will it be the messiest IsarMathLib proof ever? Only time will tell.

lemma (in int1**)** Int_ZF_2_4_L7: **assumes** A1: f ∈ $\mathcal{S}_+$ **and**
A2: ∀m∈$\mathbb{Z}_+$. f^{-1}(m)-1 ∈ $\mathbb{Z}_+$
shows
∃U∈$\mathbb{Z}$. ∀m∈$\mathbb{Z}_+$. ∀n∈$\mathbb{Z}_+$. f(f^{-1}(m+n)-f^{-1}(m)-f^{-1}(n)) ≤ U
∃N∈$\mathbb{Z}$. ∀m∈$\mathbb{Z}_+$. ∀n∈$\mathbb{Z}_+$. N ≤ f(f^{-1}(m+n)-f^{-1}(m)-f^{-1}(n))
⟨*proof*⟩

The expression $f^{-1}(m+n)-f^{-1}(m)-f^{-1}(n)$ is uniformly bounded for all pairs $\langle m,n\rangle \in \mathbb{Z}_+\times\mathbb{Z}_+$. Recall that in the int1 context ε(f,x) is defined so that $\varepsilon(f,\langle m,n\rangle) = f^{-1}(m+n)-f^{-1}(m)-f^{-1}(n)$.

lemma (in int1**)** Int_ZF_2_4_L8: **assumes** A1: f ∈ $\mathcal{S}_+$ **and**
A2: ∀m∈$\mathbb{Z}_+$. f^{-1}(m)-1 ∈ $\mathbb{Z}_+$
shows ∃M. ∀x∈$\mathbb{Z}_+\times\mathbb{Z}_+$. abs($\varepsilon$(f,x)) ≤ M
⟨*proof*⟩

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 ∈ $\mathcal{S}_+$ **and** A2: g = {⟨p,f^{-1}(p)⟩. p∈$\mathbb{Z}_+$}
shows
g : $\mathbb{Z}_+\to\mathbb{Z}_+$
g : $\mathbb{Z}_+\to\mathbb{Z}$
⟨*proof*⟩

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 ∈ $\mathcal{S}_+$ **and** A2: g = {⟨p,f^{-1}(p)⟩. p∈$\mathbb{Z}_+$} **and** A3: p∈$\mathbb{Z}_+$
shows g(p) = f^{-1}(p)

⟨*proof*⟩

The (candidate for) the inverse of a positive slope is a slope.

lemma (in int1**) Int_ZF_2_4_L11: assumes** A1: f ∈ $\mathcal{S}_+$ **and**
A2: ∀m∈$\mathbb{Z}_+$. f^{-1}(m)-**1** ∈ $\mathbb{Z}_+$ **and**
A3: g = {⟨p,f^{-1}(p)⟩. p∈$\mathbb{Z}_+$}
shows OddExtension($\mathbb{Z}$,IntegerAddition,IntegerOrder,g) ∈ $\mathcal{S}$
⟨*proof*⟩

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 ∈ $\mathcal{S}_+$ **and**
A2: ∀m∈$\mathbb{Z}_+$. f^{-1}(m)-**1** ∈ $\mathbb{Z}_+$
shows ∃h∈$\mathcal{S}$. f∘h ∼ id($\mathbb{Z}$)
⟨*proof*⟩

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 ∈ $\mathcal{S}_+$
shows ∃g∈$\mathcal{S}$. f∼g ∧ (∃h∈$\mathcal{S}$. g∘h ∼ id($\mathbb{Z}$))
⟨*proof*⟩

26.5 Completeness

In this section we consider properties of slopes that are needed for the proof of completeness of real numbers constructred 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 ∈ $\mathbb{Z}$
shows
∀n ∈ $\mathbb{Z}$. (m^S)(n) = m·n
m^S ∈ $\mathcal{S}$
⟨*proof*⟩

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 $\in \mathcal{S}$
shows $\exists$m$\in\mathbb{Z}$. $\exists$g$\in\mathcal{S}$. (m$^S\sim$g $\wedge$ (f$\sim$g $\vee$ g+(-f) $\in \mathcal{S}_+$))
⟨*proof*⟩

The negative of an integer embeds in slopes as a negative of the orgiginal embedding.

lemma (in int1**)** Int_ZF_2_5_L3: **assumes** A1: m $\in \mathbb{Z}$
shows (-m)^S = -(m^S)
⟨*proof*⟩

The sum of embeddings is the embeding of the sum.

lemma (in int1**)** Int_ZF_2_5_L3A: **assumes** A1: m$\in\mathbb{Z}$ k$\in\mathbb{Z}$
shows (m^S) + (k^S) = ((m+k)^S)
⟨*proof*⟩

The composition of embeddings is the embeding of the product.

lemma (in int1**)** Int_ZF_2_5_L3B: **assumes** A1: m$\in\mathbb{Z}$ k$\in\mathbb{Z}$
shows (m^S) $\circ$ (k^S) = ((m$\cdot$k)^S)
⟨*proof*⟩

Embedding integers in slopes preserves order.

lemma (in int1**)** Int_ZF_2_5_L4: **assumes** A1: m$\leq$n
shows (m^S) $\sim$ (n^S) $\vee$ (n^S)+(-(m^S)) $\in \mathcal{S}_+$
⟨*proof*⟩

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($\mathbb{Z}$,$\mathbb{Z}$)
shows m=**0**
⟨*proof*⟩

Embeddings of two integers are almost equal only if the integers are equal.

lemma (in int1**)** Int_ZF_2_5_L6:
assumes A1: m$\in\mathbb{Z}$ k$\in\mathbb{Z}$ **and** A2: (m^S) $\sim$ (k^S)
shows m=k
⟨*proof*⟩

Embedding of 1 is the identity slope and embedding of zero is a finite range function.

lemma (in int1**)** Int_ZF_2_5_L7: **shows**
1^S = id($\mathbb{Z}$)
0^S $\in$ FinRangeFunctions($\mathbb{Z}$,$\mathbb{Z}$)
⟨*proof*⟩

A somewhat technical condition for a 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 $\in \mathcal{S}$ **and** A2: N $\in \mathbb{Z}$ M $\in \mathbb{Z}$ **and**
  A3: $\forall$n$\in\mathbb{Z}_+$. M·n $\leq$ f(N·n)
  shows $M^S \sim f\circ(N^S) \vee$ (f∘(N^S)) + (-(M^S)) $\in \mathcal{S}_+$
⟨*proof*⟩

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 $\in \mathcal{S}$ **and** A2: N $\in \mathbb{Z}$ M $\in \mathbb{Z}$ **and**
  A3: $\forall$n$\in\mathbb{Z}_+$. f(N·n) $\leq$ M·n
  shows $f\circ(N^S) \sim (M^S) \vee (M^S)$ + (-(f∘(N^S))) $\in \mathcal{S}_+$
⟨*proof*⟩

end

27 Real_ZF.thy

theory `Real_ZF` **imports** `Int_ZF Ring_ZF_1`

begin

The goal of the `Real_ZF` series of theory files is to provide a contruction 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 contruction 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 $s : Z \to Z$ such that the set $\{s(m+n) - s(m) - s(n)\}_{n,m \in Z}$ is finite (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 $s(Z)$ 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.

27.1 The definition of real numbers

First we define slopes and real numbers as the set of their classes. The definition of slopes references the notion of almost homomorphisms defined in `Group_ZF_2.thy`: slopes are defined as almost homomorphisms on integers with integer addition as the operation. Similarly the notions of the first and second operation on slopes (which is really the addition and composition of slopes) is derived as a special case of the first and second operation on almost homomorphisms.

constdefs

```
Slopes ≡ AlmostHoms(int,IntegerAddition)

SlopeOp1 ≡ AlHomOp1(int,IntegerAddition)

SlopeOp2 ≡ AlHomOp2(int,IntegerAddition)

BoundedIntMaps ≡ FinRangeFunctions(int,int)

SlopeEquivalenceRel ≡ QuotientGroupRel(Slopes,SlopeOp1,BoundedIntMaps)

RealNumbers ≡ Slopes//SlopeEquivalenceRel

RealAddition ≡ ProjFun2(Slopes,SlopeEquivalenceRel,SlopeOp1)
```

```
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)`
⟨*proof*⟩

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: IsAring(RealNumbers,RealAddition,RealMultiplication)`
⟨*proof*⟩

We can use theorems proven in `group0` and `group1` contexts applied to the group of real numbers.

lemma `Real_ZF_1_L2:`
```
  group0(RealNumbers,RealAddition)
  RealAddition {is commutative on} RealNumbers
  group1(RealNumbers,RealAddition)
```
⟨*proof*⟩

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 :: i⇒i (- _ 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 ($\mathbb{R}_0$)
defines non_zero_def[simp]: $\mathbb{R}_0$ ≡ $\mathbb{R}$-{**0**}

fixes inv ($\_^{-1}$ [90] 91)
defines inv_def[simp]:
a^{-1} ≡ GroupInv($\mathbb{R}_0$,restrict(RealMultiplication,$\mathbb{R}_0\times\mathbb{R}_0$))(a)

In real0 context all theorems proven in the ring0, context are valid.

lemma (in real0) Real_ZF_1_L3: **shows**
ring0($\mathbb{R}$,RealAddition,RealMultiplication)
⟨*proof*⟩

Lets try out our notation to see that zero and one are real numbers.

lemma (in real0) Real_ZF_1_L4: **shows** **0**∈$\mathbb{R}$ **1**∈$\mathbb{R}$
⟨*proof*⟩

The lemma below lists some properties that require one real number to state.

lemma (in real0) Real_ZF_1_L5: **assumes** A1: a∈$\mathbb{R}$
shows
(-a) ∈ $\mathbb{R}$
(-(-a)) = a
a+**0** = a
0+a = a
a·**1** = a
1·a = a
a-a = **0**
a-**0** = a
⟨*proof*⟩

The lemma below lists some properties that require two real numbers to state.

lemma (in real0) Real_ZF_1_L6: **assumes** a∈$\mathbb{R}$ b∈$\mathbb{R}$
shows
a+b ∈ $\mathbb{R}$
a-b ∈ $\mathbb{R}$
a·b ∈ $\mathbb{R}$
a+b = b+a
(-a)·b = -(a·b)
a·(-b) = -(a·b)
⟨*proof*⟩

Multiplication of reals is associative.

lemma (in real0) Real_ZF_1_L6A: **assumes** a∈$\mathbb{R}$ b∈$\mathbb{R}$ c∈$\mathbb{R}$

shows a·(b·c) = (a·b)·c
⟨*proof*⟩

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
⟨*proof*⟩

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
⟨*proof*⟩

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 appriopriate 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)
⟨*proof*⟩

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 appriopriate 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
⟨*proof*⟩

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*⟩

We will need a couple of results from Group_ZF_3.thy The first two that state that the definition of addition and multiplication of real numbers are consistent, that is the result does not depend on the choice of the slopes representing the numbers. The second one implies that what we call SlopeEquivalenceRel is actually an equivalence relation on the set of slopes. We also show that the neutral element of the multiplicative operation on reals (in short number 1) is the class of the identity function on integers.

lemma Real_ZF_1_L11: **shows**
Congruent2(SlopeEquivalenceRel,SlopeOp1)
Congruent2(SlopeEquivalenceRel,SlopeOp2)
SlopeEquivalenceRel ⊆ Slopes × Slopes
equiv(Slopes, SlopeEquivalenceRel)
SlopeEquivalenceRel{id(int)} =
TheNeutralElement(RealNumbers,RealMultiplication)
BoundedIntMaps ⊆ Slopes
⟨*proof*⟩

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**
⟨*proof*⟩

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) O s} = -(r{s})
⟨*proof*⟩

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
⟨*proof*⟩

Identity function on integers is a slope.

lemma Real_ZF_1_L14: **shows** id(int) ∈ Slopes
⟨*proof*⟩

This 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.

end

28 Real_ZF_1.thy

theory Real_ZF_1 **imports** Real_ZF Int_ZF_2 OrderedField_ZF

begin

In this theory file we continue the construction of real numbers started in Real_ZF.thy to a succesful conclusion. We put here those parts of the construction that can not be done in the general settings of abelian groups and require integers.

28.1 Definitions and notation

In this section we define notions and notation needed for the rest of the construction.

The order on the set of real numbers is constructed by specifying the set of positive reals. This is defined as the projection of the set of positive slopes. A slope is positive if it takes an infinite number of posititive values on the positive integers (see Int_ZF_2.thy for properties of positive slopes). The order relation on real numbers is defined by prescribing the set of positive numbers (see section "Alternative definitions" in OrderedGroup_ZF.thy.).

constdefs

```
PositiveSlopes ≡ {s ∈ Slopes.
s(PositiveIntegers) ∩  PositiveIntegers ∉ Fin(int)}

PositiveReals ≡ {SlopeEquivalenceRel{s}. s ∈ PositiveSlopes}

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 $\mathtt{m}^R$. For a real number a notation $\lfloor a \rfloor$ means the largest integer m such that the real version of it (that is, m^R) is not greater than a. For an integer m and a subset of reals S the expression $\Gamma(S, m)$ is defined as $\max\{\lfloor p^R \cdot x \rfloor : x \in S\}$. This is plays a role in the proof of completeness of real numbers. We also reuse some notation defined in the int0 context, like $\mathbb{Z}_+$ (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 ∼ r ≡ ⟨s,r⟩ ∈ SlopeEquivalenceRel
```

fixes slope_add (**infix** + 70)
defines slope_add_def [simp]:
s + r ≡ SlopeOp1⟨s,r⟩

fixes slope_comp (**infix** ∘ 71)
defines slope_comp_def [simp]: s ∘ r ≡ SlopeOp2⟨s,r⟩

fixes slopes ($\mathcal{S}$)
defines slopes_def [simp]: $\mathcal{S}$ ≡ AlmostHoms(int,IntegerAddition)

fixes posslopes ($\mathcal{S}_+$)
defines posslopes_def [simp]: $\mathcal{S}_+$ ≡ PositiveSlopes

fixes slope_class ([_])
defines slope_class_def [simp]: [f] ≡ SlopeEquivalenceRel{f}

fixes slope_neg :: i⇒i (-_ [90] 91)
defines slope_neg_def [simp]: -s ≡ GroupInv(int,IntegerAddition) 0 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 ($\mathbb{R}_+$)
defines positivereals_def [simp]: $\mathbb{R}_+$ ≡ PositiveSet($\mathbb{R}$,RealAddition,OrderOnReals)

fixes intembed ($\_^R$ [90] 91)
defines intembed_def [simp]:
m^R ≡ [{⟨n,IntegerMultiplication⟨m,n⟩ ⟩. n ∈ int}]

fixes floor (⌊ _ ⌋)
defines floor_def [simp]:
⌊a⌋ ≡ Maximum(IntegerOrder,{m ∈ int. m^R ≤ a})

fixes Γ
defines Γ_def [simp]: Γ(S,p) ≡ Maximum(IntegerOrder,{⌊p^R·x⌋. x∈S})

fixes ia (**infixl** + 69)
defines ia_def [simp]: a+b ≡ IntegerAddition<a,b>

fixes iminus :: i⇒i (- _ 72)
defines rminus_def [simp]: -a ≡ GroupInv(int,IntegerAddition)(a)

fixes isub (**infixl** - 69)
defines isub_def [simp]: a-b ≡ a+ (- b)

fixes intpositives ($\mathbb{Z}_+$)
defines intpositives_def [simp]:
$\mathbb{Z}_+ \equiv$ PositiveSet(int,IntegerAddition,IntegerOrder)

fixes zlesseq (**infix** $\leq$ 60)
defines lesseq_def [simp]: m $\leq$ n $\equiv \langle$m,n$\rangle \in$ IntegerOrder

fixes imult (**infixl** $\cdot$ 70)
defines imult_def [simp]: a$\cdot$b $\equiv$ IntegerMultiplication<a,b>

fixes izero ($\mathbf{0}_Z$)
defines izero_def [simp]: $\mathbf{0}_Z \equiv$ TheNeutralElement(int,IntegerAddition)

fixes ione ($\mathbf{1}_Z$)
defines ione_def [simp]: $\mathbf{1}_Z \equiv$ TheNeutralElement(int,IntegerMultiplication)

fixes itwo ($\mathbf{2}_Z$)
defines itwo_def [simp]: $\mathbf{2}_Z \equiv \mathbf{1}_Z$+$\mathbf{1}_Z$

fixes abs
defines abs_def [simp]:
abs(m) $\equiv$ AbsoluteValue(int,IntegerAddition,IntegerOrder)(m)

fixes δ
defines δ_def [simp] : δ(s,m,n) $\equiv$ s(m+n)-s(m)-s(n)

28.2 Multiplication of real numbers

Multiplication of real numbers is defined as a projection of composition of slopes onto the space of equivalence classes of slopes. Thus, the product of the real numbers given as classes of slopes s and r is defined as the class of $s \circ r$. The goal of this section is to show that multiplication defined this way is commutative.

Let's recall a theorem from Int_ZF_2.thy that states that if f, g are slopes, then $f \circ g$ is equivalent to $g \circ f$. Here we conclude from that that the classes of $f \circ g$ and $g \circ f$ are the same.

lemma (in real1) Real_ZF_1_1_L2: **assumes** A1: f $\in \mathcal{S}$ g $\in \mathcal{S}$
  shows [f$\circ$g] = [g$\circ$f]
⟨*proof*⟩

Classes of slopes are real numbers.

lemma (in real1) Real_ZF_1_1_L3: **assumes** A1: f $\in \mathcal{S}$
  shows [f] $\in \mathbb{R}$
⟨*proof*⟩

Each real number is a class of a slope.

lemma (in real1**)** Real_ZF_1_1_L3A: **assumes** A1: a∈ℝ
shows ∃f∈$\mathcal{S}$. a = [f]
⟨*proof*⟩

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 ∈ $\mathcal{S}$ g ∈ $\mathcal{S}$
shows
[f] + [g] = [f+g]
[f] · [g] = [f∘g]
⟨*proof*⟩

The next lemma is essentially the same as Real_ZF_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 ∈ $\mathcal{S}$
shows [-f] = -[f]
⟨*proof*⟩

Subtracting real numbers correspods to adding the opposite slope.

lemma (in real1**)** Real_ZF_1_1_L4B: **assumes** A1: f ∈ $\mathcal{S}$ g ∈ $\mathcal{S}$
shows [f] - [g] = [f+(-g)]
⟨*proof*⟩

Multiplication of real numbers is commutative.

theorem (in real1**)** real_mult_commute: **assumes** A1: a∈ℝ b∈ℝ
shows a·b = b·a
⟨*proof*⟩

Multiplication is commutative on reals.

lemma real_mult_commutative: **shows**
RealMultiplication {is commutative on} RealNumbers
⟨*proof*⟩

The neutral element of multiplication of reals (denoted as **1** in the real1 context) is the class of identity function on integers. This is really shown in Real_ZF_1_L11, here we only rewrite it in the notation used in the real1 context.

lemma (in real1**)** real_one_cl_identity: **shows** [id(int)] = **1**
⟨*proof*⟩

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 ∈ BoundedIntMaps **shows** [f] = **0**
⟨*proof*⟩

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 ∈ $\mathcal{S}$ g ∈ $\mathcal{S}$
shows [f] = [g] ⟷ f ∼ g
⟨*proof*⟩

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 ∼ g
shows [f] = [g]
⟨*proof*⟩

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 id(int) ∈ $\mathcal{S}$
⟨*proof*⟩

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 ∈ BoundedIntMaps
shows ¬(id(int) ∼ f)
⟨*proof*⟩

Zero is not one.

lemma (in real1) real_zero_not_one: shows 1≠0
⟨*proof*⟩

Negative of a real number is a real number. Property of groups.

lemma (in real1) Real_ZF_1_1_L8: assumes a∈$\mathbb{R}$ **shows** (-a) ∈ $\mathbb{R}$
⟨*proof*⟩

An identity with three real numbers.

lemma (in real1) Real_ZF_1_1_L9: assumes a∈$\mathbb{R}$ b∈$\mathbb{R}$ c∈$\mathbb{R}$
shows a·(b·c) = a·c·b
⟨*proof*⟩

28.3 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 $\subseteq$ Slopes
  PositiveReals $\subseteq$ RealNumbers
⟨*proof*⟩

Positive reals are the same as classes of a positive slopes.

lemma (in real1**)** Real_ZF_1_2_L2:
  shows a $\in$ PositiveReals $\longleftrightarrow$ ($\exists$f$\in\mathcal{S}_+$. a = [f])
⟨*proof*⟩

Let's recall from Int_ZF_2.thy that the sum and composition of positive slopes is a positive slope.

lemma (in real1**)** Real_ZF_1_2_L3:
  assumes f$\in\mathcal{S}_+$ g$\in\mathcal{S}_+$
  shows
  f+g $\in \mathcal{S}_+$
  f$\circ$g $\in \mathcal{S}_+$
  ⟨*proof*⟩

Bounded integer maps are not positive slopes.

lemma (in real1**)** Real_ZF_1_2_L5:
  assumes f $\in$ BoundedIntMaps
  shows f $\notin \mathcal{S}_+$
  ⟨*proof*⟩

The set of positive reals is closed under addition and multiplication. Zero (the neutral element of addition) is not a positive number.

lemma (in real1**)** Real_ZF_1_2_L6: **shows**
  PositiveReals {is closed under} RealAddition
  PositiveReals {is closed under} RealMultiplication
  0 $\notin$ PositiveReals
⟨*proof*⟩

If a class of a slope f is not zero, then either f is a positive slope or $-f$ is a positive slope. The real proof is in Int_ZF_2.thy.

lemma (in real1**)** Real_ZF_1_2_L7:
  assumes A1: f $\in \mathcal{S}$ **and** A2: [f] $\neq$ **0**
  shows (f $\in \mathcal{S}_+$) Xor ((-f) $\in \mathcal{S}_+$)
  ⟨*proof*⟩

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 $\in \mathcal{S}$ g $\in \mathcal{S}$
  and A2: (f $\in \mathcal{S}_+$) Xor (g $\in \mathcal{S}_+$)
  shows ([f] $\in$ PositiveReals) Xor ([g] $\in$ PositiveReals)

⟨*proof*⟩

The trichotomy law for the (potential) order on reals: if $a \neq 0$, then either a is positive or $-a$ is potitive.

lemma (in real1**)** Real_ZF_1_2_L9:
  assumes A1: a∈ℝ **and** A2: a≠**0**
  shows (a ∈ PositiveReals) Xor ((-a) ∈ PositiveReals)
⟨*proof*⟩

Finally we are ready to prove that real numbers form an ordered ring. with no zero divisors.

theorem reals_are_ord_ring: **shows**
  IsAnOrdRing(RealNumbers,RealAddition,RealMultiplication,OrderOnReals)
  OrderOnReals {is total on} RealNumbers
  PositiveSet(RealNumbers,RealAddition,OrderOnReals) = PositiveReals
  HasNoZeroDivs(RealNumbers,RealAddition,RealMultiplication)
⟨*proof*⟩

All theorems proven in the ring1 (about ordered rings), group3 (about ordered groups) and group1 (about groups) contexts are valid as applied to ordered real numbers with addition and (real) order.

lemma Real_ZF_1_2_L10: **shows**
  ring1(RealNumbers,RealAddition,RealMultiplication,OrderOnReals)
  IsAnOrdGroup(RealNumbers,RealAddition,OrderOnReals)
  group3(RealNumbers,RealAddition,OrderOnReals)
  OrderOnReals {is total on} RealNumbers
⟨*proof*⟩

If $a = b$ or $b - a$ is positive, then a is less or equal b.

lemma (in real1**)** Real_ZF_1_2_L11: **assumes** A1: a∈ℝ b∈ℝ **and**
  A3: a=b ∨ b-a ∈ PositiveReals
  shows a≤b
  ⟨*proof*⟩

A sufficient condition for two classes to be in the real order.

lemma (in real1**)** Real_ZF_1_2_L12: **assumes** A1: f ∈ $\mathcal{S}$ g ∈ $\mathcal{S}$ **and**
  A2: f∼g ∨ (g + (-f)) ∈ $\mathcal{S}_+$
  shows [f] ≤ [g]
⟨*proof*⟩

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∈ℝ **and** A2: (-a) ≤ b
  shows (-b) ≤ a
  ⟨*proof*⟩

Real order is antisymmetric.

lemma (in real1) real_ord_antisym:
assumes A1: a≤b b≤a **shows** a=b
⟨*proof*⟩

Real order is transitive.

lemma (in real1) real_ord_transitive: **assumes** A1: a≤b b≤c
shows a≤c
⟨*proof*⟩

We can multiply both sides of an inequality by a nonnegative real number.

lemma (in real1) Real_ZF_1_2_L14:
assumes a≤b **and** **0**≤c
shows
a·c ≤ b·c
c·a ≤ c·b
⟨*proof*⟩

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∈$\mathbb{R}_+$
shows c·a ≤ c·b
⟨*proof*⟩

In the real1 context notation $a \leq b$ implies that a and b are real numbers.

lemma (in real1) Real_ZF_1_2_L15: **assumes** a≤b **shows** a∈$\mathbb{R}$ b∈$\mathbb{R}$
⟨*proof*⟩

$a \leq b$ implies that $0 \leq b - a$.

lemma (in real1) Real_ZF_1_2_L16: **assumes** a≤b
shows **0** ≤ b-a
⟨*proof*⟩

A sum of nonnegative elements is nonnegative.

lemma (in real1) Real_ZF_1_2_L17: **assumes** **0**≤a **0**≤b
shows **0** ≤ a+b
⟨*proof*⟩

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
⟨*proof*⟩

The order on real is reflexive.

lemma (in real1) real_ord_refl: **assumes** a∈$\mathbb{R}$ **shows** a≤a
⟨*proof*⟩

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
⟨*proof*⟩

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
⟨*proof*⟩

What happens when one real number is not greater or equal than another?

lemma (in real1) Real_ZF_1_2_L20: assumes a∈ℝ b∈ℝ **and** ¬(a≤b)
shows b < a
⟨*proof*⟩

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∈ℝ b∈ℝ **and** c ≤ a-b
shows c+b ≤ a
⟨*proof*⟩

The order on reals is a relation on reals.

lemma (in real1) Real_ZF_1_2_L22: shows OrderOnReals ⊆ ℝ×ℝ
⟨*proof*⟩

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 ⊆ ℝ
⟨*proof*⟩

Properties of the maximum of three real numbers.

lemma (in real1) Real_ZF_1_2_L24:
assumes A1: a∈ℝ b∈ℝ c∈ℝ
shows
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})
⟨*proof*⟩

lemma (in real1) real_strict_ord_transit:
assumes A1: a≤b **and** A2: b<c
shows a<c

⟨*proof*⟩

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 ∈ $\mathbb{R}_+$ **and** a≤b **and** 1<c
shows a<b·c
⟨*proof*⟩

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
⟨*proof*⟩

Real order is translation invariant.

lemma (in real1**)** real_ord_transl_inv**:**
assumes a≤b **and** c∈ℝ
shows c+a ≤ c+b
⟨*proof*⟩

It is convenient to have the transitivity of the order on integers in the notation 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*⟩

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⊆ℝ **and** ¬HasAmaximum(OrderOnReals,A) **and** x∈A
shows ∃y∈A. x<y
⟨*proof*⟩

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*⟩

If a real number is less than another, then the secon 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*⟩

28.4 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 ∈ $\mathcal{S}_+$
shows ∃g∈$\mathcal{S}$. f∼g ∧ (∃h∈$\mathcal{S}$. g∘h ∼ id(int))
⟨*proof*⟩

The set of real numbers we are constructing is an ordered field.

theorem (in real1**)** reals_are_ord_field**: shows**
IsAnOrdField(RealNumbers,RealAddition,RealMultiplication,OrderOnReals)
⟨*proof*⟩

Reals form a field.

lemma reals_are_field**:**
shows IsAfield(RealNumbers,RealAddition,RealMultiplication)
⟨*proof*⟩

Theorem proven in field0 and field1 contexts are valid as applied to real numbers.

lemma field_cntxts_ok**: shows**
field0(RealNumbers,RealAddition,RealMultiplication)
field1(RealNumbers,RealAddition,RealMultiplication,OrderOnReals)
⟨*proof*⟩

If a is positive, then a^{-1} is also positive.

lemma (in real1**)** Real_ZF_1_3_L1**: assumes** a ∈ $\mathbb{R}_+$
shows a^{-1} ∈ $\mathbb{R}_+$ a^{-1} ∈ $\mathbb{R}$
⟨*proof*⟩

A technical fact about multiplying strict inequality by the inverse of one of the sides.

lemma (in real1) Real_ZF_1_3_L2:
assumes $a \in \mathbb{R}_+$ **and** $a^{-1} < b$
shows $1 < b \cdot a$
⟨*proof*⟩

If $a < b$, then $(b-a)^{-1}$ is positive.

lemma (in real1) Real_ZF_1_3_L3: assumes $a<b$
shows $(b-a)^{-1} \in \mathbb{R}_+$
⟨*proof*⟩

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: $a \in \mathbb{R}$ $b \in \mathbb{R}_+$ **and A2:** $a \cdot b < c$
shows $a < c \cdot b^{-1}$
⟨*proof*⟩

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: $b \in \mathbb{R}$ $c \in \mathbb{R}_+$ **and A2:** $a < b \cdot c$
shows $a \cdot c^{-1} < b$
⟨*proof*⟩

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: $b \in \mathbb{R}$ $c \in \mathbb{R}_+$ **and A2:** $a \leq b \cdot c$
shows $a \cdot c^{-1} \leq b$
⟨*proof*⟩

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: $a \in \mathbb{R}$ $b \in \mathbb{R}_+$ **and A2:** $a \cdot b \leq c$
shows $a \leq c \cdot b^{-1}$
⟨*proof*⟩

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 $a<b$ **and** $(b-a)^{-1} < c$
shows $1 + a \cdot c < b \cdot c$
⟨*proof*⟩

We can multiply an inequality by the inverse of a positive number.

lemma (in real1) Real_ZF_1_3_L6:

assumes a≤b **and** c∈$\mathbb{R}_+$ **shows** a·c^{-1} ≤ b·c^{-1}
⟨*proof*⟩

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∈$\mathbb{R}_+$ **shows**
a·c < b·c
c·a < c·b
a·c^{-1} < b·c^{-1}
⟨*proof*⟩

An identity with three real numbers, inverse and cancelling.

lemma (in real1**)** Real_ZF_1_3_L8: **assumes**a∈$\mathbb{R}$ b∈$\mathbb{R}$ b≠**0** c∈$\mathbb{R}$
shows a·b·(c·b^{-1}) = a·c
⟨*proof*⟩

28.5 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 ∈ int
shows m^R ∈ $\mathbb{R}$
⟨*proof*⟩

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 ∈ int
shows $(-m)^R$ = -(m^R)
⟨*proof*⟩

The embedding of sum of integers is the sum of embeddings.

lemma (in real1**)** Real_ZF_1_4_L1A: **assumes** m ∈ int k ∈ int
shows m^R + k^R = $((m+k)^R)$
⟨*proof*⟩

The embedding of a difference of integers is the difference of embeddings.

lemma (in real1**)** Real_ZF_1_4_L1B: **assumes** A1: m ∈ int k ∈ int
shows m^R - k^R = $(m-k)^R$
⟨*proof*⟩

The embedding of the product of integers is the product of embeddings.

lemma (in real1**)** Real_ZF_1_4_L1C: **assumes** m ∈ int k ∈ int
shows m^R · k^R = $(m \cdot k)^R$

⟨*proof*⟩

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 ∃m∈int. a ≤ m^R
⟨*proof*⟩

For any real numbers there is an integer whose real version (embedding) is less or equal.

lemma (in real1) Real_ZF_1_4_L3: assumes A1: a∈ℝ
shows {m ∈ int. m^R ≤ a} ≠ 0
⟨*proof*⟩

Embeddings of two integers are equal only if the integers are equal.

lemma (in real1) Real_ZF_1_4_L4:
assumes A1: m ∈ int k ∈ int and A2: m^R = k^R
shows m=k
⟨*proof*⟩

The embedding of integers preserves the order.

lemma (in real1) Real_ZF_1_4_L5: assumes A1: m≤k
shows m^R ≤ k^R
⟨*proof*⟩

The embedding of integers preserves the strict order.

lemma (in real1) Real_ZF_1_4_L5A: assumes A1: m≤k m≠k
shows m^R < k^R
⟨*proof*⟩

For any real number there is a positive integer whose real version is (strictly) greater. This is Lemma 14 i) in [2].

lemma (in real1) Arthan_Lemma14i: assumes A1: a∈ℝ
shows ∃n∈$\mathbb{Z}_+$. a < n^R
⟨*proof*⟩

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 ∈ int m ∈ int and A2: m^R ≤ k^R
shows m≤k
⟨*proof*⟩

The floor function is well defined and has expected properties.

lemma (in real1) Real_ZF_1_4_L7: assumes A1: a∈ℝ
shows
IsBoundedAbove({m ∈ int. m^R ≤ a},IntegerOrder)

{m ∈ int. $m^R \leq a$} ≠ 0
$\lfloor a \rfloor \in$ int
$\lfloor a \rfloor^R \leq a$
⟨*proof*⟩

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 ∈ int **and** A2: $m^R \leq a$
shows $m \leq \lfloor a \rfloor$
⟨*proof*⟩

Integer zero and one embed as real zero and one.

lemma (in real1) int_0_1_are_real_zero_one:
shows $\mathbf{0}_Z{}^R = \mathbf{0}$ $\mathbf{1}_Z{}^R = \mathbf{1}$
⟨*proof*⟩

Integer two embeds as the real two.

lemma (in real1) int_two_is_real_two: **shows** $\mathbf{2}_Z{}^R = \mathbf{2}$
⟨*proof*⟩

A positive integer embeds as a positive (hence nonnegative) real.

lemma (in real1) int_pos_is_real_pos: **assumes** A1: p∈$\mathbb{Z}_+$
shows
$p^R \in \mathbb{R}$
$\mathbf{0} \leq p^R$
$p^R \in \mathbb{R}_+$
⟨*proof*⟩

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 $x < M^R y$. This is Lemma 14 ii) in [2].

lemma (in real1) Arthan_Lemma14ii: **assumes** A1: x∈$\mathbb{R}$ y ∈ $\mathbb{R}_+$
shows ∃M∈$\mathbb{Z}_+$. x < M^R·y
⟨*proof*⟩

Taking the floor function preserves the order.

lemma (in real1) Real_ZF_1_4_L9: **assumes** A1: a≤b
shows $\lfloor a \rfloor \leq \lfloor b \rfloor$
⟨*proof*⟩

If S is bounded above and p is a positive intereger, then $\Gamma(S, p)$ is well defined.

lemma (in real1) Real_ZF_1_4_L10:
assumes A1: IsBoundedAbove(S,OrderOnReals) S≠0 **and** A2: p∈$\mathbb{Z}_+$
shows
IsBoundedAbove({$\lfloor p^R \cdot x \rfloor$. x∈S},IntegerOrder)

Γ(S,p) ∈ {⌊p^R·x⌋. x∈S}
Γ(S,p) ∈ int
⟨*proof*⟩

If p is a positive integer, then for all $s \in S$ the floor of $p \cdot x$ is not greater that $\Gamma(S, p)$.

lemma (in real1**)** Real_ZF_1_4_L11:
assumes A1: IsBoundedAbove(S,OrderOnReals) **and** A2: x∈S **and** A3: p∈$\mathbb{Z}_+$
shows ⌊p^R·x⌋ ≤ Γ(S,p)
⟨*proof*⟩

The candidate for supremum is an integer mapping with values given by Γ.

lemma (in real1**)** Real_ZF_1_4_L12:
assumes A1: IsBoundedAbove(S,OrderOnReals) S≠0 **and**
A2: g = {⟨p,Γ(S,p)⟩. p∈$\mathbb{Z}_+$}
shows
g : $\mathbb{Z}_+$→int
∀n∈$\mathbb{Z}_+$. g(n) = Γ(S,n)
⟨*proof*⟩

Every integer is equal to the floor of its embedding.

lemma (in real1**)** Real_ZF_1_4_L14: **assumes** A1: m ∈ int
shows ⌊m^R⌋ = m
⟨*proof*⟩

Floor of (real) zero is (integer) zero.

lemma (in real1**)** floor_01_is_zero_one: **shows**
⌊**0**⌋ = $\mathbf{0}_Z$ ⌊**1**⌋ = $\mathbf{1}_Z$
⟨*proof*⟩

Floor of (real) two is (integer) two.

lemma (in real1**)** floor_2_is_two: **shows** ⌊**2**⌋ = $\mathbf{2}_Z$
⟨*proof*⟩

Floor of a product of embeddings of integers is equal to the product of integers.

lemma (in real1**)** Real_ZF_1_4_L14A: **assumes** A1: m ∈ int k ∈ int
shows ⌊m^R·k^R⌋ = m·k
⟨*proof*⟩

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∈$\mathbb{R}$ **and** A2: p ∈ int
shows ⌊x + p^R⌋ = ⌊x⌋ + p
⟨*proof*⟩

Floor of the difference of a number and the embedding of an integer is the floor of the number minus the integer.

lemma (in real1) Real_ZF_1_4_L16: assumes A1: x∈ℝ and A2: p ∈ int
shows $\lfloor x - p^R \rfloor = \lfloor x \rfloor - p$
⟨*proof*⟩

The floor of sum of embeddings is the sum of the integers.

lemma (in real1) Real_ZF_1_4_L17: assumes m ∈ int n ∈ int
shows $\lfloor (m^R) + n^R \rfloor = m + n$
⟨*proof*⟩

A lemma about adding one to floor.

lemma (in real1) Real_ZF_1_4_L17A: assumes A1: a∈ℝ
shows $\mathbf{1} + \lfloor a \rfloor^R = (\mathbf{1}_Z + \lfloor a \rfloor)^R$
⟨*proof*⟩

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∈ℝ
shows
$a - \lfloor a \rfloor^R < \mathbf{1}$
$a < (\mathbf{1}_Z + \lfloor a \rfloor)^R$
⟨*proof*⟩

The next lemma corresponds to Lemma 14 iii) in [2]. It says that we can find a rational number between any two different real numbers.

lemma (in real1) Arthan_Lemma14iii: assumes A1: x<y
shows $\exists M \in int.\ \exists N \in \mathbb{Z}_+.\ \ x \cdot N^R < M^R \wedge M^R < y \cdot N^R$
⟨*proof*⟩

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
$\mathrm{abs}(\lfloor x+y \rfloor - \lfloor x \rfloor - \lfloor y \rfloor) \leq \mathbf{2}_Z$
⟨*proof*⟩

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: IsBoundedAbove(S,OrderOnReals) S≠0 and
A2: n∈$\mathbb{Z}_+$ x∈S **and**
A3: Γ(S,n) = $\lfloor n^R \cdot x \rfloor$ **and**
A4: y∈S x≤y
shows $\Gamma(S,n) = \lfloor n^R \cdot y \rfloor$
⟨*proof*⟩

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≠0 **and**
A2: m∈$\mathbb{Z}_+$ n∈$\mathbb{Z}_+$
shows abs(Γ(S,m+n) - Γ(S,m) - Γ(S,n)) ≤ $\mathbf{2}_Z$
⟨*proof*⟩

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 δ in the real1 context is bounded on positive integers. This is really proven in Int_ZF_2.thy, but we restate it here for convenience. Recall from Group_ZF_3.thy that OddExtension of a function defined on the set of positive elements (of an ordered group) is the only odd function that is equal to the given one when restricted to positive elements.

lemma (in real1) Real_ZF_1_4_L21A:
assumes A1: f:$\mathbb{Z}_+$→int ∀a∈$\mathbb{Z}_+$. ∀b∈$\mathbb{Z}_+$. abs(δ(f,a,b)) ≤ L
shows OddExtension(int,IntegerAddition,IntegerOrder,f) ∈ $\mathcal{S}$
⟨*proof*⟩

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 A1: IsBoundedAbove(S,OrderOnReals) S≠0 **and**
A2: g = {⟨p,Γ(S,p)⟩. p∈$\mathbb{Z}_+$}
shows OddExtension(int,IntegerAddition,IntegerOrder,g) ∈ $\mathcal{S}$
⟨*proof*⟩

A technical lemma used in the proof that all elements of S are less or equal than the candidate for supremum of S.

lemma (in real1) Real_ZF_1_4_L23:
assumes A1: f ∈ $\mathcal{S}$ **and** A2: N ∈ int M ∈ int **and**
A3: ∀n∈$\mathbb{Z}_+$. M·n ≤ f(N·n)
shows M^R ≤ [f]·(N^R)
⟨*proof*⟩

A technical lemma aimed used in the proof the candidate for supremum of S is less or equal than any upper bound for S.

lemma (in real1) Real_ZF_1_4_L23A:
assumes A1: f ∈ $\mathcal{S}$ **and** A2: N ∈ int M ∈ int **and**
A3: ∀n∈$\mathbb{Z}_+$. f(N·n) ≤ M·n
shows [f]·(N^R) ≤ M^R
⟨*proof*⟩

The essential condition to claim that the candidate for supremum of S is greater or equal than all elements of S.

lemma (in real1) Real_ZF_1_4_L24:
assumes A1: IsBoundedAbove(S,OrderOnReals) **and**
A2: x<y y∈S **and**

A4: N ∈ $\mathbb{Z}_+$ M ∈ int **and**
A5: M^R < y·N^R **and** A6: p ∈ $\mathbb{Z}_+$
shows p·M ≤ Γ(S,p·N)
⟨*proof*⟩

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≠0 **and** A2: p ∈ $\mathbb{Z}_+$
and A3:
h = OddExtension(int,IntegerAddition,IntegerOrder,{⟨p,Γ(S,p)⟩. p∈$\mathbb{Z}_+$})
shows h(p) = Γ(S,p)
⟨*proof*⟩

The candidate for the supremum of S is not smaller than any element of S.

lemma (in real1**)** Real_ZF_1_4_L25:
assumes A1: IsBoundedAbove(S,OrderOnReals) **and**
A2: ¬HasAmaximum(OrderOnReals,S) **and**
A3: x∈S **and** A4:
h = OddExtension(int,IntegerAddition,IntegerOrder,{⟨p,Γ(S,p)⟩. p∈$\mathbb{Z}_+$})
shows x ≤ [h]
⟨*proof*⟩

The essential condition to claim that the candidate for supremum of S is less or equal than any upper bound of S.

lemma (in real1**)** Real_ZF_1_4_L26:
assumes A1: IsBoundedAbove(S,OrderOnReals) **and**
A2: x≤y x∈S **and**
A4: N ∈ $\mathbb{Z}_+$ M ∈ int **and**
A5: y·N^R < M^R **and** A6: p ∈ $\mathbb{Z}_+$
shows $\lfloor (N\cdot p)^R \cdot x \rfloor$ ≤ M·p
⟨*proof*⟩

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≠0 **and**
h = OddExtension(int,IntegerAddition,IntegerOrder,{⟨p,Γ(S,p)⟩. p∈$\mathbb{Z}_+$})
and p ∈ $\mathbb{Z}_+$
shows ∃x∈S. h(p) = $\lfloor p^R \cdot x \rfloor$
⟨*proof*⟩

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≠0

and A2: $\forall$x∈S. x≤y **and** A3:
h = OddExtension(int,IntegerAddition,IntegerOrder,{⟨p,Γ(S,p)⟩. p∈$\mathbb{Z}_+$})
shows [h] ≤ y
⟨*proof*⟩

Now we can prove that every nonempty subset of reals that is bounded above has a supremum.

lemma (in real1) real_order_complete:
assumes A1: IsBoundedAbove(S,OrderOnReals) S≠0
shows HasAminimum(OrderOnReals,$\bigcap$a∈S. OrderOnReals{a})
⟨*proof*⟩

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.

theorem eudoxus_reals_are_reals: **shows**
IsAmodelOfReals(RealNumbers,RealAddition,RealMultiplication,OrderOnReals)
⟨*proof*⟩

This completes the construction. It was fun.

end

29 Complex_ZF.thy

theory `Complex_ZF` **imports** `OrderedField_ZF`

begin

The goal of this theory is to define complex numbers and prove that the Metamath complex numbers axioms hold.

29.1 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 ($\mathbb{C} = R \times R$), binary oparations of addition and multiplication of complex numbers and the order raletion on $\mathbb{R} = R \times 0$. The `ImCxAdd, ReCxAdd, ImCxMul, 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 `CplxAdd` and `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 $\{\langle x, 0\rangle : x \in R\}$ `ComplexReals(R,A)` and the strict order relation `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, lets repeat this slower. We start with the order relation r on a (model of) real numbers. We want to define an order relation on a subset of complex numbers, namely on $R \times \{0\}$. To do that we use the notion of a relation induced by a mapping. The mapping here is $f : R \times \{0\} \to R, f\langle x, 0\rangle = x$ which is defined under a name of `SliceProjection` in `func_ZF.thy`. This defines a relation r_1 (called `InducedRelation(f,r)`, see `func_ZF`) on $R \times \{0\}$ such that $\langle\langle x, 0\rangle, \langle y, 0\rangle\rangle \in r_1$ iff $\langle x, y\rangle \in r$. This way we get what we call `CplxROrder(R,A,r)`. However, this is not the end of the story, because Metamath uses strict inequalities, 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 $<_{\mathbb{R}}$ in the definition of `complex0` context.

constdefs

`ReCxAdd(R,A,a,b)` $\equiv$ `A`$\langle$`fst(a),fst(b)`$\rangle$

```
ImCxAdd(R,A,a,b) ≡ A⟨snd(a),snd(b)⟩

CplxAdd(R,A) ≡
{⟨p, ⟨ ReCxAdd(R,A,fst(p),snd(p)),ImCxAdd(R,A,fst(p),snd(p)) ⟩ ⟩.
p∈(R×R)×(R×R)}

ImCxMul(R,A,M,a,b) ≡ A⟨M⟨fst(a),snd(b)⟩, M⟨snd(a),fst(b)⟩ ⟩

ReCxMul(R,A,M,a,b) ≡
A⟨M⟨fst(a),fst(b)⟩,GroupInv(R,A)(M⟨snd(a),snd(b)⟩)⟩

CplxMul(R,A,M) ≡
{ ⟨p, ⟨ReCxMul(R,A,M,fst(p),snd(p)),ImCxMul(R,A,M,fst(p),snd(p))⟩ ⟩.

p ∈ (R×R)×(R×R)}

ComplexReals(R,A) ≡ R×{TheNeutralElement(R,A)}

CplxROrder(R,A,r) ≡
InducedRelation(SliceProjection(ComplexReals(R,A)),r)
```

The next locale defines proof context and notation that will be used for complex numbers.

```
locale complex0 =
  fixes R and A and M and r
  assumes R_are_reals: IsAmodelOfReals(R,A,M,r)

  fixes complex (ℂ)
  defines complex_def[simp]: ℂ ≡ R×R

  fixes rone (1_R)
  defines rone_def[simp]: 1_R ≡ TheNeutralElement(R,M)

  fixes rzero (0_R)
  defines rzero_def[simp]: 0_R ≡ TheNeutralElement(R,A)

  fixes one (1)
  defines one_def[simp]: 1 ≡ ⟨1_R, 0_R⟩

  fixes zero (0)
  defines zero_def[simp]: 0 ≡ ⟨0_R, 0_R⟩

  fixes iunit (i)
  defines iunit_def[simp]: i ≡ ⟨0_R,1_R⟩

  fixes creal (ℝ)
  defines creal_def[simp]: ℝ ≡ {⟨r,0_R⟩. r∈R}

  fixes ca (infixl + 69)
```

defines ca_def[simp]: a + b ≡ CplxAdd(R,A)⟨a,b⟩

fixes cm (**infixl** · 71)
defines cm_def[simp]: a · b ≡ CplxMul(R,A,M)⟨a,b⟩

fixes rmul (**infixl** · 71)
defines rmul_def[simp]: a · b ≡ M⟨a,b⟩

fixes radd (**infixl** + 69)
defines radd_def[simp]: a + b ≡ A⟨a,b⟩

fixes rneg :: i⇒i (- _ 70)
defines rneg_def[simp]: - a ≡ GroupInv(R,A)(a)

fixes lessr (**infix** $<_{\mathbb{R}}$ 68)
defines lessr_def[simp]:
a $<_{\mathbb{R}}$ b ≡ ⟨a,b⟩ ∈ StrictVersion(CplxROrder(R,A,r))

fixes cpnf ($+\infty$)
defines cpnf_def[simp]: $+\infty \equiv \mathbb{C}$

fixes cmnf ($-\infty$)
defines cmnf_def[simp]: $-\infty \equiv \{\mathbb{C}\}$

fixes cxr ($\mathbb{R}^*$)
defines cxr_def[simp]: $\mathbb{R}^* \equiv \mathbb{R} \cup \{+\infty, -\infty\}$

fixes cltrrset (<)
defines cltrrset_def[simp]:
< ≡ StrictVersion(CplxROrder(R,A,r)) ∪
$\{\langle -\infty, +\infty\rangle\} \cup (\mathbb{R}\times\{+\infty\}) \cup (\{-\infty\}\times\mathbb{R})$

29.2 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*⟩

The next lemma shows the definition of real and imaginary part of complex

sum and product in a more readable form using notation defined in `complex0` locale.

lemma (in `complex0`**)** `cplx_mul_add_defs`**: shows**
`ReCxAdd(R,A,⟨a,b⟩,⟨c,d⟩) = a + c`
`ImCxAdd(R,A,⟨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*⟩

Real and imaginary parts of sums and products of complex numbers are real.

lemma (in `complex0`**)** `cplx_mul_add_types`:
assumes `A1: z₁ ∈ ℂ  z₂ ∈ ℂ`
shows
`ReCxAdd(R,A,z₁,z₂) ∈ R`
`ImCxAdd(R,A,z₁,z₂) ∈ R`
`ImCxMul(R,A,M,z₁,z₂) ∈ R`
`ReCxMul(R,A,M,z₁,z₂) ∈ R`
⟨*proof*⟩

Complex reals are complex. Recall the definition of $\mathbb{R}$ in the `complex0` locale.

lemma (in `complex0`**)** `axresscn`**: shows** $\mathbb{R} \subseteq \mathbb{C}$
⟨*proof*⟩

Complex 1 is not complex 0.

lemma (in `complex0`**)** `ax1ne0`**: shows** $\mathbf{1} \neq \mathbf{0}$
⟨*proof*⟩

Complex addition is a complex valued binary operation on complex numbers.

lemma (in `complex0`**)** `axaddopr`**: shows** `CplxAdd(R,A)`: $\mathbb{C} \times \mathbb{C} \to \mathbb{C}$
⟨*proof*⟩

Complex multiplication is a complex valued binary operation on complex numbers.

lemma (in `complex0`**)** `axmulopr`**: shows** `CplxMul(R,A,M)`: $\mathbb{C} \times \mathbb{C} \to \mathbb{C}$
⟨*proof*⟩

What are the values of omplex addition and multiplication in terms of their real and imaginary parts?

lemma (in `complex0`**)** `cplx_mul_add_vals`:
assumes `A1: a∈R  b∈R  c∈R  d∈R`
shows
`⟨a,b⟩ + ⟨c,d⟩ = ⟨a + c, b + d⟩`
`⟨a,b⟩ · ⟨c,d⟩ = ⟨a·c + (-b·d), a·d + b·c⟩`
⟨*proof*⟩

Complex multiplication is commutative.

lemma (in complex0**)** axmulcom**: assumes** A1**:** a ∈ ℂ b ∈ ℂ
  shows a·b = b·a
  ⟨*proof*⟩

A sum of complex numbers is complex.

lemma (in complex0**)** axaddcl**: assumes** a ∈ ℂ b ∈ ℂ
  shows a+b ∈ ℂ
  ⟨*proof*⟩

A product of complex numbers is complex.

lemma (in complex0**)** axmulcl**: assumes** a ∈ ℂ b ∈ ℂ
  shows a·b ∈ ℂ
  ⟨*proof*⟩

Multiplication is distributive with respect to addition.

lemma (in complex0**)** axdistr**:**
  assumes A1**:** a ∈ ℂ b ∈ ℂ c ∈ ℂ
  shows a·(b + c) = a·b + a·c
⟨*proof*⟩

Complex addition is commutative.

lemma (in complex0**)** axaddcom**: assumes** a ∈ ℂ b ∈ ℂ
  shows a+b = b+a
  ⟨*proof*⟩

Complex addition is associative.

lemma (in complex0**)** axaddass**: assumes** A1**:** a ∈ ℂ b ∈ ℂ c ∈ ℂ
  shows a + b + c = a + (b + c)
⟨*proof*⟩

Complex multiplication is associative.

lemma (in complex0**)** axmulass**: assumes** A1**:** a ∈ ℂ b ∈ ℂ c ∈ ℂ
  shows a · b · c = a · (b · c)
⟨*proof*⟩

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** ∈ ℝ
  ⟨*proof*⟩

The imaginary unit is a "square root" of -1 (that is, $i^2 + 1 = 0$).

lemma (in complex0**)** axi2m1**: shows** i·i **+ 1 = 0**
  ⟨*proof*⟩

0 is the neutral element of complex addition.

lemma (in complex0**)** ax0id**: assumes** a ∈ ℂ
  shows a **+ 0** = a
  ⟨*proof*⟩

The imaginary unit is a complex number.

lemma (in complex0**) axicn: shows** i ∈ ℂ
⟨*proof*⟩

All complex numbers have additive inverses.

lemma (in complex0**) axnegex: assumes** A1: a ∈ ℂ
shows ∃x∈ℂ . a + x = **0**
⟨*proof*⟩

A non-zero complex number has a multiplicative inverse.

lemma (in complex0**) axrecex: assumes** A1: a ∈ ℂ **and** A2: a≠**0**
shows ∃x∈ℂ . a·x = **1**
⟨*proof*⟩

Complex 1 is a right neutral element for multiplication.

lemma (in complex0**) ax1id: assumes** A1: a ∈ ℂ
shows a·**1** = a
⟨*proof*⟩

A formula for sum of (complex) real numbers.

lemma (in complex0**) sum_of_reals: assumes** a∈ℝ b∈ℝ
shows
a + b = ⟨fst(a) + fst(b),$\mathbf{0}_R$⟩
⟨*proof*⟩

The sum of real numbers is real.

lemma (in complex0**) axaddrcl: assumes** A1: a∈ℝ b∈ℝ
shows a + b ∈ ℝ
⟨*proof*⟩

The formula for the product of (complex) real numbers.

lemma (in complex0**) prod_of_reals: assumes** A1: a∈ℝ b∈ℝ
shows a · b = ⟨fst(a)·fst(b),$\mathbf{0}_R$⟩
⟨*proof*⟩

The product of (complex) real numbers is real.

lemma (in complex0**) axmulrcl: assumes** a∈ℝ b∈ℝ
shows a · b ∈ ℝ
⟨*proof*⟩

The existence of a real negative of a real number.

lemma (in complex0**) axrnegex: assumes** A1: a∈ℝ
shows ∃ x ∈ ℝ. a + x = **0**
⟨*proof*⟩

Each nonzero real number has a real inverse

lemma (in complex0**) axrrecex:**

assumes A1: a ∈ ℝ a ≠ **0**
shows ∃x∈ℝ. a · x = **1**
⟨*proof*⟩

Our ℝ symbol is the real axis on the complex plane.

lemma (in complex0**)** real_means_real_axis: **shows** ℝ = ComplexReals(R,A)
⟨*proof*⟩

The CplxROrder thing is a relation on the complex reals.

lemma (in complex0**)** cplx_ord_on_cplx_reals:
shows CplxROrder(R,A,r) ⊆ ℝ×ℝ
⟨*proof*⟩

The strict version of the complex relation is a relation on complex reals.

lemma (in complex0**)** cplx_strict_ord_on_cplx_reals:
shows StrictVersion(CplxROrder(R,A,r)) ⊆ ℝ×ℝ
⟨*proof*⟩

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 $<_{\mathbb{R}}$ b
shows a∈ℝ b∈ℝ
⟨*proof*⟩

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,$\mathbf{0}_R$⟩ $<_{\mathbb{R}}$ ⟨y,$\mathbf{0}_R$⟩ ⟷ ⟨x,y⟩ ∈ r ∧ x≠y
⟨*proof*⟩

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*⟩

The trichotomy law for the strict order on the complex reals.

lemma (in complex0**)** cplx_strict_ord_trich:
assumes a ∈ ℝ b ∈ ℝ
shows Exactly_1_of_3_holds(a$<_{\mathbb{R}}$b, a=b, b$<_{\mathbb{R}}$a)
⟨*proof*⟩

The strict order on the complex reals is kind of antisymetric.

lemma (in complex0**)** pre_axlttri: **assumes** A1: a ∈ ℝ b ∈ ℝ
shows a $<_{\mathbb{R}}$ b ⟷ ¬(a=b ∨ b $<_{\mathbb{R}}$ a)
⟨*proof*⟩

The strict order on complex reals is transitive.

lemma (in complex0**)** cplx_strict_ord_trans:
shows trans(StrictVersion(CplxROrder(R,A,r)))
⟨*proof*⟩

The strict order on complex reals is transitive - the explicit version of cplx_strict_ord_trans.

lemma (in complex0**)** pre_axlttrn:
assumes A1: a $<_{\mathbb{R}}$ b b $<_{\mathbb{R}}$ c
shows a $<_{\mathbb{R}}$ c
⟨*proof*⟩

The strict order on complex reals is preserved by translations.

lemma (in complex0**)** pre_axltadd:
assumes A1: a $<_{\mathbb{R}}$ b **and** A2: c ∈ ℝ
shows c+a $<_{\mathbb{R}}$ c+b
⟨*proof*⟩

The set of positive complex reals is closed with respect to multiplication.

lemma (in complex0**)** pre_axmulgt0: **assumes** A1: **0** $<_{\mathbb{R}}$ a **0** $<_{\mathbb{R}}$ b
shows **0** $<_{\mathbb{R}}$ a·b
⟨*proof*⟩

The order on complex reals is linear and complete.

lemma (in complex0**)** cmplx_reals_ord_lin_compl: **shows**
CplxROrder(R,A,r) {is complete}
IsLinOrder(ℝ,CplxROrder(R,A,r))
⟨*proof*⟩

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 $<_{\mathbb{R}}$ x
shows
∃x∈ℝ. (∀y∈X. ¬(x $<_{\mathbb{R}}$ y)) ∧ (∀y∈ℝ. (y $<_{\mathbb{R}}$ x ⟶ (∃z∈X. y $<_{\mathbb{R}}$ z)))
⟨*proof*⟩

end

30 MMI_prelude.thy

theory `MMI_prelude` **imports** `equalities`

begin

In this theory file we define the context in which theorems imported from Metamath are proven and prove the logic and set theory Metamath lemmas that the proofs of Metamath theorems about real and complex numbers depend on.

30.1 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 distribution. 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_logis_and_sets.thy`. Most of the dependencies can be proven automatically by Isabelle. However, some manual work has to be done for the dependencies that Isabelle 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.

30.2 The context for Metamath theorems

We list the Metamth'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.

```
consts
   IsASet :: i⇒o (_ isASet [90] 90)

defs
  set_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.

```
locale MMIsar0 =
  fixes real (ℝ)
  fixes complex (ℂ)
  fixes one :: i (1)
  fixes zero :: i (0)
  fixes iunit :: i (i)
  fixes caddset (+)
  fixes cmulset (·)
  fixes lessrrel (<ℝ)

  fixes ca (infixl + 69)
  defines ca_def: a + b ≡ +⟨a,b⟩
  fixes cm (infixl · 71)
  defines cm_def: a · b ≡ ·⟨a,b⟩
  fixes sub (infixl - 69)
  defines sub_def: a - b ≡ ⋃ { x ∈ ℂ. b + x = a }
  fixes cneg :: i⇒i (-_ 95)
  defines cneg_def: - a ≡ 0 - a
```

fixes cdiv (**infixl** / 70)
defines cdiv_def: a / b ≡ ⋃ { x ∈ ℂ. b · x = a }
fixes cpnf ($+\infty$)
defines cpnf_def: $+\infty \equiv \mathbb{C}$
fixes cmnf ($-\infty$)
defines cmnf_def: $-\infty \equiv \{\mathbb{C}\}$
fixes cxr ($\mathbb{R}^*$)
defines cxr_def: $\mathbb{R}^* \equiv \mathbb{R} \cup \{+\infty, -\infty\}$
fixes lessr (**infix** $<_{\mathbb{R}}$ 68)
defines lessr_def: a $<_{\mathbb{R}}$ b ≡ ⟨a,b⟩ ∈ $<_{\mathbb{R}}$
fixes cltrrset (<)
defines cltrrset_def:
< ≡ ($<_{\mathbb{R}}$ ∩ ℝ×ℝ) ∪ {⟨$-\infty$,$+\infty$⟩} ∪
(ℝ×{$+\infty$}) ∪ ({$-\infty$}×ℝ)
fixes cltrr (**infix** < 68)
defines cltrr_def: a < b ≡ ⟨a,b⟩ ∈ <
fixes lsq (**infix** ≤ 68)
defines lsq_def: a ≤ b ≡ ¬ (b < a)

assumes MMI_pre_axlttri:
A ∈ ℝ ∧ B ∈ ℝ ⟶ (A $<_{\mathbb{R}}$ B ⟷ ¬(A=B ∨ B $<_{\mathbb{R}}$ A))
assumes MMI_pre_axlttrn:
A ∈ ℝ ∧ B ∈ ℝ ∧ C ∈ ℝ ⟶ ((A $<_{\mathbb{R}}$ B ∧ B $<_{\mathbb{R}}$ C) ⟶ A $<_{\mathbb{R}}$ C)
assumes MMI_pre_axltadd:
A ∈ ℝ ∧ B ∈ ℝ ∧ C ∈ ℝ ⟶ (A $<_{\mathbb{R}}$ B ⟶ C+A $<_{\mathbb{R}}$ C+B)
assumes MMI_pre_axmulgt0:
A ∈ ℝ ∧ B ∈ ℝ ⟶ (**0** $<_{\mathbb{R}}$ A ∧ **0** $<_{\mathbb{R}}$ B ⟶ **0** $<_{\mathbb{R}}$ A·B)
assumes MMI_pre_axsup:
A ⊆ ℝ ∧ A ≠ 0 ∧ (∃x∈ℝ. ∀y∈A. y $<_{\mathbb{R}}$ x) ⟶
(∃x∈ℝ. (∀y∈A. ¬(x $<_{\mathbb{R}}$ y)) ∧ (∀y∈ℝ. (y $<_{\mathbb{R}}$ x ⟶ (∃z∈A. y $<_{\mathbb{R}}$ z))))
assumes MMI_axresscn: ℝ ⊆ ℂ
assumes MMI_ax1ne0: **1** ≠ **0**
assumes MMI_axcnex: ℂ isASet
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_ax1re: **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_axrecex: A ∈ ℂ ∧ A ≠ **0** ⟶ (∃ x ∈ ℂ. A · x = **1**)
assumes MMI_ax1id: 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_axrnegex: A ∈ ℝ ⟶ (∃ x ∈ ℝ. A + x = **0**)
assumes MMI_axrrecex: A ∈ ℝ ∧ A ≠ **0** ⟶ (∃ x ∈ ℝ. A · x = **1**)

constdefs
StrictOrder (**infix** Orders 65)
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)

end

31 Metamath_interface.thy

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.

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*⟩

In complex0 context the strict version of the order relation on complex reals is a relation on complex reals.

end

32 MMI_examples.thy

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 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 MMIsar0**)** MMI_dividt:
shows (A ∈ ℂ ∧ A ≠ **0**) ⟶ (A / A) = **1**
⟨*proof*⟩

lemma (in MMIsar0**)** MMI_div0t:
shows (A ∈ ℂ ∧ A ≠ **0**) ⟶ (**0** / A) = **0**
⟨*proof*⟩

lemma (in MMIsar0**)** MMI_diveq0t:
shows (A ∈ ℂ ∧ C ∈ ℂ ∧ C ≠ **0**) ⟶
((A / C) = **0** ⟷ A = **0**)
⟨*proof*⟩

lemma (in MMIsar0**)** MMI_recrec: **assumes** A1: A ∈ ℂ **and**
A2: A ≠ **0**
shows (**1** / (**1** / A)) = A
⟨*proof*⟩

lemma (in MMIsar0**)** MMI_divid: **assumes** A1: A ∈ ℂ **and**
A2: A ≠ **0**
shows (A / A) = **1**
⟨*proof*⟩

lemma (in MMIsar0**)** MMI_div0: **assumes** A1: A ∈ ℂ **and**
A2: A ≠ **0**
shows (**0** / A) = **0**
⟨*proof*⟩

lemma (in MMIsar0**)** MMI_div1: **assumes** A1: A ∈ ℂ
shows (A / **1**) = A
⟨*proof*⟩

lemma (in MMIsar0**)** MMI_div1t:
shows A ∈ ℂ ⟶ (A / **1**) = A
⟨*proof*⟩

lemma (in MMIsar0**)** MMI_divnegt:
shows (A ∈ ℂ ∧ B ∈ ℂ ∧ B ≠ **0**) ⟶
(- (A / B)) = ((- A) / B)

⟨*proof*⟩

lemma (in MMIsar0**)** MMI_divsubdirt:
shows ((A ∈ ℂ ∧ B ∈ ℂ ∧ C ∈ ℂ) ∧ C ≠ **0**) ⟶
((A - B) / C) =
((A / C) - (B / C))
⟨*proof*⟩

end

33 Metamath_sampler.thy

theory `Metamath_sampler` **imports** `Metamath_interface MMI_Complex_ZF_1`

begin

This theory file contains some examples of theorems translated fro Metamath and formulated in the `complex0` context.

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*⟩

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`**:** `<` `Orders` $\mathbb{R}^*$
⟨*proof*⟩

end

References

[1] N. A'Campo. A natural construction for the real numbers. 2003.

[2] R. D. Arthan. The Eudoxus Real Numbers. 2004.

[3] R. S. at al. The Efficient Real Numbers. 2003.

[4] N. D. Megill. Metamath. A Computer Language for Pure Mathematics. 2004. http://us.metamath.org/.