Generalizations in Isabelle/ZF

Victor Porton

December 24, 2010

Contents

1 Link to informal document

This Isabelle/ZF theory is based on the ideas of article "Generalization in ZF".

2 Some addons to Isabelle/ZF

theory *ZF-Addons*
imports *Main-ZF*
begin

We need to add some additional lemmas to Isabelle/ZF before proceed.

lemma *inj-inj-range*: $f \in inj(A,B) \Longrightarrow f \in inj(A,range(f))$
⟨*proof*⟩

lemma *inj-bij-range*: $f \in inj(A,B) \Longrightarrow f \in bij(A,range(f))$
⟨*proof*⟩

lemma *range-subset*: $f\colon A \to B \Longrightarrow range(f) \subseteq B$
⟨*proof*⟩

lemma *lam-is-fun*: $f = (lam\ x{:}d.\ b(x)) \Longrightarrow f{:}\ d \to range(f)$
⟨*proof*⟩

lemma *comp-fun2*:
$[\![\ g{:}\ A \to B1;\ f{:}\ B0 \to C;\ B1 \subseteq B0\ ]\!] \Longrightarrow (f\ O\ g) : A \to C$
⟨*proof*⟩

This lemma superceedes the lemma *comp-eq-id-iff* in Isabelle/ZF:

lemma *comp-eq-id-iff1*:
$[\![\ g{:}\ B \to A;\ f{:}\ A \to C\ ]\!] \Longrightarrow (\forall\, y \in B.\ f`(g`y) = y) \longleftrightarrow f\ O\ g = id(B)$
⟨*proof*⟩

lemma *right-comp-id-any*: $r\ O\ id(C) = restrict(r, C)$
⟨*proof*⟩

end

3 A theory of generalization

theory *Generalization*
imports *Main-ZF ZF-Addons*
begin

This theory formalizes the intuitive notion of *generalization*.

3.1 Generalization situation

locale *generalization-situation* =
fixes *small*::*i* **and** *big*::*i*
fixes *embed*::*i*
assumes *embed-inj*: $embed \in inj(small, big)$
begin

In Isabelle 2009-2 locale params are not visible. As a workaround I define their abbreviations:

abbreviation *small-param* ≡ *small*
abbreviation *big-param* ≡ *big*
abbreviation *embed-param* ≡ *embed*

definition *small2-def*: *small2* ≡ *range*(*embed*)
definition *spec-def*: *spec* ≡ *converse*(*embed*)

lemma *spec-inj*: $spec \in inj(small2, small)$
⟨*proof*⟩

lemma *spec-fun*: $spec{:}\ small2 \to small$
⟨*proof*⟩

lemma *embed-fun*: *embed*: *small*→*big* ⟨*proof*⟩

lemma *embed-surj*: *embed*∈*surj*(*small*, *small2*)
⟨*proof*⟩

theorem *embed-bij*: *embed*∈*bij*(*small*, *small2*)
⟨*proof*⟩

theorem *small2-sub-big*: *small2*⊆*big* ⟨*proof*⟩

theorem *spec-bij*: *spec*∈*bij*(*small2*, *small*)
⟨*proof*⟩

end

3.2 Arbitrary generalizations

locale *arbitrary-generalization* = *generalization-situation* +
fixes *newbig*::*i*
fixes *move*::*i*
assumes
move-bij: *move*∈*bij*(*big*, *newbig*) **and**
move-embed: *move* *O* *embed* = *id*(*small*)
begin

In Isabelle 2009-2 locale params are not visible. As a workaround I define their abbreviations:

abbreviation *newbig-param* ≡ *newbig*
abbreviation *move-param* ≡ *move*

lemma *move-fun*: *move*: *big*→*newbig* ⟨*proof*⟩

lemma *move-inj*: *move*∈*inj*(*big*, *newbig*) ⟨*proof*⟩
lemma *move-surj*: *move*∈*surj*(*big*, *newbig*) ⟨*proof*⟩

lemma *move-domain*: *domain*(*move*) = *big* ⟨*proof*⟩

theorem *move-embed-plain* [*simp*]: *x*∈*small* ⟹ *move*‘(*embed*‘*x*) = *x*
⟨*proof*⟩

definition *ret-def*: *ret* ≡ *converse*(*move*)

lemma *ret-bij*: *ret*∈*bij*(*newbig*, *big*) ⟨*proof*⟩

lemma *ret-inj*: *ret*∈*inj*(*newbig*, *big*) ⟨*proof*⟩
lemma *ret-surj*: *ret*∈*surj*(*newbig*, *big*) ⟨*proof*⟩

lemma *ret-restrict*: *embed* = *restrict*(*ret*, *small*)

⟨*proof*⟩

end

3.3 ZF generalization

We will need this lemma to assert that ZF generalization is an arbitrary generalization:

lemma *mem-not-refl-2*: $\{t\} \notin t$
⟨*proof*⟩

locale *ZF-generalization* = *generalization-situation*
begin

definition *token-def*: $token \equiv Pow(\bigcup(\bigcup(small)))$

lemma *token-not-small*: $\langle token,x\rangle \notin small$
⟨*proof*⟩

definition *zf-move-fun*::$i \Rightarrow i$ **where** *zf-move-fun-def*: $zf\text{-}move\text{-}fun(x) \equiv$ *if* x: *small2* *then* *spec*‘x *else* $\langle token,x\rangle$
definition *zf-move-def*: $zf\text{-}move \equiv (lam\ x{:}big.\ zf\text{-}move\text{-}fun(x))$

definition *zf-newbig-def*: $zf\text{-}newbig \equiv range(zf\text{-}move)$

lemma *zf-move-domain*: $domain(zf\text{-}move) = big$ ⟨*proof*⟩

lemma *zf-move-fun*: $zf\text{-}move$: $big \to zf\text{-}newbig$ ⟨*proof*⟩

theorem *small-less-zf-newbig*: $small \subseteq zf\text{-}newbig$
⟨*proof*⟩

theorem *zf-move-inj*: $zf\text{-}move \in inj(big,\ zf\text{-}newbig)$
⟨*proof*⟩

theorem *zf-move-surj*: $zf\text{-}move \in surj(big,\ zf\text{-}newbig)$
⟨*proof*⟩

theorem *zf-move-bij*: $zf\text{-}move \in bij(big,\ zf\text{-}newbig)$
⟨*proof*⟩

theorem *zf-move-embed* [*simp*]: $x \in small \Longrightarrow zf\text{-}move`(embed`x) = x$
⟨*proof*⟩

theorem *zf-embed-move*: $zf\text{-}move\ O\ embed = id(small)$
⟨*proof*⟩

end

Next prove that ZF generalization is an arbitrary generalization:

sublocale *ZF-generalization* $\subseteq$ *arbitrary-generalization small big embed zf-newbig zf-move*
⟨*proof*⟩

end

4 An example of generalization

theory *int-obj-ex*
  imports *Generalization*
begin

In this example I show that integers can be considered as a generalization of natural numbers.

interpretation *int-interpr*: *ZF-generalization nat int* (*lam n*:*nat*. *int-of*(*n*))
⟨*proof*⟩

abbreviation *int-obj* ≡ *int-interpr.newbig-param*

Naturals are a subset of integers.

lemma *nat* $\subseteq$ *int-obj* ⟨*proof*⟩

An example of defining an operation on the generalization set:

definition
  add::[*i*,*i*]⇒*i*
  where *add*(*x*,*y*) == *int-interpr.move-param*‘(*int-interpr.ret*‘*x* $+ *int-interpr.ret*‘*y*)

end