

Recursion Theorem

Georgy Dunaev

January 28, 2023

Abstract

This document contains a proof of the recursion theorem. This is a mechanization of the proof of the recursion theorem from the text *Introduction to Set Theory*, by Karel Hrbacek and Thomas Jech. This implementation may be used as the basis for a model of Peano Arithmetic in ZF. While recursion and the natural numbers are already available in ZF, this clean development is much easier to follow.

Contents

1	Recursion Submission	1
2	Basic Set Theory	1
3	Compatible set	2
4	Partial computation	4
5	Set of functions	5
6	Lemmas for recursion theorem	6
7	Recursion theorem	7
8	Lemmas for addition	7
9	Definition of addition	8

1 Recursion Submission

Recursion Theorem is proved in the following document. It also contains the addition on natural numbers. The development is done in the context of Zermelo-Fraenkel set theory.

```
theory recursion  
  imports ZF  
begin
```

2 Basic Set Theory

Useful lemmas about sets, functions and natural numbers

lemma *pisubsig* : $\langle \text{Pi}(A,P) \subseteq \text{Pow}(\text{Sigma}(A,P)) \rangle$
<proof>

lemma *apparg*:
fixes $f A B$
assumes $T0: \langle f: A \rightarrow B \rangle$
assumes $T1: \langle f ' a = b \rangle$
assumes $T2: \langle a \in A \rangle$
shows $\langle \langle a, b \rangle \in f \rangle$
<proof>

theorem *nat-induct-bound* :
assumes $H0: \langle P(0) \rangle$
assumes $H1: \langle !!x. x \in \text{nat} \implies P(x) \implies P(\text{succ}(x)) \rangle$
shows $\langle \forall n \in \text{nat}. P(n) \rangle$
<proof>

theorem *nat-Tr* : $\langle \forall n \in \text{nat}. m \in n \longrightarrow m \in \text{nat} \rangle$
<proof>

theorem *zeroleq* : $\langle \forall n \in \text{nat}. 0 \in n \vee 0 = n \rangle$
<proof>

theorem *JH2-1ii* : $\langle m \in \text{succ}(n) \implies m \in n \vee m = n \rangle$
<proof>

theorem *nat-transitive*: $\langle \forall n \in \text{nat}. \forall k. \forall m. k \in m \wedge m \in n \longrightarrow k \in n \rangle$
<proof>

theorem *nat-xninx* : $\langle \forall n \in \text{nat}. \neg(n \in n) \rangle$
<proof>

theorem *nat-asym* : $\langle \forall n \in \text{nat}. \forall m. \neg(n \in m \wedge m \in n) \rangle$
<proof>

theorem *zerolesucc* : $\langle \forall n \in \text{nat}. 0 \in \text{succ}(n) \rangle$
<proof>

theorem *succ-le* : $\langle \forall n \in \text{nat}. \text{succ}(m) \in \text{succ}(n) \longrightarrow m \in n \rangle$
<proof>

theorem *succ-le2* : $\langle \forall n \in \text{nat}. \forall m. \text{succ}(m) \in \text{succ}(n) \longrightarrow m \in n \rangle$
<proof>

theorem *le-succ* : $\langle \forall n \in \text{nat}. m \in n \longrightarrow \text{succ}(m) \in \text{succ}(n) \rangle$

$\langle \text{proof} \rangle$

theorem *nat-linord*: $\langle \forall n \in \text{nat}. \forall m \in \text{nat}. m \in n \vee m = n \vee n \in m \rangle$
 $\langle \text{proof} \rangle$

lemma *tgb*:

assumes *knat*: $\langle k \in \text{nat} \rangle$

assumes *D*: $\langle t \in k \rightarrow A \rangle$

shows $\langle t \in \text{Pow}(\text{nat} \times A) \rangle$

$\langle \text{proof} \rangle$

3 Compatible set

Union of compatible set of functions is a function.

definition *compat* :: $\langle [i, i] \Rightarrow o \rangle$

where $\text{compat}(f1, f2) == \forall x. \forall y1. \forall y2. \langle x, y1 \rangle \in f1 \wedge \langle x, y2 \rangle \in f2 \rightarrow y1 = y2$

lemma *compatI* [*intro*]:

assumes *H*: $\langle \bigwedge x y1 y2. [\langle x, y1 \rangle \in f1; \langle x, y2 \rangle \in f2] \Longrightarrow y1 = y2 \rangle$

shows $\langle \text{compat}(f1, f2) \rangle$

$\langle \text{proof} \rangle$

lemma *compatD*:

assumes *H*: $\langle \text{compat}(f1, f2) \rangle$

shows $\langle \bigwedge x y1 y2. [\langle x, y1 \rangle \in f1; \langle x, y2 \rangle \in f2] \Longrightarrow y1 = y2 \rangle$

$\langle \text{proof} \rangle$

lemma *compatE*:

assumes *H*: $\langle \text{compat}(f1, f2) \rangle$

and *W*: $\langle \bigwedge x y1 y2. [\langle x, y1 \rangle \in f1; \langle x, y2 \rangle \in f2] \Longrightarrow y1 = y2 \rangle \Longrightarrow E \rangle$

shows $\langle E \rangle$

$\langle \text{proof} \rangle$

definition *compatset* :: $\langle i \Rightarrow o \rangle$

where $\text{compatset}(S) == \forall f1 \in S. \forall f2 \in S. \text{compat}(f1, f2)$

lemma *compatsetI* [*intro*] :

assumes *I*: $\langle \bigwedge f1 f2. [f1 \in S; f2 \in S] \Longrightarrow \text{compat}(f1, f2) \rangle$

shows $\langle \text{compatset}(S) \rangle$

$\langle \text{proof} \rangle$

lemma *compatsetD*:

assumes *H*: $\langle \text{compatset}(S) \rangle$

shows $\langle \bigwedge f1 f2. [f1 \in S; f2 \in S] \Longrightarrow \text{compat}(f1, f2) \rangle$

$\langle \text{proof} \rangle$

lemma *compatsetE*:

assumes $H: \langle \text{compatset}(S) \rangle$
and $W: \langle (\bigwedge f1 f2. \llbracket f1 \in S; f2 \in S \rrbracket \implies \text{compat}(f1, f2)) \implies E \rangle$
shows $\langle E \rangle$
 $\langle \text{proof} \rangle$

theorem $\text{upairI1} : \langle a \in \{a, b\} \rangle$
 $\langle \text{proof} \rangle$

theorem $\text{upairI2} : \langle b \in \{a, b\} \rangle$
 $\langle \text{proof} \rangle$

theorem $\text{sinup} : \langle \{x\} \in \langle x, xa \rangle \rangle$
 $\langle \text{proof} \rangle$

theorem $\text{compatsetunionfun} :$
fixes S
assumes $H0: \langle \text{compatset}(S) \rangle$
shows $\langle \text{function}(\bigcup S) \rangle$
 $\langle \text{proof} \rangle$

theorem $\text{mkel} :$
assumes $1: \langle A \rangle$
assumes $2: \langle A \implies B \rangle$
shows $\langle B \rangle$
 $\langle \text{proof} \rangle$

theorem $\text{valofunion} :$
fixes S
assumes $H0: \langle \text{compatset}(S) \rangle$
assumes $W: \langle f \in S \rangle$
assumes $Q: \langle f: A \rightarrow B \rangle$
assumes $T: \langle a \in A \rangle$
assumes $P: \langle f \ ' a = v \rangle$
shows $N: \langle (\bigcup S) \ ' a = v \rangle$
 $\langle \text{proof} \rangle$

4 Partial computation

definition $\text{satpc} :: \langle [i, i, i] \Rightarrow o \rangle$
where $\langle \text{satpc}(t, \alpha, g) == \forall n \in \alpha . t' \text{succ}(n) = g \ ' \langle t'n, n \rangle \rangle$
 m -step computation based on a and g

definition $\text{partcomp} :: \langle [i, i, i, i] \Rightarrow o \rangle$
where $\langle \text{partcomp}(A, t, m, a, g) == (t: \text{succ}(m) \rightarrow A) \wedge (t'0 = a) \wedge \text{satpc}(t, m, g) \rangle$

lemma partcompI [*intro*]:
assumes $H1: \langle (t: \text{succ}(m) \rightarrow A) \rangle$
assumes $H2: \langle (t'0 = a) \rangle$
assumes $H3: \langle \text{satpc}(t, m, g) \rangle$

shows $\langle \text{partcomp}(A, t, m, a, g) \rangle$
 $\langle \text{proof} \rangle$

lemma *partcompD1*: $\langle \text{partcomp}(A, t, m, a, g) \implies t \in \text{succ}(m) \rightarrow A \rangle$
 $\langle \text{proof} \rangle$

lemma *partcompD2*: $\langle \text{partcomp}(A, t, m, a, g) \implies (t'0=a) \rangle$
 $\langle \text{proof} \rangle$

lemma *partcompD3*: $\langle \text{partcomp}(A, t, m, a, g) \implies \text{satpc}(t, m, g) \rangle$
 $\langle \text{proof} \rangle$

lemma *partcompE* [*elim*] :
assumes $1: \langle \text{partcomp}(A, t, m, a, g) \rangle$
and $2: \langle \llbracket (t: \text{succ}(m) \rightarrow A) ; (t'0=a) ; \text{satpc}(t, m, g) \rrbracket \implies E \rangle$
shows $\langle E \rangle$
 $\langle \text{proof} \rangle$

If we add ordered pair in the middle of partial computation then it will not change.

lemma *addmiddle*:

assumes $m \text{ nat}: \langle m \in \text{nat} \rangle$
assumes $F: \langle \text{partcomp}(A, t, m, a, g) \rangle$
assumes $x \text{ in } m: \langle x \in m \rangle$
shows $\langle \text{cons}(\langle \text{succ}(x), g \langle t \langle x, x \rangle \rangle, t) = t \rangle$
 $\langle \text{proof} \rangle$

5 Set of functions

It is denoted as F on page 48 in "Introduction to Set Theory".

definition *pcs* :: $\langle [i, i, i] \Rightarrow i \rangle$
where $\langle \text{pcs}(A, a, g) == \{t \in \text{Pow}(\text{nat} * A). \exists m \in \text{nat}. \text{partcomp}(A, t, m, a, g)\} \rangle$

lemma *pcs-uniq* :
assumes $F1: \langle m1 \in \text{nat} \rangle$
assumes $F2: \langle m2 \in \text{nat} \rangle$
assumes $H1: \langle \text{partcomp}(A, f1, m1, a, g) \rangle$
assumes $H2: \langle \text{partcomp}(A, f2, m2, a, g) \rangle$
shows $\langle \forall n \in \text{nat}. n \in \text{succ}(m1) \wedge n \in \text{succ}(m2) \longrightarrow f1'n = f2'n \rangle$
 $\langle \text{proof} \rangle$

lemma *domainsubsetfunc* :
assumes $Q: \langle f1 \subseteq f2 \rangle$
shows $\langle \text{domain}(f1) \subseteq \text{domain}(f2) \rangle$
 $\langle \text{proof} \rangle$

lemma *natdomfunc*:

assumes $1: \langle q \in A \rangle$
assumes $J0: \langle f1 \in \text{Pow}(\text{nat} \times A) \rangle$
assumes $U: \langle m1 \in \text{domain}(f1) \rangle$
shows $\langle m1 \in \text{nat} \rangle$
 $\langle \text{proof} \rangle$

lemma *pcs-lem* :
assumes $1: \langle q \in A \rangle$
shows $\langle \text{compatset}(\text{pcs}(A, a, g)) \rangle$
 $\langle \text{proof} \rangle$

theorem *fuissu* : $\langle f \in X \rightarrow Y \implies f \subseteq X \times Y \rangle$
 $\langle \text{proof} \rangle$

theorem *recuniq* :
fixes f
assumes $H0: \langle f \in \text{nat} \rightarrow A \wedge f \text{ ' } 0 = a \wedge \text{satpc}(f, \text{nat}, g) \rangle$
fixes t
assumes $H1: \langle t \in \text{nat} \rightarrow A \wedge t \text{ ' } 0 = a \wedge \text{satpc}(t, \text{nat}, g) \rangle$
fixes x
shows $\langle f=t \rangle$
 $\langle \text{proof} \rangle$

6 Lemmas for recursion theorem

locale *recthm* =
fixes $A :: i$
and $a :: i$
and $g :: i$
assumes $\text{hyp1} : \langle a \in A \rangle$
and $\text{hyp2} : \langle g : ((A * \text{nat}) \rightarrow A) \rangle$
begin

lemma *l3*: $\langle \text{function}(\bigcup \text{pcs}(A, a, g)) \rangle$
 $\langle \text{proof} \rangle$

lemma *l1* : $\langle \bigcup \text{pcs}(A, a, g) \subseteq \text{nat} \times A \rangle$
 $\langle \text{proof} \rangle$

lemma *le1*:
assumes $H: \langle x \in 1 \rangle$
shows $\langle x=0 \rangle$
 $\langle \text{proof} \rangle$

lemma *lsinglfun* : $\langle \text{function}(\{\langle 0, a \rangle\}) \rangle$
 $\langle \text{proof} \rangle$

lemma *singlsatpc*: $\langle \text{satpc}(\{\langle 0, a \rangle\}, 0, g) \rangle$
 $\langle \text{proof} \rangle$

lemma *zerostep* :
shows $\langle \text{partcomp}(A, \{\langle 0, a \rangle\}, 0, a, g) \rangle$
 $\langle \text{proof} \rangle$

lemma *zainupcs* : $\langle \langle 0, a \rangle \in \bigcup \text{pcs}(A, a, g) \rangle$
 $\langle \text{proof} \rangle$

lemma *l2'*: $\langle 0 \in \text{domain}(\bigcup \text{pcs}(A, a, g)) \rangle$
 $\langle \text{proof} \rangle$

Push an ordered pair to the end of partial computation t and obtain another partial computation.

lemma *shortlem* :
assumes $m \text{ nat} : \langle m \in \text{nat} \rangle$
assumes $F : \langle \text{partcomp}(A, t, m, a, g) \rangle$
shows $\langle \text{partcomp}(A, \text{cons}(\langle \text{succ}(m), g \text{ ' } \langle t' m, m \rangle \rangle), t), \text{succ}(m), a, g \rangle$
 $\langle \text{proof} \rangle$

lemma *l2*: $\langle \text{nat} \subseteq \text{domain}(\bigcup \text{pcs}(A, a, g)) \rangle$
 $\langle \text{proof} \rangle$

lemma *useful* : $\langle \forall m \in \text{nat}. \exists t. \text{partcomp}(A, t, m, a, g) \rangle$
 $\langle \text{proof} \rangle$

lemma *l4* : $\langle (\bigcup \text{pcs}(A, a, g)) \in \text{nat} \rightarrow A \rangle$
 $\langle \text{proof} \rangle$

lemma *l5*: $\langle (\bigcup \text{pcs}(A, a, g)) \text{ ' } 0 = a \rangle$
 $\langle \text{proof} \rangle$

lemma *ballE2*:
assumes $\langle \forall x \in A. P(x) \rangle$
assumes $\langle x \in A \rangle$
assumes $\langle P(x) ==> Q \rangle$
shows Q
 $\langle \text{proof} \rangle$

Recall that $\text{satpc}(t, \alpha, g) == \forall n \in \alpha . t' \text{succ}(n) = g \text{ ' } \langle t' n, n \rangle \text{ part-}$
 $\text{comp}(A, t, m, a, g) == (t : \text{succ}(m) \rightarrow A) \wedge (t' 0 = a) \wedge \text{satpc}(t, m, g) \text{ pcs}(A, a, g)$
 $== \{t \in \text{Pow}(\text{nat} * A). \exists m. \text{partcomp}(A, t, m, a, g)\}$

lemma *l6new*: $\langle \text{satpc}(\bigcup \text{pcs}(A, a, g), \text{nat}, g) \rangle$
 $\langle \text{proof} \rangle$

7 Recursion theorem

theorem *recursionthm*:
shows $\langle \exists ! f. ((f \in (\text{nat} \rightarrow A)) \wedge ((f' 0) = a) \wedge \text{satpc}(f, \text{nat}, g)) \rangle$

⟨proof⟩

end

8 Lemmas for addition

Let's define function $t(x) = (a+x)$. Firstly we need to define a function $g: nat \times nat \rightarrow nat$, such that $g\langle t'n, n \rangle = t'succ(n) = a + (n + 1) = (a + n) + 1 = (t'n) + 1$ So $g\langle a, b \rangle = a + 1$ and $g(p) = succ(pr1(p))$ and $satpc(t, \alpha, g) \iff \forall n \in \alpha . t'succ(n) = succ(t'n)$.

definition $addg :: \langle i \rangle$

where $addg-def : \langle addg == \lambda x \in (nat * nat) . succ(fst(x)) \rangle$

lemma $addgfun: \langle function(addg) \rangle$

⟨proof⟩

lemma $addgsubpow : \langle addg \in Pow((nat \times nat) \times nat) \rangle$

⟨proof⟩

lemma $addgdom : \langle nat \times nat \subseteq domain(addg) \rangle$

⟨proof⟩

lemma $plussucc:$

assumes $F: \langle f \in (nat \rightarrow nat) \rangle$

assumes $H: \langle satpc(f, nat, addg) \rangle$

shows $\langle \forall n \in nat . f'succ(n) = succ(f'n) \rangle$

⟨proof⟩

9 Definition of addition

Theorem that addition of natural numbers exists and unique in some sense. Due to theorem 'plussucc' the term $satpc(f, nat, addg)$ can be replaced here with $\forall n \in nat . f'succ(n) = succ(f'n)$.

theorem $addition:$

assumes $\langle a \in nat \rangle$

shows

$\langle \exists ! f . ((f \in (nat \rightarrow nat)) \wedge ((f'0) = a) \wedge satpc(f, nat, addg)) \rangle$

⟨proof⟩

end