

# Recursion Theorem

Georgy Dunaev

June 16, 2024

## 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

<b>1</b>	<b>Recursion Submission</b>	<b>1</b>
<b>2</b>	<b>Basic Set Theory</b>	<b>1</b>
<b>3</b>	<b>Compatible set</b>	<b>9</b>
<b>4</b>	<b>Partial computation</b>	<b>12</b>
<b>5</b>	<b>Set of functions</b>	<b>13</b>
<b>6</b>	<b>Lemmas for recursion theorem</b>	<b>19</b>
<b>7</b>	<b>Recursion theorem</b>	<b>29</b>
<b>8</b>	<b>Lemmas for addition</b>	<b>30</b>
<b>9</b>	<b>Definition of addition</b>	<b>32</b>

## 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* :  $\langle Pi(A,P) \subseteq Pow(Sigma(A,P)) \rangle$

**proof**

**fix** *x*

**assume**  $\langle x \in Pi(A,P) \rangle$

**hence**  $\langle x \in \{f \in Pow(Sigma(A,P)). A \subseteq domain(f) \ \& \ function(f)\} \rangle$

**by** (*unfold Pi-def*)

**thus**  $\langle x \in Pow(Sigma(A, P)) \rangle$

**by** (*rule CollectD1*)

**qed**

**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**(*rule iffD2[OF func.apply-iff], rule T0*)

**show** *T*:  $\langle a \in A \wedge f \ ' \ a = b \rangle$

**by** (*rule conjI[OF T2 T1]*)

**qed**

**theorem** *nat-induct-bound* :

**assumes** *H0*:  $\langle P(0) \rangle$

**assumes** *H1*:  $\langle \forall x. x \in nat \implies P(x) \implies P(succ(x)) \rangle$

**shows**  $\langle \forall n \in nat. P(n) \rangle$

**proof**(*rule ballI*)

**fix** *n*

**assume** *H2*:  $\langle n \in nat \rangle$

**show**  $\langle P(n) \rangle$

**proof**(*rule nat-induct[of n]*)

**from** *H2* **show**  $\langle n \in nat \rangle$  **by** *assumption*

**next**

**show**  $\langle P(0) \rangle$  **by** (*rule H0*)

**next**

**fix** *x*

**assume** *H3*:  $\langle x \in nat \rangle$

**assume** *H4*:  $\langle P(x) \rangle$

**show**  $\langle P(succ(x)) \rangle$  **by** (*rule H1[OF H3 H4]*)

**qed**

**qed**

**theorem** *nat-Tr* :  $\langle \forall n \in nat. m \in n \longrightarrow m \in nat \rangle$

**proof**(*rule nat-induct-bound*)

**show**  $\langle m \in 0 \longrightarrow m \in nat \rangle$  **by** *auto*

**next**

**fix** *x*

```

assume  $H0: \langle x \in \text{nat} \rangle$ 
assume  $H1: \langle m \in x \longrightarrow m \in \text{nat} \rangle$ 
show  $\langle m \in \text{succ}(x) \longrightarrow m \in \text{nat} \rangle$ 
proof(rule impI)
  assume  $H2: \langle m \in \text{succ}(x) \rangle$ 
  show  $\langle m \in \text{nat} \rangle$ 
  proof(rule succE[OF H2])
    assume  $H3: \langle m = x \rangle$ 
    from  $H0$  and  $H3$  show  $\langle m \in \text{nat} \rangle$ 
    by auto
  next
  assume  $H4: \langle m \in x \rangle$ 
  show  $\langle m \in \text{nat} \rangle$ 
  by(rule mp[OF H1 H4])
qed
qed
qed

theorem zeroleq :  $\langle \forall n \in \text{nat}. 0 \in n \vee 0 = n \rangle$ 
proof(rule ballI)
  fix  $n$ 
  assume  $H1: \langle n \in \text{nat} \rangle$ 
  show  $\langle 0 \in n \vee 0 = n \rangle$ 
  proof(rule nat-induct[of n])
    from  $H1$  show  $\langle n \in \text{nat} \rangle$  by assumption
  next
  show  $\langle 0 \in 0 \vee 0 = 0 \rangle$  by (rule disjI2, rule refl)
  next
  fix  $x$ 
  assume  $H2: \langle x \in \text{nat} \rangle$ 
  assume  $H3: \langle 0 \in x \vee 0 = x \rangle$ 
  show  $\langle 0 \in \text{succ}(x) \vee 0 = \text{succ}(x) \rangle$ 
  proof(rule disjE[OF H3])
    assume  $H4: \langle 0 \in x \rangle$ 
    show  $\langle 0 \in \text{succ}(x) \vee 0 = \text{succ}(x) \rangle$ 
    proof(rule disjI1)
      show  $\langle 0 \in \text{succ}(x) \rangle$ 
      by (rule succI2[OF H4])
    qed
  next
  assume  $H4: \langle 0 = x \rangle$ 
  show  $\langle 0 \in \text{succ}(x) \vee 0 = \text{succ}(x) \rangle$ 
  proof(rule disjI1)
    have  $q: \langle x \in \text{succ}(x) \rangle$  by auto
    from  $q$  and  $H4$  show  $\langle 0 \in \text{succ}(x) \rangle$  by auto
  qed
qed
qed

```

qed

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

**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**(*rule nat-induct-bound*)

show  $\langle \forall k. \forall m. k \in m \wedge m \in 0 \longrightarrow k \in 0 \rangle$   
**proof**(*rule allI, rule allI, rule impI*)

fix *k m*  
assume *H*:  $\langle k \in m \wedge m \in 0 \rangle$   
then have *H*:  $\langle m \in 0 \rangle$  by *auto*  
then show  $\langle k \in 0 \rangle$  by *auto*

qed

next

fix *n*  
assume *H0*:  $\langle n \in \text{nat} \rangle$   
assume *H1*:  $\langle \forall k. \forall m. \langle k \in m \wedge m \in n \longrightarrow k \in n \rangle \rangle$

show  $\langle \forall k. \forall m. \langle k \in m \wedge m \in \text{succ}(n) \longrightarrow k \in \text{succ}(n) \rangle \rangle$   
**proof**(*rule allI, rule allI, rule impI*)

fix *k m*  
assume *H4*:  $\langle k \in m \wedge m \in \text{succ}(n) \rangle$   
hence *H4'*:  $\langle m \in \text{succ}(n) \rangle$  by (*rule conjunct2*)  
hence *H4''*:  $\langle m \in n \vee m = n \rangle$  by (*rule succE, auto*)  
from *H4* have *Q*:  $\langle k \in m \rangle$  by (*rule conjunct1*)  
have *H1S*:  $\langle \forall m. k \in m \wedge m \in n \longrightarrow k \in n \rangle$   
by (*rule spec[OF H1]*)

have *H1S*:  $\langle k \in m \wedge m \in n \longrightarrow k \in n \rangle$   
by (*rule spec[OF H1S]*)

show  $\langle k \in \text{succ}(n) \rangle$   
**proof**(*rule disjE[OF H4'']*)

assume *L*:  $\langle m \in n \rangle$   
from *Q* and *L* have *QL*:  $\langle k \in m \wedge m \in n \rangle$  by *auto*  
have *G*:  $\langle k \in n \rangle$  by (*rule mp [OF H1S QL]*)  
show  $\langle k \in \text{succ}(n) \rangle$   
by (*rule succI2[OF G]*)

next

assume *L*:  $\langle m = n \rangle$   
from *Q* have *F*:  $\langle k \in \text{succ}(m) \rangle$  by *auto*  
from *L* and *Q* show  $\langle k \in \text{succ}(n) \rangle$  by *auto*

qed

qed

qed

```

theorem nat-xninx :  $\langle \forall n \in \text{nat}. \neg(n \in n) \rangle$ 
proof(rule nat-induct-bound)
  show  $\langle 0 \notin 0 \rangle$ 
    by auto
next
  fix x
  assume H0:  $\langle x \in \text{nat} \rangle$ 
  assume H1:  $\langle x \notin x \rangle$ 
  show  $\langle \text{succ}(x) \notin \text{succ}(x) \rangle$ 
  proof(rule contrapos[OF H1])
    assume Q:  $\langle \text{succ}(x) \in \text{succ}(x) \rangle$ 
    have D:  $\langle \text{succ}(x) \in x \vee \text{succ}(x) = x \rangle$ 
      by (rule JH2-1ii[OF Q])
    show  $\langle x \in x \rangle$ 
  proof(rule disjE[OF D])
    assume Y1:  $\langle \text{succ}(x) \in x \rangle$ 
    have U:  $\langle x \in \text{succ}(x) \rangle$  by (rule succI1)
    have T:  $\langle x \in \text{succ}(x) \wedge \text{succ}(x) \in x \longrightarrow x \in x \rangle$ 
      by (rule spec[OF spec[OF bspec[OF nat-transitive H0]]])
    have R:  $\langle x \in \text{succ}(x) \wedge \text{succ}(x) \in x \rangle$ 
      by (rule conjI[OF U Y1])
    show  $\langle x \in x \rangle$ 
      by (rule mp[OF T R])
  next
    assume Y1:  $\langle \text{succ}(x) = x \rangle$ 
    show  $\langle x \in x \rangle$ 
      by (rule subst[OF Y1], rule Q)
  qed
qed
qed

theorem nat-asym :  $\langle \forall n \in \text{nat}. \forall m. \neg(n \in m \wedge m \in n) \rangle$ 
proof(rule ballI, rule allI)
  fix n m
  assume H0:  $\langle n \in \text{nat} \rangle$ 
  have Q:  $\langle \neg(n \in n) \rangle$ 
    by(rule bspec[OF nat-xninx H0])
  show  $\langle \neg(n \in m \wedge m \in n) \rangle$ 
  proof(rule contrapos[OF Q])
    assume W:  $\langle n \in m \wedge m \in n \rangle$ 
    show  $\langle n \in n \rangle$ 
      by (rule mp[OF spec[OF spec[OF bspec[OF nat-transitive H0]]] W])
  qed
qed

theorem zerolesucc :  $\langle \forall n \in \text{nat}. 0 \in \text{succ}(n) \rangle$ 
proof(rule nat-induct-bound)
  show  $\langle 0 \in 1 \rangle$ 

```

```

    by auto
next
fix x
assume H0:⟨x∈nat⟩
assume H1:⟨0∈succ(x)⟩
show ⟨0∈succ(succ(x))⟩
proof
  assume J:⟨0 ∉ succ(x)⟩
  show ⟨0 = succ(x)⟩
  by(rule notE[OF J H1])
qed
qed

theorem succ-le : ⟨∀ n∈nat. succ(m)∈succ(n) ⟶ m∈n⟩
proof(rule nat-induct-bound)
  show ⟨succ(m) ∈ 1 ⟶ m ∈ 0⟩
  by blast
next
fix x
assume H0:⟨x ∈ nat⟩
assume H1:⟨succ(m) ∈ succ(x) ⟶ m ∈ x⟩
show ⟨succ(m) ∈
      succ(succ(x)) ⟶
      m ∈ succ(x)⟩
proof(rule impI)
  assume J0:⟨succ(m) ∈ succ(succ(x))⟩
  show ⟨m ∈ succ(x)⟩
  proof(rule succE[OF J0])
    assume R:⟨succ(m) = succ(x)⟩
    hence R:⟨m=x⟩ by (rule upair.succ-inject)
    from R and succI1 show ⟨m ∈ succ(x)⟩ by auto
  next
    assume R:⟨succ(m) ∈ succ(x)⟩
    have R:⟨m∈x⟩ by (rule mp[OF H1 R])
    then show ⟨m ∈ succ(x)⟩ by auto
  qed
qed
qed

theorem succ-le2 : ⟨∀ n∈nat. ∀ m. succ(m)∈succ(n) ⟶ m∈n⟩
proof
  fix n
  assume H:⟨n∈nat⟩
  show ⟨∀ m. succ(m) ∈ succ(n) ⟶ m ∈ n⟩
  proof
    fix m
    from succ-le and H show ⟨succ(m) ∈ succ(n) ⟶ m ∈ n⟩ by auto
  qed
qed

```

```

theorem le-succ :  $\langle \forall n \in \text{nat}. m \in n \longrightarrow \text{succ}(m) \in \text{succ}(n) \rangle$ 
proof(rule nat-induct-bound)
  show  $\langle m \in 0 \longrightarrow \text{succ}(m) \in 1 \rangle$ 
    by auto
next
  fix x
  assume H0:  $\langle x \in \text{nat} \rangle$ 
  assume H1:  $\langle m \in x \longrightarrow \text{succ}(m) \in \text{succ}(x) \rangle$ 
  show  $\langle m \in \text{succ}(x) \longrightarrow$ 
     $\text{succ}(m) \in \text{succ}(\text{succ}(x)) \rangle$ 
  proof(rule impI)
    assume HR1:  $\langle m \in \text{succ}(x) \rangle$ 
    show  $\langle \text{succ}(m) \in \text{succ}(\text{succ}(x)) \rangle$ 
    proof(rule succE[OF HR1])
      assume Q:  $\langle m = x \rangle$ 
      from Q show  $\langle \text{succ}(m) \in \text{succ}(\text{succ}(x)) \rangle$ 
        by auto
    next
      assume Q:  $\langle m \in x \rangle$ 
      have Q:  $\langle \text{succ}(m) \in \text{succ}(x) \rangle$ 
        by (rule mp[OF H1 Q])
      from Q show  $\langle \text{succ}(m) \in \text{succ}(\text{succ}(x)) \rangle$ 
        by (rule succI2)
    qed
  qed
qed

theorem nat-linord:  $\langle \forall n \in \text{nat}. \forall m \in \text{nat}. m \in n \vee m = n \vee n \in m \rangle$ 
proof(rule ballI)
  fix n
  assume H1:  $\langle n \in \text{nat} \rangle$ 
  show  $\langle \forall m \in \text{nat}. m \in n \vee m = n \vee n \in m \rangle$ 
  proof(rule nat-induct[of n])
    from H1 show  $\langle n \in \text{nat} \rangle$  by assumption
  next
    show  $\langle \forall m \in \text{nat}. m \in 0 \vee m = 0 \vee 0 \in m \rangle$ 
    proof
      fix m
      assume J:  $\langle m \in \text{nat} \rangle$ 
      show  $\langle m \in 0 \vee m = 0 \vee 0 \in m \rangle$ 
      proof(rule disjI2)
        have Q:  $\langle 0 \in m \vee 0 = m \rangle$  by (rule bspec[OF zeroleq J])
        show  $\langle m = 0 \vee 0 \in m \rangle$ 
          by (rule disjE[OF Q], auto)
      qed
    qed
  next
    fix x

```

```

assume  $K:\langle x \in \text{nat} \rangle$ 
assume  $M:\langle \forall m \in \text{nat}. m \in x \vee m = x \vee x \in m \rangle$ 
show  $\langle \forall m \in \text{nat}. m \in \text{succ}(x) \vee m = \text{succ}(x) \vee \text{succ}(x) \in m \rangle$ 
proof(rule nat-induct-bound)
  show  $\langle 0 \in \text{succ}(x) \vee 0 = \text{succ}(x) \vee \text{succ}(x) \in 0 \rangle$ 
  proof(rule disjI1)
    show  $\langle 0 \in \text{succ}(x) \rangle$ 
    by (rule bspec[OF zerosucc  $K$ ])
  qed
next
fix  $y$ 
assume  $H0:\langle y \in \text{nat} \rangle$ 
assume  $H1:\langle y \in \text{succ}(x) \vee y = \text{succ}(x) \vee \text{succ}(x) \in y \rangle$ 
show  $\langle \text{succ}(y) \in \text{succ}(x) \vee \text{succ}(y) = \text{succ}(x) \vee \text{succ}(x) \in \text{succ}(y) \rangle$ 
proof(rule disjE[OF H1])
  assume  $W:\langle y \in \text{succ}(x) \rangle$ 
  show  $\langle \text{succ}(y) \in \text{succ}(x) \vee \text{succ}(y) = \text{succ}(x) \vee \text{succ}(x) \in \text{succ}(y) \rangle$ 
  proof(rule succE[OF W])
    assume  $G:\langle y = x \rangle$ 
    show  $\langle \text{succ}(y) \in \text{succ}(x) \vee \text{succ}(y) = \text{succ}(x) \vee \text{succ}(x) \in \text{succ}(y) \rangle$ 
    by (rule disjI2, rule disjI1, rule subst[OF G], rule refl)
  next
  assume  $G:\langle y \in x \rangle$ 
  have  $R:\langle \text{succ}(y) \in \text{succ}(x) \rangle$ 
  by (rule mp[OF bspec[OF le-succ  $K$ ]  $G$ ])
  show  $\langle \text{succ}(y) \in \text{succ}(x) \vee \text{succ}(y) = \text{succ}(x) \vee \text{succ}(x) \in \text{succ}(y) \rangle$ 
  by(rule disjI1, rule R)
  qed
next
assume  $W:\langle y = \text{succ}(x) \vee \text{succ}(x) \in y \rangle$ 
show  $\langle \text{succ}(y) \in \text{succ}(x) \vee \text{succ}(y) = \text{succ}(x) \vee \text{succ}(x) \in \text{succ}(y) \rangle$ 
proof(rule disjE[OF W])
  assume  $W:\langle y = \text{succ}(x) \rangle$ 
  show  $\langle \text{succ}(y) \in \text{succ}(x) \vee \text{succ}(y) = \text{succ}(x) \vee \text{succ}(x) \in \text{succ}(y) \rangle$ 

```



```

    by (rule disjI2, rule disjI2, rule subst[OF W], rule succI1)
  next
    assume W:⟨succ(x)∈y⟩
    show ⟨succ(y) ∈ succ(x) ∨
          succ(y) = succ(x) ∨
          succ(x) ∈ succ(y)⟩
      by (rule disjI2, rule disjI2, rule succI2[OF W])
  qed
qed
qed
qed
qed

```

lemma *tgb*:

```

  assumes knat: ⟨k∈nat⟩
  assumes D: ⟨t ∈ k → A⟩
  shows ⟨t ∈ Pow(nat × A)⟩
proof -
  from D
  have q:⟨t∈{t∈Pow(Sigma(k,%-.A)). k⊆domain(t) & function(t)}⟩
    by(unfold Pi-def)
  have J:⟨t ∈ Pow(k × A)⟩
    by (rule CollectD1[OF q])
  have G:⟨k × A ⊆ nat × A⟩
  proof(rule func.Sigma-mono)
    from knat
    show ⟨k⊆nat⟩
      by (rule QUniv.naturals-subset-nat)
  next
    show ⟨∧x. x ∈ k ⇒ A ⊆ A⟩
      by auto
  qed
  show ⟨t ∈ Pow(nat × A)⟩
    by (rule subsetD, rule func.Pow-mono[OF G], rule J)
qed

```

### 3 Compatible set

Union of compatible set of functions is a function.

**definition** *compat* :: ⟨[i,i]⇒o⟩

where  $compat(f1,f2) == \forall x.\forall y1.\forall y2.\langle x,y1\rangle \in f1 \wedge \langle x,y2\rangle \in f2 \longrightarrow y1=y2$

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 ⟨*compat*(*f1*,*f2*)⟩

**proof**(*unfold compat-def*)

show ⟨ $\forall x y1 y2. \langle x, y1\rangle \in f1 \wedge \langle x, y2\rangle \in f2 \longrightarrow y1 = y2$ ⟩

**proof**(*rule allI | rule impI*)+

```

    fix x y1 y2
    assume K:⟨x, y1⟩ ∈ f1 ∧ ⟨x, y2⟩ ∈ f2⟩
    have K1:⟨x, y1⟩ ∈ f1⟩ by (rule conjunct1[OF K])
    have K2:⟨x, y2⟩ ∈ f2⟩ by (rule conjunct2[OF K])
    show ⟨y1 = y2⟩ by (rule H[OF K1 K2])
  qed
qed

lemma compatD:
  assumes H: ⟨compat(f1,f2)⟩
  shows ⟨∧x y1 y2. [[⟨x,y1⟩ ∈ f1; ⟨x,y2⟩ ∈ f2]] ⇒ y1=y2⟩
proof -
  fix x y1 y2
  assume Q1:⟨x, y1⟩ ∈ f1⟩
  assume Q2:⟨x, y2⟩ ∈ f2⟩
  from H have H:⟨∀x y1 y2. ⟨x, y1⟩ ∈ f1 ∧ ⟨x, y2⟩ ∈ f2 ⟶ y1 = y2⟩
    by (unfold compat-def)
  show ⟨y1=y2⟩
  proof(rule mp[OF spec[OF spec[OF spec[OF H]]]])
    show ⟨x, y1⟩ ∈ f1 ∧ ⟨x, y2⟩ ∈ f2⟩
    by(rule conjI[OF Q1 Q2])
  qed
qed

lemma compatE:
  assumes H: ⟨compat(f1,f2)⟩
  and W:⟨(∧x y1 y2. [[⟨x,y1⟩ ∈ f1; ⟨x,y2⟩ ∈ f2]] ⇒ y1=y2) ⇒ E⟩
  shows ⟨E⟩
  by (rule W, rule compatD[OF H], assumption+)

definition compatset :: ⟨i⇒o⟩
  where compatset(S) == ∀f1∈S.∀f2∈S. compat(f1,f2)

lemma compatsetI [intro] :
  assumes 1:⟨∧f1 f2. [[f1∈S;f2∈S]] ⇒ compat(f1,f2)⟩
  shows ⟨compatset(S)⟩
  by (unfold compatset-def, rule ballI, rule ballI, rule 1, assumption+)

lemma compatsetD:
  assumes H: ⟨compatset(S)⟩
  shows ⟨∧f1 f2. [[f1∈S; f2∈S]] ⇒ compat(f1,f2)⟩
proof -
  fix f1 f2
  assume H1:⟨f1∈S⟩
  assume H2:⟨f2∈S⟩
  from H have H:⟨∀f1∈S.∀f2∈S. compat(f1,f2)⟩
    by (unfold compatset-def)
  show ⟨compat(f1,f2)⟩

```

by (rule bspec[OF bspec[OF H H1] H2])  
 qed

**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  $\langle E \rangle$   
 by (rule W, rule compatsetD[OF H], assumption+)

**theorem** *upairI1* :  $\langle a \in \{a, b\} \rangle$   
**proof**  
 assume  $\langle a \notin \{b\} \rangle$   
 show  $\langle a = a \rangle$  by (rule refl)  
 qed

**theorem** *upairI2* :  $\langle b \in \{a, b\} \rangle$   
**proof**  
 assume  $H: \langle b \notin \{b\} \rangle$   
 have  $Y: \langle b \in \{b\} \rangle$  by (rule upair.singletonI)  
 show  $\langle b = a \rangle$  by (rule notE[OF H Y])  
 qed

**theorem** *sinup* :  $\langle \{x\} \in \langle x, xa \rangle \rangle$   
**proof** (*unfold Pair-def*)  
 show  $\langle \{x\} \in \{\{x, x\}, \{x, xa\}\} \rangle$   
**proof** (*rule IFOL.subst*)  
 show  $\langle \{x\} \in \{\{x\}, \{x, xa\}\} \rangle$   
 by (rule upairI1)  
 next  
 show  $\langle \{\{x\}, \{x, xa\}\} = \{\{x, x\}, \{x, xa\}\} \rangle$   
 by *blast*  
 qed  
 qed

**theorem** *compatsetunionfun* :  
 fixes  $S$   
 assumes  $H0: \langle \text{compatset}(S) \rangle$   
 shows  $\langle \text{function}(\bigcup S) \rangle$   
**proof** (*unfold function-def*)  
 show  $\langle \forall x y1. \langle x, y1 \rangle \in \bigcup S \longrightarrow$   
 $(\forall y2. \langle x, y2 \rangle \in \bigcup S \longrightarrow y1 = y2) \rangle$   
**proof** (*rule allI, rule allI, rule impI, rule allI, rule impI*)  
 fix  $x y1 y2$   
 assume  $F1: \langle \langle x, y1 \rangle \in \bigcup S \rangle$   
 assume  $F2: \langle \langle x, y2 \rangle \in \bigcup S \rangle$   
 show  $\langle y1 = y2 \rangle$   
**proof** (*rule UnionE[OF F1], rule UnionE[OF F2]*)  
 fix  $f1 f2$   
 assume  $J1: \langle \langle x, y1 \rangle \in f1 \rangle$

```

    assume J2:⟨x, y2⟩ ∈ f2⟩
    assume K1:⟨f1 ∈ S⟩
    assume K2:⟨f2 ∈ S⟩
    have R:⟨compat(f1,f2)⟩
      by (rule compatsetD[OF H0 K1 K2])
    show ⟨y1=y2⟩
      by(rule compatD[OF R J1 J2])
  qed
qed
qed

```

```

theorem mkel :
  assumes 1:⟨A⟩
  assumes 2:⟨A⇒B⟩
  shows ⟨B⟩
  by (rule 2, rule 1)

```

```

theorem valofunion :
  fixes S
  assumes H0:⟨compatset(S)⟩
  assumes W:⟨f∈S⟩
  assumes Q:⟨f:A→B⟩
  assumes T:⟨a∈A⟩
  assumes P:⟨f ‘ a = v⟩
  shows N:⟨(∪ S) ‘ a = v⟩
proof -
  have K:⟨⟨a, v⟩ ∈ f⟩
    by (rule apparg[OF Q P T])
  show N:⟨(∪ S) ‘ a = v⟩
  proof(rule function-apply-equality)
    show ⟨function(∪ S)⟩
      by(rule compatsetunionfun[OF H0])
  next
    show ⟨⟨a, v⟩ ∈ ∪ S⟩
      by(rule UnionI[OF W K ])
  qed
qed

```

## 4 Partial computation

```

definition satpc :: ⟨[i,i,i] ⇒ o ⟩
  where ⟨satpc(t,α,g) == ∀ n ∈ α . t‘succ(n) = g ‘ <t‘n, n>⟩

```

*m*-step computation based on *a* and *g*

```

definition partcomp :: ⟨[i,i,i,i] ⇒ o ⟩
  where ⟨partcomp(A,t,m,a,g) == (t:succ(m)→A) ∧ (t‘0=a) ∧ satpc(t,m,g)⟩

```

```

lemma partcompI [intro]:
  assumes H1:⟨(t:succ(m)→A)⟩

```

```

assumes  $H2: \langle t'0=a \rangle$ 
assumes  $H3: \langle satpc(t,m,g) \rangle$ 
shows  $\langle partcomp(A,t,m,a,g) \rangle$ 
proof (unfold partcomp-def, auto)
  show  $\langle t \in succ(m) \rightarrow A \rangle$  by (rule H1)
  show  $\langle t'0=a \rangle$  by (rule H2)
  show  $\langle satpc(t,m,g) \rangle$  by (rule H3)
qed

```

```

lemma partcompD1:  $\langle partcomp(A,t,m,a,g) \implies t \in succ(m) \rightarrow A \rangle$ 
by (unfold partcomp-def, auto)

```

```

lemma partcompD2:  $\langle partcomp(A,t,m,a,g) \implies (t'0=a) \rangle$ 
by (unfold partcomp-def, auto)

```

```

lemma partcompD3:  $\langle partcomp(A,t,m,a,g) \implies satpc(t,m,g) \rangle$ 
by (unfold partcomp-def, auto)

```

```

lemma partcompE [elim] :
  assumes  $1: \langle partcomp(A,t,m,a,g) \rangle$ 
  and  $2: \langle \llbracket (t: succ(m) \rightarrow A) ; (t'0=a) ; satpc(t,m,g) \rrbracket \implies E \rangle$ 
  shows  $\langle E \rangle$ 
by (rule 2, rule partcompD1[OF 1], rule partcompD2[OF 1], rule partcompD3[OF 1])

```

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

**lemma addmiddle:**

```

assumes  $mnat: \langle m \in nat \rangle$ 
assumes  $F: \langle partcomp(A,t,m,a,g) \rangle$ 
assumes  $xinm: \langle x \in m \rangle$ 
shows  $\langle cons(\langle succ(x), g \langle t'x, x \rangle \rangle, t) = t \rangle$ 
proof (rule partcompE[OF F])
  assume  $F1: \langle t \in succ(m) \rightarrow A \rangle$ 
  assume  $F2: \langle t'0 = a \rangle$ 
  assume  $F3: \langle satpc(t, m, g) \rangle$ 
from  $F3$ 
have  $W: \langle \forall n \in m. t' succ(n) = g \langle t'n, n \rangle \rangle$ 
  by (unfold satpc-def)
have  $U: \langle t' succ(x) = g \langle t'x, x \rangle \rangle$ 
  by (rule bspec[OF W xinm])
have  $E: \langle \langle succ(x), (g \langle t'x, x \rangle) \rangle \in t \rangle$ 
proof (rule apparg[OF F1 U])
  show  $\langle succ(x) \in succ(m) \rangle$ 
  by (rule mp[OF bspec[OF le-succ mnat] xinm])
qed
show ?thesis
by (rule equalities.cons-absorb[OF E])

```

qed

## 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**  $\langle pcs(A, a, g) == \{t \in Pow(nat * A). \exists m \in nat. partcomp(A, t, m, a, g)\} \rangle$

**lemma**  $pcs\text{-}uniq :$   
**assumes**  $F1 : \langle m1 \in nat \rangle$   
**assumes**  $F2 : \langle m2 \in nat \rangle$   
**assumes**  $H1 : \langle partcomp(A, f1, m1, a, g) \rangle$   
**assumes**  $H2 : \langle partcomp(A, f2, m2, a, g) \rangle$   
**shows**  $\langle \forall n \in nat. n \in succ(m1) \wedge n \in succ(m2) \longrightarrow f1'n = f2'n \rangle$   
**proof**( $rule\ partcompE[OF\ H1], rule\ partcompE[OF\ H2]$ )  
**assume**  $H11 : \langle f1 \in succ(m1) \rightarrow A \rangle$   
**assume**  $H12 : \langle f1 ' 0 = a \rangle$   
**assume**  $H13 : \langle satpc(f1, m1, g) \rangle$   
**assume**  $H21 : \langle f2 \in succ(m2) \rightarrow A \rangle$   
**assume**  $H22 : \langle f2 ' 0 = a \rangle$   
**assume**  $H23 : \langle satpc(f2, m2, g) \rangle$   
**show**  $\langle \forall n \in nat. n \in succ(m1) \wedge n \in succ(m2) \longrightarrow f1'n = f2'n \rangle$   
**proof**( $rule\ nat\text{-}induct\text{-}bound$ )  
**from**  $H12$  **and**  $H22$   
**show**  $\langle 0 \in succ(m1) \wedge 0 \in succ(m2) \longrightarrow f1 ' 0 = f2 ' 0 \rangle$   
**by**  $auto$   
**next**  
**fix**  $x$   
**assume**  $J0 : \langle x \in nat \rangle$   
**assume**  $J1 : \langle x \in succ(m1) \wedge x \in succ(m2) \longrightarrow f1 ' x = f2 ' x \rangle$   
**from**  $H13$  **have**  $G1 : \langle \forall n \in m1 . f1'succ(n) = g ' \langle f1'n, n \rangle \rangle$   
**by** ( $unfold\ satpc\text{-}def, auto$ )  
**from**  $H23$  **have**  $G2 : \langle \forall n \in m2 . f2'succ(n) = g ' \langle f2'n, n \rangle \rangle$   
**by** ( $unfold\ satpc\text{-}def, auto$ )  
**show**  $\langle succ(x) \in succ(m1) \wedge succ(x) \in succ(m2) \longrightarrow$   
 $f1 ' succ(x) = f2 ' succ(x) \rangle$   
**proof**  
**assume**  $K : \langle succ(x) \in succ(m1) \wedge succ(x) \in succ(m2) \rangle$   
**from**  $K$  **have**  $K1 : \langle succ(x) \in succ(m1) \rangle$  **by**  $auto$   
**from**  $K$  **have**  $K2 : \langle succ(x) \in succ(m2) \rangle$  **by**  $auto$   
**have**  $K1' : \langle x \in m1 \rangle$  **by** ( $rule\ mp[OF\ bspec[OF\ succ\text{-}le\ F1]\ K1$ )  
**have**  $K2' : \langle x \in m2 \rangle$  **by** ( $rule\ mp[OF\ bspec[OF\ succ\text{-}le\ F2]\ K2$ )  
**have**  $U1 : \langle x \in succ(m1) \rangle$   
**by** ( $rule\ Nat.succ\text{-}in\text{-}naturalD[OF\ K1\ Nat.nat\text{-}succI[OF\ F1]$ )  
**have**  $U2 : \langle x \in succ(m2) \rangle$   
**by** ( $rule\ Nat.succ\text{-}in\text{-}naturalD[OF\ K2\ Nat.nat\text{-}succI[OF\ F2]$ )  
**have**  $Y1 : \langle f1'succ(x) = g ' \langle f1'x, x \rangle \rangle$   
**by** ( $rule\ bspec[OF\ G1\ K1\ \uparrow]$ )

```

have Y2:⟨f2'succ(x) = g ' <f2'x, x>⟩
  by (rule bspec[OF G2 K2 ])
have ⟨f1 ' x = f2 ' x⟩
  by(rule mp[OF J1 conjI[OF U1 U2]])
then have Y:⟨g ' <f1'x, x> = g ' <f2'x, x>⟩ by auto
from Y1 and Y2 and Y
show ⟨f1 ' succ(x) = f2 ' succ(x)⟩
  by auto
qed
qed
qed

```

```

lemma domainsubsetfunc :
  assumes Q:⟨f1 ⊆ f2⟩
  shows ⟨domain(f1) ⊆ domain(f2)⟩
proof
  fix x
  assume H:⟨x ∈ domain(f1)⟩
  show ⟨x ∈ domain(f2)⟩
  proof(rule domainE[OF H])
    fix y
    assume W:⟨(x, y) ∈ f1⟩
    have ⟨(x, y) ∈ f2⟩
      by(rule subsetD[OF Q W])
    then show ⟨x ∈ domain(f2)⟩
      by(rule domainI)
  qed
qed

```

```

lemma natdomfunc:
  assumes 1:⟨q ∈ A⟩
  assumes J0:⟨f1 ∈ Pow(nat × A)⟩
  assumes U:⟨m1 ∈ domain(f1)⟩
  shows ⟨m1 ∈ nat⟩
proof -
  from J0 have J0 : ⟨f1 ⊆ nat × A⟩
    by auto
  have J0:⟨domain(f1) ⊆ domain(nat × A)⟩
    by(rule func.domain-mono[OF J0])
  have F:⟨m1 ∈ domain(nat × A)⟩
    by(rule subsetD[OF J0 U])
  have R:⟨domain(nat × A) = nat⟩
    by (rule equalities.domain-of-prod[OF 1])
  show ⟨m1 ∈ nat⟩
    by(rule subst[OF R], rule F)
qed

```

```

lemma pcs-lem :
  assumes 1:⟨q ∈ A⟩

```

```

shows ⟨compatset(pcs(A, a, g))⟩
proof
  fix f1 f2
  assume H1:⟨f1 ∈ pcs(A, a, g)⟩
  then have H1':⟨f1 ∈ {t∈Pow(nat*A)}. ∃ m∈nat. partcomp(A,t,m,a,g)}⟩ by (unfold
  pcs-def)
  hence H1'A:⟨f1 ∈ Pow(nat*A)⟩ by auto
  hence H1'A':⟨f1 ⊆ (nat*A)⟩ by auto
  assume H2:⟨f2 ∈ pcs(A, a, g)⟩
  then have H2':⟨f2 ∈ {t∈Pow(nat*A)}. ∃ m∈nat. partcomp(A,t,m,a,g)}⟩ by (unfold
  pcs-def)
  show ⟨compat(f1, f2)⟩
  proof(rule compatI)
    fix x y1 y2
    assume P1:⟨⟨x, y1⟩ ∈ f1⟩
    assume P2:⟨⟨x, y2⟩ ∈ f2⟩
    show ⟨y1 = y2⟩
    proof(rule CollectE[OF H1'], rule CollectE[OF H2'])
      assume J0:⟨f1 ∈ Pow(nat × A)⟩
      assume J1:⟨f2 ∈ Pow(nat × A)⟩
      assume J2:⟨∃ m∈nat. partcomp(A, f1, m, a, g)⟩
      assume J3:⟨∃ m∈nat. partcomp(A, f2, m, a, g)⟩
      show ⟨y1 = y2⟩
      proof(rule bexE[OF J2], rule bexE[OF J3])
        fix m1 m2
        assume K1:⟨partcomp(A, f1, m1, a, g)⟩
        assume K2:⟨partcomp(A, f2, m2, a, g)⟩
        hence K2':⟨(f2:succ(m2)→A) ∧ (f2'0=a) ∧ satpc(f2,m2,g)⟩
          by (unfold partcomp-def)
        from K1 have K1'A:⟨(f1:succ(m1)→A)⟩ by (rule partcompD1)
        from K2' have K2'A:⟨(f2:succ(m2)→A)⟩ by auto
        from K1'A have K1'AD:⟨domain(f1) = succ(m1)⟩
          by(rule domain-of-fun)
        from K2'A have K2'AD:⟨domain(f2) = succ(m2)⟩
          by(rule domain-of-fun)
        have L1:⟨f1'x=y1⟩
          by (rule func.apply-equality[OF P1], rule K1'A)
        have L2:⟨f2'x=y2⟩
          by(rule func.apply-equality[OF P2], rule K2'A)
        have m1nat:⟨m1∈nat⟩
        proof(rule natdomfunc[OF 1 J0])
          show ⟨m1 ∈ domain(f1)⟩
            by (rule ssubst[OF K1'AD], auto)
        qed
        have m2nat:⟨m2∈nat⟩
        proof(rule natdomfunc[OF 1 J1])
          show ⟨m2 ∈ domain(f2)⟩
            by (rule ssubst[OF K2'AD], auto)
        qed
      qed
    qed
  qed

```



```

have G1:⟨x, y1⟩ ∈ (nat*A)
  by(rule subsetD[OF H1 'A P1])
have KK:⟨x∈nat⟩
  by(rule SigmaE[OF G1], auto)

have W:⟨f1'x=f2'x⟩
proof(rule mp[OF bspec[OF pcs-uniq KK] ])
  show ⟨m1 ∈ nat⟩
    by (rule m1nat)
  next
  show ⟨m2 ∈ nat⟩
    by (rule m2nat)
  next
  show ⟨partcomp(A, f1, m1, a, g)⟩
    by (rule K1)
  next
  show ⟨partcomp(A, f2, m2, a, g)⟩
    by (rule K2)
  next
  have U1:⟨x ∈ succ(m1)⟩
    by (rule func.domain-type[OF P1 K1 'A])
  have U2:⟨x ∈ succ(m2)⟩
    by (rule func.domain-type[OF P2 K2 'A])
  show ⟨x ∈ succ(m1) ∧ x ∈ succ(m2)⟩
    by (rule conjI[OF U1 U2])
qed
from L1 and W and L2
show ⟨y1 = y2⟩ by auto
qed
qed
qed
qed

```

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

**proof**

**fix** *w*

**assume** *H1* :  $\langle f \in X \rightarrow Y \rangle$

**then have** *J1*: $\langle f \in \{q \in \text{Pow}(\text{Sigma}(X, \lambda. Y)). X \subseteq \text{domain}(q) \ \& \ \text{function}(q)\} \rangle$

**by** (*unfold Pi-def*)

**then have** *J2*: $\langle f \in \text{Pow}(\text{Sigma}(X, \lambda. Y)) \rangle$

**by** *auto*

**then have** *J3*: $\langle f \subseteq \text{Sigma}(X, \lambda. Y) \rangle$

**by** *auto*

**assume** *H2* :  $\langle w \in f \rangle$

**from** *J3* **and** *H2* **have**  $\langle w \in \text{Sigma}(X, \lambda. Y) \rangle$

**by** *auto*

**then have** *J4*: $\langle w \in (\bigcup x \in X. (\bigcup y \in Y. \{x, y\})) \rangle$

**by** *auto*

```

show  $\langle w \in X * Y \rangle$ 
proof (rule UN-E[OF J4])
  fix  $x$ 
  assume  $V1: \langle x \in X \rangle$ 
  assume  $V2: \langle w \in (\bigcup_{y \in Y}. \{\langle x, y \rangle\}) \rangle$ 
  show  $\langle w \in X \times Y \rangle$ 
  proof (rule UN-E[OF V2])
    fix  $y$ 
    assume  $V3: \langle y \in Y \rangle$ 
    assume  $V4: \langle w \in \{\langle x, y \rangle\} \rangle$ 
    then have  $V4: \langle w = \langle x, y \rangle \rangle$ 
      by auto
    have  $v5: \langle \langle x, y \rangle \in \text{Sigma}(X, \lambda-. Y) \rangle$ 
    proof(rule SigmaI)
      show  $\langle x \in X \rangle$  by (rule V1)
    next
      show  $\langle y \in Y \rangle$  by (rule V3)
    qed
    then have  $V5: \langle \langle x, y \rangle \in X * Y \rangle$ 
      by auto
    from  $V4$  and  $V5$  show  $\langle w \in X \times Y \rangle$  by auto
  qed
qed
qed

```

**theorem** *recuniq* :

```

fixes  $f$ 
assumes  $H0: \langle f \in \text{nat} \rightarrow A \wedge f ' 0 = a \wedge \text{satpc}(f, \text{nat}, g) \rangle$ 
fixes  $t$ 
assumes  $H1: \langle t \in \text{nat} \rightarrow A \wedge t ' 0 = a \wedge \text{satpc}(t, \text{nat}, g) \rangle$ 
fixes  $x$ 
shows  $\langle f=t \rangle$ 
proof -
  from  $H0$  have  $H02: \langle \forall n \in \text{nat}. f' \text{succ}(n) = g ' \langle f'n, n \rangle \rangle$  by (unfold satpc-def,
  auto)
  from  $H0$  have  $H01: \langle f ' 0 = a \rangle$  by auto
  from  $H0$  have  $H00: \langle f \in \text{nat} \rightarrow A \rangle$  by auto
  from  $H1$  have  $H12: \langle \forall n \in \text{nat}. t' \text{succ}(n) = g ' \langle t'n, n \rangle \rangle$  by (unfold satpc-def,
  auto)
  from  $H1$  have  $H11: \langle t ' 0 = a \rangle$  by auto
  from  $H1$  have  $H10: \langle t \in \text{nat} \rightarrow A \rangle$  by auto
  show  $\langle f=t \rangle$ 
  proof (rule fun-extension[OF H00 H10])
    fix  $x$ 
    assume  $K: \langle x \in \text{nat} \rangle$ 
    show  $\langle f ' x = t ' x \rangle$ 
    proof(rule nat-induct[of x])
      show  $\langle x \in \text{nat} \rangle$  by (rule K)
    next

```

```

    from H01 and H11 show  $\langle f' 0 = t' 0 \rangle$ 
      by auto
  next
  fix x
  assume A:  $\langle x \in \text{nat} \rangle$ 
  assume B:  $\langle f' x = t' x \rangle$ 
  show  $\langle f' \text{succ}(x) = t' \text{succ}(x) \rangle$ 
  proof -
    from H02 and A have H02':  $\langle f' \text{succ}(x) = g' \langle f' x, x \rangle \rangle$ 
      by (rule bspec)
    from H12 and A have H12':  $\langle t' \text{succ}(x) = g' \langle t' x, x \rangle \rangle$ 
      by (rule bspec)
    from B and H12' have H12'':  $\langle t' \text{succ}(x) = g' \langle f' x, x \rangle \rangle$  by auto
    from H12'' and H02' show  $\langle f' \text{succ}(x) = t' \text{succ}(x) \rangle$  by auto
  qed
qed
qed
qed

```

## 6 Lemmas for recursion theorem

locale *recthm* =

```

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

```

begin

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

by (rule compatsetunionfun, rule pcs-lem, rule hyp1)

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

proof

fix x

assume H:  $\langle x \in \bigcup \text{pcs}(A, a, g) \rangle$

hence H:  $\langle x \in \bigcup \{t \in \text{Pow}(\text{nat} * A). \exists m \in \text{nat}. \text{partcomp}(A, t, m, a, g)\} \rangle$

by (unfold pcs-def)

show  $\langle x \in \text{nat} \times A \rangle$

proof(rule UnionE[OF H])

fix B

assume J1:  $\langle x \in B \rangle$

assume J2:  $\langle B \in \{t \in \text{Pow}(\text{nat} \times A) .$

$\exists m \in \text{nat}. \text{partcomp}(A, t, m, a, g)\} \rangle$

hence J2:  $\langle B \in \text{Pow}(\text{nat} \times A) \rangle$  by auto

hence J2:  $\langle B \subseteq \text{nat} \times A \rangle$  by auto

from J1 and J2 show  $\langle x \in \text{nat} \times A \rangle$

by auto

qed

qed

lemma *le1*:

assumes  $H:\langle x \in 1 \rangle$

shows  $\langle x = 0 \rangle$

proof

show  $\langle x \subseteq 0 \rangle$

proof

fix  $z$

assume  $J:\langle z \in x \rangle$

show  $\langle z \in 0 \rangle$

proof(rule *succE*[*OF*  $H$ ])

assume  $J:\langle x \in 0 \rangle$

show  $\langle z \in 0 \rangle$

by (rule *notE*[*OF* *not-mem-empty*  $J$ ])

next

assume  $K:\langle x = 0 \rangle$

from  $J$  and  $K$  show  $\langle z \in 0 \rangle$

by *auto*

qed

qed

next

show  $\langle 0 \subseteq x \rangle$  by *auto*

qed

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

proof(*unfold* *function-def*)

show  $\langle \forall x y. \langle x, y \rangle \in \{\langle 0, a \rangle\} \longrightarrow$

$\langle \forall y'. \langle x, y' \rangle \in \{\langle 0, a \rangle\} \longrightarrow$

$y = y' \rangle$

proof(rule *allI*,rule *allI*,rule *impI*,rule *allI*,rule *impI*)

fix  $x y y'$

assume  $H0:\langle \langle x, y \rangle \in \{\langle 0, a \rangle\} \rangle$

assume  $H1:\langle \langle x, y' \rangle \in \{\langle 0, a \rangle\} \rangle$

show  $\langle y = y' \rangle$

proof(rule *upair.singletonE*[*OF*  $H0$ ],rule *upair.singletonE*[*OF*  $H1$ ])

assume  $H0:\langle \langle x, y \rangle = \langle 0, a \rangle \rangle$

assume  $H1:\langle \langle x, y' \rangle = \langle 0, a \rangle \rangle$

from  $H0$  and  $H1$  have  $H:\langle \langle x, y \rangle = \langle x, y' \rangle \rangle$  by *auto*

then show  $\langle y = y' \rangle$  by *auto*

qed

qed

qed

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

proof(*unfold* *satpc-def*)

show  $\langle \forall n \in 0. \{\langle 0, a \rangle\} \text{ ' succ}(n) =$

$g \text{ ' } \{\langle 0, a \rangle\} \text{ ' } n, n \rangle$

by *auto*

qed

lemma zerostep :

shows  $\langle \text{partcomp}(A, \{\langle 0, a \rangle\}, 0, a, g) \rangle$

proof (unfold partcomp-def)

show  $\langle \{\langle 0, a \rangle\} \in 1 \rightarrow A \wedge \{\langle 0, a \rangle\} \text{ ' } 0 = a \wedge \text{satpc}(\{\langle 0, a \rangle\}, 0, g) \rangle$

proof

show  $\langle \{\langle 0, a \rangle\} \in 1 \rightarrow A \rangle$

proof (unfold Pi-def)

show  $\langle \{\langle 0, a \rangle\} \in \{f \in \text{Pow}(1 \times A) . 1 \subseteq \text{domain}(f) \wedge \text{function}(f)\} \rangle$

proof

show  $\langle \{\langle 0, a \rangle\} \in \text{Pow}(1 \times A) \rangle$

proof (rule PowI, rule equalities.singleton-subsetI)

show  $\langle \langle 0, a \rangle \in 1 \times A \rangle$

proof

show  $\langle 0 \in 1 \rangle$  by auto

next

show  $\langle a \in A \rangle$  by (rule hyp1)

qed

qed

next

show  $\langle 1 \subseteq \text{domain}(\{\langle 0, a \rangle\}) \wedge \text{function}(\{\langle 0, a \rangle\}) \rangle$

proof

show  $\langle 1 \subseteq \text{domain}(\{\langle 0, a \rangle\}) \rangle$

proof

fix  $x$

assume  $W: \langle x \in 1 \rangle$

from  $W$  have  $W: \langle x = 0 \rangle$  by (rule le1)

have  $Y: \langle 0 \in \text{domain}(\{\langle 0, a \rangle\}) \rangle$

by auto

from  $W$  and  $Y$

show  $\langle x \in \text{domain}(\{\langle 0, a \rangle\}) \rangle$

by auto

qed

next

show  $\langle \text{function}(\{\langle 0, a \rangle\}) \rangle$

by (rule lsinglfun)

qed

qed

qed

show  $\langle \{\langle 0, a \rangle\} \text{ ' } 0 = a \wedge \text{satpc}(\{\langle 0, a \rangle\}, 0, g) \rangle$

proof

show  $\langle \{\langle 0, a \rangle\} \text{ ' } 0 = a \rangle$

by (rule func.singleton-apply)

next

show  $\langle \text{satpc}(\{\langle 0, a \rangle\}, 0, g) \rangle$

by (rule singlsatpc)

qed

qed

**qed**

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

**proof**

**show**  $\langle \langle 0, a \rangle \in \{ \langle 0, a \rangle \} \rangle$

**by** *auto*

**next**

**show**  $\langle \{ \langle 0, a \rangle \} \in pcs(A, a, g) \rangle$

**proof**(*unfold pcs-def*)

**show**  $\langle \{ \langle 0, a \rangle \} \in \{ t \in Pow(nat \times A) . \exists m \in nat. partcomp(A, t, m, a, g) \} \rangle$

**proof**

**show**  $\langle \{ \langle 0, a \rangle \} \in Pow(nat \times A) \rangle$

**proof**(*rule PowI, rule equalities.singleton-subsetI*)

**show**  $\langle \langle 0, a \rangle \in nat \times A \rangle$

**proof**

**show**  $\langle 0 \in nat \rangle$  **by** *auto*

**next**

**show**  $\langle a \in A \rangle$  **by** (*rule hyp1*)

**qed**

**qed**

**next**

**show**  $\langle \exists m \in nat. partcomp(A, \{ \langle 0, a \rangle \}, m, a, g) \rangle$

**proof**

**show**  $\langle partcomp(A, \{ \langle 0, a \rangle \}, 0, a, g) \rangle$

**by** (*rule zerostep*)

**next**

**show**  $\langle 0 \in nat \rangle$  **by** *auto*

**qed**

**qed**

**qed**

**qed**

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

**proof**

**show**  $\langle \langle 0, a \rangle \in \bigcup pcs(A, a, g) \rangle$

**by** (*rule zainupcs*)

**qed**

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

**lemma** *shortlem* :

**assumes** *m**nat*:  $\langle m \in nat \rangle$

**assumes** *F*:  $\langle partcomp(A, t, m, a, g) \rangle$

**shows**  $\langle partcomp(A, cons(\langle succ(m), g \text{ ‘ } \langle t \text{ ‘ } m, m \rangle \rangle, t), succ(m), a, g) \rangle$

**proof**(*rule partcompE[OF F]*)

**assume** *F1*:  $\langle t \in succ(m) \rightarrow A \rangle$

**assume** *F2*:  $\langle t \text{ ‘ } 0 = a \rangle$

**assume** *F3*:  $\langle satpc(t, m, g) \rangle$

```

show ?thesis
proof
  have  $ljk: \langle \text{cons}(\langle \text{succ}(m), g \text{ ' } \langle t \text{ ' } m, m \rangle \rangle), t \rangle \in (\text{cons}(\text{succ}(m), \text{succ}(m)) \rightarrow A) \rangle$ 
  proof(rule func.fun-extend3[OF F1])
    show  $\langle \text{succ}(m) \notin \text{succ}(m) \rangle$ 
    by (rule upair.mem-not-refl)
    have  $tmA: \langle t \text{ ' } m \in A \rangle$ 
    by (rule func.apply-funtype[OF F1], auto)
    show  $\langle g \text{ ' } \langle t \text{ ' } m, m \rangle \in A \rangle$ 
    by(rule func.apply-funtype[OF hyp2], auto, rule tmA, rule mnat)
  qed
  have  $\langle \text{cons}(\langle \text{succ}(m), g \text{ ' } \langle t \text{ ' } m, m \rangle \rangle), t \rangle \in (\text{cons}(\text{succ}(m), \text{succ}(m)) \rightarrow A) \rangle$ 
  by (rule ljk)
  then have  $\langle \text{cons}(\langle \text{cons}(m, m), g \text{ ' } \langle t \text{ ' } m, m \rangle \rangle), t \rangle \in \text{cons}(\text{cons}(m, m), \text{cons}(m, m)) \rightarrow A \rangle$ 
  by (unfold succ-def)
  then show  $\langle \text{cons}(\langle \text{succ}(m), g \text{ ' } \langle t \text{ ' } m, m \rangle \rangle), t \rangle \in \text{succ}(\text{succ}(m)) \rightarrow A \rangle$ 
  by (unfold succ-def, assumption)
  show  $\langle \text{cons}(\langle \text{succ}(m), g \text{ ' } \langle t \text{ ' } m, m \rangle \rangle), t \text{ ' } 0 = a \rangle$ 
  proof(rule trans, rule func.fun-extend-apply[OF F1])
    show  $\langle \text{succ}(m) \notin \text{succ}(m) \rangle$  by (rule upair.mem-not-refl)
    show  $\langle (\text{if } 0 = \text{succ}(m) \text{ then } g \text{ ' } \langle t \text{ ' } m, m \rangle \text{ else } t \text{ ' } 0) = a \rangle$ 
    by(rule trans, rule upair.if-not-P, auto, rule F2)
  qed
  show  $\langle \text{satpc}(\text{cons}(\langle \text{succ}(m), g \text{ ' } \langle t \text{ ' } m, m \rangle \rangle), t), \text{succ}(m), g \rangle$ 
  proof(unfold satpc-def, rule ballI)
    fix  $n$ 
    assume  $Q: \langle n \in \text{succ}(m) \rangle$ 
    show  $\langle \text{cons}(\langle \text{succ}(m), g \text{ ' } \langle t \text{ ' } m, m \rangle \rangle), t \rangle \text{ ' } \text{succ}(n)$ 
    =  $g \text{ ' } \langle \text{cons}(\langle \text{succ}(m), g \text{ ' } \langle t \text{ ' } m, m \rangle \rangle), t \rangle \text{ ' } n, n \rangle$ 
    proof(rule trans, rule func.fun-extend-apply[OF F1], rule upair.mem-not-refl)
      show  $\langle (\text{if } \text{succ}(n) = \text{succ}(m) \text{ then } g \text{ ' } \langle t \text{ ' } m, m \rangle \text{ else } t \text{ ' } \text{succ}(n)) =$ 
       $g \text{ ' } \langle \text{cons}(\langle \text{succ}(m), g \text{ ' } \langle t \text{ ' } m, m \rangle \rangle), t \rangle \text{ ' } n, n \rangle$ 
      proof(rule upair.succE[OF Q])
        assume  $Y: \langle n = m \rangle$ 
        show  $\langle (\text{if } \text{succ}(n) = \text{succ}(m) \text{ then } g \text{ ' } \langle t \text{ ' } m, m \rangle \text{ else } t \text{ ' } \text{succ}(n)) =$ 
         $g \text{ ' } \langle \text{cons}(\langle \text{succ}(m), g \text{ ' } \langle t \text{ ' } m, m \rangle \rangle), t \rangle \text{ ' } n, n \rangle$ 
        proof(rule trans, rule upair.if-P)
          from  $Y$  show  $\langle \text{succ}(n) = \text{succ}(m) \rangle$  by auto
        next
          have  $L1: \langle t \text{ ' } m = \text{cons}(\langle \text{succ}(m), g \text{ ' } \langle t \text{ ' } m, m \rangle \rangle), t \rangle \text{ ' } n \rangle$ 
          proof(rule sym, rule trans, rule func.fun-extend-apply[OF F1], rule upair.mem-not-refl)
            show  $\langle (\text{if } n = \text{succ}(m) \text{ then } g \text{ ' } \langle t \text{ ' } m, m \rangle \text{ else } t \text{ ' } n) = t \text{ ' } m \rangle$ 
            proof(rule trans, rule upair.if-not-P)
              from  $Y$  show  $\langle t \text{ ' } n = t \text{ ' } m \rangle$  by auto
            show  $\langle n \neq \text{succ}(m) \rangle$ 
            proof(rule not-sym)
              show  $\langle \text{succ}(m) \neq n \rangle$ 

```

```

      by(rule subst, rule sym, rule Y, rule upair.succ-neq-self)
    qed
  qed
  qed
  from Y
  have L2:⟨m = n⟩
    by auto
  have L:⟨t ‘ m, m⟩ = ⟨cons(⟨succ(m), g ‘ ⟨t ‘ m, m⟩⟩, t) ‘ n, n⟩
    by(rule subst-context2[OF L1 L2])
  show ⟨g ‘ ⟨t ‘ m, m⟩ = g ‘ ⟨cons(⟨succ(m), g ‘ ⟨t ‘ m, m⟩⟩, t) ‘ n, n⟩⟩
    by(rule subst-context[OF L])
  qed
next
  assume Y:⟨n ∈ m⟩
  show ⟨(if succ(n) = succ(m) then g ‘ ⟨t ‘ m, m⟩ else t ‘ succ(n)) =
    g ‘ ⟨cons(⟨succ(m), g ‘ ⟨t ‘ m, m⟩⟩, t) ‘ n, n⟩⟩
  proof(rule trans, rule upair.if-not-P)
    show ⟨succ(n) ≠ succ(m)⟩
  by(rule contrapos, rule upair.mem-imp-not-eq, rule Y, rule upair.succ-inject,
  assumption)
  next
  have X:⟨cons(⟨succ(m), g ‘ ⟨t ‘ m, m⟩⟩, t) ‘ n = t ‘ n⟩
  proof(rule trans, rule func.fun-extend-apply[OF F1], rule upair.mem-not-refl)
    show ⟨(if n = succ(m) then g ‘ ⟨t ‘ m, m⟩ else t ‘ n) = t ‘ n⟩
  proof(rule upair.if-not-P)
    show ⟨n ≠ succ(m)⟩
  proof(rule contrapos)
    assume q:n=succ(m)
    from q and Y have M:⟨succ(m) ∈ m⟩
    by auto
    show ⟨m ∈ m⟩
  by(rule Nat.succ-in-naturalD[OF M mnat])
  next
  show ⟨m ∉ m⟩ by (rule upair.mem-not-refl)
  qed
  qed
  qed
  from F3
  have W:⟨∀ n ∈ m. t ‘ succ(n) = g ‘ ⟨t ‘ n, n⟩⟩
    by (unfold satpc-def)
  have U:⟨t ‘ succ(n) = g ‘ ⟨t ‘ n, n⟩⟩
    by (rule bspec[OF W Y])
  show ⟨t ‘ succ(n) = g ‘ ⟨cons(⟨succ(m), g ‘ ⟨t ‘ m, m⟩⟩, t) ‘ n, n⟩⟩
    by (rule trans, rule U, rule sym, rule subst-context[OF X])
  qed
  qed
  qed
  qed
  qed

```



qed

```
lemma l2:⟨nat ⊆ domain(⋃ pcs(A, a, g))⟩
proof
  fix x
  assume G:⟨x ∈ nat⟩
  show ⟨x ∈ domain(⋃ pcs(A, a, g))⟩
  proof(rule nat-induct[of x])
    show ⟨x ∈ nat⟩ by (rule G)
  next
  fix x
  assume Q1:⟨x ∈ nat⟩
  assume Q2:⟨x ∈ domain(⋃ pcs(A, a, g))⟩
  show ⟨succ(x) ∈ domain(⋃ pcs(A, a, g))⟩
  proof(rule domainE[OF Q2])
    fix y
    assume W1:⟨x, y⟩ ∈ (⋃ pcs(A, a, g))
    show ⟨succ(x) ∈ domain(⋃ pcs(A, a, g))⟩
    proof(rule UnionE[OF W1])
      fix t
      assume E1:⟨x, y⟩ ∈ t
      assume E2:⟨t ∈ pcs(A, a, g)⟩
      hence E2:⟨t ∈ {t ∈ Pow(nat * A). ∃ m ∈ nat. partcomp(A, t, m, a, g)}⟩
        by(unfold pcs-def)
      have E21:⟨t ∈ Pow(nat * A)⟩
        by(rule CollectD1[OF E2])
      have E22m:⟨∃ m ∈ nat. partcomp(A, t, m, a, g)⟩
        by(rule CollectD2[OF E2])
      show ⟨succ(x) ∈ domain(⋃ pcs(A, a, g))⟩
      proof(rule bexE[OF E22m])
        fix m
        assume mnat:⟨m ∈ nat⟩
        assume E22P:⟨partcomp(A, t, m, a, g)⟩
        hence E22:⟨((t.succ(m) → A) ∧ (t'0 = a)) ∧ satpc(t, m, g)⟩
          by(unfold partcomp-def, auto)
        hence E223:⟨satpc(t, m, g)⟩ by auto
        hence E223:⟨∀ n ∈ m . t'succ(n) = g ' <t'n, n>⟩
          by(unfold satpc-def, auto)
        from E22 have E221:⟨(t.succ(m) → A)⟩
          by auto
        from E221 have domt:⟨domain(t) = succ(m)⟩
          by (rule func.domain-of-fun)
        from E1 have xind:⟨x ∈ domain(t)⟩
          by (rule equalities.domainI)
        from xind and domt have xinsm:⟨x ∈ succ(m)⟩
          by auto
        show ⟨succ(x) ∈ domain(⋃ pcs(A, a, g))⟩
      proof
```

```

show  $\langle \text{succ}(x), g \text{ ' } \langle t'x, x \rangle \in (\bigcup \text{pcs}(A, a, g)) \rangle$ 
proof

  show  $\langle \text{cons}(\langle \text{succ}(x), g \text{ ' } \langle t'x, x \rangle \rangle, t) \in \text{pcs}(A, a, g) \rangle$ 
  proof(unfold pcs-def, rule CollectI)
    from E21
    have  $L1: \langle t \subseteq \text{nat} \times A \rangle$ 
      by auto
    from Q1 have  $J1: \langle \text{succ}(x) \in \text{nat} \rangle$ 
      by auto
    have  $txA: \langle t \text{ ' } x \in A \rangle$ 
      by (rule func.apply-type[OF E221 xinsm])
    from  $txA$  and Q1 have  $txx: \langle t \text{ ' } x, x \rangle \in A \times \text{nat} \rangle$ 
      by auto
    have  $secp: \langle g \text{ ' } \langle t \text{ ' } x, x \rangle \in A \rangle$ 
      by(rule func.apply-type[OF hyp2 txx])
    from  $J1$  and  $secp$ 
    have  $L2: \langle \langle \text{succ}(x), g \text{ ' } \langle t \text{ ' } x, x \rangle \rangle \in \text{nat} \times A \rangle$ 
      by auto
    show  $\langle \text{cons}(\langle \text{succ}(x), g \text{ ' } \langle t \text{ ' } x, x \rangle \rangle, t) \in \text{Pow}(\text{nat} \times A) \rangle$ 
    proof(rule PowI)
      show  $\langle \text{cons}(\langle \text{succ}(x), g \text{ ' } \langle t \text{ ' } x, x \rangle \rangle, t) \subseteq \text{nat} \times A \rangle$ 
      proof
        show  $\langle \langle \text{succ}(x), g \text{ ' } \langle t \text{ ' } x, x \rangle \rangle \in \text{nat} \times A \wedge t \subseteq \text{nat} \times A \rangle$ 
        by (rule conjI[OF L2 L1])
      qed
    qed
  next
  show  $\langle \exists m \in \text{nat}. \text{partcomp}(A, \text{cons}(\langle \text{succ}(x), g \text{ ' } \langle t \text{ ' } x, x \rangle \rangle, t), m, a,$ 
 $g) \rangle$ 
    proof(rule succE[OF xinsm])
      assume  $xeqm: \langle x = m \rangle$ 
      show  $\langle \exists m \in \text{nat}. \text{partcomp}(A, \text{cons}(\langle \text{succ}(x), g \text{ ' } \langle t \text{ ' } x, x \rangle \rangle, t), m,$ 
 $a, g) \rangle$ 
        proof
          show  $\langle \text{partcomp}(A, \text{cons}(\langle \text{succ}(x), g \text{ ' } \langle t \text{ ' } x, x \rangle \rangle, t), \text{succ}(x), a, g) \rangle$ 
          proof(rule shortlem[OF Q1])
            show  $\langle \text{partcomp}(A, t, x, a, g) \rangle$ 
            proof(rule subst[of m x], rule sym, rule xeqm)
              show  $\langle \text{partcomp}(A, t, m, a, g) \rangle$ 
              by (rule E22P)
            qed
          qed
        next
        from Q1 show  $\langle \text{succ}(x) \in \text{nat} \rangle$  by auto
        qed
      next
      assume  $xinm: \langle x \in m \rangle$ 
      have  $lmm: \langle \text{cons}(\langle \text{succ}(x), g \text{ ' } \langle t \text{ ' } x, x \rangle \rangle, t) = t \rangle$ 

```

```

    by (rule addmiddle[OF mnat E22P xinm])
    show  $\langle \exists m \in \text{nat}. \text{partcomp}(A, \text{cons}(\langle \text{succ}(x), g \text{ ' } \langle t \text{ ' } x, x \rangle \rangle, t), m, a, g) \rangle$ 
  next
    by (rule subst[of t], rule sym, rule lmm, rule E22m)
  qed
  qed
  next
  show  $\langle \langle \text{succ}(x), g \text{ ' } \langle t \text{ ' } x, x \rangle \rangle \in \text{cons}(\langle \text{succ}(x), g \text{ ' } \langle t \text{ ' } x, x \rangle \rangle, t) \rangle$ 
  by auto
  qed
  qed
  qed
  next
  show  $\langle 0 \in \text{domain}(\bigcup \text{pcs}(A, a, g)) \rangle$ 
  by (rule l2')
  qed
  qed

lemma useful :  $\langle \forall m \in \text{nat}. \exists t. \text{partcomp}(A, t, m, a, g) \rangle$ 
proof (rule nat-induct-bound)
  show  $\langle \exists t. \text{partcomp}(A, t, 0, a, g) \rangle$ 
  proof
    show  $\langle \text{partcomp}(A, \{\langle 0, a \rangle\}, 0, a, g) \rangle$ 
    by (rule zerostep)
  qed
  next
  fix m
  assume mnat :  $\langle m \in \text{nat} \rangle$ 
  assume G :  $\langle \exists t. \text{partcomp}(A, t, m, a, g) \rangle$ 
  show  $\langle \exists t. \text{partcomp}(A, t, \text{succ}(m), a, g) \rangle$ 
  proof (rule exE[OF G])
    fix t
    assume G :  $\langle \text{partcomp}(A, t, m, a, g) \rangle$ 
    show  $\langle \exists t. \text{partcomp}(A, t, \text{succ}(m), a, g) \rangle$ 
    proof
      show  $\langle \text{partcomp}(A, \text{cons}(\langle \text{succ}(m), g \text{ ' } \langle t \text{ ' } m, m \rangle \rangle, t), \text{succ}(m), a, g) \rangle$ 
      by (rule shortlem[OF mnat G])
    qed
  qed
  qed
  qed

lemma l4 :  $\langle (\bigcup \text{pcs}(A, a, g)) \in \text{nat} \rightarrow A \rangle$ 
proof (unfold Pi-def)
  show  $\langle \bigcup \text{pcs}(A, a, g) \in \{f \in \text{Pow}(\text{nat} \times A) . \text{nat} \subseteq \text{domain}(f) \wedge \text{function}(f)\} \rangle$ 
  proof
    show  $\langle \bigcup \text{pcs}(A, a, g) \in \text{Pow}(\text{nat} \times A) \rangle$ 
    proof

```

```

    show  $\langle \bigcup pcs(A, a, g) \subseteq nat \times A \rangle$ 
      by (rule l1)
  qed
next
  show  $\langle nat \subseteq domain(\bigcup pcs(A, a, g)) \wedge function(\bigcup pcs(A, a, g)) \rangle$ 
  proof
    show  $\langle nat \subseteq domain(\bigcup pcs(A, a, g)) \rangle$ 
      by (rule l2)
    next
      show  $\langle function(\bigcup pcs(A, a, g)) \rangle$ 
        by (rule l3)
    qed
  qed
qed

```

```

lemma l5:  $\langle (\bigcup pcs(A, a, g)) \text{ ' } 0 = a \rangle$ 
proof (rule func.function-apply-equality)
  show  $\langle function(\bigcup pcs(A, a, g)) \rangle$ 
    by (rule l3)
  next
    show  $\langle \langle 0, a \rangle \in \bigcup pcs(A, a, g) \rangle$ 
      by (rule zainupcs)
  qed

```

```

lemma ballE2:
  assumes  $\langle \forall x \in AA. P(x) \rangle$ 
  assumes  $\langle x \in AA \rangle$ 
  assumes  $\langle P(x) ==> Q \rangle$ 
  shows  $Q$ 
  by (rule assms(3), rule bspec, rule assms(1), rule assms(2))

```

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

```

lemma l6new:  $\langle satpc(\bigcup pcs(A, a, g), nat, g) \rangle$ 
proof (unfold satpc-def, rule ballI)
  fix n
  assume nnat:  $\langle n \in nat \rangle$ 
  hence snnat:  $\langle succ(n) \in nat \rangle$  by auto

  show  $\langle (\bigcup pcs(A, a, g)) \text{ ' } succ(n) = g \text{ ' } \langle (\bigcup pcs(A, a, g)) \text{ ' } n, n \rangle \rangle$ 
  proof (rule ballE2[OF useful snnat], erule exE)
    fix t
    assume Y:  $\langle \text{ part-comp}(A, t, succ(n), a, g) \rangle$ 
    show  $\langle (\bigcup pcs(A, a, g)) \text{ ' } succ(n) = g \text{ ' } \langle (\bigcup pcs(A, a, g)) \text{ ' } n, n \rangle \rangle$ 
    proof (rule partcompE[OF Y])
      assume Y1:  $\langle t \in succ(succ(n)) \rightarrow A \rangle$ 
      assume Y2:  $\langle t \text{ ' } 0 = a \rangle$ 
      assume Y3:  $\langle satpc(t, succ(n), g) \rangle$ 

```

```

hence  $Y3: \langle \forall x \in \text{succ}(n) . t' \text{succ}(x) = g' \langle t'x, x \rangle \rangle$ 
  by (unfold satpc-def)
hence  $Y3: \langle t' \text{succ}(n) = g' \langle t'n, n \rangle \rangle$ 
  by (rule bspec, auto)
have  $e1: \langle (\bigcup \text{pcs}(A, a, g))' \text{succ}(n) = t' \text{succ}(n) \rangle$ 
proof(rule valofunion, rule pcs-lem, rule hyp1)
  show  $\langle t \in \text{pcs}(A, a, g) \rangle$ 
  proof(unfold pcs-def, rule CollectI)
    show  $\langle t \in \text{Pow}(\text{nat} \times A) \rangle$ 
    proof(rule tgb)
      show  $\langle t \in \text{succ}(\text{succ}(n)) \rightarrow A \rangle$  by (rule Y1)
    next
      from snnat
      show  $\langle \text{succ}(\text{succ}(n)) \in \text{nat} \rangle$  by auto
    qed
  next
    show  $\langle \exists m \in \text{nat} . \text{partcomp}(A, t, m, a, g) \rangle$ 
    by(rule beXI, rule Y, rule snnat)
  qed
next
  show  $\langle t \in \text{succ}(\text{succ}(n)) \rightarrow A \rangle$  by (rule Y1)
next
  show  $\langle \text{succ}(n) \in \text{succ}(\text{succ}(n)) \rangle$  by auto
next
  show  $\langle t' \text{succ}(n) = t' \text{succ}(n) \rangle$  by (rule refl)
qed
have  $e2: \langle (\bigcup \text{pcs}(A, a, g))' n = t' n \rangle$ 
proof(rule valofunion, rule pcs-lem, rule hyp1)
  show  $\langle t \in \text{pcs}(A, a, g) \rangle$ 
  proof(unfold pcs-def, rule CollectI)
    show  $\langle t \in \text{Pow}(\text{nat} \times A) \rangle$ 
    proof(rule tgb)
      show  $\langle t \in \text{succ}(\text{succ}(n)) \rightarrow A \rangle$  by (rule Y1)
    next
      from snnat
      show  $\langle \text{succ}(\text{succ}(n)) \in \text{nat} \rangle$  by auto
    qed
  next
    show  $\langle \exists m \in \text{nat} . \text{partcomp}(A, t, m, a, g) \rangle$ 
    by(rule beXI, rule Y, rule snnat)
  qed
next
  show  $\langle t \in \text{succ}(\text{succ}(n)) \rightarrow A \rangle$  by (rule Y1)
next
  show  $\langle n \in \text{succ}(\text{succ}(n)) \rangle$  by auto
next
  show  $\langle t' n = t' n \rangle$  by (rule refl)
qed
have  $e3: \langle g' \langle (\bigcup \text{pcs}(A, a, g))' n, n \rangle = g' \langle t' n, n \rangle \rangle$ 

```

```

    by (rule subst[OF e2], rule refl)
  show ⟨(⋃ pcs(A, a, g)) ‘ succ(n) = g ‘ (⋃ pcs(A, a, g)) ‘ n, n⟩
    by (rule trans, rule e1, rule trans, rule Y3, rule sym, rule e3)
  qed
qed
qed

```

## 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**

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

**proof**

**show**  $\langle (\bigcup \text{pcs}(A, a, g)) \in \text{nat} \rightarrow A \wedge (\bigcup \text{pcs}(A, a, g)) ' 0 = a \wedge \text{satpc}(\bigcup \text{pcs}(A, a, g), \text{nat}, g) \rangle$

**proof**

**show**  $\langle \bigcup \text{pcs}(A, a, g) \in \text{nat} \rightarrow A \rangle$

**by** (rule l4)

**next**

**show**  $\langle (\bigcup \text{pcs}(A, a, g)) ' 0 = a \wedge \text{satpc}(\bigcup \text{pcs}(A, a, g), \text{nat}, g) \rangle$

**proof**

**show**  $\langle (\bigcup \text{pcs}(A, a, g)) ' 0 = a \rangle$

**by** (rule l5)

**next**

**show**  $\langle \text{satpc}(\bigcup \text{pcs}(A, a, g), \text{nat}, g) \rangle$

**by** (rule l6new)

**qed**

**qed**

**qed**

**next**

**show**  $\langle \bigwedge f y. f \in \text{nat} \rightarrow A \wedge$

$f'0 = a \wedge$

$\text{satpc}(f, \text{nat}, g) \implies$

$y \in \text{nat} \rightarrow A \wedge$

$y'0 = a \wedge$

$\text{satpc}(y, \text{nat}, g) \implies$

$f = y \rangle$

**by** (rule recuniq)

**qed**

**end**

## 8 Lemmas for addition

Let's define function  $t(x) = (a+x)$ . Firstly we need to define a function  $g: \text{nat} \times \text{nat} \rightarrow \text{nat}$ , such that  $g'(t'n, n) = t'succ(n) = a + (n + 1) = (a$

$+ n) + 1 = (t'n) + 1$  So  $g\langle a, b \rangle = a + 1$  and  $g(p) = \text{succ}(\text{pr1}(p))$  and  $\text{satpc}(t, \alpha, g) \iff \forall n \in \alpha . t'\text{succ}(n) = \text{succ}(t'n)$ .

**definition**  $\text{addg} :: \langle i \rangle$

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

**lemma**  $\text{addgfun} : \langle \text{function}(\text{addg}) \rangle$

by  $(\text{unfold } \text{addg-def}, \text{rule } \text{func.function-lam})$

**lemma**  $\text{addgsubpow} : \langle \text{addg} \in \text{Pow}((\text{nat} \times \text{nat}) \times \text{nat}) \rangle$

**proof**  $(\text{unfold } \text{addg-def}, \text{rule } \text{subsetD})$

show  $\langle (\lambda x \in \text{nat} \times \text{nat}. \text{succ}(\text{fst}(x))) \in \text{nat} \times \text{nat} \rightarrow \text{nat} \rangle$

**proof**  $(\text{rule } \text{func.lam-type})$

fix  $x$

assume  $\langle x \in \text{nat} \times \text{nat} \rangle$

hence  $\langle \text{fst}(x) \in \text{nat} \rangle$  by *auto*

thus  $\langle \text{succ}(\text{fst}(x)) \in \text{nat} \rangle$  by *auto*

**qed**

**next**

show  $\langle \text{nat} \times \text{nat} \rightarrow \text{nat} \subseteq \text{Pow}((\text{nat} \times \text{nat}) \times \text{nat}) \rangle$

by  $(\text{rule } \text{pisubsig})$

**qed**

**lemma**  $\text{addgdom} : \langle \text{nat} \times \text{nat} \subseteq \text{domain}(\text{addg}) \rangle$

**proof**  $(\text{unfold } \text{addg-def})$

have  $e : \langle \text{domain}(\lambda x \in \text{nat} \times \text{nat}. \text{succ}(\text{fst}(x))) = \text{nat} \times \text{nat} \rangle$

by  $(\text{rule } \text{domain-lam})$

show  $\langle \text{nat} \times \text{nat} \subseteq \text{domain}(\lambda x \in \text{nat} \times \text{nat}. \text{succ}(\text{fst}(x))) \rangle$

by  $(\text{rule } \text{subst}, \text{rule } \text{sym}, \text{rule } e, \text{auto})$

**qed**

**lemma**  $\text{plussucc}$ :

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

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

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

**proof**

fix  $n$

assume  $J : \langle n \in \text{nat} \rangle$

from  $H$

have  $H : \langle \forall n \in \text{nat} . f'\text{succ}(n) = (\lambda x \in (\text{nat} * \text{nat}). \text{succ}(\text{fst}(x)))' \langle f'n, n \rangle \rangle$

by  $(\text{unfold } \text{satpc-def}, \text{unfold } \text{addg-def})$

have  $H : \langle f'\text{succ}(n) = (\lambda x \in (\text{nat} * \text{nat}). \text{succ}(\text{fst}(x)))' \langle f'n, n \rangle \rangle$

by  $(\text{rule } \text{bspec}[OF H J])$

have  $Q : \langle (\lambda x \in (\text{nat} * \text{nat}). \text{succ}(\text{fst}(x)))' \langle f'n, n \rangle = \text{succ}(\text{fst}(\langle f'n, n \rangle)) \rangle$

**proof**  $(\text{rule } \text{func.beta})$

show  $\langle \langle f' n, n \rangle \in \text{nat} \times \text{nat} \rangle$

**proof**

show  $\langle f' n \in \text{nat} \rangle$

by  $(\text{rule } \text{func.apply-funtype}[OF F J])$

```

    show ⟨n ∈ nat⟩
      by (rule J)
  qed
  have HQ: ⟨f'succ(n) = succ(fst(⟨f'n, n⟩))⟩
    by (rule trans[OF H Q])
  have K: ⟨fst(⟨f'n, n⟩) = f'n⟩
    by auto
  hence K': ⟨succ(fst(⟨f'n, n⟩)) = succ(f'n)⟩
    by (rule subst-context)
  show ⟨f'succ(n) = succ(f'n)⟩
    by (rule trans[OF HQ K])
  qed

```

## 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 ⟨a ∈ nat⟩
  shows
    ⟨∃!f. ((f ∈ (nat → nat)) ∧ ((f'0) = a) ∧ satpc(f, nat, addg))⟩
proof(rule recthm.recursionthm, unfold recthm-def)
  show ⟨a ∈ nat ∧ addg ∈ nat × nat → nat⟩
  proof
    show ⟨a ∈ nat⟩ by (rule assms(1))
  next
    show ⟨addg ∈ nat × nat → nat⟩
  proof(unfold Pi-def, rule CollectI)
    show ⟨addg ∈ Pow((nat × nat) × nat)⟩
      by (rule addgsubpow)
  next
    have A2: ⟨nat × nat ⊆ domain(addg)⟩
      by(rule addgdom)
    have A3: ⟨function(addg)⟩
      by (rule addgfun)
    show ⟨nat × nat ⊆ domain(addg) ∧ function(addg)⟩
      by(rule conjI[OF A2 A3])
  qed
qed
qed
end

```