

From Abstract to Concrete Gödel's Incompleteness  
Theorems—Part II

Andrei Popescu      Dmitriy Traytel

February 6, 2026

## Abstract

We validate an abstract formulation of Gödel's Second Incompleteness Theorem from a [separate AFP entry](#) by instantiating it to the case of *finite consistent extensions of the Hereditarily Finite (HF) Set theory*, i.e., consistent FOL theories extending the HF Set theory with a finite set of axioms.

The instantiation draws heavily on infrastructure previously developed by Larry Paulson in his [direct formalisation of the concrete result](#). It strengthens Paulson's formalization of Gödel's Second from that entry by *not* assuming soundness, and in fact not relying on any notion of model or semantic interpretation. The strengthening was obtained by first replacing some of Paulson's semantic arguments with proofs within his HF calculus, and then plugging in some of Paulson's (modified) lemmas to instantiate our soundness-free Gödel's Second locale.

# Contents

<b>1</b>	<b>Syntax of Terms and Formulas using Nominal Logic</b>	<b>2</b>
1.1	Terms and Formulas . . . . .	2
1.1.1	Hf is a pure permutation type . . . . .	2
1.1.2	The datatypes . . . . .	2
1.1.3	Substitution . . . . .	3
1.1.4	Derived syntax . . . . .	4
1.1.5	Derived logical connectives . . . . .	5
1.2	Axioms and Theorems . . . . .	5
1.2.1	Logical axioms . . . . .	5
1.2.2	Concrete variables . . . . .	6
1.2.3	The HF axioms . . . . .	6
1.2.4	Equality axioms . . . . .	6
1.2.5	The proof system . . . . .	7
1.2.6	Derived rules of inference . . . . .	7
1.2.7	The Deduction Theorem . . . . .	10
1.2.8	Cut rules . . . . .	10
1.3	Miscellaneous logical rules . . . . .	10
1.3.1	Quantifier reasoning . . . . .	13
1.3.2	Congruence rules . . . . .	14
1.4	Equality reasoning . . . . .	15
1.4.1	The congruence property for ( <i>EQ</i> ), and other basic properties of equality . . . . .	15
1.4.2	The congruence property for ( <i>IN</i> ) . . . . .	15
1.4.3	The congruence properties for <i>Eats</i> and <i>HPair</i> . . . . .	15
1.4.4	Substitution for Equalities . . . . .	15
1.4.5	Congruence Rules for Predicates . . . . .	16
1.5	Zero and Falsity . . . . .	16
1.5.1	The Formula <i>FIs</i> . . . . .	17
1.5.2	More properties of <i>Zero</i> . . . . .	17
1.5.3	Basic properties of <i>Eats</i> . . . . .	18
1.6	Bounded Quantification involving <i>Eats</i> . . . . .	19
1.7	Induction . . . . .	20
<b>2</b>	<b>De Bruijn Syntax, Quotations, Codes, V-Codes</b>	<b>21</b>
2.1	de Bruijn Indices (locally-nameless version) . . . . .	21
2.2	Characterising the Well-Formed de Bruijn Formulas . . . . .	23
2.2.1	Well-Formed Terms . . . . .	23
2.2.2	Well-Formed Formulas . . . . .	24
2.3	Well formed terms and formulas (de Bruijn representation) . . . . .	24
2.4	Quotations . . . . .	26
2.4.1	Quotations of de Bruijn terms . . . . .	26
2.4.2	Quotations of de Bruijn formulas . . . . .	26
2.5	Definitions Involving Coding . . . . .	28

2.5.1	The set $\Gamma$ of Definition 1.1, constant terms used for coding	29
2.6	V-Coding for terms and formulas, for the Second Theorem	29
<b>3</b>	<b>Basic Predicates</b>	<b>31</b>
3.1	The Subset Relation	31
3.2	Extensionality	32
3.3	The Disjointness Relation	33
3.4	The Foundation Theorem	34
3.5	The Ordinal Property	34
3.6	Induction on Ordinals	36
3.7	Linearity of Ordinals	36
3.8	The predicate <i>OrdNotEqP</i>	37
3.9	Predecessor of an Ordinal	37
3.10	Case Analysis and Zero/SUCC Induction	38
3.11	The predicate <i>HFun_Sigma</i>	38
3.12	The predicate <i>HDomain_Incl</i>	39
3.13	<i>HPair</i> is Provably Injective	40
3.14	<i>SUCC</i> is Provably Injective	41
3.15	The predicate <i>LstSeqP</i>	41
<b>4</b>	<b>Sigma-Formulas and Theorem 2.5</b>	<b>43</b>
4.1	Ground Terms and Formulas	43
4.2	Sigma Formulas	44
4.2.1	Strict Sigma Formulas	44
4.2.2	Closure properties for Sigma-formulas	44
4.3	Lemma 2.2: Atomic formulas are Sigma-formulas	44
4.4	Universal Quantification Bounded by an Arbitrary Term	46
4.5	Lemma 2.3: Sequence-related concepts are Sigma-formulas	46
<b>5</b>	<b>Predicates for Terms, Formulas and Substitution</b>	<b>47</b>
5.1	Predicates for atomic terms	47
5.1.1	Free Variables	47
5.1.2	De Bruijn Indexes	47
5.1.3	Various syntactic lemmas	48
5.2	The predicate <i>SeqCTermP</i> , for Terms and Constants	48
5.3	The predicates <i>TermP</i> and <i>ConstP</i>	49
5.3.1	Definition	49
5.3.2	Correctness properties for constants	49
5.4	Abstraction over terms	49
5.4.1	Defining the syntax: main predicate	50
5.5	Substitution over terms	51
5.5.1	Defining the syntax	51
5.6	Abstraction over formulas	51
5.6.1	The predicate <i>AbstAtomicP</i>	51
5.6.2	The predicate <i>AbsMakeForm</i>	52
5.6.3	Defining the syntax: the main <i>AbstForm</i> predicate	53
5.7	Substitution over formulas	53
5.7.1	The predicate <i>SubstAtomicP</i>	53
5.7.2	The predicate <i>SubstMakeForm</i>	54
5.7.3	Defining the syntax: the main <i>SubstForm</i> predicate	54
5.8	The predicate <i>AtomicP</i>	55
5.9	The predicate <i>MakeForm</i>	55
5.10	The predicate <i>SeqFormP</i>	56
5.11	The predicate <i>FormP</i>	56

5.11.1	Definition . . . . .	56
5.11.2	The predicate <i>VarNonOccFormP</i> (Derived from <i>SubstFormP</i> ) . . . . .	57
<b>6</b>	<b>Formalizing Provability</b>	<b>58</b>
6.1	Section 4 Predicates (Leading up to Pf) . . . . .	58
6.1.1	The predicate <i>SentP</i> , for the Sentential (Boolean) Axioms . . . . .	58
6.1.2	The predicate <i>Equality_axP</i> , for the Equality Axioms . . . . .	58
6.1.3	The predicate <i>HF_axP</i> , for the HF Axioms . . . . .	58
6.1.4	The specialisation axioms . . . . .	59
6.1.5	The induction axioms . . . . .	59
6.1.6	The predicate <i>AxiomP</i> , for any Axioms . . . . .	60
6.1.7	The predicate <i>ModPonP</i> , for the inference rule Modus Ponens . . . . .	60
6.1.8	The predicate <i>ExistsP</i> , for the existential rule . . . . .	60
6.1.9	The predicate <i>SubstP</i> , for the substitution rule . . . . .	61
6.1.10	The predicate <i>PrfP</i> . . . . .	61
6.1.11	The predicate <i>PfP</i> . . . . .	62
<b>7</b>	<b>Syntactic Preliminaries for the Second Incompleteness Theorem</b>	<b>63</b>
7.1	NotInDom . . . . .	63
7.2	Restriction of a Sequence to a Domain . . . . .	64
7.3	Applications to LstSeqP . . . . .	65
7.4	Ordinal Addition . . . . .	65
7.4.1	Predicate form, defined on sequences . . . . .	65
7.4.2	Proving that these relations are functions . . . . .	66
7.5	A Shifted Sequence . . . . .	67
7.6	Union of Two Sets . . . . .	68
7.7	Append on Sequences . . . . .	69
7.8	LstSeqP and SeqAppendP . . . . .	70
7.9	Substitution and Abstraction on Terms . . . . .	70
7.9.1	Atomic cases . . . . .	70
7.9.2	Non-atomic cases . . . . .	71
7.9.3	Substitution over a constant . . . . .	71
7.10	Substitution on Formulas . . . . .	71
7.10.1	Membership . . . . .	71
7.10.2	Equality . . . . .	72
7.10.3	Negation . . . . .	72
7.10.4	Disjunction . . . . .	72
7.10.5	Existential . . . . .	72
7.11	Constant Terms . . . . .	72
7.12	Proofs . . . . .	73
7.13	Formulas . . . . .	73
7.14	Abstraction on Formulas . . . . .	73
7.14.1	Membership . . . . .	73
7.14.2	Equality . . . . .	74
7.14.3	Negation . . . . .	74
7.14.4	Disjunction . . . . .	74
7.14.5	Existential . . . . .	74
7.14.6	MakeForm . . . . .	76
7.14.7	Negation . . . . .	76
7.14.8	Disjunction . . . . .	76
7.14.9	Existential . . . . .	76

<b>8 Pseudo-Coding: Section 7 Material</b>	<b>78</b>
8.1 General Lemmas . . . . .	78
8.2 Simultaneous Substitution . . . . .	79
8.3 The Main Theorems of Section 7 . . . . .	81
<b>9 Quotations of the Free Variables</b>	<b>83</b>
9.1 Sequence version of the “Special p-Function, $F^*$ ” . . . . .	83
9.1.1 Defining the syntax: quantified body . . . . .	83
9.1.2 Correctness properties . . . . .	84
9.2 The “special function” itself . . . . .	84
9.2.1 Correctness properties . . . . .	84
9.3 The Operator <i>quote_all</i> . . . . .	85
9.3.1 Definition and basic properties . . . . .	85
9.3.2 Transferring theorems to the level of derivability . . . . .	85
9.4 Star Property. Equality and Membership: Lemmas 9.3 and 9.4 . . . . .	87
9.5 Star Property. Universal Quantifier: Lemma 9.7 . . . . .	87
9.6 The Derivability Condition, Theorem 9.1 . . . . .	87
<b>10 Uniqueness Results: Syntactic Relations are Functions</b>	<b>89</b>
10.0.1 <i>SeqStTermP</i> . . . . .	89
10.0.2 <i>SubstAtomicP</i> . . . . .	90
10.0.3 <i>SeqSubstFormP</i> . . . . .	90
10.0.4 <i>SubstFormP</i> . . . . .	90
<b>11 Section 6 Material and Gödel’s First Incompleteness Theorem</b>	<b>91</b>
11.1 The Function W and Lemma 6.1 . . . . .	91
11.1.1 Predicate form, defined on sequences . . . . .	91
11.1.2 Predicate form of W . . . . .	92
11.1.3 Proving that these relations are functions . . . . .	92
11.2 The Function HF and Lemma 6.2 . . . . .	93
11.2.1 Defining the syntax: quantified body . . . . .	93
11.2.2 Defining the syntax: main predicate . . . . .	94
11.2.3 Proving that these relations are functions . . . . .	94
11.3 The Function K and Lemma 6.3 . . . . .	95
<b>12 The Instantiation</b>	<b>96</b>

# Chapter 1

## Syntax of Terms and Formulas using Nominal Logic

```
theory SyntaxN
imports Nominal2.Nominal2 HereditarilyFinite.OrdArith
begin
```

### 1.1 Terms and Formulas

#### 1.1.1 Hf is a pure permutation type

```
instantiation hf :: pt
```

```
begin
```

```
  definition p · (s::hf) = s
```

```
  instance
```

```
    ⟨proof⟩
```

```
end
```

```
instance hf :: pure
```

```
  ⟨proof⟩
```

```
atom_decl name
```

```
declare fresh_set_empty [simp]
```

```
lemma supp_name [simp]: fixes i::name shows supp i = {atom i}
```

```
  ⟨proof⟩
```

#### 1.1.2 The datatypes

```
nominal_datatype tm = Zero | Var name | Eats tm tm
```

```
nominal_datatype fm =
```

```
  Mem tm tm (infixr ⟨IN⟩ 150)
```

```
  | Eq tm tm (infixr ⟨EQ⟩ 150)
```

```
  | Disj fm fm (infixr ⟨OR⟩ 130)
```

```
  | Neg fm
```

```
  | Ex x::name f::fm binds x in f
```

Mem, Eq are atomic formulas; Disj, Neg, Ex are non-atomic

```
declare tm.supp [simp] fm.supp [simp]
```

### 1.1.3 Substitution

**nominal\_function** *subst* :: *name*  $\Rightarrow$  *tm*  $\Rightarrow$  *tm*  $\Rightarrow$  *tm*

**where**

*subst* *i* *x* *Zero* = *Zero*  
| *subst* *i* *x* (*Var* *k*) = (if *i=k* then *x* else *Var* *k*)  
| *subst* *i* *x* (*Eats* *t* *u*) = *Eats* (*subst* *i* *x* *t*) (*subst* *i* *x* *u*)  
⟨*proof*⟩

**nominal\_termination** (*eqvt*)

⟨*proof*⟩

**lemma** *fresh\_subst\_if* [*simp*]:

$j \# \text{subst } i \ x \ t \iff (\text{atom } i \# t \wedge j \# t) \vee (j \# x \wedge (j \# t \vee j = \text{atom } i))$   
⟨*proof*⟩

**lemma** *forget\_subst\_tm* [*simp*]: *atom* *a*  $\#$  *tm*  $\implies$  *subst* *a* *x* *tm* = *tm*

⟨*proof*⟩

**lemma** *subst\_tm\_id* [*simp*]: *subst* *a* (*Var* *a*) *tm* = *tm*

⟨*proof*⟩

**lemma** *subst\_tm\_commute* [*simp*]:

*atom* *j*  $\#$  *tm*  $\implies$  *subst* *j* *u* (*subst* *i* *t* *tm*) = *subst* *i* (*subst* *j* *u* *t*) *tm*  
⟨*proof*⟩

**lemma** *subst\_tm\_commute2* [*simp*]:

*atom* *j*  $\#$  *t*  $\implies$  *atom* *i*  $\#$  *u*  $\implies$   $i \neq j \implies$  *subst* *j* *u* (*subst* *i* *t* *tm*) = *subst* *i* *t* (*subst* *j* *u* *tm*)  
⟨*proof*⟩

**lemma** *repeat\_subst\_tm* [*simp*]: *subst* *i* *u* (*subst* *i* *t* *tm*) = *subst* *i* (*subst* *i* *u* *t*) *tm*

⟨*proof*⟩

**nominal\_function** *subst\_fm* :: *fm*  $\Rightarrow$  *name*  $\Rightarrow$  *tm*  $\Rightarrow$  *fm* (⟨'\_(\_::=\_)⟩ [1000, 0, 0] 200)

**where**

*Mem*: (*Mem* *t* *u*)(*i*::=*x*) = *Mem* (*subst* *i* *x* *t*) (*subst* *i* *x* *u*)  
| *Eq*: (*Eq* *t* *u*)(*i*::=*x*) = *Eq* (*subst* *i* *x* *t*) (*subst* *i* *x* *u*)  
| *Disj*: (*Disj* *A* *B*)(*i*::=*x*) = *Disj* (*A*(*i*::=*x*)) (*B*(*i*::=*x*))  
| *Neg*: (*Neg* *A*)(*i*::=*x*) = *Neg* (*A*(*i*::=*x*))  
| *Ex*: *atom* *j*  $\#$  (*i*, *x*)  $\implies$  (*Ex* *j* *A*)(*i*::=*x*) = *Ex* *j* (*A*(*i*::=*x*))

⟨*proof*⟩

**nominal\_termination** (*eqvt*)

⟨*proof*⟩

**lemma** *size\_subst\_fm* [*simp*]: *size* (*A*(*i*::=*x*)) = *size* *A*

⟨*proof*⟩

**lemma** *forget\_subst\_fm* [*simp*]: *atom* *a*  $\#$  *A*  $\implies$  *A*(*a*::=*x*) = *A*

⟨*proof*⟩

**lemma** *subst\_fm\_id* [*simp*]: *A*(*a*::=*Var* *a*) = *A*

⟨*proof*⟩

**lemma** *fresh\_subst\_fm\_if* [*simp*]:

$j \# (A(i::=x)) \iff (\text{atom } i \# A \wedge j \# A) \vee (j \# x \wedge (j \# A \vee j = \text{atom } i))$   
⟨*proof*⟩

**lemma** *subst\_fm\_commute* [*simp*]:

$atom\ j \# A \implies (A(i::=t))(j::=u) = A(i ::= subst\ j\ u\ t)$   
 ⟨proof⟩

**lemma** *repeat\_subst\_fm [simp]*:  $(A(i::=t))(i::=u) = A(i ::= subst\ i\ u\ t)$   
 ⟨proof⟩

**lemma** *subst\_fm\_Ex\_with\_renaming*:  
 $atom\ i' \# (A, i, j, t) \implies (Ex\ i\ A)(j ::= t) = Ex\ i' (((i \leftrightarrow i') \cdot A)(j ::= t))$   
 ⟨proof⟩

the simplifier cannot apply the rule above, because it introduces a new variable at the right hand side.

⟨ML⟩

### 1.1.4 Derived syntax

#### Ordered pairs

**definition** *HPair* ::  $tm \Rightarrow tm \Rightarrow tm$   
**where**  $HPair\ a\ b = Eats\ (Eats\ Zero\ (Eats\ (Eats\ Zero\ b)\ a))\ (Eats\ (Eats\ Zero\ a)\ a)$

**lemma** *HPair\_eqvt [eqvt]*:  $(p \cdot HPair\ a\ b) = HPair\ (p \cdot a)\ (p \cdot b)$   
 ⟨proof⟩

**lemma** *fresh\_HPpair [simp]*:  $x \# HPair\ a\ b \longleftrightarrow (x \# a \wedge x \# b)$   
 ⟨proof⟩

**lemma** *HPair\_injective\_iff [iff]*:  $HPair\ a\ b = HPair\ a'\ b' \longleftrightarrow (a = a' \wedge b = b')$   
 ⟨proof⟩

**lemma** *subst\_tm\_HPpair [simp]*:  $subst\ i\ x\ (HPair\ a\ b) = HPair\ (subst\ i\ x\ a)\ (subst\ i\ x\ b)$   
 ⟨proof⟩

#### Ordinals

**definition**  
 $SUCC :: tm \Rightarrow tm$  **where**  
 $SUCC\ x \equiv Eats\ x\ x$

**fun** *ORD\_OF* ::  $nat \Rightarrow tm$   
**where**  
 $ORD\_OF\ 0 = Zero$   
 $| ORD\_OF\ (Suc\ k) = SUCC\ (ORD\_OF\ k)$

**lemma** *SUCC\_fresh\_iff [simp]*:  $a \# SUCC\ t \longleftrightarrow a \# t$   
 ⟨proof⟩

**lemma** *SUCC\_eqvt [eqvt]*:  $(p \cdot SUCC\ a) = SUCC\ (p \cdot a)$   
 ⟨proof⟩

**lemma** *SUCC\_subst [simp]*:  $subst\ i\ t\ (SUCC\ k) = SUCC\ (subst\ i\ t\ k)$   
 ⟨proof⟩

**lemma** *ORD\_OF\_fresh [simp]*:  $a \# ORD\_OF\ n$   
 ⟨proof⟩

**lemma** *ORD\_OF\_eqvt [eqvt]*:  $(p \cdot ORD\_OF\ n) = ORD\_OF\ (p \cdot n)$   
 ⟨proof⟩

## 1.1.5 Derived logical connectives

**abbreviation**  $Imp :: fm \Rightarrow fm \Rightarrow fm$  (**infixr**  $\langle IMP \rangle$  125)  
**where**  $Imp\ A\ B \equiv Disj\ (Neg\ A)\ B$

**abbreviation**  $All :: name \Rightarrow fm \Rightarrow fm$   
**where**  $All\ i\ A \equiv Neg\ (Ex\ i\ (Neg\ A))$

**abbreviation**  $All2 :: name \Rightarrow tm \Rightarrow fm \Rightarrow fm$  — bounded universal quantifier, for Sigma formulas  
**where**  $All2\ i\ t\ A \equiv All\ i\ ((Var\ i\ IN\ t)\ IMP\ A)$

### Conjunction

**definition**  $Conj :: fm \Rightarrow fm \Rightarrow fm$  (**infixr**  $\langle AND \rangle$  135)  
**where**  $Conj\ A\ B \equiv Neg\ (Disj\ (Neg\ A)\ (Neg\ B))$

**lemma**  $Conj\_eqvt$  [eqvt]:  $p \cdot (A\ AND\ B) = (p \cdot A)\ AND\ (p \cdot B)$   
(proof)

**lemma**  $fresh\_Conj$  [simp]:  $a \# A\ AND\ B \longleftrightarrow (a \# A \wedge a \# B)$   
(proof)

**lemma**  $supp\_Conj$  [simp]:  $supp\ (A\ AND\ B) = supp\ A \cup supp\ B$   
(proof)

**lemma**  $size\_Conj$  [simp]:  $size\ (A\ AND\ B) = size\ A + size\ B + 4$   
(proof)

**lemma**  $Conj\_injective\_iff$  [iff]:  $(A\ AND\ B) = (A'\ AND\ B') \longleftrightarrow (A = A' \wedge B = B')$   
(proof)

**lemma**  $subst\_fm\_Conj$  [simp]:  $(A\ AND\ B)(i::=x) = (A(i::=x))\ AND\ (B(i::=x))$   
(proof)

### If and only if

**definition**  $Iff :: fm \Rightarrow fm \Rightarrow fm$  (**infixr**  $\langle IFF \rangle$  125)  
**where**  $Iff\ A\ B = Conj\ (Imp\ A\ B)\ (Imp\ B\ A)$

**lemma**  $Iff\_eqvt$  [eqvt]:  $p \cdot (A\ IFF\ B) = (p \cdot A)\ IFF\ (p \cdot B)$   
(proof)

**lemma**  $fresh\_Iff$  [simp]:  $a \# A\ IFF\ B \longleftrightarrow (a \# A \wedge a \# B)$   
(proof)

**lemma**  $size\_Iff$  [simp]:  $size\ (A\ IFF\ B) = 2*(size\ A + size\ B) + 8$   
(proof)

**lemma**  $Iff\_injective\_iff$  [iff]:  $(A\ IFF\ B) = (A'\ IFF\ B') \longleftrightarrow (A = A' \wedge B = B')$   
(proof)

**lemma**  $subst\_fm\_Iff$  [simp]:  $(A\ IFF\ B)(i::=x) = (A(i::=x))\ IFF\ (B(i::=x))$   
(proof)

## 1.2 Axioms and Theorems

### 1.2.1 Logical axioms

**inductive\_set**  $boolean\_axioms :: fm\ set$

**where**

*Ident*:  $A \text{ IMP } A \in \text{boolean\_axioms}$   
| *DisjI1*:  $A \text{ IMP } (A \text{ OR } B) \in \text{boolean\_axioms}$   
| *DisjCont*:  $(A \text{ OR } A) \text{ IMP } A \in \text{boolean\_axioms}$   
| *DisjAssoc*:  $(A \text{ OR } (B \text{ OR } C)) \text{ IMP } ((A \text{ OR } B) \text{ OR } C) \in \text{boolean\_axioms}$   
| *DisjConj*:  $(C \text{ OR } A) \text{ IMP } ((\text{Neg } C) \text{ OR } B) \text{ IMP } (A \text{ OR } B) \in \text{boolean\_axioms}$

**inductive\_set special\_axioms :: fm set where**

*I*:  $A(i::=x) \text{ IMP } (\text{Ex } i \ A) \in \text{special\_axioms}$

**inductive\_set induction\_axioms :: fm set where**

*ind*:

*atom* ( $j::\text{name}$ )  $\#$  ( $i, A$ )  
 $\implies A(i::=\text{Zero}) \text{ IMP } ((\text{All } i \ (\text{All } j \ (A \text{ IMP } (A(i::=\text{Var } j) \text{ IMP } A(i::=\text{Eats}(\text{Var } i)(\text{Var } j)))))) \text{ IMP } (\text{All } i \ A))$   
 $\in \text{induction\_axioms}$

## 1.2.2 Concrete variables

**declare** *Abs\_name\_inject*[*simp*]

**abbreviation**

$X0 \equiv \text{Abs\_name } (\text{Atom } (\text{Sort } \text{"SyntaxN.name"} \ [])) \ 0$

**abbreviation**

$X1 \equiv \text{Abs\_name } (\text{Atom } (\text{Sort } \text{"SyntaxN.name"} \ [])) \ (\text{Suc } 0)$

— We prefer *Suc 0* because simplification will transform 1 to that form anyway.

**abbreviation**

$X2 \equiv \text{Abs\_name } (\text{Atom } (\text{Sort } \text{"SyntaxN.name"} \ [])) \ 2$

**abbreviation**

$X3 \equiv \text{Abs\_name } (\text{Atom } (\text{Sort } \text{"SyntaxN.name"} \ [])) \ 3$

**abbreviation**

$X4 \equiv \text{Abs\_name } (\text{Atom } (\text{Sort } \text{"SyntaxN.name"} \ [])) \ 4$

## 1.2.3 The HF axioms

**definition** *HF1* :: fm where — the axiom  $(z = 0) = (\forall x. x \notin z)$

$\text{HF1} = (\text{Var } X0 \ \text{EQ } \text{Zero}) \ \text{IFF} \ (\text{All } X1 \ (\text{Neg } (\text{Var } X1 \ \text{IN } \text{Var } X0)))$

**definition** *HF2* :: fm where — the axiom  $(z = x \triangleleft y) = (\forall u. (u \in z) = (u \in x \vee u = y))$

$\text{HF2} \equiv \text{Var } X0 \ \text{EQ } \text{Eats } (\text{Var } X1) \ (\text{Var } X2) \ \text{IFF}$   
 $\text{All } X3 \ (\text{Var } X3 \ \text{IN } \text{Var } X0 \ \text{IFF } \text{Var } X3 \ \text{IN } \text{Var } X1 \ \text{OR } \text{Var } X3 \ \text{EQ } \text{Var } X2)$

**definition** *HF\_axioms* where  $\text{HF\_axioms} = \{\text{HF1}, \text{HF2}\}$

## 1.2.4 Equality axioms

**definition** *refl\_ax* :: fm where

$\text{refl\_ax} = \text{Var } X1 \ \text{EQ } \text{Var } X1$

**definition** *eq\_cong\_ax* :: fm where

$\text{eq\_cong\_ax} = ((\text{Var } X1 \ \text{EQ } \text{Var } X2) \ \text{AND} \ (\text{Var } X3 \ \text{EQ } \text{Var } X4)) \ \text{IMP}$   
 $((\text{Var } X1 \ \text{EQ } \text{Var } X3) \ \text{IMP} \ (\text{Var } X2 \ \text{EQ } \text{Var } X4))$

**definition** *mem\_cong\_ax* :: fm where

$\text{mem\_cong\_ax} = ((\text{Var } X1 \ \text{EQ } \text{Var } X2) \ \text{AND} \ (\text{Var } X3 \ \text{EQ } \text{Var } X4)) \ \text{IMP}$

$((\text{Var } X1 \text{ IN } \text{Var } X3) \text{ IMP } (\text{Var } X2 \text{ IN } \text{Var } X4))$

**definition** *eats\_cong\_ax* :: *fm* **where**

$\text{eats\_cong\_ax} = ((\text{Var } X1 \text{ EQ } \text{Var } X2) \text{ AND } (\text{Var } X3 \text{ EQ } \text{Var } X4)) \text{ IMP}$   
 $((\text{Eats } (\text{Var } X1) (\text{Var } X3)) \text{ EQ } (\text{Eats } (\text{Var } X2) (\text{Var } X4)))$

**definition** *equality\_axioms* :: *fm set* **where**

$\text{equality\_axioms} = \{\text{refl\_ax}, \text{eq\_cong\_ax}, \text{mem\_cong\_ax}, \text{eats\_cong\_ax}\}$

### 1.2.5 The proof system

This arbitrary additional axiom generalises the statements of the incompleteness theorems and other results to any formal system stronger than the HF theory. The additional axiom could be the conjunction of any finite number of assertions. Any more general extension must be a form that can be formalised for the proof predicate.

**consts** *extra\_axiom* :: *fm*

**inductive** *hftm* :: *fm set*  $\Rightarrow$  *fm*  $\Rightarrow$  *bool* (**infixl**  $\langle \vdash \rangle$  55)

**where**

*Hyp*:  $A \in H \Longrightarrow H \vdash A$   
*Extra*:  $H \vdash \text{extra\_axiom}$   
*Bool*:  $A \in \text{boolean\_axioms} \Longrightarrow H \vdash A$   
*Eq*:  $A \in \text{equality\_axioms} \Longrightarrow H \vdash A$   
*Spec*:  $A \in \text{special\_axioms} \Longrightarrow H \vdash A$   
*HF*:  $A \in \text{HF\_axioms} \Longrightarrow H \vdash A$   
*Ind*:  $A \in \text{induction\_axioms} \Longrightarrow H \vdash A$   
*MP*:  $H \vdash A \text{ IMP } B \Longrightarrow H' \vdash A \Longrightarrow H \cup H' \vdash B$   
*Exists*:  $H \vdash A \text{ IMP } B \Longrightarrow \text{atom } i \nmid B \Longrightarrow \forall C \in H. \text{atom } i \nmid C \Longrightarrow H \vdash (\text{Ex } i \ A) \text{ IMP } B$

### 1.2.6 Derived rules of inference

**lemma** *contraction*:  $\text{insert } A (\text{insert } A H) \vdash B \Longrightarrow \text{insert } A H \vdash B$   
 $\langle \text{proof} \rangle$

**lemma** *thin\_Un*:  $H \vdash A \Longrightarrow H \cup H' \vdash A$   
 $\langle \text{proof} \rangle$

**lemma** *thin*:  $H \vdash A \Longrightarrow H \subseteq H' \Longrightarrow H' \vdash A$   
 $\langle \text{proof} \rangle$

**lemma** *thin0*:  $\{\} \vdash A \Longrightarrow H \vdash A$   
 $\langle \text{proof} \rangle$

**lemma** *thin1*:  $H \vdash B \Longrightarrow \text{insert } A H \vdash B$   
 $\langle \text{proof} \rangle$

**lemma** *thin2*:  $\text{insert } A1 H \vdash B \Longrightarrow \text{insert } A1 (\text{insert } A2 H) \vdash B$   
 $\langle \text{proof} \rangle$

**lemma** *thin3*:  $\text{insert } A1 (\text{insert } A2 H) \vdash B \Longrightarrow \text{insert } A1 (\text{insert } A2 (\text{insert } A3 H)) \vdash B$   
 $\langle \text{proof} \rangle$

**lemma** *thin4*:  
 $\text{insert } A1 (\text{insert } A2 (\text{insert } A3 H)) \vdash B$   
 $\Longrightarrow \text{insert } A1 (\text{insert } A2 (\text{insert } A3 (\text{insert } A4 H))) \vdash B$   
 $\langle \text{proof} \rangle$

**lemma** *rotate2*:  $\text{insert } A2 (\text{insert } A1 H) \vdash B \Longrightarrow \text{insert } A1 (\text{insert } A2 H) \vdash B$



**lemma rotate13:**

$insert\ A13\ (insert\ A1\ (insert\ A2\ (insert\ A3\ (insert\ A4\ (insert\ A5\ (insert\ A6\ (insert\ A7\ (insert\ A8\ (insert\ A9\ (insert\ A10\ (insert\ A11\ (insert\ A12\ H)))))))))) \vdash B$   
 $\implies insert\ A1\ (insert\ A2\ (insert\ A3\ (insert\ A4\ (insert\ A5\ (insert\ A6\ (insert\ A7\ (insert\ A8\ (insert\ A9\ (insert\ A10\ (insert\ A11\ (insert\ A12\ (insert\ A13\ H)))))))))) \vdash B$   
(proof)

**lemma rotate14:**

$insert\ A14\ (insert\ A1\ (insert\ A2\ (insert\ A3\ (insert\ A4\ (insert\ A5\ (insert\ A6\ (insert\ A7\ (insert\ A8\ (insert\ A9\ (insert\ A10\ (insert\ A11\ (insert\ A12\ (insert\ A13\ H)))))))))) \vdash B$   
 $\implies insert\ A1\ (insert\ A2\ (insert\ A3\ (insert\ A4\ (insert\ A5\ (insert\ A6\ (insert\ A7\ (insert\ A8\ (insert\ A9\ (insert\ A10\ (insert\ A11\ (insert\ A12\ (insert\ A13\ (insert\ A14\ H)))))))))) \vdash B$   
(proof)

**lemma rotate15:**

$insert\ A15\ (insert\ A1\ (insert\ A2\ (insert\ A3\ (insert\ A4\ (insert\ A5\ (insert\ A6\ (insert\ A7\ (insert\ A8\ (insert\ A9\ (insert\ A10\ (insert\ A11\ (insert\ A12\ (insert\ A13\ (insert\ A14\ H)))))))))) \vdash B$   
 $\implies insert\ A1\ (insert\ A2\ (insert\ A3\ (insert\ A4\ (insert\ A5\ (insert\ A6\ (insert\ A7\ (insert\ A8\ (insert\ A9\ (insert\ A10\ (insert\ A11\ (insert\ A12\ (insert\ A13\ (insert\ A14\ (insert\ A15\ H)))))))))) \vdash B$   
(proof)

**lemma MP\_same:**  $H \vdash A \text{ IMP } B \implies H \vdash A \implies H \vdash B$   
(proof)

**lemma MP\_thin:**  $HA \vdash A \text{ IMP } B \implies HB \vdash A \implies HA \cup HB \subseteq H \implies H \vdash B$   
(proof)

**lemma MP\_null:**  $\{\} \vdash A \text{ IMP } B \implies H \vdash A \implies H \vdash B$   
(proof)

**lemma Disj\_commute:**  $H \vdash B \text{ OR } A \implies H \vdash A \text{ OR } B$   
(proof)

**lemma S: assumes**  $H \vdash A \text{ IMP } (B \text{ IMP } C)$  ***H'***  $\vdash A \text{ IMP } B$  **shows**  $H \cup H' \vdash A \text{ IMP } C$   
(proof)

**lemma Assume:**  $insert\ A\ H \vdash A$   
(proof)

**lemmas**  $AssumeH = Assume\ Assume\ [THEN\ rotate2]\ Assume\ [THEN\ rotate3]\ Assume\ [THEN\ rotate4]\ Assume\ [THEN\ rotate5]\ Assume\ [THEN\ rotate6]\ Assume\ [THEN\ rotate7]\ Assume\ [THEN\ rotate8]\ Assume\ [THEN\ rotate9]\ Assume\ [THEN\ rotate10]\ Assume\ [THEN\ rotate11]\ Assume\ [THEN\ rotate12]$

**declare**  $AssumeH\ [intro!]$

**lemma Imp\_triv\_I:**  $H \vdash B \implies H \vdash A \text{ IMP } B$   
(proof)

**lemma DisjAssoc1:**  $H \vdash A \text{ OR } (B \text{ OR } C) \implies H \vdash (A \text{ OR } B) \text{ OR } C$   
(proof)

**lemma DisjAssoc2:**  $H \vdash (A \text{ OR } B) \text{ OR } C \implies H \vdash A \text{ OR } (B \text{ OR } C)$   
(proof)

**lemma Disj\_commute\_Imp:**  $H \vdash (B \text{ OR } A) \text{ IMP } (A \text{ OR } B)$   
(proof)

**lemma** *Disj\_Semicong\_1*:  $H \vdash A \text{ OR } C \implies H \vdash A \text{ IMP } B \implies H \vdash B \text{ OR } C$   
*<proof>*

**lemma** *Imp\_Imp\_commute*:  $H \vdash B \text{ IMP } (A \text{ IMP } C) \implies H \vdash A \text{ IMP } (B \text{ IMP } C)$   
*<proof>*

### 1.2.7 The Deduction Theorem

**lemma** *deduction\_Diff*: **assumes**  $H \vdash B$  **shows**  $H - \{C\} \vdash C \text{ IMP } B$   
*<proof>*

**theorem** *Imp\_I* [*intro!*]: *insert*  $A \ H \vdash B \implies H \vdash A \text{ IMP } B$   
*<proof>*

**lemma** *anti\_deduction*:  $H \vdash A \text{ IMP } B \implies \text{insert } A \ H \vdash B$   
*<proof>*

### 1.2.8 Cut rules

**lemma** *cut*:  $H \vdash A \implies \text{insert } A \ H' \vdash B \implies H \cup H' \vdash B$   
*<proof>*

**lemma** *cut\_same*:  $H \vdash A \implies \text{insert } A \ H \vdash B \implies H \vdash B$   
*<proof>*

**lemma** *cut\_thin*:  $HA \vdash A \implies \text{insert } A \ HB \vdash B \implies HA \cup HB \subseteq H \implies H \vdash B$   
*<proof>*

**lemma** *cut0*:  $\{\} \vdash A \implies \text{insert } A \ H \vdash B \implies H \vdash B$   
*<proof>*

**lemma** *cut1*:  $\{A\} \vdash B \implies H \vdash A \implies H \vdash B$   
*<proof>*

**lemma** *rcut1*:  $\{A\} \vdash B \implies \text{insert } B \ H \vdash C \implies \text{insert } A \ H \vdash C$   
*<proof>*

**lemma** *cut2*:  $[[\{A,B\} \vdash C; H \vdash A; H \vdash B]] \implies H \vdash C$   
*<proof>*

**lemma** *rcut2*:  $\{A,B\} \vdash C \implies \text{insert } C \ H \vdash D \implies H \vdash B \implies \text{insert } A \ H \vdash D$   
*<proof>*

**lemma** *cut3*:  $[[\{A,B,C\} \vdash D; H \vdash A; H \vdash B; H \vdash C]] \implies H \vdash D$   
*<proof>*

**lemma** *cut4*:  $[[\{A,B,C,D\} \vdash E; H \vdash A; H \vdash B; H \vdash C; H \vdash D]] \implies H \vdash E$   
*<proof>*

## 1.3 Miscellaneous logical rules

**lemma** *Disj\_I1*:  $H \vdash A \implies H \vdash A \text{ OR } B$   
*<proof>*

**lemma** *Disj\_I2*:  $H \vdash B \implies H \vdash A \text{ OR } B$   
*<proof>*

**lemma** *Peirce*:  $H \vdash (\text{Neg } A) \text{ IMP } A \implies H \vdash A$   
 ⟨proof⟩

**lemma** *Contra*:  $\text{insert } (\text{Neg } A) H \vdash A \implies H \vdash A$   
 ⟨proof⟩

**lemma** *Imp\_Neg\_I*:  $H \vdash A \text{ IMP } B \implies H \vdash A \text{ IMP } (\text{Neg } B) \implies H \vdash \text{Neg } A$   
 ⟨proof⟩

**lemma** *NegNeg\_I*:  $H \vdash A \implies H \vdash \text{Neg } (\text{Neg } A)$   
 ⟨proof⟩

**lemma** *NegNeg\_D*:  $H \vdash \text{Neg } (\text{Neg } A) \implies H \vdash A$   
 ⟨proof⟩

**lemma** *Neg\_D*:  $H \vdash \text{Neg } A \implies H \vdash A \implies H \vdash B$   
 ⟨proof⟩

**lemma** *Disj\_Neg\_1*:  $H \vdash A \text{ OR } B \implies H \vdash \text{Neg } B \implies H \vdash A$   
 ⟨proof⟩

**lemma** *Disj\_Neg\_2*:  $H \vdash A \text{ OR } B \implies H \vdash \text{Neg } A \implies H \vdash B$   
 ⟨proof⟩

**lemma** *Neg\_Disj\_I*:  $H \vdash \text{Neg } A \implies H \vdash \text{Neg } B \implies H \vdash \text{Neg } (A \text{ OR } B)$   
 ⟨proof⟩

**lemma** *Conj\_I* [intro!]:  $H \vdash A \implies H \vdash B \implies H \vdash A \text{ AND } B$   
 ⟨proof⟩

**lemma** *Conj\_E1*:  $H \vdash A \text{ AND } B \implies H \vdash A$   
 ⟨proof⟩

**lemma** *Conj\_E2*:  $H \vdash A \text{ AND } B \implies H \vdash B$   
 ⟨proof⟩

**lemma** *Conj\_commute*:  $H \vdash B \text{ AND } A \implies H \vdash A \text{ AND } B$   
 ⟨proof⟩

**lemma** *Conj\_E*: **assumes**  $\text{insert } A (\text{insert } B H) \vdash C$  **shows**  $\text{insert } (A \text{ AND } B) H \vdash C$   
 ⟨proof⟩

**lemmas** *Conj\_EH* = *Conj\_E* *Conj\_E* [THEN rotate2] *Conj\_E* [THEN rotate3] *Conj\_E* [THEN rotate4] *Conj\_E* [THEN rotate5]  
           *Conj\_E* [THEN rotate6] *Conj\_E* [THEN rotate7] *Conj\_E* [THEN rotate8] *Conj\_E* [THEN rotate9] *Conj\_E* [THEN rotate10]

**declare** *Conj\_EH* [intro!]

**lemma** *Neg\_I0*: **assumes**  $(\bigwedge B. \text{atom } i \nmid B \implies \text{insert } A H \vdash B)$  **shows**  $H \vdash \text{Neg } A$   
 ⟨proof⟩

**lemma** *Neg\_mono*:  $\text{insert } A H \vdash B \implies \text{insert } (\text{Neg } B) H \vdash \text{Neg } A$   
 ⟨proof⟩

**lemma** *Conj\_mono*:  $\text{insert } A H \vdash B \implies \text{insert } C H \vdash D \implies \text{insert } (A \text{ AND } C) H \vdash B \text{ AND } D$   
 ⟨proof⟩

**lemma** *Disj\_mono*:

**assumes**  $insert\ A\ H \vdash B\ insert\ C\ H \vdash D$  **shows**  $insert\ (A\ OR\ C)\ H \vdash B\ OR\ D$   
(*proof*)

**lemma** *Disj\_E*:

**assumes**  $A: insert\ A\ H \vdash C$  **and**  $B: insert\ B\ H \vdash C$  **shows**  $insert\ (A\ OR\ B)\ H \vdash C$   
(*proof*)

**lemmas**  $Disj\_EH = Disj\_E\ Disj\_E\ [THEN\ rotate2]\ Disj\_E\ [THEN\ rotate3]\ Disj\_E\ [THEN\ rotate4]\ Disj\_E\ [THEN\ rotate5]$

$Disj\_E\ [THEN\ rotate6]\ Disj\_E\ [THEN\ rotate7]\ Disj\_E\ [THEN\ rotate8]\ Disj\_E\ [THEN\ rotate9]\ Disj\_E\ [THEN\ rotate10]$

**declare** *Disj\_EH* [*intro!*]

**lemma** *Contra'*:  $insert\ A\ H \vdash Neg\ A \implies H \vdash Neg\ A$

(*proof*)

**lemma** *NegNeg\_E* [*intro!*]:  $insert\ A\ H \vdash B \implies insert\ (Neg\ (Neg\ A))\ H \vdash B$

(*proof*)

**declare** *NegNeg\_E* [*THEN rotate2, intro!*]

**declare** *NegNeg\_E* [*THEN rotate3, intro!*]

**declare** *NegNeg\_E* [*THEN rotate4, intro!*]

**declare** *NegNeg\_E* [*THEN rotate5, intro!*]

**declare** *NegNeg\_E* [*THEN rotate6, intro!*]

**declare** *NegNeg\_E* [*THEN rotate7, intro!*]

**declare** *NegNeg\_E* [*THEN rotate8, intro!*]

**lemma** *Imp\_E*:

**assumes**  $A: H \vdash A$  **and**  $B: insert\ B\ H \vdash C$  **shows**  $insert\ (A\ IMP\ B)\ H \vdash C$

(*proof*)

**lemma** *Imp\_cut*:

**assumes**  $insert\ C\ H \vdash A\ IMP\ B\ \{A\} \vdash C$

**shows**  $H \vdash A\ IMP\ B$

(*proof*)

**lemma** *Iff\_I* [*intro!*]:  $insert\ A\ H \vdash B \implies insert\ B\ H \vdash A \implies H \vdash A\ IFF\ B$

(*proof*)

**lemma** *Iff\_MP\_same*:  $H \vdash A\ IFF\ B \implies H \vdash A \implies H \vdash B$

(*proof*)

**lemma** *Iff\_MP2\_same*:  $H \vdash A\ IFF\ B \implies H \vdash B \implies H \vdash A$

(*proof*)

**lemma** *Iff\_refl* [*intro!*]:  $H \vdash A\ IFF\ A$

(*proof*)

**lemma** *Iff\_sym*:  $H \vdash A\ IFF\ B \implies H \vdash B\ IFF\ A$

(*proof*)

**lemma** *Iff\_trans*:  $H \vdash A\ IFF\ B \implies H \vdash B\ IFF\ C \implies H \vdash A\ IFF\ C$

(*proof*)

**lemma** *Iff\_E*:

$insert\ A\ (insert\ B\ H) \vdash C \implies insert\ (Neg\ A)\ (insert\ (Neg\ B)\ H) \vdash C \implies insert\ (A\ IFF\ B)\ H \vdash C$

(*proof*)

**lemma** *Iff\_E1*:  
**assumes**  $A: H \vdash A$  **and**  $B: \text{insert } B \ H \vdash C$  **shows**  $\text{insert } (A \text{ IFF } B) \ H \vdash C$   
 $\langle \text{proof} \rangle$

**lemma** *Iff\_E2*:  
**assumes**  $A: H \vdash A$  **and**  $B: \text{insert } B \ H \vdash C$  **shows**  $\text{insert } (B \text{ IFF } A) \ H \vdash C$   
 $\langle \text{proof} \rangle$

**lemma** *Iff\_MP\_left*:  $H \vdash A \text{ IFF } B \implies \text{insert } A \ H \vdash C \implies \text{insert } B \ H \vdash C$   
 $\langle \text{proof} \rangle$

**lemma** *Iff\_MP\_left'*:  $H \vdash A \text{ IFF } B \implies \text{insert } B \ H \vdash C \implies \text{insert } A \ H \vdash C$   
 $\langle \text{proof} \rangle$

**lemma** *Swap*:  $\text{insert } (\text{Neg } B) \ H \vdash A \implies \text{insert } (\text{Neg } A) \ H \vdash B$   
 $\langle \text{proof} \rangle$

**lemma** *Cases*:  $\text{insert } A \ H \vdash B \implies \text{insert } (\text{Neg } A) \ H \vdash B \implies H \vdash B$   
 $\langle \text{proof} \rangle$

**lemma** *Neg\_Conj\_E*:  $H \vdash B \implies \text{insert } (\text{Neg } A) \ H \vdash C \implies \text{insert } (\text{Neg } (A \text{ AND } B)) \ H \vdash C$   
 $\langle \text{proof} \rangle$

**lemma** *Disj\_CI*:  $\text{insert } (\text{Neg } B) \ H \vdash A \implies H \vdash A \text{ OR } B$   
 $\langle \text{proof} \rangle$

**lemma** *Disj\_3I*:  $\text{insert } (\text{Neg } A) \ (\text{insert } (\text{Neg } C) \ H) \vdash B \implies H \vdash A \text{ OR } B \text{ OR } C$   
 $\langle \text{proof} \rangle$

**lemma** *Contrapos1*:  $H \vdash A \text{ IMP } B \implies H \vdash \text{Neg } B \text{ IMP } \text{Neg } A$   
 $\langle \text{proof} \rangle$

**lemma** *Contrapos2*:  $H \vdash (\text{Neg } B) \text{ IMP } (\text{Neg } A) \implies H \vdash A \text{ IMP } B$   
 $\langle \text{proof} \rangle$

**lemma** *ContraAssumeN* [*intro*]:  $B \in H \implies \text{insert } (\text{Neg } B) \ H \vdash A$   
 $\langle \text{proof} \rangle$

**lemma** *ContraAssume*:  $\text{Neg } B \in H \implies \text{insert } B \ H \vdash A$   
 $\langle \text{proof} \rangle$

**lemma** *ContraProve*:  $H \vdash B \implies \text{insert } (\text{Neg } B) \ H \vdash A$   
 $\langle \text{proof} \rangle$

**lemma** *Disj\_IE1*:  $\text{insert } B \ H \vdash C \implies \text{insert } (A \text{ OR } B) \ H \vdash A \text{ OR } C$   
 $\langle \text{proof} \rangle$

**lemmas**  $\text{Disj\_IE1H} = \text{Disj\_IE1 } \text{Disj\_IE1 } [\text{THEN rotate2}] \text{Disj\_IE1 } [\text{THEN rotate3}] \text{Disj\_IE1 } [\text{THEN rotate4}] \text{Disj\_IE1 } [\text{THEN rotate5}]$   
 $\text{Disj\_IE1 } [\text{THEN rotate6}] \text{Disj\_IE1 } [\text{THEN rotate7}] \text{Disj\_IE1 } [\text{THEN rotate8}]$

**declare** *Disj\_IE1H* [*intro!*]

### 1.3.1 Quantifier reasoning

**lemma** *Ex\_I*:  $H \vdash A(i::=x) \implies H \vdash \text{Ex } i \ A$   
 $\langle \text{proof} \rangle$

**lemma** *Ex\_E*:

**assumes**  $insert\ A\ H \vdash B\ atom\ i \# B \forall C \in H. atom\ i \# C$   
**shows**  $insert\ (Ex\ i\ A)\ H \vdash B$   
 $\langle proof \rangle$

**lemma**  $Ex\_E\_with\_renaming$ :

**assumes**  $insert\ ((i \leftrightarrow i') \cdot A)\ H \vdash B\ atom\ i' \# (A, i, B) \forall C \in H. atom\ i' \# C$   
**shows**  $insert\ (Ex\ i\ A)\ H \vdash B$   
 $\langle proof \rangle$

**lemmas**  $Ex\_EH = Ex\_E\ Ex\_E\ [THEN\ rotate2]\ Ex\_E\ [THEN\ rotate3]\ Ex\_E\ [THEN\ rotate4]\ Ex\_E$   
 $[THEN\ rotate5]$

$Ex\_E\ [THEN\ rotate6]\ Ex\_E\ [THEN\ rotate7]\ Ex\_E\ [THEN\ rotate8]\ Ex\_E\ [THEN\ rotate9]$   
 $Ex\_E\ [THEN\ rotate10]$

**declare**  $Ex\_EH\ [intro!]$

**lemma**  $Ex\_mono$ :  $insert\ A\ H \vdash B \implies \forall C \in H. atom\ i \# C \implies insert\ (Ex\ i\ A)\ H \vdash (Ex\ i\ B)$   
 $\langle proof \rangle$

**lemma**  $All\_I\ [intro!]$ :  $H \vdash A \implies \forall C \in H. atom\ i \# C \implies H \vdash All\ i\ A$   
 $\langle proof \rangle$

**lemma**  $All\_D$ :  $H \vdash All\ i\ A \implies H \vdash A(i::=x)$   
 $\langle proof \rangle$

**lemma**  $All\_E$ :  $insert\ (A(i::=x))\ H \vdash B \implies insert\ (All\ i\ A)\ H \vdash B$   
 $\langle proof \rangle$

**lemma**  $All\_E'$ :  $H \vdash All\ i\ A \implies insert\ (A(i::=x))\ H \vdash B \implies H \vdash B$   
 $\langle proof \rangle$

**lemma**  $All2\_E$ :  $\llbracket atom\ i \# t; H \vdash x\ IN\ t; insert\ (A(i::=x))\ H \vdash B \rrbracket \implies insert\ (All2\ i\ t\ A)\ H \vdash B$   
 $\langle proof \rangle$

**lemma**  $All2\_E'$ :  $\llbracket H \vdash All2\ i\ t\ A; H \vdash x\ IN\ t; insert\ (A(i::=x))\ H \vdash B; atom\ i \# t \rrbracket \implies H \vdash B$   
 $\langle proof \rangle$

### 1.3.2 Congruence rules

**lemma**  $Neg\_cong$ :  $H \vdash A\ IFF\ A' \implies H \vdash Neg\ A\ IFF\ Neg\ A'$   
 $\langle proof \rangle$

**lemma**  $Disj\_cong$ :  $H \vdash A\ IFF\ A' \implies H \vdash B\ IFF\ B' \implies H \vdash A\ OR\ B\ IFF\ A'\ OR\ B'$   
 $\langle proof \rangle$

**lemma**  $Conj\_cong$ :  $H \vdash A\ IFF\ A' \implies H \vdash B\ IFF\ B' \implies H \vdash A\ AND\ B\ IFF\ A'\ AND\ B'$   
 $\langle proof \rangle$

**lemma**  $Imp\_cong$ :  $H \vdash A\ IFF\ A' \implies H \vdash B\ IFF\ B' \implies H \vdash (A\ IMP\ B)\ IFF\ (A'\ IMP\ B')$   
 $\langle proof \rangle$

**lemma**  $Iff\_cong$ :  $H \vdash A\ IFF\ A' \implies H \vdash B\ IFF\ B' \implies H \vdash (A\ IFF\ B)\ IFF\ (A'\ IFF\ B')$   
 $\langle proof \rangle$

**lemma**  $Ex\_cong$ :  $H \vdash A\ IFF\ A' \implies \forall C \in H. atom\ i \# C \implies H \vdash (Ex\ i\ A)\ IFF\ (Ex\ i\ A')$   
 $\langle proof \rangle$

**lemma**  $All\_cong$ :  $H \vdash A\ IFF\ A' \implies \forall C \in H. atom\ i \# C \implies H \vdash (All\ i\ A)\ IFF\ (All\ i\ A')$   
 $\langle proof \rangle$

**lemma** *Subst*:  $H \vdash A \implies \forall B \in H. \text{atom } i \# B \implies H \vdash A (i ::= x)$   
 ⟨proof⟩

## 1.4 Equality reasoning

### 1.4.1 The congruence property for (*EQ*), and other basic properties of equality

**lemma** *Eq\_cong1*:  $\{\} \vdash (t \text{ EQ } t' \text{ AND } u \text{ EQ } u') \text{ IMP } (t \text{ EQ } u \text{ IMP } t' \text{ EQ } u')$   
 ⟨proof⟩

**lemma** *Refl [iff]*:  $H \vdash t \text{ EQ } t$   
 ⟨proof⟩

Apparently necessary in order to prove the congruence property.

**lemma** *Sym*: **assumes**  $H \vdash t \text{ EQ } u$  **shows**  $H \vdash u \text{ EQ } t$   
 ⟨proof⟩

**lemma** *Sym\_L*:  $\text{insert } (t \text{ EQ } u) H \vdash A \implies \text{insert } (u \text{ EQ } t) H \vdash A$   
 ⟨proof⟩

**lemma** *Trans*: **assumes**  $H \vdash x \text{ EQ } y$   $H \vdash y \text{ EQ } z$  **shows**  $H \vdash x \text{ EQ } z$   
 ⟨proof⟩

**lemma** *Eq\_cong*:  
**assumes**  $H \vdash t \text{ EQ } t'$   $H \vdash u \text{ EQ } u'$  **shows**  $H \vdash t \text{ EQ } u \text{ IFF } t' \text{ EQ } u'$   
 ⟨proof⟩

**lemma** *Eq\_Trans\_E*:  $H \vdash x \text{ EQ } u \implies \text{insert } (t \text{ EQ } u) H \vdash A \implies \text{insert } (x \text{ EQ } t) H \vdash A$   
 ⟨proof⟩

### 1.4.2 The congruence property for (*IN*)

**lemma** *Mem\_cong1*:  $\{\} \vdash (t \text{ EQ } t' \text{ AND } u \text{ EQ } u') \text{ IMP } (t \text{ IN } u \text{ IMP } t' \text{ IN } u')$   
 ⟨proof⟩

**lemma** *Mem\_cong*:  
**assumes**  $H \vdash t \text{ EQ } t'$   $H \vdash u \text{ EQ } u'$  **shows**  $H \vdash t \text{ IN } u \text{ IFF } t' \text{ IN } u'$   
 ⟨proof⟩

### 1.4.3 The congruence properties for *Eats* and *HPair*

**lemma** *Eats\_cong1*:  $\{\} \vdash (t \text{ EQ } t' \text{ AND } u \text{ EQ } u') \text{ IMP } (Eats \ t \ u \ \text{EQ } Eats \ t' \ u')$   
 ⟨proof⟩

**lemma** *Eats\_cong*:  $\llbracket H \vdash t \text{ EQ } t'; H \vdash u \text{ EQ } u' \rrbracket \implies H \vdash Eats \ t \ u \ \text{EQ } Eats \ t' \ u'$   
 ⟨proof⟩

**lemma** *HPair\_cong*:  $\llbracket H \vdash t \text{ EQ } t'; H \vdash u \text{ EQ } u' \rrbracket \implies H \vdash HPair \ t \ u \ \text{EQ } HPair \ t' \ u'$   
 ⟨proof⟩

**lemma** *SUCC\_cong*:  $H \vdash t \text{ EQ } t' \implies H \vdash SUCC \ t \ \text{EQ } SUCC \ t'$   
 ⟨proof⟩

### 1.4.4 Substitution for Equalities

**lemma** *Eq\_subst\_tm\_Iff*:  $\{t \text{ EQ } u\} \vdash \text{subst } i \ t \ tm \ \text{EQ } \text{subst } i \ u \ tm$

*<proof>*

**lemma** *Eq\_subst\_fm\_Iff*:  $\text{insert } (t \text{ EQ } u) \ H \vdash A(i::=t) \text{ IFF } A(i::=u)$   
*<proof>*

**lemma** *Var\_Eq\_subst\_Iff*:  $\text{insert } (\text{Var } i \text{ EQ } t) \ H \vdash A(i::=t) \text{ IFF } A$   
*<proof>*

**lemma** *Var\_Eq\_imp\_subst\_Iff*:  $H \vdash \text{Var } i \text{ EQ } t \implies H \vdash A(i::=t) \text{ IFF } A$   
*<proof>*

## 1.4.5 Congruence Rules for Predicates

**lemma** *P1\_cong*:

**fixes** *tms* :: *tm list*

**assumes**  $\bigwedge i \ t \ x. \text{atom } i \ \# \ tms \implies (P \ t)(i::=x) = P \ (\text{subst } i \ x \ t)$  **and**  $H \vdash x \text{ EQ } x'$   
**shows**  $H \vdash P \ x \text{ IFF } P \ x'$

*<proof>*

**lemma** *P2\_cong*:

**fixes** *tms* :: *tm list*

**assumes** *sub*:  $\bigwedge i \ t \ u \ v \ x. \text{atom } i \ \# \ tms \implies (P \ t \ u)(i::=x) = P \ (\text{subst } i \ x \ t) \ (\text{subst } i \ x \ u)$   
**and** *eq*:  $H \vdash x \text{ EQ } x' \ H \vdash y \text{ EQ } y'$   
**shows**  $H \vdash P \ x \ y \text{ IFF } P \ x' \ y'$

*<proof>*

**lemma** *P3\_cong*:

**fixes** *tms* :: *tm list*

**assumes** *sub*:  $\bigwedge i \ t \ u \ v \ x. \text{atom } i \ \# \ tms \implies$   
 $(P \ t \ u \ v)(i::=x) = P \ (\text{subst } i \ x \ t) \ (\text{subst } i \ x \ u) \ (\text{subst } i \ x \ v)$   
**and** *eq*:  $H \vdash x \text{ EQ } x' \ H \vdash y \text{ EQ } y' \ H \vdash z \text{ EQ } z'$   
**shows**  $H \vdash P \ x \ y \ z \text{ IFF } P \ x' \ y' \ z'$

*<proof>*

**lemma** *P4\_cong*:

**fixes** *tms* :: *tm list*

**assumes** *sub*:  $\bigwedge i \ t1 \ t2 \ t3 \ t4 \ x. \text{atom } i \ \# \ tms \implies$   
 $(P \ t1 \ t2 \ t3 \ t4)(i::=x) = P \ (\text{subst } i \ x \ t1) \ (\text{subst } i \ x \ t2) \ (\text{subst } i \ x \ t3) \ (\text{subst } i \ x \ t4)$   
**and** *eq*:  $H \vdash x1 \text{ EQ } x1' \ H \vdash x2 \text{ EQ } x2' \ H \vdash x3 \text{ EQ } x3' \ H \vdash x4 \text{ EQ } x4'$   
**shows**  $H \vdash P \ x1 \ x2 \ x3 \ x4 \text{ IFF } P \ x1' \ x2' \ x3' \ x4'$

*<proof>*

## 1.5 Zero and Falsity

**lemma** *Mem\_Zero\_iff*:

**assumes** *atom*  $i \ \# \ t$  **shows**  $H \vdash (t \text{ EQ } \text{Zero}) \text{ IFF } (\text{All } i \ (\text{Neg } ((\text{Var } i) \text{ IN } t)))$

*<proof>*

**lemma** *Mem\_Zero\_E* [*intro!*]:  $\text{insert } (x \text{ IN } \text{Zero}) \ H \vdash A$

*<proof>*

**declare** *Mem\_Zero\_E* [*THEN rotate2, intro!*]

**declare** *Mem\_Zero\_E* [*THEN rotate3, intro!*]

**declare** *Mem\_Zero\_E* [*THEN rotate4, intro!*]

**declare** *Mem\_Zero\_E* [*THEN rotate5, intro!*]

**declare** *Mem\_Zero\_E* [*THEN rotate6, intro!*]

**declare** *Mem\_Zero\_E* [*THEN rotate7, intro!*]

**declare** *Mem\_Zero\_E* [*THEN rotate8, intro!*]

### 1.5.1 The Formula $Fls$

**definition**  $Fls$  where  $Fls \equiv Zero$  IN  $Zero$

**lemma**  $Fls\_eqvt$  [*eqvt*]:  $(p \cdot Fls) = Fls$   
 ⟨*proof*⟩

**lemma**  $Fls\_fresh$  [*simp*]:  $a \# Fls$   
 ⟨*proof*⟩

**lemma**  $Neg\_I$  [*intro!*]:  $insert\ A\ H \vdash Fls \implies H \vdash Neg\ A$   
 ⟨*proof*⟩

**lemma**  $Neg\_E$  [*intro!*]:  $H \vdash A \implies insert\ (Neg\ A)\ H \vdash Fls$   
 ⟨*proof*⟩

**declare**  $Neg\_E$  [*THEN rotate2, intro!*]  
**declare**  $Neg\_E$  [*THEN rotate3, intro!*]  
**declare**  $Neg\_E$  [*THEN rotate4, intro!*]  
**declare**  $Neg\_E$  [*THEN rotate5, intro!*]  
**declare**  $Neg\_E$  [*THEN rotate6, intro!*]  
**declare**  $Neg\_E$  [*THEN rotate7, intro!*]  
**declare**  $Neg\_E$  [*THEN rotate8, intro!*]

We need these because  $Neg\ (A\ IMP\ B)$  doesn't have to be syntactically a conjunction.

**lemma**  $Neg\_Imp\_I$  [*intro!*]:  $H \vdash A \implies insert\ B\ H \vdash Fls \implies H \vdash Neg\ (A\ IMP\ B)$   
 ⟨*proof*⟩

**lemma**  $Neg\_Imp\_E$  [*intro!*]:  $insert\ (Neg\ B)\ (insert\ A\ H) \vdash C \implies insert\ (Neg\ (A\ IMP\ B))\ H \vdash C$   
 ⟨*proof*⟩

**declare**  $Neg\_Imp\_E$  [*THEN rotate2, intro!*]  
**declare**  $Neg\_Imp\_E$  [*THEN rotate3, intro!*]  
**declare**  $Neg\_Imp\_E$  [*THEN rotate4, intro!*]  
**declare**  $Neg\_Imp\_E$  [*THEN rotate5, intro!*]  
**declare**  $Neg\_Imp\_E$  [*THEN rotate6, intro!*]  
**declare**  $Neg\_Imp\_E$  [*THEN rotate7, intro!*]  
**declare**  $Neg\_Imp\_E$  [*THEN rotate8, intro!*]

**lemma**  $Fls\_E$  [*intro!*]:  $insert\ Fls\ H \vdash A$   
 ⟨*proof*⟩

**declare**  $Fls\_E$  [*THEN rotate2, intro!*]  
**declare**  $Fls\_E$  [*THEN rotate3, intro!*]  
**declare**  $Fls\_E$  [*THEN rotate4, intro!*]  
**declare**  $Fls\_E$  [*THEN rotate5, intro!*]  
**declare**  $Fls\_E$  [*THEN rotate6, intro!*]  
**declare**  $Fls\_E$  [*THEN rotate7, intro!*]  
**declare**  $Fls\_E$  [*THEN rotate8, intro!*]

**lemma**  $truth\_provable$ :  $H \vdash (Neg\ Fls)$   
 ⟨*proof*⟩

**lemma**  $ExFalse$ :  $H \vdash Fls \implies H \vdash A$   
 ⟨*proof*⟩

### 1.5.2 More properties of $Zero$

**lemma**  $Eq\_Zero\_D$ :

**assumes**  $H \vdash t \text{ EQ Zero } H \vdash u \text{ IN } t$  **shows**  $H \vdash A$   
 ⟨proof⟩

**lemma** *Eq\_Zero\_thm*:  
**assumes**  $\text{atom } i \# t$  **shows**  $\{ \text{All } i \text{ (Neg ((Var } i \text{ IN } t)) ) \} \vdash t \text{ EQ Zero}$   
 ⟨proof⟩

**lemma** *Eq\_Zero\_I*:  
**assumes**  $\text{insi: insert ((Var } i \text{ IN } t) H \vdash \text{Fls and } i1: \text{atom } i \# t \text{ and } i2: \forall B \in H. \text{atom } i \# B$   
**shows**  $H \vdash t \text{ EQ Zero}$   
 ⟨proof⟩

### 1.5.3 Basic properties of *Eats*

**lemma** *Eq\_Eats\_iff*:  
**assumes**  $\text{atom } i \# (z, t, u)$   
**shows**  $H \vdash (z \text{ EQ Eats } t \ u) \text{ IFF } (\text{All } i \text{ (Var } i \text{ IN } z \text{ IFF Var } i \text{ IN } t \text{ OR Var } i \text{ EQ } u))$   
 ⟨proof⟩

**lemma** *Eq\_Eats\_I*:  
 $H \vdash \text{All } i \text{ (Var } i \text{ IN } z \text{ IFF Var } i \text{ IN } t \text{ OR Var } i \text{ EQ } u) \implies \text{atom } i \# (z, t, u) \implies H \vdash z \text{ EQ Eats } t \ u$   
 ⟨proof⟩

**lemma** *Mem\_Eats\_Iff*:  
 $H \vdash x \text{ IN } (\text{Eats } t \ u) \text{ IFF } x \text{ IN } t \text{ OR } x \text{ EQ } u$   
 ⟨proof⟩

**lemma** *Mem\_Eats\_I1*:  $H \vdash u \text{ IN } t \implies H \vdash u \text{ IN Eats } t \ z$   
 ⟨proof⟩

**lemma** *Mem\_Eats\_I2*:  $H \vdash u \text{ EQ } z \implies H \vdash u \text{ IN Eats } t \ z$   
 ⟨proof⟩

**lemma** *Mem\_Eats\_E*:  
**assumes**  $A: \text{insert } (u \text{ IN } t) H \vdash C$  **and**  $B: \text{insert } (u \text{ EQ } z) H \vdash C$   
**shows**  $\text{insert } (u \text{ IN Eats } t \ z) H \vdash C$   
 ⟨proof⟩

**lemmas**  $\text{Mem\_Eats\_EH} = \text{Mem\_Eats\_E Mem\_Eats\_E [THEN rotate2] Mem\_Eats\_E [THEN rotate3] Mem\_Eats\_E [THEN rotate4] Mem\_Eats\_E [THEN rotate5] Mem\_Eats\_E [THEN rotate6] Mem\_Eats\_E [THEN rotate7] Mem\_Eats\_E [THEN rotate8]}$   
**declare** *Mem\_Eats\_EH* [intro!]

**lemma** *Mem\_SUCC\_I1*:  $H \vdash u \text{ IN } t \implies H \vdash u \text{ IN SUCC } t$   
 ⟨proof⟩

**lemma** *Mem\_SUCC\_I2*:  $H \vdash u \text{ EQ } t \implies H \vdash u \text{ IN SUCC } t$   
 ⟨proof⟩

**lemma** *Mem\_SUCC\_Refl* [simp]:  $H \vdash k \text{ IN SUCC } k$   
 ⟨proof⟩

**lemma** *Mem\_SUCC\_E*:  
**assumes**  $\text{insert } (u \text{ IN } t) H \vdash C$   $\text{insert } (u \text{ EQ } t) H \vdash C$  **shows**  $\text{insert } (u \text{ IN SUCC } t) H \vdash C$   
 ⟨proof⟩

**lemmas**  $\text{Mem\_SUCC\_EH} = \text{Mem\_SUCC\_E Mem\_SUCC\_E [THEN rotate2] Mem\_SUCC\_E [THEN$

rotate3] Mem\_SUCC\_E [THEN rotate4] Mem\_SUCC\_E [THEN rotate5]  
 Mem\_SUCC\_E [THEN rotate6] Mem\_SUCC\_E [THEN rotate7] Mem\_SUCC\_E [THEN  
 rotate8]

**lemma** *Eats\_EQ\_Zero\_E*: insert (Eats t u EQ Zero) H ⊢ A  
 ⟨proof⟩

**lemmas** *Eats\_EQ\_Zero\_EH* = Eats\_EQ\_Zero\_E Eats\_EQ\_Zero\_E [THEN rotate2] Eats\_EQ\_Zero\_E  
 [THEN rotate3] Eats\_EQ\_Zero\_E [THEN rotate4] Eats\_EQ\_Zero\_E [THEN rotate5]  
 Eats\_EQ\_Zero\_E [THEN rotate6] Eats\_EQ\_Zero\_E [THEN rotate7] Eats\_EQ\_Zero\_E  
 [THEN rotate8]

**declare** *Eats\_EQ\_Zero\_EH* [intro!]

**lemma** *Eats\_EQ\_Zero\_E2*: insert (Zero EQ Eats t u) H ⊢ A  
 ⟨proof⟩

**lemmas** *Eats\_EQ\_Zero\_E2H* = Eats\_EQ\_Zero\_E2 Eats\_EQ\_Zero\_E2 [THEN rotate2] Eats\_EQ\_Zero\_E2  
 [THEN rotate3] Eats\_EQ\_Zero\_E2 [THEN rotate4] Eats\_EQ\_Zero\_E2 [THEN rotate5]  
 Eats\_EQ\_Zero\_E2 [THEN rotate6] Eats\_EQ\_Zero\_E2 [THEN rotate7] Eats\_EQ\_Zero\_E2  
 [THEN rotate8]

**declare** *Eats\_EQ\_Zero\_E2H* [intro!]

## 1.6 Bounded Quantification involving *Eats*

**lemma** *All2\_cong*:  $H \vdash t \text{ EQ } t' \implies H \vdash A \text{ IFF } A' \implies \forall C \in H. \text{atom } i \# C \implies H \vdash (\text{All2 } i \ t \ A) \text{ IFF } (\text{All2 } i \ t' \ A')$   
 ⟨proof⟩

**lemma** *All2\_Zero\_E* [intro!]:  $H \vdash B \implies \text{insert } (\text{All2 } i \ \text{Zero } A) \ H \vdash B$   
 ⟨proof⟩

**lemma** *All2\_Eats\_I\_D*:  
 $\text{atom } i \# (t, u) \implies \{ \text{All2 } i \ t \ A, A(i::=u) \} \vdash (\text{All2 } i \ (\text{Eats } t \ u) \ A)$   
 ⟨proof⟩

**lemma** *All2\_Eats\_I*:  
 $\llbracket \text{atom } i \# (t, u); H \vdash \text{All2 } i \ t \ A; H \vdash A(i::=u) \rrbracket \implies H \vdash (\text{All2 } i \ (\text{Eats } t \ u) \ A)$   
 ⟨proof⟩

**lemma** *All2\_Eats\_E1*:  
 $\llbracket \text{atom } i \# (t, u); \forall C \in H. \text{atom } i \# C \rrbracket \implies \text{insert } (\text{All2 } i \ (\text{Eats } t \ u) \ A) \ H \vdash \text{All2 } i \ t \ A$   
 ⟨proof⟩

**lemma** *All2\_Eats\_E2*:  
 $\llbracket \text{atom } i \# (t, u); \forall C \in H. \text{atom } i \# C \rrbracket \implies \text{insert } (\text{All2 } i \ (\text{Eats } t \ u) \ A) \ H \vdash A(i::=u)$   
 ⟨proof⟩

**lemma** *All2\_Eats\_E*:  
**assumes** *i*:  $\text{atom } i \# (t, u)$   
**and** *B*:  $\text{insert } (\text{All2 } i \ t \ A) \ (\text{insert } (A(i::=u)) \ H) \vdash B$   
**shows**  $\text{insert } (\text{All2 } i \ (\text{Eats } t \ u) \ A) \ H \vdash B$   
 ⟨proof⟩

**lemma** *All2\_SUCC\_I*:  
 $\text{atom } i \# t \implies H \vdash \text{All2 } i \ t \ A \implies H \vdash A(i::=t) \implies H \vdash (\text{All2 } i \ (\text{SUCC } t) \ A)$   
 ⟨proof⟩

**lemma** *All2\_SUCC\_E*:

**assumes**  $atom\ i \# t$   
**and**  $insert\ (All2\ i\ t\ A)\ (insert\ (A(i::=t))\ H) \vdash B$   
**shows**  $insert\ (All2\ i\ (SUCC\ t)\ A)\ H \vdash B$   
 $\langle proof \rangle$

**lemma**  $All2\_SUCC\_E'$ :  
**assumes**  $H \vdash u\ EQ\ SUCC\ t$   
**and**  $atom\ i \# t \forall C \in H. atom\ i \# C$   
**and**  $insert\ (All2\ i\ t\ A)\ (insert\ (A(i::=t))\ H) \vdash B$   
**shows**  $insert\ (All2\ i\ u\ A)\ H \vdash B$   
 $\langle proof \rangle$

## 1.7 Induction

**lemma**  $Ind$ :  
**assumes**  $j: atom\ (j::name) \# (i, A)$   
**and**  $prems: H \vdash A(i::=Zero)\ H \vdash All\ i\ (All\ j\ (A\ IMP\ (A(i::=Var\ j)\ IMP\ A(i::=Eats(Var\ i)(Var\ j))))))$   
**shows**  $H \vdash A$   
 $\langle proof \rangle$   
**end**

## Chapter 2

# De Bruijn Syntax, Quotations, Codes, V-Codes

```
theory Coding
imports SyntaxN
begin

declare fresh_Nil [iff]
```

### 2.1 de Bruijn Indices (locally-nameless version)

```
nominal_datatype dbtm = DBZero | DBVar name | DBInd nat | DBEats dbtm dbtm
```

```
nominal_datatype dbfm =
  DBMem dbtm dbtm
| DBEq dbtm dbtm
| DBDisj dbfm dbfm
| DBNeg dbfm
| DBEx dbfm
```

```
declare dbtm.supp [simp]
declare dbfm.supp [simp]
```

```
fun lookup :: name list ⇒ nat ⇒ name ⇒ dbtm
  where
  lookup [] n x = DBVar x
  | lookup (y # ys) n x = (if x = y then DBInd n else (lookup ys (Suc n) x))
```

```
lemma fresh_imp_notin_env: atom name # e ⇒ name ∉ set e
  <proof>
```

```
lemma lookup_notin: x ∉ set e ⇒ lookup e n x = DBVar x
  <proof>
```

```
lemma lookup_in:
  x ∈ set e ⇒ ∃ k. lookup e n x = DBInd k ∧ n ≤ k ∧ k < n + length e
  <proof>
```

```
lemma lookup_fresh: x # lookup e n y ↔ y ∈ set e ∨ x ≠ atom y
  <proof>
```

```
lemma lookup_eqvt[eqvt]: (p · lookup xs n x) = lookup (p · xs) (p · n) (p · x)
```

*<proof>*

**lemma** *lookup\_inject* [iff]:  $(\text{lookup } e \ n \ x = \text{lookup } e \ n \ y) \longleftrightarrow x = y$   
*<proof>*

**nominal\_function** *trans\_tm* :: *name list*  $\Rightarrow$  *tm*  $\Rightarrow$  *dbtm*

**where**

*trans\_tm* *e* *Zero* = *DBZero*  
| *trans\_tm* *e* (*Var* *k*) = *lookup* *e* 0 *k*  
| *trans\_tm* *e* (*Eats* *t* *u*) = *DBEats* (*trans\_tm* *e* *t*) (*trans\_tm* *e* *u*)  
*<proof>*

**nominal\_termination** (*eqvt*)

*<proof>*

**lemma** *fresh\_trans\_tm\_iff* [simp]:  $i \# \text{trans\_tm } e \ t \longleftrightarrow i \# t \vee i \in \text{atom } ' \text{set } e$   
*<proof>*

**lemma** *trans\_tm\_forget*:  $\text{atom } i \# t \Longrightarrow \text{trans\_tm } [i] \ t = \text{trans\_tm } [] \ t$   
*<proof>*

**nominal\_function** (*invariant*  $\lambda(xs, \_) \ y. \text{atom } ' \text{set } xs \ \#* \ y$ )

*trans\_fm* :: *name list*  $\Rightarrow$  *fm*  $\Rightarrow$  *dbfm*

**where**

*trans\_fm* *e* (*Mem* *t* *u*) = *DBMem* (*trans\_tm* *e* *t*) (*trans\_tm* *e* *u*)  
| *trans\_fm* *e* (*Eq* *t* *u*) = *DBEq* (*trans\_tm* *e* *t*) (*trans\_tm* *e* *u*)  
| *trans\_fm* *e* (*Disj* *A* *B*) = *DBDisj* (*trans\_fm* *e* *A*) (*trans\_fm* *e* *B*)  
| *trans\_fm* *e* (*Neg* *A*) = *DBNeg* (*trans\_fm* *e* *A*)  
|  $\text{atom } k \ \# \ e \Longrightarrow \text{trans\_fm } e \ (\text{Ex } k \ A) = \text{DBEx} \ (\text{trans\_fm } (k\#e) \ A)$   
*<proof>*

**nominal\_termination** (*eqvt*)

*<proof>*

**lemma** *fresh\_trans\_fm* [simp]:  $i \# \text{trans\_fm } e \ A \longleftrightarrow i \# A \vee i \in \text{atom } ' \text{set } e$   
*<proof>*

**abbreviation** *DBConj* :: *dbfm*  $\Rightarrow$  *dbfm*  $\Rightarrow$  *dbfm*

**where** *DBConj* *t* *u*  $\equiv$  *DBNeg* (*DBDisj* (*DBNeg* *t*) (*DBNeg* *u*))

**lemma** *trans\_fm\_Conj* [simp]:  $\text{trans\_fm } e \ (\text{Conj } A \ B) = \text{DBConj} \ (\text{trans\_fm } e \ A) \ (\text{trans\_fm } e \ B)$   
*<proof>*

**lemma** *trans\_tm\_inject* [iff]:  $(\text{trans\_tm } e \ t = \text{trans\_tm } e \ u) \longleftrightarrow t = u$   
*<proof>*

**lemma** *trans\_fm\_inject* [iff]:  $(\text{trans\_fm } e \ A = \text{trans\_fm } e \ B) \longleftrightarrow A = B$   
*<proof>*

**lemma** *trans\_fm\_perm*:

**assumes** *c*:  $\text{atom } c \ \# \ (i, j, A, B)$

**and** *t*:  $\text{trans\_fm } [i] \ A = \text{trans\_fm } [j] \ B$

**shows**  $(i \leftrightarrow c) \cdot A = (j \leftrightarrow c) \cdot B$

*<proof>*

## 2.2 Characterising the Well-Formed de Bruijn Formulas

### 2.2.1 Well-Formed Terms

**inductive** *wf\_dbtm* :: *dbtm*  $\Rightarrow$  *bool*

**where**

*Zero*: *wf\_dbtm* *DBZero*  
| *Var*: *wf\_dbtm* (*DBVar* *name*)  
| *Eats*: *wf\_dbtm* *t1*  $\Longrightarrow$  *wf\_dbtm* *t2*  $\Longrightarrow$  *wf\_dbtm* (*DBEats* *t1* *t2*)

**equivariance** *wf\_dbtm*

**inductive\_cases** *Zero\_wf\_dbtm* [*elim!*]: *wf\_dbtm* *DBZero*  
**inductive\_cases** *Var\_wf\_dbtm* [*elim!*]: *wf\_dbtm* (*DBVar* *name*)  
**inductive\_cases** *Ind\_wf\_dbtm* [*elim!*]: *wf\_dbtm* (*DBInd* *i*)  
**inductive\_cases** *Eats\_wf\_dbtm* [*elim!*]: *wf\_dbtm* (*DBEats* *t1* *t2*)

**declare** *wf\_dbtm.intros* [*intro*]

**lemma** *wf\_dbtm\_imp\_is\_tm*:

**assumes** *wf\_dbtm* *x*

**shows**  $\exists t::tm. x = trans\_tm \ [] \ t$

*<proof>*

**lemma** *wf\_dbtm\_trans\_tm*: *wf\_dbtm* (*trans\_tm*  $\ [] \ t$ )

*<proof>*

**theorem** *wf\_dbtm\_iff\_is\_tm*: *wf\_dbtm* *x*  $\longleftrightarrow$  ( $\exists t::tm. x = trans\_tm \ [] \ t$ )

*<proof>*

**nominal\_function** *abst\_dbtm* :: *name*  $\Rightarrow$  *nat*  $\Rightarrow$  *dbtm*  $\Rightarrow$  *dbtm*

**where**

*abst\_dbtm* *name* *i* *DBZero* = *DBZero*  
| *abst\_dbtm* *name* *i* (*DBVar* *name'*) = (if *name* = *name'* then *DBInd* *i* else *DBVar* *name'*)  
| *abst\_dbtm* *name* *i* (*DBInd* *j*) = *DBInd* *j*  
| *abst\_dbtm* *name* *i* (*DBEats* *t1* *t2*) = *DBEats* (*abst\_dbtm* *name* *i* *t1*) (*abst\_dbtm* *name* *i* *t2*)

*<proof>*

**nominal\_termination** (*eqvt*)

*<proof>*

**nominal\_function** *subst\_dbtm* :: *dbtm*  $\Rightarrow$  *name*  $\Rightarrow$  *dbtm*  $\Rightarrow$  *dbtm*

**where**

*subst\_dbtm* *u* *i* *DBZero* = *DBZero*  
| *subst\_dbtm* *u* *i* (*DBVar* *name*) = (if *i* = *name* then *u* else *DBVar* *name*)  
| *subst\_dbtm* *u* *i* (*DBInd* *j*) = *DBInd* *j*  
| *subst\_dbtm* *u* *i* (*DBEats* *t1* *t2*) = *DBEats* (*subst\_dbtm* *u* *i* *t1*) (*subst\_dbtm* *u* *i* *t2*)

*<proof>*

**nominal\_termination** (*eqvt*)

*<proof>*

**lemma** *fresh\_iff\_non\_subst\_dbtm*: *subst\_dbtm* *DBZero* *i* *t* = *t*  $\longleftrightarrow$  *atom* *i*  $\#$  *t*

*<proof>*

**lemma** *lookup\_append*: *lookup* (*e* @ [*i*]) *n* *j* = *abst\_dbtm* *i* (*length* *e* + *n*) (*lookup* *e* *n* *j*)

*<proof>*

**lemma** *trans\_tm\_abs*: *trans\_tm* (*e*@[*name*]) *t* = *abst\_dbtm* *name* (*length* *e*) (*trans\_tm* *e* *t*)

*<proof>*

## 2.2.2 Well-Formed Formulas

**nominal\_function** *abst\_dbfm* :: *name*  $\Rightarrow$  *nat*  $\Rightarrow$  *dbfm*  $\Rightarrow$  *dbfm*

**where**

*abst\_dbfm* *name* *i* (*DBMem* *t1* *t2*) = *DBMem* (*abst\_dbtm* *name* *i* *t1*) (*abst\_dbtm* *name* *i* *t2*)  
| *abst\_dbfm* *name* *i* (*DBEq* *t1* *t2*) = *DBEq* (*abst\_dbtm* *name* *i* *t1*) (*abst\_dbtm* *name* *i* *t2*)  
| *abst\_dbfm* *name* *i* (*DBDisj* *A1* *A2*) = *DBDisj* (*abst\_dbfm* *name* *i* *A1*) (*abst\_dbfm* *name* *i* *A2*)  
| *abst\_dbfm* *name* *i* (*DBNeg* *A*) = *DBNeg* (*abst\_dbfm* *name* *i* *A*)  
| *abst\_dbfm* *name* *i* (*DBEx* *A*) = *DBEx* (*abst\_dbfm* *name* (*i*+1) *A*)

*<proof>*

**nominal\_termination** (*eqvt*)

*<proof>*

**nominal\_function** *subst\_dbfm* :: *dbtm*  $\Rightarrow$  *name*  $\Rightarrow$  *dbfm*  $\Rightarrow$  *dbfm*

**where**

*subst\_dbfm* *u* *i* (*DBMem* *t1* *t2*) = *DBMem* (*subst\_dbtm* *u* *i* *t1*) (*subst\_dbtm* *u* *i* *t2*)  
| *subst\_dbfm* *u* *i* (*DBEq* *t1* *t2*) = *DBEq* (*subst\_dbtm* *u* *i* *t1*) (*subst\_dbtm* *u* *i* *t2*)  
| *subst\_dbfm* *u* *i* (*DBDisj* *A1* *A2*) = *DBDisj* (*subst\_dbfm* *u* *i* *A1*) (*subst\_dbfm* *u* *i* *A2*)  
| *subst\_dbfm* *u* *i* (*DBNeg* *A*) = *DBNeg* (*subst\_dbfm* *u* *i* *A*)  
| *subst\_dbfm* *u* *i* (*DBEx* *A*) = *DBEx* (*subst\_dbfm* *u* *i* *A*)

*<proof>*

**nominal\_termination** (*eqvt*)

*<proof>*

**lemma** *fresh\_iff\_non\_subst\_dbfm*: *subst\_dbfm* *DBZero* *i* *t* = *t*  $\longleftrightarrow$  *atom* *i*  $\#$  *t*

*<proof>*

## 2.3 Well formed terms and formulas (de Bruijn representation)

**inductive** *wf\_dbfm* :: *dbfm*  $\Rightarrow$  *bool*

**where**

*Mem*: *wf\_dbtm* *t1*  $\Longrightarrow$  *wf\_dbtm* *t2*  $\Longrightarrow$  *wf\_dbfm* (*DBMem* *t1* *t2*)  
| *Eq*: *wf\_dbtm* *t1*  $\Longrightarrow$  *wf\_dbtm* *t2*  $\Longrightarrow$  *wf\_dbfm* (*DBEq* *t1* *t2*)  
| *Disj*: *wf\_dbfm* *A1*  $\Longrightarrow$  *wf\_dbfm* *A2*  $\Longrightarrow$  *wf\_dbfm* (*DBDisj* *A1* *A2*)  
| *Neg*: *wf\_dbfm* *A*  $\Longrightarrow$  *wf\_dbfm* (*DBNeg* *A*)  
| *Ex*: *wf\_dbfm* *A*  $\Longrightarrow$  *wf\_dbfm* (*DBEx* (*abst\_dbfm* *name* 0 *A*))

**equivariance** *wf\_dbfm*

**lemma** *atom\_fresh\_abst\_dbtm* [*simp*]: *atom* *i*  $\#$  *abst\_dbtm* *i* *n* *t*

*<proof>*

**lemma** *atom\_fresh\_abst\_dbfm* [*simp*]: *atom* *i*  $\#$  *abst\_dbfm* *i* *n* *A*

*<proof>*

Setting up strong induction: "avoiding" for name. Necessary to allow some proofs to go through

**nominal\_inductive** *wf\_dbfm*

**avoids** *Ex*: *name*

*<proof>*

**inductive\_cases** *Mem\_wf\_dbfm* [*elim!*]: *wf\_dbfm* (*DBMem* *t1* *t2*)

**inductive\_cases** *Eq\_wf\_dbfm* [*elim!*]: *wf\_dbfm* (*DBEq* *t1* *t2*)

**inductive\_cases** *Disj\_wf\_dbfm* [elim!]:  $wf\_dbfm (DBDisj A1 A2)$   
**inductive\_cases** *Neg\_wf\_dbfm* [elim!]:  $wf\_dbfm (DBNeg A)$   
**inductive\_cases** *Ex\_wf\_dbfm* [elim!]:  $wf\_dbfm (DBEx z)$

**declare**  $wf\_dbfm.intros$  [intro]

**lemma** *trans\_fm\_abs*:  $trans\_fm (e@[name]) A = abst\_dbfm\ name\ (length\ e)\ (trans\_fm\ e\ A)$   
 ⟨proof⟩

**lemma** *abst\_trans\_fm*:  $abst\_dbfm\ name\ 0\ (trans\_fm\ []\ A) = trans\_fm\ [name]\ A$   
 ⟨proof⟩

**lemma** *abst\_trans\_fm2*:  $i \neq j \implies abst\_dbfm\ i\ (Suc\ 0)\ (trans\_fm\ [j]\ A) = trans\_fm\ [j,i]\ A$   
 ⟨proof⟩

**lemma** *wf\_dbfm\_imp\_is\_fm*:  
**assumes**  $wf\_dbfm\ x$  **shows**  $\exists A::fm.\ x = trans\_fm\ []\ A$   
 ⟨proof⟩

**lemma** *wf\_dbfm\_trans\_fm*:  $wf\_dbfm\ (trans\_fm\ []\ A)$   
 ⟨proof⟩

**lemma** *wf\_dbfm\_iff\_is\_fm*:  $wf\_dbfm\ x \longleftrightarrow (\exists A::fm.\ x = trans\_fm\ []\ A)$   
 ⟨proof⟩

**lemma** *dbtm\_abst\_ignore* [simp]:  
 $abst\_dbtm\ name\ i\ (abst\_dbtm\ name\ j\ t) = abst\_dbtm\ name\ j\ t$   
 ⟨proof⟩

**lemma** *abst\_dbtm\_fresh\_ignore* [simp]:  $atom\ name\ \# u \implies abst\_dbtm\ name\ j\ u = u$   
 ⟨proof⟩

**lemma** *dbtm\_subst\_ignore* [simp]:  
 $subst\_dbtm\ u\ name\ (abst\_dbtm\ name\ j\ t) = abst\_dbtm\ name\ j\ t$   
 ⟨proof⟩

**lemma** *dbtm\_abst\_swap\_subst*:  
 $name \neq name' \implies atom\ name' \# u \implies$   
 $subst\_dbtm\ u\ name\ (abst\_dbtm\ name'\ j\ t) = abst\_dbtm\ name'\ j\ (subst\_dbtm\ u\ name\ t)$   
 ⟨proof⟩

**lemma** *dbfm\_abst\_swap\_subst*:  
 $name \neq name' \implies atom\ name' \# u \implies$   
 $subst\_dbfm\ u\ name\ (abst\_dbfm\ name'\ j\ A) = abst\_dbfm\ name'\ j\ (subst\_dbfm\ u\ name\ A)$   
 ⟨proof⟩

**lemma** *subst\_trans\_commute* [simp]:  
 $atom\ i \# e \implies subst\_dbtm\ (trans\_tm\ e\ u)\ i\ (trans\_tm\ e\ t) = trans\_tm\ e\ (subst\ i\ u\ t)$   
 ⟨proof⟩

**lemma** *subst\_fm\_trans\_commute* [simp]:  
 $subst\_dbfm\ (trans\_tm\ []\ u)\ name\ (trans\_fm\ []\ A) = trans\_fm\ []\ (A\ (name::=u))$   
 ⟨proof⟩

**lemma** *subst\_fm\_trans\_commute\_eq*:  
 $du = trans\_tm\ []\ u \implies subst\_dbfm\ du\ i\ (trans\_fm\ []\ A) = trans\_fm\ []\ (A\ (i::=u))$   
 ⟨proof⟩

## 2.4 Quotations

```

fun HTuple :: nat ⇒ tm where
  HTuple 0 = HPair Zero Zero
  | HTuple (Suc k) = HPair Zero (HTuple k)

```

```

lemma fresh_HTuple [simp]: x # HTuple n
  ⟨proof⟩

```

```

lemma HTuple_eqvt[eqvt]: (p · HTuple n) = HTuple (p · n)
  ⟨proof⟩

```

### 2.4.1 Quotations of de Bruijn terms

```

definition nat_of_name :: name ⇒ nat
  where nat_of_name x = nat_of (atom x)

```

```

lemma nat_of_name_inject [simp]: nat_of_name n1 = nat_of_name n2 ⟷ n1 = n2
  ⟨proof⟩

```

```

definition name_of_nat :: nat ⇒ name
  where name_of_nat n ≡ Abs_name (Atom (Sort "SyntaxN.name" [])) n

```

```

lemma nat_of_name_Abs_eq [simp]: nat_of_name (Abs_name (Atom (Sort "SyntaxN.name" [])) n)
  = n
  ⟨proof⟩

```

```

lemma nat_of_name_name_eq [simp]: nat_of_name (name_of_nat n) = n
  ⟨proof⟩

```

```

lemma name_of_nat_nat_of_name [simp]: name_of_nat (nat_of_name i) = i
  ⟨proof⟩

```

```

lemma HPair_neq_ORD_OF [simp]: HPair x y ≠ ORD_OF i
  ⟨proof⟩

```

Infinite support, so we cannot use nominal primrec.

```

function quot_dbtm :: dbtm ⇒ tm
  where
    quot_dbtm DBZero = Zero
  | quot_dbtm (DBVar name) = ORD_OF (Suc (nat_of_name name))
  | quot_dbtm (DBInd k) = HPair (HTuple 6) (ORD_OF k)
  | quot_dbtm (DBEats t u) = HPair (HTuple 1) (HPair (quot_dbtm t) (quot_dbtm u))
  ⟨proof⟩

```

```

termination
  ⟨proof⟩

```

### 2.4.2 Quotations of de Bruijn formulas

Infinite support, so we cannot use nominal primrec.

```

function quot_dbfm :: dbfm ⇒ tm
  where
    quot_dbfm (DBMem t u) = HPair (HTuple 0) (HPair (quot_dbtm t) (quot_dbtm u))
  | quot_dbfm (DBEq t u) = HPair (HTuple 2) (HPair (quot_dbtm t) (quot_dbtm u))
  | quot_dbfm (DBDisj A B) = HPair (HTuple 3) (HPair (quot_dbfm A) (quot_dbfm B))
  | quot_dbfm (DBNeg A) = HPair (HTuple 4) (quot_dbfm A)
  | quot_dbfm (DBEx A) = HPair (HTuple 5) (quot_dbfm A)

```

*<proof>*

**termination**

*<proof>*

**lemma** *HTuple\_minus\_1*:  $n > 0 \implies \text{HTuple } n = \text{HPair } \text{Zero } (\text{HTuple } (n - 1))$

*<proof>*

**lemmas** *HTS* = *HTuple\_minus\_1* *HTuple.simps* — for freeness reasoning on codes

**class** *quot* =

**fixes** *quot* :: 'a  $\Rightarrow$  tm ( $\langle \_ \rangle$ )

**instantiation** *tm* :: *quot*

**begin**

**definition** *quot\_tm* :: *tm*  $\Rightarrow$  *tm*

**where** *quot\_tm* *t* = *quot\_dbtm* (*trans\_tm* [] *t*)

**instance** *<proof>*

**end**

**lemma** *quot\_dbtm\_fresh* [*simp*]:  $s \# (\text{quot\_dbtm } t)$

*<proof>*

**lemma** *quot\_tm\_fresh* [*simp*]: **fixes** *t::tm* **shows**  $s \# \langle t \rangle$

*<proof>*

**lemma** *quot\_Zero* [*simp*]:  $\langle \text{Zero} \rangle = \text{Zero}$

*<proof>*

**lemma** *quot\_Var*:  $\langle \text{Var } x \rangle = \text{SUCC } (\text{ORD\_OF } (\text{nat\_of\_name } x))$

*<proof>*

**lemma** *quot\_Eats*:  $\langle \text{Eats } x \ y \rangle = \text{HPair } (\text{HTuple } 1) (\text{HPair } \langle x \rangle \langle y \rangle)$

*<proof>*

**instantiation** *fm* :: *quot*

**begin**

**definition** *quot\_fm* :: *fm*  $\Rightarrow$  *tm*

**where** *quot\_fm* *A* = *quot\_dbfm* (*trans\_fm* [] *A*)

**instance** *<proof>*

**end**

**lemma** *quot\_dbfm\_fresh* [*simp*]:  $s \# (\text{quot\_dbfm } A)$

*<proof>*

**lemma** *quot\_fm\_fresh* [*simp*]: **fixes** *A::fm* **shows**  $s \# \langle A \rangle$

*<proof>*

**lemma** *quot\_fm\_permute* [*simp*]: **fixes** *A::fm* **shows**  $p \cdot \langle A \rangle = \langle A \rangle$

*<proof>*

**lemma** *quot\_Mem*:  $\langle x \text{ IN } y \rangle = \text{HPair } (\text{HTuple } 0) (\text{HPair } \langle x \rangle \langle y \rangle)$

*<proof>*

**lemma** *quot\_Eq*:  $\langle x \text{ EQ } y \rangle = \text{HPair } (\text{HTuple } 2) (\text{HPair } \langle x \rangle \langle y \rangle)$

*<proof>*

**lemma** *quot\_Disj*: « $A \text{ OR } B$ » = *HPair* (*HTuple* 3) (*HPair* (« $A$ ») (« $B$ »))  
*<proof>*

**lemma** *quot\_Neg*: « $\text{Neg } A$ » = *HPair* (*HTuple* 4) (« $A$ »)  
*<proof>*

**lemma** *quot\_Ex*: « $\text{Ex } i \ A$ » = *HPair* (*HTuple* 5) (*quot\_dbfm* (*trans\_fm* [i] A))  
*<proof>*

**lemmas** *quot\_simps* = *quot\_Var quot\_Eats quot\_Eq quot\_Mem quot\_Disj quot\_Neg quot\_Ex*

## 2.5 Definitions Involving Coding

**abbreviation** *Q\_Eats* ::  $tm \Rightarrow tm \Rightarrow tm$   
**where** *Q\_Eats*  $t \ u \equiv \text{HPair } (\text{HTuple } (\text{Suc } 0)) (\text{HPair } t \ u)$

**abbreviation** *Q\_Succ* ::  $tm \Rightarrow tm$   
**where** *Q\_Succ*  $t \equiv \text{Q\_Eats } t \ t$

**lemma** *quot\_Succ*: « $\text{SUCC } x$ » = *Q\_Succ* « $x$ »  
*<proof>*

**abbreviation** *Q\_HPPair* ::  $tm \Rightarrow tm \Rightarrow tm$   
**where** *Q\_HPPair*  $t \ u \equiv$   
    *Q\_Eats* (*Q\_Eats* Zero (*Q\_Eats* (*Q\_Eats* Zero  $u$ )  $t$ ))  
    (*Q\_Eats* (*Q\_Eats* Zero  $t$ )  $t$ )

**abbreviation** *Q\_Mem* ::  $tm \Rightarrow tm \Rightarrow tm$   
**where** *Q\_Mem*  $t \ u \equiv \text{HPair } (\text{HTuple } 0) (\text{HPair } t \ u)$

**abbreviation** *Q\_Eq* ::  $tm \Rightarrow tm \Rightarrow tm$   
**where** *Q\_Eq*  $t \ u \equiv \text{HPair } (\text{HTuple } 2) (\text{HPair } t \ u)$

**abbreviation** *Q\_Disj* ::  $tm \Rightarrow tm \Rightarrow tm$   
**where** *Q\_Disj*  $t \ u \equiv \text{HPair } (\text{HTuple } 3) (\text{HPair } t \ u)$

**abbreviation** *Q\_Neg* ::  $tm \Rightarrow tm$   
**where** *Q\_Neg*  $t \equiv \text{HPair } (\text{HTuple } 4) \ t$

**abbreviation** *Q\_Conj* ::  $tm \Rightarrow tm \Rightarrow tm$   
**where** *Q\_Conj*  $t \ u \equiv \text{Q\_Neg } (\text{Q\_Disj } (\text{Q\_Neg } t) (\text{Q\_Neg } u))$

**abbreviation** *Q\_Imp* ::  $tm \Rightarrow tm \Rightarrow tm$   
**where** *Q\_Imp*  $t \ u \equiv \text{Q\_Disj } (\text{Q\_Neg } t) \ u$

**abbreviation** *Q\_Ex* ::  $tm \Rightarrow tm$   
**where** *Q\_Ex*  $t \equiv \text{HPair } (\text{HTuple } 5) \ t$

**abbreviation** *Q\_All* ::  $tm \Rightarrow tm$   
**where** *Q\_All*  $t \equiv \text{Q\_Neg } (\text{Q\_Ex } (\text{Q\_Neg } t))$

**lemma** *quot\_subst\_eq*: « $A(i::=t)$ » = *quot\_dbfm* (*subst\_dbfm* (*trans\_tm* []  $t$ )  $i$  (*trans\_fm* [] A))  
*<proof>*

**lemma** *Q\_Succ\_cong*:  $H \vdash x \text{ EQ } x' \implies H \vdash \text{Q\_Succ } x \text{ EQ } \text{Q\_Succ } x'$   
*<proof>*

## 2.5.1 The set $\Gamma$ of Definition 1.1, constant terms used for coding

**inductive** *coding\_tm* :: *tm*  $\Rightarrow$  *bool*  
**where**  
*Ord*:  $\exists i. x = \text{ORD\_OF } i \implies \text{coding\_tm } x$   
| *HPair*:  $\text{coding\_tm } x \implies \text{coding\_tm } y \implies \text{coding\_tm } (\text{HPair } x \ y)$

**declare** *coding\_tm.intros* [*intro*]

**lemma** *coding\_tm\_Zero* [*intro*]: *coding\_tm Zero*  
 $\langle \text{proof} \rangle$

**lemma** *coding\_tm\_HTuple* [*intro*]: *coding\_tm (HTuple k)*  
 $\langle \text{proof} \rangle$

**inductive\_simps** *coding\_tm\_HPpair* [*simp*]: *coding\_tm (HPair x y)*

**lemma** *quot\_dbtm\_coding* [*simp*]: *coding\_tm (quot\_dbtm t)*  
 $\langle \text{proof} \rangle$

**lemma** *quot\_dbfm\_coding* [*simp*]: *coding\_tm (quot\_dbfm fm)*  
 $\langle \text{proof} \rangle$

**lemma** *quot\_fm\_coding*: **fixes** *A::fm* **shows** *coding\_tm «A»*  
 $\langle \text{proof} \rangle$

## 2.6 V-Coding for terms and formulas, for the Second Theorem

Infinite support, so we cannot use nominal primrec.

**function** *vquot\_dbtm* :: *name set*  $\Rightarrow$  *dbtm*  $\Rightarrow$  *tm*  
**where**  
*vquot\_dbtm V DBZero* = *Zero*  
| *vquot\_dbtm V (DBVar name)* = (*if name*  $\in$  *V* *then Var name*  
                                  *else ORD\_OF (Suc (nat\_of\_name name))*)  
| *vquot\_dbtm V (DBInd k)* = *HPair (HTuple 6) (ORD\_OF k)*  
| *vquot\_dbtm V (DBEats t u)* = *HPair (HTuple 1) (HPair (vquot\_dbtm V t) (vquot\_dbtm V u))*  
 $\langle \text{proof} \rangle$

**termination**  
 $\langle \text{proof} \rangle$

**lemma** *fresh\_vquot\_dbtm* [*simp*]:  $i \# \text{vquot\_dbtm } V \ \text{tm} \longleftrightarrow i \# \ \text{tm} \vee i \notin \text{atom } \text{‘ } V$   
 $\langle \text{proof} \rangle$

Infinite support, so we cannot use nominal primrec.

**function** *vquot\_dbfm* :: *name set*  $\Rightarrow$  *dbfm*  $\Rightarrow$  *tm*  
**where**  
*vquot\_dbfm V (DBMem t u)* = *HPair (HTuple 0) (HPair (vquot\_dbtm V t) (vquot\_dbtm V u))*  
| *vquot\_dbfm V (DBEq t u)* = *HPair (HTuple 2) (HPair (vquot\_dbtm V t) (vquot\_dbtm V u))*  
| *vquot\_dbfm V (DBDisj A B)* = *HPair (HTuple 3) (HPair (vquot\_dbfm V A) (vquot\_dbfm V B))*  
| *vquot\_dbfm V (DBNeg A)* = *HPair (HTuple 4) (vquot\_dbfm V A)*  
| *vquot\_dbfm V (DBEx A)* = *HPair (HTuple 5) (vquot\_dbfm V A)*  
 $\langle \text{proof} \rangle$

**termination**  
 $\langle \text{proof} \rangle$

```

lemma fresh_vquot_dbfm [simp]:  $i \# vquot\_dbfm\ V\ fm \longleftrightarrow i \# fm \vee i \notin atom\ 'V$ 
  <proof>

class vquot =
  fixes vquot :: 'a  $\Rightarrow$  name set  $\Rightarrow$  tm (⟨ $\_$ ⟩ [0,1000]1000)

instantiation tm :: vquot
begin
  definition vquot_tm :: tm  $\Rightarrow$  name set  $\Rightarrow$  tm
    where vquot_tm t V = vquot_dbtm V (trans_tm [] t)
  instance <proof>
end

lemma vquot_dbtm_empty [simp]: vquot_dbtm {} t = quot_dbtm t
  <proof>

lemma vquot_tm_empty [simp]: fixes t::tm shows [t]{ } = «t»
  <proof>

lemma vquot_dbtm_eq: atom ' V  $\cap$  supp t = atom ' W  $\cap$  supp t  $\Longrightarrow$  vquot_dbtm V t = vquot_dbtm
  W t
  <proof>

instantiation fm :: vquot
begin
  definition vquot_fm :: fm  $\Rightarrow$  name set  $\Rightarrow$  tm
    where vquot_fm A V = vquot_dbfm V (trans_fm [] A)
  instance <proof>
end

lemma vquot_fm_fresh [simp]: fixes A::fm shows  $i \# [A]\ V \longleftrightarrow i \# A \vee i \notin atom\ 'V$ 
  <proof>

lemma vquot_dbfm_empty [simp]: vquot_dbfm {} A = quot_dbfm A
  <proof>

lemma vquot_fm_empty [simp]: fixes A::fm shows [A]{ } = «A»
  <proof>

lemma vquot_dbfm_eq: atom ' V  $\cap$  supp A = atom ' W  $\cap$  supp A  $\Longrightarrow$  vquot_dbfm V A = vquot_dbfm
  W A
  <proof>

lemma vquot_fm_insert:
  fixes A::fm shows atom i  $\notin$  supp A  $\Longrightarrow$  [A](insert i V) = [A]V
  <proof>

declare HTuple.simps [simp del]

end

```

# Chapter 3

## Basic Predicates

```
theory Predicates
imports SyntaxN
begin
```

### 3.1 The Subset Relation

```
nominal_function Subset :: tm  $\Rightarrow$  tm  $\Rightarrow$  fm (infixr <SUBS> 150)
  where atom z  $\#$  (t, u)  $\Longrightarrow$  t SUBS u = All2 z t ((Var z) IN u)
  <proof>
```

```
nominal_termination (eqvt)
  <proof>
```

```
declare Subset.simps [simp del]
```

```
lemma Subset_fresh_iff [simp]: a  $\#$  t SUBS u  $\longleftrightarrow$  a  $\#$  t  $\wedge$  a  $\#$  u
  <proof>
```

```
lemma subst_fm_Subset [simp]: (t SUBS u)(i::=x) = (subst i x t) SUBS (subst i x u)
  <proof>
```

```
lemma Subset_I:
  assumes insert ((Var i) IN t) H  $\vdash$  (Var i) IN u atom i  $\#$  (t,u)  $\forall B \in H. atom i \# B$ 
  shows H  $\vdash$  t SUBS u
  <proof>
```

```
lemma Subset_D:
  assumes major: H  $\vdash$  t SUBS u and minor: H  $\vdash$  a IN t shows H  $\vdash$  a IN u
  <proof>
```

```
lemma Subset_E: H  $\vdash$  t SUBS u  $\Longrightarrow$  H  $\vdash$  a IN t  $\Longrightarrow$  insert (a IN u) H  $\vdash$  A  $\Longrightarrow$  H  $\vdash$  A
  <proof>
```

```
lemma Subset_cong: H  $\vdash$  t EQ t'  $\Longrightarrow$  H  $\vdash$  u EQ u'  $\Longrightarrow$  H  $\vdash$  t SUBS u IFF t' SUBS u'
  <proof>
```

```
lemma Set_MP: x SUBS y  $\in$  H  $\Longrightarrow$  z IN x  $\in$  H  $\Longrightarrow$  insert (z IN y) H  $\vdash$  A  $\Longrightarrow$  H  $\vdash$  A
  <proof>
```

```
lemma Zero_Subset_I [intro!]: H  $\vdash$  Zero SUBS t
  <proof>
```

**lemma** *Zero\_SubsetE*:  $H \vdash A \implies \text{insert } (\text{Zero SUBS } X) H \vdash A$   
 ⟨proof⟩

**lemma** *Subset\_Zero\_D*:  
**assumes**  $H \vdash t \text{ SUBS } \text{Zero}$  **shows**  $H \vdash t \text{ EQ } \text{Zero}$   
 ⟨proof⟩

**lemma** *Subset\_refl*:  $H \vdash t \text{ SUBS } t$   
 ⟨proof⟩

**lemma** *Eats\_Subset\_Iff*:  $H \vdash \text{Eats } x \ y \ \text{SUBS } z \text{ IFF } (x \ \text{SUBS } z) \ \text{AND } (y \ \text{IN } z)$   
 ⟨proof⟩

**lemma** *Eats\_Subset\_I* [intro!]:  $H \vdash x \ \text{SUBS } z \implies H \vdash y \ \text{IN } z \implies H \vdash \text{Eats } x \ y \ \text{SUBS } z$   
 ⟨proof⟩

**lemma** *Eats\_Subset\_E* [intro!]:  
 $\text{insert } (x \ \text{SUBS } z) (\text{insert } (y \ \text{IN } z) H) \vdash C \implies \text{insert } (\text{Eats } x \ y \ \text{SUBS } z) H \vdash C$   
 ⟨proof⟩

A surprising proof: a consequence of  $?H \vdash \text{Eats } ?x \ ?y \ \text{SUBS } ?z \text{ IFF } ?x \ \text{SUBS } ?z \ \text{AND } ?y \ \text{IN } ?z$  and reflexivity!

**lemma** *Subset\_Eats\_I* [intro!]:  $H \vdash x \ \text{SUBS } \text{Eats } x \ y$   
 ⟨proof⟩

**lemma** *SUCC\_Subset\_I* [intro!]:  $H \vdash x \ \text{SUBS } z \implies H \vdash x \ \text{IN } z \implies H \vdash \text{SUCC } x \ \text{SUBS } z$   
 ⟨proof⟩

**lemma** *SUCC\_Subset\_E* [intro!]:  
 $\text{insert } (x \ \text{SUBS } z) (\text{insert } (x \ \text{IN } z) H) \vdash C \implies \text{insert } (\text{SUCC } x \ \text{SUBS } z) H \vdash C$   
 ⟨proof⟩

**lemma** *Subset\_trans0*:  $\{ a \ \text{SUBS } b, b \ \text{SUBS } c \} \vdash a \ \text{SUBS } c$   
 ⟨proof⟩

**lemma** *Subset\_trans*:  $H \vdash a \ \text{SUBS } b \implies H \vdash b \ \text{SUBS } c \implies H \vdash a \ \text{SUBS } c$   
 ⟨proof⟩

**lemma** *Subset\_SUCC*:  $H \vdash a \ \text{SUBS } (\text{SUCC } a)$   
 ⟨proof⟩

**lemma** *All2\_Subset\_lemma*:  $\text{atom } l \ \sharp (k', k) \implies \{P\} \vdash P' \implies \{\text{All2 } l \ k \ P, k' \ \text{SUBS } k\} \vdash \text{All2 } l \ k' \ P'$   
 ⟨proof⟩

**lemma** *All2\_Subset*:  $\llbracket H \vdash \text{All2 } l \ k \ P; H \vdash k' \ \text{SUBS } k; \{P\} \vdash P'; \text{atom } l \ \sharp (k', k) \rrbracket \implies H \vdash \text{All2 } l \ k' \ P'$   
 ⟨proof⟩

## 3.2 Extensionality

**lemma** *Extensionality*:  $H \vdash x \ \text{EQ } y \text{ IFF } x \ \text{SUBS } y \ \text{AND } y \ \text{SUBS } x$   
 ⟨proof⟩

**lemma** *Equality\_I*:  $H \vdash y \ \text{SUBS } x \implies H \vdash x \ \text{SUBS } y \implies H \vdash x \ \text{EQ } y$   
 ⟨proof⟩

**lemma** *EQ\_imp\_SUBS*:  $\text{insert } (t \ \text{EQ } u) H \vdash (t \ \text{SUBS } u)$   
 ⟨proof⟩

**lemma** *EQ\_imp\_SUBS2*:  $\text{insert } (u \text{ EQ } t) H \vdash (t \text{ SUBS } u)$   
 ⟨proof⟩

**lemma** *Equality\_E*:  $\text{insert } (t \text{ SUBS } u) (\text{insert } (u \text{ SUBS } t) H) \vdash A \implies \text{insert } (t \text{ EQ } u) H \vdash A$   
 ⟨proof⟩

### 3.3 The Disjointness Relation

The following predicate is defined in order to prove Lemma 2.3, Foundation

**nominal\_function** *Disjoint* ::  $tm \Rightarrow tm \Rightarrow fm$   
**where**  $\text{atom } z \# (t, u) \implies \text{Disjoint } t \ u = \text{All2 } z \ t \ (\text{Neg } ((\text{Var } z) \text{ IN } u))$   
 ⟨proof⟩

**nominal\_termination** (*eqt*)  
 ⟨proof⟩

**declare** *Disjoint.simps* [*simp del*]

**lemma** *Disjoint\_fresh\_iff* [*simp*]:  $a \# \text{Disjoint } t \ u \longleftrightarrow a \# t \wedge a \# u$   
 ⟨proof⟩

**lemma** *subst\_fm\_Disjoint* [*simp*]:  
 $(\text{Disjoint } t \ u)(i::=x) = \text{Disjoint } (\text{subst } i \ x \ t) (\text{subst } i \ x \ u)$   
 ⟨proof⟩

**lemma** *Disjoint\_cong*:  $H \vdash t \text{ EQ } t' \implies H \vdash u \text{ EQ } u' \implies H \vdash \text{Disjoint } t \ u \text{ IFF } \text{Disjoint } t' \ u'$   
 ⟨proof⟩

**lemma** *Disjoint\_I*:  
**assumes**  $\text{insert } ((\text{Var } i) \text{ IN } t) (\text{insert } ((\text{Var } i) \text{ IN } u) H) \vdash \text{Fls}$   
 $\text{atom } i \# (t, u) \ \forall B \in H. \text{atom } i \# B$   
**shows**  $H \vdash \text{Disjoint } t \ u$   
 ⟨proof⟩

**lemma** *Disjoint\_E*:  
**assumes** *major*:  $H \vdash \text{Disjoint } t \ u$  **and** *minor*:  $H \vdash a \text{ IN } t \ H \vdash a \text{ IN } u$  **shows**  $H \vdash A$   
 ⟨proof⟩

**lemma** *Disjoint\_commute*:  $\{ \text{Disjoint } t \ u \} \vdash \text{Disjoint } u \ t$   
 ⟨proof⟩

**lemma** *Disjoint\_commute\_I*:  $H \vdash \text{Disjoint } t \ u \implies H \vdash \text{Disjoint } u \ t$   
 ⟨proof⟩

**lemma** *Disjoint\_commute\_D*:  $\text{insert } (\text{Disjoint } t \ u) H \vdash A \implies \text{insert } (\text{Disjoint } u \ t) H \vdash A$   
 ⟨proof⟩

**lemma** *Zero\_Disjoint\_I1* [*iff*]:  $H \vdash \text{Disjoint } \text{Zero } t$   
 ⟨proof⟩

**lemma** *Zero\_Disjoint\_I2* [*iff*]:  $H \vdash \text{Disjoint } t \ \text{Zero}$   
 ⟨proof⟩

**lemma** *Disjoint\_Eats\_D1*:  $\{ \text{Disjoint } (\text{Eats } x \ y) \ z \} \vdash \text{Disjoint } x \ z$   
 ⟨proof⟩

**lemma** *Disjoint\_Eats\_D2*:  $\{ \text{Disjoint } (\text{Eats } x \ y) \ z \} \vdash \text{Neg}(y \text{ IN } z)$

*<proof>*

**lemma** *Disjoint\_Eats\_E*:

*insert (Disjoint x z) (insert (Neg(y IN z)) H) ⊢ A ⇒ insert (Disjoint (Eats x y) z) H ⊢ A*  
*<proof>*

**lemma** *Disjoint\_Eats\_E2*:

*insert (Disjoint z x) (insert (Neg(y IN z)) H) ⊢ A ⇒ insert (Disjoint z (Eats x y)) H ⊢ A*  
*<proof>*

**lemma** *Disjoint\_Eats\_Imp*: { *Disjoint x z, Neg(y IN z)* } ⊢ *Disjoint (Eats x y) z*

*<proof>*

**lemma** *Disjoint\_Eats\_I* [intro!]: *H ⊢ Disjoint x z ⇒ insert (y IN z) H ⊢ Fls ⇒ H ⊢ Disjoint (Eats x y) z*

*<proof>*

**lemma** *Disjoint\_Eats\_I2* [intro!]: *H ⊢ Disjoint z x ⇒ insert (y IN z) H ⊢ Fls ⇒ H ⊢ Disjoint z (Eats x y)*

*<proof>*

### 3.4 The Foundation Theorem

**lemma** *Foundation\_lemma*:

**assumes** *i: atom i # z*

**shows** { *All2 i z (Neg (Disjoint (Var i) z))* } ⊢ *Neg (Var i IN z) AND Disjoint (Var i) z*

*<proof>*

**theorem** *Foundation*: *atom i # z ⇒ {} ⊢ All2 i z (Neg (Disjoint (Var i) z)) IMP z EQ Zero*

*<proof>*

**lemma** *Mem\_Neg\_refl*: { } ⊢ *Neg (x IN x)*

*<proof>*

**lemma** *Mem\_refl\_E* [intro!]: *insert (x IN x) H ⊢ A*

*<proof>*

**lemma** *Mem\_non\_refl*: **assumes** *H ⊢ x IN x* **shows** *H ⊢ A*

*<proof>*

**lemma** *Mem\_Neg\_sym*: { *x IN y, y IN x* } ⊢ *Fls*

*<proof>*

**lemma** *Mem\_not\_sym*: *insert (x IN y) (insert (y IN x) H) ⊢ A*

*<proof>*

### 3.5 The Ordinal Property

**nominal\_function** *OrdP* :: *tm ⇒ fm*

**where**  $\llbracket \text{atom } y \# (x, z); \text{atom } z \# x \rrbracket \Longrightarrow$

*OrdP x = All2 y x ((Var y) SUBS x AND All2 z (Var y) ((Var z) SUBS (Var y)))*

*<proof>*

**nominal\_termination** (*eqvt*)

*<proof>*

**lemma**

**shows** *OrdP\_fresh\_iff* [simp]:  $a \# \text{OrdP } x \longleftrightarrow a \# x$  (is ?thesis1)  
 <proof>

**lemma** *subst\_fm\_OrdP* [simp]:  $(\text{OrdP } t)(i::=x) = \text{OrdP } (\text{subst } i \ x \ t)$   
 <proof>

**lemma** *OrdP\_cong*:  $H \vdash x \text{ EQ } x' \implies H \vdash \text{OrdP } x \text{ IFF } \text{OrdP } x'$   
 <proof>

**lemma** *OrdP\_Mem\_lemma*:

**assumes**  $z: \text{atom } z \# (k,l)$  **and**  $l: \text{insert } (\text{OrdP } k) \ H \vdash l \text{ IN } k$

**shows**  $\text{insert } (\text{OrdP } k) \ H \vdash l \text{ SUBS } k$  **AND**  $\text{All2 } z \ l \ (\text{Var } z \ \text{SUBS } l)$

<proof>

**lemma** *OrdP\_Mem\_E*:

**assumes**  $\text{atom } z \# (k,l)$

$\text{insert } (\text{OrdP } k) \ H \vdash l \text{ IN } k$

$\text{insert } (l \ \text{SUBS } k) \ (\text{insert } (\text{All2 } z \ l \ (\text{Var } z \ \text{SUBS } l)) \ H) \vdash A$

**shows**  $\text{insert } (\text{OrdP } k) \ H \vdash A$

<proof>

**lemma** *OrdP\_Mem\_imp\_Subset*:

**assumes**  $k: H \vdash k \text{ IN } l$  **and**  $l: H \vdash \text{OrdP } l$  **shows**  $H \vdash k \text{ SUBS } l$

<proof>

**lemma** *SUCC\_Subset\_Ord\_lemma*:  $\{ k' \text{ IN } k, \text{OrdP } k \} \vdash \text{SUCC } k' \ \text{SUBS } k$

<proof>

**lemma** *SUCC\_Subset\_Ord*:  $H \vdash k' \text{ IN } k \implies H \vdash \text{OrdP } k \implies H \vdash \text{SUCC } k' \ \text{SUBS } k$

<proof>

**lemma** *OrdP\_Trans\_lemma*:  $\{ \text{OrdP } k, i \text{ IN } j, j \text{ IN } k \} \vdash i \text{ IN } k$

<proof>

**lemma** *OrdP\_Trans*:  $H \vdash \text{OrdP } k \implies H \vdash i \text{ IN } j \implies H \vdash j \text{ IN } k \implies H \vdash i \text{ IN } k$

<proof>

**lemma** *Ord\_IN\_Ord0*:

**assumes**  $l: H \vdash l \text{ IN } k$

**shows**  $\text{insert } (\text{OrdP } k) \ H \vdash \text{OrdP } l$

<proof>

**lemma** *Ord\_IN\_Ord*:  $H \vdash l \text{ IN } k \implies H \vdash \text{OrdP } k \implies H \vdash \text{OrdP } l$

<proof>

**lemma** *OrdP\_I*:

**assumes**  $\text{insert } (\text{Var } y \ \text{IN } x) \ H \vdash (\text{Var } y) \ \text{SUBS } x$

**and**  $\text{insert } (\text{Var } z \ \text{IN } \text{Var } y) \ (\text{insert } (\text{Var } y \ \text{IN } x) \ H) \vdash (\text{Var } z) \ \text{SUBS } (\text{Var } y)$

**and**  $\text{atom } y \# (x, z) \ \forall B \in H. \text{atom } y \# B \ \text{atom } z \# x \ \forall B \in H. \text{atom } z \# B$

**shows**  $H \vdash \text{OrdP } x$

<proof>

**lemma** *OrdP\_Zero* [simp]:  $H \vdash \text{OrdP } \text{Zero}$

<proof>

**lemma** *OrdP\_SUCC\_I0*:  $\{ \text{OrdP } k \} \vdash \text{OrdP } (\text{SUCC } k)$

<proof>

**lemma** *OrdP\_SUCC\_I*:  $H \vdash \text{OrdP } k \implies H \vdash \text{OrdP } (\text{SUCC } k)$   
 ⟨proof⟩

**lemma** *Zero\_In\_OrdP*:  $\{ \text{OrdP } x \} \vdash x \text{ EQ Zero OR Zero IN } x$   
 ⟨proof⟩

**lemma** *OrdP\_HPaiRE*:  $\text{insert } (\text{OrdP } (\text{HPair } x y)) H \vdash A$   
 ⟨proof⟩

**lemmas** *OrdP\_HPaiREH* = *OrdP\_HPaiRE OrdP\_HPaiRE [THEN rotate2] OrdP\_HPaiRE [THEN rotate3] OrdP\_HPaiRE [THEN rotate4] OrdP\_HPaiRE [THEN rotate5] OrdP\_HPaiRE [THEN rotate6] OrdP\_HPaiRE [THEN rotate7] OrdP\_HPaiRE [THEN rotate8] OrdP\_HPaiRE [THEN rotate9] OrdP\_HPaiRE [THEN rotate10]*  
**declare** *OrdP\_HPaiREH* [intro!]

**lemma** *Zero\_Eq\_HPaiRE*:  $\text{insert } (\text{Zero EQ HPair } x y) H \vdash A$   
 ⟨proof⟩

**lemmas** *Zero\_Eq\_HPaiREH* = *Zero\_Eq\_HPaiRE Zero\_Eq\_HPaiRE [THEN rotate2] Zero\_Eq\_HPaiRE [THEN rotate3] Zero\_Eq\_HPaiRE [THEN rotate4] Zero\_Eq\_HPaiRE [THEN rotate5] Zero\_Eq\_HPaiRE [THEN rotate6] Zero\_Eq\_HPaiRE [THEN rotate7] Zero\_Eq\_HPaiRE [THEN rotate8] Zero\_Eq\_HPaiRE [THEN rotate9] Zero\_Eq\_HPaiRE [THEN rotate10]*  
**declare** *Zero\_Eq\_HPaiREH* [intro!]

**lemma** *HPair\_Eq\_ZeroE*:  $\text{insert } (\text{HPair } x y \text{ EQ Zero}) H \vdash A$   
 ⟨proof⟩

**lemmas** *HPair\_Eq\_ZeroEH* = *HPair\_Eq\_ZeroE HPair\_Eq\_ZeroE [THEN rotate2] HPair\_Eq\_ZeroE [THEN rotate3] HPair\_Eq\_ZeroE [THEN rotate4] HPair\_Eq\_ZeroE [THEN rotate5] HPair\_Eq\_ZeroE [THEN rotate6] HPair\_Eq\_ZeroE [THEN rotate7] HPair\_Eq\_ZeroE [THEN rotate8] HPair\_Eq\_ZeroE [THEN rotate9] HPair\_Eq\_ZeroE [THEN rotate10]*  
**declare** *HPair\_Eq\_ZeroEH* [intro!]

### 3.6 Induction on Ordinals

**lemma** *OrdInd\_lemma*:  
**assumes**  $j$ : *atom* ( $j::\text{name}$ )  $\#$  ( $i, A$ )  
**shows**  $\{ \text{OrdP } (\text{Var } i) \} \vdash (\text{All } i (\text{OrdP } (\text{Var } i) \text{ IMP } ((\text{All2 } j (\text{Var } i) (A(i::= \text{Var } j))) \text{ IMP } A))) \text{ IMP } A$   
 ⟨proof⟩

**lemma** *OrdInd*:  
**assumes**  $j$ : *atom* ( $j::\text{name}$ )  $\#$  ( $i, A$ )  
**and**  $x$ :  $H \vdash \text{OrdP } (\text{Var } i)$  **and** *step*:  $H \vdash \text{All } i (\text{OrdP } (\text{Var } i) \text{ IMP } (\text{All2 } j (\text{Var } i) (A(i::= \text{Var } j))) \text{ IMP } A)$   
**shows**  $H \vdash A$   
 ⟨proof⟩

**lemma** *OrdIndH*:  
**assumes** *atom* ( $j::\text{name}$ )  $\#$  ( $i, A$ )  
**and**  $H \vdash \text{All } i (\text{OrdP } (\text{Var } i) \text{ IMP } (\text{All2 } j (\text{Var } i) (A(i::= \text{Var } j))) \text{ IMP } A)$   
**shows**  $\text{insert } (\text{OrdP } (\text{Var } i)) H \vdash A$   
 ⟨proof⟩

### 3.7 Linearity of Ordinals

**lemma** *OrdP\_linear\_lemma*:

**assumes**  $j$ :  $\text{atom } j \# i$   
**shows**  $\{ \text{OrdP } ( \text{Var } i ) \} \vdash \text{All } j ( \text{OrdP } ( \text{Var } j ) \text{ IMP } ( \text{Var } i \text{ IN } \text{Var } j \text{ OR } \text{Var } i \text{ EQ } \text{Var } j \text{ OR } \text{Var } j \text{ IN } \text{Var } i ) )$   
**(is**  $\_ \vdash ?\text{scheme}$ )  
 $\langle \text{proof} \rangle$

**lemma**  $\text{OrdP\_linear\_imp}$ :  $\{ \} \vdash \text{OrdP } x \text{ IMP } \text{OrdP } y \text{ IMP } x \text{ IN } y \text{ OR } x \text{ EQ } y \text{ OR } y \text{ IN } x$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{OrdP\_linear}$ :  
**assumes**  $H \vdash \text{OrdP } x \ H \vdash \text{OrdP } y$   
 $\text{insert } (x \text{ IN } y) \ H \vdash A \ \text{insert } (x \text{ EQ } y) \ H \vdash A \ \text{insert } (y \text{ IN } x) \ H \vdash A$   
**shows**  $H \vdash A$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{Zero\_In\_SUCC}$ :  $\{ \text{OrdP } k \} \vdash \text{Zero IN SUCC } k$   
 $\langle \text{proof} \rangle$

### 3.8 The predicate $\text{OrdNotEqP}$

**nominal\_function**  $\text{OrdNotEqP} :: \text{tm} \Rightarrow \text{tm} \Rightarrow \text{fm}$  (**infixr**  $\langle \text{NEQ} \rangle$  150)  
**where**  $\text{OrdNotEqP } x \ y = \text{OrdP } x \ \text{AND} \ \text{OrdP } y \ \text{AND} \ (x \ \text{IN } y \ \text{OR} \ y \ \text{IN } x)$   
 $\langle \text{proof} \rangle$

**nominal\_termination** ( $\text{eqvt}$ )  
 $\langle \text{proof} \rangle$

**lemma**  $\text{OrdNotEqP\_fresh\_iff}$  [ $\text{simp}$ ]:  $a \# \text{OrdNotEqP } x \ y \longleftrightarrow a \# x \wedge a \# y$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{OrdNotEqP\_subst}$  [ $\text{simp}$ ]:  $(\text{OrdNotEqP } x \ y)(i::=t) = \text{OrdNotEqP } (\text{subst } i \ t \ x) (\text{subst } i \ t \ y)$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{OrdNotEqP\_cong}$ :  $H \vdash x \ \text{EQ} \ x' \Longrightarrow H \vdash y \ \text{EQ} \ y' \Longrightarrow H \vdash \text{OrdNotEqP } x \ y \ \text{IFF} \ \text{OrdNotEqP } x' \ y'$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{OrdNotEqP\_self\_contra}$ :  $\{x \ \text{NEQ} \ x\} \vdash \text{Fls}$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{OrdNotEqP\_OrdP\_E}$ :  $\text{insert } (\text{OrdP } x) (\text{insert } (\text{OrdP } y) \ H) \vdash A \Longrightarrow \text{insert } (x \ \text{NEQ} \ y) \ H \vdash A$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{OrdNotEqP\_I}$ :  $\text{insert } (x \ \text{EQ} \ y) \ H \vdash \text{Fls} \Longrightarrow H \vdash \text{OrdP } x \Longrightarrow H \vdash \text{OrdP } y \Longrightarrow H \vdash x \ \text{NEQ} \ y$   
 $\langle \text{proof} \rangle$

**declare**  $\text{OrdNotEqP.simps}$  [ $\text{simp del}$ ]

**lemma**  $\text{OrdNotEqP\_imp\_Neg\_Eq}$ :  $\{x \ \text{NEQ} \ y\} \vdash \text{Neg } (x \ \text{EQ} \ y)$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{OrdNotEqP\_E}$ :  $H \vdash x \ \text{EQ} \ y \Longrightarrow \text{insert } (x \ \text{NEQ} \ y) \ H \vdash A$   
 $\langle \text{proof} \rangle$

### 3.9 Predecessor of an Ordinal

**lemma**  $\text{OrdP\_set\_max\_lemma}$ :

**assumes**  $j$ :  $\text{atom } (j::\text{name}) \# i$  **and**  $k$ :  $\text{atom } (k::\text{name}) \# (i,j)$   
**shows**  $\{ \} \vdash (\text{Neg } (\text{Var } i \text{ EQ Zero}) \text{ AND } (\text{All2 } j (\text{Var } i) (\text{OrdP } (\text{Var } j)))) \text{ IMP}$   
 $(\text{Ex } j (\text{Var } j \text{ IN } \text{Var } i \text{ AND } (\text{All2 } k (\text{Var } i) (\text{Var } k \text{ SUBS } \text{Var } j))))$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{OrdP\_max\_imp}$ :  
**assumes**  $j$ :  $\text{atom } j \# (x)$  **and**  $k$ :  $\text{atom } k \# (x,j)$   
**shows**  $\{ \text{OrdP } x, \text{Neg } (x \text{ EQ Zero}) \} \vdash \text{Ex } j (\text{Var } j \text{ IN } x \text{ AND } (\text{All2 } k x (\text{Var } k \text{ SUBS } \text{Var } j)))$   
 $\langle \text{proof} \rangle$

**declare**  $\text{OrdP.simps}$  [ $\text{simp del}$ ]

### 3.10 Case Analysis and Zero/SUCC Induction

**lemma**  $\text{OrdP\_cases\_lemma}$ :  
**assumes**  $p$ :  $\text{atom } p \# x$   
**shows**  $\{ \text{OrdP } x, \text{Neg } (x \text{ EQ Zero}) \} \vdash \text{Ex } p (\text{OrdP } (\text{Var } p) \text{ AND } x \text{ EQ SUCC } (\text{Var } p))$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{OrdP\_cases\_disj}$ :  
**assumes**  $p$ :  $\text{atom } p \# x$   
**shows**  $\text{insert } (\text{OrdP } x) H \vdash x \text{ EQ Zero OR } \text{Ex } p (\text{OrdP } (\text{Var } p) \text{ AND } x \text{ EQ SUCC } (\text{Var } p))$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{OrdP\_cases\_E}$ :  
 $\llbracket \text{insert } (x \text{ EQ Zero}) H \vdash A;$   
 $\text{insert } (x \text{ EQ SUCC } (\text{Var } k)) (\text{insert } (\text{OrdP } (\text{Var } k)) H) \vdash A;$   
 $\text{atom } k \# (x,A); \quad \forall C \in H. \text{atom } k \# C \rrbracket$   
 $\implies \text{insert } (\text{OrdP } x) H \vdash A$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{OrdInd2\_lemma}$ :  
 $\{ \text{OrdP } (\text{Var } i), A(i::= \text{Zero}), (\text{All } i (\text{OrdP } (\text{Var } i) \text{ IMP } A \text{ IMP } (A(i::= \text{SUCC } (\text{Var } i)))) \} \vdash A$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{OrdInd2}$ :  
**assumes**  $H \vdash \text{OrdP } (\text{Var } i)$   
**and**  $H \vdash A(i::= \text{Zero})$   
**and**  $H \vdash \text{All } i (\text{OrdP } (\text{Var } i) \text{ IMP } A \text{ IMP } (A(i::= \text{SUCC } (\text{Var } i))))$   
**shows**  $H \vdash A$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{OrdInd2H}$ :  
**assumes**  $H \vdash A(i::= \text{Zero})$   
**and**  $H \vdash \text{All } i (\text{OrdP } (\text{Var } i) \text{ IMP } A \text{ IMP } (A(i::= \text{SUCC } (\text{Var } i))))$   
**shows**  $\text{insert } (\text{OrdP } (\text{Var } i)) H \vdash A$   
 $\langle \text{proof} \rangle$

### 3.11 The predicate $\text{HFun\_Sigma}$

To characterise the concept of a function using only bounded universal quantifiers.

See the note after the proof of Lemma 2.3.

**definition**  $\text{hfun\_sigma}$  **where**  
 $\text{hfun\_sigma } r \equiv \forall z \in r. \forall z' \in r. \exists x y x' y'. z = \langle x,y \rangle \wedge z' = \langle x',y' \rangle \wedge (x=x' \longrightarrow y=y')$

**definition**  $\text{hfun\_sigma\_ord}$  **where**

$hfun\_sigma\_ord\ r \equiv \forall z \in r. \forall z' \in r. \exists x\ y\ x'\ y'. z = \langle x, y \rangle \wedge z' = \langle x', y' \rangle \wedge Ord\ x \wedge Ord\ x' \wedge (x=x' \rightarrow y=y')$

**nominal\_function**  $HFun\_Sigma :: tm \Rightarrow fm$

**where**  $\llbracket atom\ z \# (r, z', x, y, x', y'); atom\ z' \# (r, x, y, x', y'); atom\ x \# (r, y, x', y'); atom\ y \# (r, x', y'); atom\ x' \# (r, y'); atom\ y' \# (r) \rrbracket \Longrightarrow$   
 $HFun\_Sigma\ r =$   
 $All2\ z\ r\ (All2\ z'\ r\ (Ex\ x\ (Ex\ y\ (Ex\ x'\ (Ex\ y'$   
 $(Var\ z\ EQ\ HPair\ (Var\ x)\ (Var\ y)\ AND\ Var\ z'\ EQ\ HPair\ (Var\ x')\ (Var\ y')$   
 $AND\ OrdP\ (Var\ x)\ AND\ OrdP\ (Var\ x')\ AND$   
 $((Var\ x\ EQ\ Var\ x')\ IMP\ (Var\ y\ EQ\ Var\ y'))))))))$

$\langle proof \rangle$

**nominal\_termination**  $(eqvt)$

$\langle proof \rangle$

**lemma**

**shows**  $HFun\_Sigma\_fresh\_iff\ [simp]: a \# HFun\_Sigma\ r \longleftrightarrow a \# r\ (\text{is } ?thesis1)$

$\langle proof \rangle$

**lemma**  $HFun\_Sigma\_subst\ [simp]: (HFun\_Sigma\ r)(i::=t) = HFun\_Sigma\ (subst\ i\ t\ r)$

$\langle proof \rangle$

**lemma**  $HFun\_Sigma\_Zero: H \vdash HFun\_Sigma\ Zero$

$\langle proof \rangle$

**lemma**  $Subset\_HFun\_Sigma: \{HFun\_Sigma\ s, s'\ SUBS\ s\} \vdash HFun\_Sigma\ s'$

$\langle proof \rangle$

Captures the property of being a relation, using fewer variables than the full definition

**lemma**  $HFun\_Sigma\_Mem\_imp\_HPair:$

**assumes**  $H \vdash HFun\_Sigma\ r\ H \vdash a\ IN\ r$

**and**  $xy: atom\ x \# (y, a, r)\ atom\ y \# (a, r)$

**shows**  $H \vdash (Ex\ x\ (Ex\ y\ (a\ EQ\ HPair\ (Var\ x)\ (Var\ y))))\ (\text{is } \_ \vdash ?concl)$

$\langle proof \rangle$

### 3.12 The predicate $HDomain\_Incl$

This is an internal version of  $\forall x \in d. \exists y\ z. z \in r \wedge z = \langle x, y \rangle$ .

**nominal\_function**  $HDomain\_Incl :: tm \Rightarrow tm \Rightarrow fm$

**where**  $\llbracket atom\ x \# (r, d, y, z); atom\ y \# (r, d, z); atom\ z \# (r, d) \rrbracket \Longrightarrow$

$HDomain\_Incl\ r\ d = All2\ x\ d\ (Ex\ y\ (Ex\ z\ (Var\ z\ IN\ r\ AND\ Var\ z\ EQ\ HPair\ (Var\ x)\ (Var\ y))))$

$\langle proof \rangle$

**nominal\_termination**  $(eqvt)$

$\langle proof \rangle$

**lemma**

**shows**  $HDomain\_Incl\_fresh\_iff\ [simp]:$

$a \# HDomain\_Incl\ r\ d \longleftrightarrow a \# r \wedge a \# d\ (\text{is } ?thesis1)$

$\langle proof \rangle$

**lemma**  $HDomain\_Incl\_subst\ [simp]:$

$(HDomain\_Incl\ r\ d)(i::=t) = HDomain\_Incl\ (subst\ i\ t\ r)\ (subst\ i\ t\ d)$

$\langle proof \rangle$

**lemma**  $HDomain\_Incl\_Subset\_lemma: \{HDomain\_Incl\ r\ k, k'\ SUBS\ k\} \vdash HDomain\_Incl\ r\ k'$

*<proof>*

**lemma** *HDomain\_Incl\_Subset*:  $H \vdash \text{HDomain\_Incl } r \ k \implies H \vdash k' \ \text{SUBS } k \implies H \vdash \text{HDomain\_Incl } r \ k'$

*<proof>*

**lemma** *HDomain\_Incl\_Mem\_Ord*:  $H \vdash \text{HDomain\_Incl } r \ k \implies H \vdash k' \ \text{IN } k \implies H \vdash \text{OrdP } k \implies H \vdash \text{HDomain\_Incl } r \ k'$

*<proof>*

**lemma** *HDomain\_Incl\_Zero [simp]*:  $H \vdash \text{HDomain\_Incl } r \ \text{Zero}$

*<proof>*

**lemma** *HDomain\_Incl\_Eats*:  $\{ \text{HDomain\_Incl } r \ d \} \vdash \text{HDomain\_Incl } (\text{Eats } r \ (\text{HPair } d \ d')) \ (\text{SUCC } d)$

*<proof>*

**lemma** *HDomain\_Incl\_Eats\_I*:  $H \vdash \text{HDomain\_Incl } r \ d \implies H \vdash \text{HDomain\_Incl } (\text{Eats } r \ (\text{HPair } d \ d'))$

*(SUCC d)*

*<proof>*

### 3.13 *HPair* is Provably Injective

**lemma** *Doubleton\_E*:

**assumes**  $\text{insert } (a \ \text{EQ } c) \ (\text{insert } (b \ \text{EQ } d) \ H) \vdash A$

$\text{insert } (a \ \text{EQ } d) \ (\text{insert } (b \ \text{EQ } c) \ H) \vdash A$

**shows**  $\text{insert } ((\text{Eats } (\text{Eats } \text{Zero } b) \ a) \ \text{EQ } (\text{Eats } (\text{Eats } \text{Zero } d) \ c)) \ H \vdash A$

*<proof>*

**lemma** *HFST*:  $\{ \text{HPair } a \ b \ \text{EQ } \text{HPair } c \ d \} \vdash a \ \text{EQ } c$

*<proof>*

**lemma** *b\_EQ\_d\_1*:  $\{ a \ \text{EQ } c, a \ \text{EQ } d, b \ \text{EQ } c \} \vdash b \ \text{EQ } d$

*<proof>*

**lemma** *HSND*:  $\{ \text{HPair } a \ b \ \text{EQ } \text{HPair } c \ d \} \vdash b \ \text{EQ } d$

*<proof>*

**lemma** *HPair\_E [intro!]*:

**assumes**  $\text{insert } (a \ \text{EQ } c) \ (\text{insert } (b \ \text{EQ } d) \ H) \vdash A$

**shows**  $\text{insert } (\text{HPair } a \ b \ \text{EQ } \text{HPair } c \ d) \ H \vdash A$

*<proof>*

**declare** *HPair\_E [THEN rotate2, intro!]*

**declare** *HPair\_E [THEN rotate3, intro!]*

**declare** *HPair\_E [THEN rotate4, intro!]*

**declare** *HPair\_E [THEN rotate5, intro!]*

**declare** *HPair\_E [THEN rotate6, intro!]*

**declare** *HPair\_E [THEN rotate7, intro!]*

**declare** *HPair\_E [THEN rotate8, intro!]*

**lemma** *HFun\_Sigma\_E*:

**assumes**  $r: H \vdash \text{HFun\_Sigma } r$

**and**  $b: H \vdash \text{HPair } a \ b \ \text{IN } r$

**and**  $b': H \vdash \text{HPair } a \ b' \ \text{IN } r$

**shows**  $H \vdash b \ \text{EQ } b'$

*<proof>*

### 3.14 *SUCC* is Provably Injective

**lemma** *SUCC\_SUBS\_lemma*:  $\{SUCC\ x\ SUBS\ SUCC\ y\} \vdash x\ SUBS\ y$   
 ⟨*proof*⟩

**lemma** *SUCC\_SUBS*:  $insert\ (SUCC\ x\ SUBS\ SUCC\ y)\ H \vdash x\ SUBS\ y$   
 ⟨*proof*⟩

**lemma** *SUCC\_inject*:  $insert\ (SUCC\ x\ EQ\ SUCC\ y)\ H \vdash x\ EQ\ y$   
 ⟨*proof*⟩

**lemma** *SUCC\_inject\_E* [*intro!*]:  $insert\ (x\ EQ\ y)\ H \vdash A \implies insert\ (SUCC\ x\ EQ\ SUCC\ y)\ H \vdash A$   
 ⟨*proof*⟩

**declare** *SUCC\_inject\_E* [*THEN rotate2, intro!*]  
**declare** *SUCC\_inject\_E* [*THEN rotate3, intro!*]  
**declare** *SUCC\_inject\_E* [*THEN rotate4, intro!*]  
**declare** *SUCC\_inject\_E* [*THEN rotate5, intro!*]  
**declare** *SUCC\_inject\_E* [*THEN rotate6, intro!*]  
**declare** *SUCC\_inject\_E* [*THEN rotate7, intro!*]  
**declare** *SUCC\_inject\_E* [*THEN rotate8, intro!*]

**lemma** *OrdP\_IN\_SUCC\_lemma*:  $\{OrdP\ x,\ y\ IN\ x\} \vdash SUCC\ y\ IN\ SUCC\ x$   
 ⟨*proof*⟩

**lemma** *OrdP\_IN\_SUCC*:  $H \vdash OrdP\ x \implies H \vdash y\ IN\ x \implies H \vdash SUCC\ y\ IN\ SUCC\ x$   
 ⟨*proof*⟩

**lemma** *OrdP\_IN\_SUCC\_D\_lemma*:  $\{OrdP\ x,\ SUCC\ y\ IN\ SUCC\ x\} \vdash y\ IN\ x$   
 ⟨*proof*⟩

**lemma** *OrdP\_IN\_SUCC\_D*:  $H \vdash OrdP\ x \implies H \vdash SUCC\ y\ IN\ SUCC\ x \implies H \vdash y\ IN\ x$   
 ⟨*proof*⟩

**lemma** *OrdP\_IN\_SUCC\_Iff*:  $H \vdash OrdP\ y \implies H \vdash SUCC\ x\ IN\ SUCC\ y\ IFF\ x\ IN\ y$   
 ⟨*proof*⟩

### 3.15 The predicate *LstSeqP*

**lemma** *hfun\_sigma\_ord\_iff*:  $hfun\_sigma\_ord\ s \longleftrightarrow OrdDom\ s \wedge hfun\_sigma\ s$   
 ⟨*proof*⟩

**lemma** *hfun\_sigma\_iff*:  $hfun\_sigma\ r \longleftrightarrow hfunction\ r \wedge hrelation\ r$   
 ⟨*proof*⟩

**lemma** *Seq\_iff*:  $Seq\ r\ d \longleftrightarrow d \leq hdomain\ r \wedge hfun\_sigma\ r$   
 ⟨*proof*⟩

**lemma** *LstSeq\_iff*:  $LstSeq\ s\ k\ y \longleftrightarrow succ\ k \leq hdomain\ s \wedge \langle k, y \rangle \in s \wedge hfun\_sigma\_ord\ s$   
 ⟨*proof*⟩

**nominal\_function** *LstSeqP* ::  $tm \Rightarrow tm \Rightarrow tm \Rightarrow fm$   
**where**

$LstSeqP\ s\ k\ y = OrdP\ k\ AND\ HDomain\_Incl\ s\ (SUCC\ k)\ AND\ HFun\_Sigma\ s\ AND\ HPair\ k\ y\ IN\ s$   
 ⟨*proof*⟩

**nominal\_termination** (*eqvt*)  
 ⟨*proof*⟩

**lemma**

**shows**  $LstSeqP\_fresh\_iff$  [simp]:

$$a \# LstSeqP\ s\ k\ y \longleftrightarrow a \# s \wedge a \# k \wedge a \# y \quad (\text{is } ?thesis1)$$

$\langle proof \rangle$

**lemma**  $LstSeqP\_subst$  [simp]:

$$(LstSeqP\ s\ k\ y)(i::=t) = LstSeqP\ (subst\ i\ t\ s)\ (subst\ i\ t\ k)\ (subst\ i\ t\ y)$$

$\langle proof \rangle$

**lemma**  $LstSeqP\_E$ :

**assumes**  $insert\ (HDomain\_Incl\ s\ (SUCC\ k))$   
 $(insert\ (OrdP\ k)\ (insert\ (HFun\_Sigma\ s)$   
 $(insert\ (HPair\ k\ y\ IN\ s)\ H))) \vdash B$

**shows**  $insert\ (LstSeqP\ s\ k\ y)\ H \vdash B$

$\langle proof \rangle$

**declare**  $LstSeqP.simps$  [simp del]

**lemma**  $LstSeqP\_cong$ :

**assumes**  $H \vdash s\ EQ\ s'\ H \vdash k\ EQ\ k'\ H \vdash y\ EQ\ y'$

**shows**  $H \vdash LstSeqP\ s\ k\ y\ IFF\ LstSeqP\ s'\ k'\ y'$

$\langle proof \rangle$

**lemma**  $LstSeqP\_OrdP$ :  $H \vdash LstSeqP\ r\ k\ y \implies H \vdash OrdP\ k$

$\langle proof \rangle$

**lemma**  $LstSeqP\_Mem\_lemma$ :  $\{ LstSeqP\ r\ k\ y, HPair\ k'\ z\ IN\ r, k'\ IN\ k \} \vdash LstSeqP\ r\ k'\ z$

$\langle proof \rangle$

**lemma**  $LstSeqP\_Mem$ :  $H \vdash LstSeqP\ r\ k\ y \implies H \vdash HPair\ k'\ z\ IN\ r \implies H \vdash k'\ IN\ k \implies H \vdash LstSeqP\ r\ k'\ z$

$\langle proof \rangle$

**lemma**  $LstSeqP\_imp\_Mem$ :  $H \vdash LstSeqP\ s\ k\ y \implies H \vdash HPair\ k\ y\ IN\ s$

$\langle proof \rangle$

**lemma**  $LstSeqP\_SUCC$ :  $H \vdash LstSeqP\ r\ (SUCC\ d)\ y \implies H \vdash HPair\ d\ z\ IN\ r \implies H \vdash LstSeqP\ r\ d\ z$

$\langle proof \rangle$

**lemma**  $LstSeqP\_EQ$ :  $\llbracket H \vdash LstSeqP\ s\ k\ y; H \vdash HPair\ k\ y'\ IN\ s \rrbracket \implies H \vdash y\ EQ\ y'$

$\langle proof \rangle$

**end**

# Chapter 4

## Sigma-Formulas and Theorem 2.5

```
theory Sigma
imports Predicates
begin
```

### 4.1 Ground Terms and Formulas

```
definition ground_aux :: tm  $\Rightarrow$  atom set  $\Rightarrow$  bool
  where ground_aux t S  $\equiv$  (supp t  $\subseteq$  S)
```

```
abbreviation ground :: tm  $\Rightarrow$  bool
  where ground t  $\equiv$  ground_aux t {}
```

```
definition ground_fm_aux :: fm  $\Rightarrow$  atom set  $\Rightarrow$  bool
  where ground_fm_aux A S  $\equiv$  (supp A  $\subseteq$  S)
```

```
abbreviation ground_fm :: fm  $\Rightarrow$  bool
  where ground_fm A  $\equiv$  ground_fm_aux A {}
```

```
lemma ground_aux_simps[simp]:
  ground_aux Zero S = True
  ground_aux (Var k) S = (if atom k  $\in$  S then True else False)
  ground_aux (Eats t u) S = (ground_aux t S  $\wedge$  ground_aux u S)
<proof>
```

```
lemma ground_fm_aux_simps[simp]:
  ground_fm_aux Fls S = True
  ground_fm_aux (t IN u) S = (ground_aux t S  $\wedge$  ground_aux u S)
  ground_fm_aux (t EQ u) S = (ground_aux t S  $\wedge$  ground_aux u S)
  ground_fm_aux (A OR B) S = (ground_fm_aux A S  $\wedge$  ground_fm_aux B S)
  ground_fm_aux (A AND B) S = (ground_fm_aux A S  $\wedge$  ground_fm_aux B S)
  ground_fm_aux (A IFF B) S = (ground_fm_aux A S  $\wedge$  ground_fm_aux B S)
  ground_fm_aux (Neg A) S = (ground_fm_aux A S)
  ground_fm_aux (Ex x A) S = (ground_fm_aux A (S  $\cup$  {atom x}))
<proof>
```

```
lemma ground_fresh[simp]:
  ground t  $\implies$  atom i  $\#$  t
  ground_fm A  $\implies$  atom i  $\#$  A
<proof>
```

## 4.2 Sigma Formulas

Section 2 material

### 4.2.1 Strict Sigma Formulas

Definition 2.1

**inductive**  $ss\_fm :: fm \Rightarrow bool$  **where**  
   $MemI: ss\_fm (Var\ i\ IN\ Var\ j)$   
   $| DisjI: ss\_fm\ A \Longrightarrow ss\_fm\ B \Longrightarrow ss\_fm\ (A\ OR\ B)$   
   $| ConjI: ss\_fm\ A \Longrightarrow ss\_fm\ B \Longrightarrow ss\_fm\ (A\ AND\ B)$   
   $| ExI: ss\_fm\ A \Longrightarrow ss\_fm\ (Ex\ i\ A)$   
   $| All2I: ss\_fm\ A \Longrightarrow atom\ j\ \#(i,A) \Longrightarrow ss\_fm\ (All2\ i\ (Var\ j)\ A)$

**equivariance**  $ss\_fm$

**nominal\_inductive**  $ss\_fm$

**avoids**  $ExI: i \mid All2I: i$

$\langle proof \rangle$

**declare**  $ss\_fm.intros$   $[intro]$

**definition**  $Sigma\_fm :: fm \Rightarrow bool$

**where**  $Sigma\_fm\ A \longleftrightarrow (\exists B. ss\_fm\ B \wedge supp\ B \subseteq supp\ A \wedge \{\} \vdash A\ IFF\ B)$

**lemma**  $Sigma\_fm\_Iff: [\{\} \vdash B\ IFF\ A; supp\ A \subseteq supp\ B; Sigma\_fm\ A] \Longrightarrow Sigma\_fm\ B$   
 $\langle proof \rangle$

**lemma**  $ss\_fm\_imp\_Sigma\_fm$   $[intro]: ss\_fm\ A \Longrightarrow Sigma\_fm\ A$   
 $\langle proof \rangle$

**lemma**  $Sigma\_fm\_Fls$   $[iff]: Sigma\_fm\ Fls$   
 $\langle proof \rangle$

### 4.2.2 Closure properties for Sigma-formulas

**lemma**

**assumes**  $Sigma\_fm\ A\ Sigma\_fm\ B$   
**shows**  $Sigma\_fm\_AND$   $[intro!]: Sigma\_fm\ (A\ AND\ B)$   
**and**  $Sigma\_fm\_OR$   $[intro!]: Sigma\_fm\ (A\ OR\ B)$   
**and**  $Sigma\_fm\_Ex$   $[intro!]: Sigma\_fm\ (Ex\ i\ A)$

$\langle proof \rangle$

**lemma**  $Sigma\_fm\_All2\_Var:$

**assumes**  $H0: Sigma\_fm\ A$  **and**  $ij: atom\ j\ \#(i,A)$   
**shows**  $Sigma\_fm\ (All2\ i\ (Var\ j)\ A)$

$\langle proof \rangle$

## 4.3 Lemma 2.2: Atomic formulas are Sigma-formulas

**lemma**  $Eq\_Eats\_Iff:$

**assumes**  $[unfolded\ fresh\_Pair,\ simp]: atom\ i\ \#(z,x,y)$

**shows**  $\{\} \vdash z\ EQ\ Eats\ x\ y\ IFF\ (All2\ i\ z\ (Var\ i\ IN\ x\ OR\ Var\ i\ EQ\ y))\ AND\ x\ SUBS\ z\ AND\ y\ IN\ z$

$\langle proof \rangle$

**lemma**  $Subset\_Zero\_sf: Sigma\_fm\ (Var\ i\ SUBS\ Zero)$

*<proof>*

**lemma** *Eq\_Zero\_sf*:  $\text{Sigma\_fm } (\text{Var } i \text{ EQ Zero})$

*<proof>*

**lemma** *theorem\_sf*: **assumes**  $\{\}$  **shows**  $\text{Sigma\_fm } A$

*<proof>*

The subset relation

**lemma** *Var\_Subset\_sf*:  $\text{Sigma\_fm } (\text{Var } i \text{ SUBS } \text{Var } j)$

*<proof>*

**lemma** *Zero\_Mem\_sf*:  $\text{Sigma\_fm } (\text{Zero } \text{IN } \text{Var } i)$

*<proof>*

**lemma** *ijk*:  $i + k < \text{Suc } (i + j + k)$

*<proof>*

**lemma** *All2\_term\_Iff\_fresh*:  $i \neq j \implies \text{atom } j' \# (i, j, A) \implies$

$\{\} \vdash (\text{All2 } i (\text{Var } j) A) \text{ IFF } \text{Ex } j' (\text{Var } j \text{ EQ } \text{Var } j' \text{ AND } \text{All2 } i (\text{Var } j') A)$

*<proof>*

**lemma** *Sigma\_fm\_All2\_fresh*:

**assumes**  $\text{Sigma\_fm } A \ i \neq j$

**shows**  $\text{Sigma\_fm } (\text{All2 } i (\text{Var } j) A)$

*<proof>*

**lemma** *Subset\_Eats\_sf*:

**assumes**  $\bigwedge j::\text{name}. \text{Sigma\_fm } (\text{Var } j \text{ IN } t)$

**and**  $\bigwedge k::\text{name}. \text{Sigma\_fm } (\text{Var } k \text{ EQ } u)$

**shows**  $\text{Sigma\_fm } (\text{Var } i \text{ SUBS } \text{Eats } t \ u)$

*<proof>*

**lemma** *Eq\_Eats\_sf*:

**assumes**  $\bigwedge j::\text{name}. \text{Sigma\_fm } (\text{Var } j \text{ EQ } t)$

**and**  $\bigwedge k::\text{name}. \text{Sigma\_fm } (\text{Var } k \text{ EQ } u)$

**shows**  $\text{Sigma\_fm } (\text{Var } i \text{ EQ } \text{Eats } t \ u)$

*<proof>*

**lemma** *Eats\_Mem\_sf*:

**assumes**  $\bigwedge j::\text{name}. \text{Sigma\_fm } (\text{Var } j \text{ EQ } t)$

**and**  $\bigwedge k::\text{name}. \text{Sigma\_fm } (\text{Var } k \text{ EQ } u)$

**shows**  $\text{Sigma\_fm } (\text{Eats } t \ u \ \text{IN } \text{Var } i)$

*<proof>*

**lemma** *Subset\_Mem\_sf\_lemma*:

$\text{size } t + \text{size } u < n \implies \text{Sigma\_fm } (t \ \text{SUBS } u) \wedge \text{Sigma\_fm } (t \ \text{IN } u)$

*<proof>*

**lemma** *Subset\_sf [iff]*:  $\text{Sigma\_fm } (t \ \text{SUBS } u)$

*<proof>*

**lemma** *Mem\_sf [iff]*:  $\text{Sigma\_fm } (t \ \text{IN } u)$

*<proof>*

The equality relation is a Sigma-Formula

**lemma** *Equality\_sf [iff]*:  $\text{Sigma\_fm } (t \ \text{EQ } u)$

*<proof>*

## 4.4 Universal Quantification Bounded by an Arbitrary Term

**lemma** *All2\_term\_Iff*:  $atom\ i \# t \implies atom\ j \# (i, t, A) \implies$   
 $\{\} \vdash (All2\ i\ t\ A)\ IFF\ Ex\ j\ (Var\ j\ EQ\ t\ AND\ All2\ i\ (Var\ j)\ A)$

*<proof>*

**lemma** *Sigma\_fm\_All2 [intro!]*:  
**assumes** *Sigma\_fm A atom i # t*  
**shows** *Sigma\_fm (All2 i t A)*

*<proof>*

## 4.5 Lemma 2.3: Sequence-related concepts are Sigma-formulas

**lemma** *OrdP\_sf [iff]*: *Sigma\_fm (OrdP t)*

*<proof>*

**lemma** *OrdNotEqP\_sf [iff]*: *Sigma\_fm (OrdNotEqP t u)*

*<proof>*

**lemma** *HDomain\_Incl\_sf [iff]*: *Sigma\_fm (HDomain\_Incl t u)*

*<proof>*

**lemma** *HFun\_Sigma\_Iff*:

**assumes**  $atom\ z \# (r, z', x, y, x', y')$   $atom\ z' \# (r, x, y, x', y')$   
 $atom\ x \# (r, y, x', y')$   $atom\ y \# (r, x', y')$   
 $atom\ x' \# (r, y')$   $atom\ y' \# (r)$

**shows**

$\{\} \vdash HFun\_Sigma\ r\ IFF$   
 $All2\ z\ r\ (All2\ z'\ r\ (Ex\ x\ (Ex\ y\ (Ex\ x'\ (Ex\ y'$   
 $(Var\ z\ EQ\ HPair\ (Var\ x)\ (Var\ y)\ AND\ Var\ z'\ EQ\ HPair\ (Var\ x')\ (Var\ y')$   
 $AND\ OrdP\ (Var\ x)\ AND\ OrdP\ (Var\ x')\ AND$   
 $((Var\ x\ NEQ\ Var\ x')\ OR\ (Var\ y\ EQ\ Var\ y'))))))))$

*<proof>*

**lemma** *HFun\_Sigma\_sf [iff]*: *Sigma\_fm (HFun\_Sigma t)*

*<proof>*

**lemma** *LstSeqP\_sf [iff]*: *Sigma\_fm (LstSeqP t u v)*

*<proof>*

**end**

## Chapter 5

# Predicates for Terms, Formulas and Substitution

```
theory Coding_Predicates
imports Coding Sigma
begin
```

```
declare succ_iff [simp del]
```

This material comes from Section 3, greatly modified for de Bruijn syntax.

### 5.1 Predicates for atomic terms

#### 5.1.1 Free Variables

```
definition VarP :: tm  $\Rightarrow$  fm where VarP x  $\equiv$  OrdP x AND Zero IN x
```

```
lemma VarP_eqvt [eqvt]: (p  $\cdot$  VarP x) = VarP (p  $\cdot$  x)
<proof>
```

```
lemma VarP_fresh_iff [simp]: a  $\#$  VarP x  $\longleftrightarrow$  a  $\#$  x
<proof>
```

```
lemma VarP_sf [iff]: Sigma_fm (VarP x)
<proof>
```

```
lemma VarP_subst [simp]: (VarP x)(i::=t) = VarP (subst i t x)
<proof>
```

```
lemma VarP_cong: H  $\vdash$  x EQ x'  $\Longrightarrow$  H  $\vdash$  VarP x IFF VarP x'
<proof>
```

```
lemma VarP_HPairE [intro!]: insert (VarP (HPair x y)) H  $\vdash$  A
<proof>
```

#### 5.1.2 De Bruijn Indexes

```
abbreviation Q_Ind :: tm  $\Rightarrow$  tm
where Q_Ind k  $\equiv$  HPair (HTuple 6) k
```

```
nominal_function IndP :: tm  $\Rightarrow$  fm
where atom m  $\#$  x  $\Longrightarrow$ 
```

$IndP\ x = Ex\ m\ (OrdP\ (Var\ m)\ AND\ x\ EQ\ HPair\ (HTuple\ 6)\ (Var\ m))$   
 ⟨proof⟩

**nominal\_termination** (eqvt)  
 ⟨proof⟩

**lemma**  
**shows**  $IndP\_fresh\_iff\ [simp]: a \# IndP\ x \longleftrightarrow a \# x$  (is ?thesis1)  
**and**  $IndP\_sf\ [iff]: Sigma\_fm\ (IndP\ x)$  (is ?thsf)  
**and**  $OrdP\_IndP\_Q\_Ind: \{OrdP\ x\} \vdash IndP\ (Q\_Ind\ x)$  (is ?thqind)  
 ⟨proof⟩

**lemma**  $IndP\_Q\_Ind: H \vdash OrdP\ x \Longrightarrow H \vdash IndP\ (Q\_Ind\ x)$   
 ⟨proof⟩

**lemma**  $subst\_fm\_IndP\ [simp]: (IndP\ t)(i::=x) = IndP\ (subst\ i\ x\ t)$   
 ⟨proof⟩

**lemma**  $IndP\_cong: H \vdash x\ EQ\ x' \Longrightarrow H \vdash IndP\ x\ IFF\ IndP\ x'$   
 ⟨proof⟩

### 5.1.3 Various syntactic lemmas

## 5.2 The predicate $SeqCTermP$ , for Terms and Constants

**nominal\_function**  $SeqCTermP :: bool \Rightarrow tm \Rightarrow tm \Rightarrow tm \Rightarrow fm$   
**where**  $\llbracket atom\ l \# (s, k, sl, m, n, sm, sn); atom\ sl \# (s, m, n, sm, sn);$   
 $atom\ m \# (s, n, sm, sn); atom\ n \# (s, sm, sn);$   
 $atom\ sm \# (s, sn); atom\ sn \# (s) \rrbracket \Longrightarrow$   
 $SeqCTermP\ vf\ s\ k\ t =$   
 $LstSeqP\ s\ k\ t\ AND$   
 $All2\ l\ (SUCC\ k)\ (Ex\ sl\ (HPair\ (Var\ l)\ (Var\ sl)\ IN\ s\ AND$   
 $(Var\ sl\ EQ\ Zero\ OR\ (if\ vf\ then\ VarP\ (Var\ sl)\ else\ Fls)\ OR$   
 $Ex\ m\ (Ex\ n\ (Ex\ sm\ (Ex\ sn\ (Var\ m\ IN\ Var\ l\ AND\ Var\ n\ IN\ Var\ l\ AND$   
 $HPair\ (Var\ m)\ (Var\ sm)\ IN\ s\ AND\ HPair\ (Var\ n)\ (Var\ sn)\ IN\ s\ AND$   
 $Var\ sl\ EQ\ Q\_Eats\ (Var\ sm)\ (Var\ sn))))))$   
 ⟨proof⟩

**nominal\_termination** (eqvt)  
 ⟨proof⟩

**lemma**  
**shows**  $SeqCTermP\_fresh\_iff\ [simp]:$   
 $a \# SeqCTermP\ vf\ s\ k\ t \longleftrightarrow a \# s \wedge a \# k \wedge a \# t$  (is ?thesis1)  
**and**  $SeqCTermP\_sf\ [iff]:$   
 $Sigma\_fm\ (SeqCTermP\ vf\ s\ k\ t)$  (is ?thsf)  
**and**  $SeqCTermP\_imp\_LstSeqP:$   
 $\{SeqCTermP\ vf\ s\ k\ t\} \vdash LstSeqP\ s\ k\ t$  (is ?thlstseq)  
**and**  $SeqCTermP\_imp\_OrdP\ [simp]:$   
 $\{SeqCTermP\ vf\ s\ k\ t\} \vdash OrdP\ k$  (is ?thord)  
 ⟨proof⟩

**lemma**  $SeqCTermP\_subst\ [simp]:$   
 $(SeqCTermP\ vf\ s\ k\ t)(j::=w) = SeqCTermP\ vf\ (subst\ j\ w\ s)\ (subst\ j\ w\ k)\ (subst\ j\ w\ t)$   
 ⟨proof⟩

**declare**  $SeqCTermP.simps\ [simp\ del]$

**abbreviation**  $SeqTermP :: tm \Rightarrow tm \Rightarrow tm \Rightarrow fm$   
**where**  $SeqTermP \equiv SeqCTermP \text{ True}$

**abbreviation**  $SeqConstP :: tm \Rightarrow tm \Rightarrow tm \Rightarrow fm$   
**where**  $SeqConstP \equiv SeqCTermP \text{ False}$

**lemma**  $SeqConstP\_imp\_SeqTermP: \{SeqConstP \ s \ k \ t\} \vdash SeqTermP \ s \ k \ t$   
 $\langle proof \rangle$

## 5.3 The predicates $TermP$ and $ConstP$

### 5.3.1 Definition

**nominal\_function**  $CTermP :: bool \Rightarrow tm \Rightarrow fm$   
**where**  $\llbracket atom \ k \ \# \ (s,t); atom \ s \ \# \ t \rrbracket \Longrightarrow$   
 $CTermP \ vf \ t = Ex \ s \ (Ex \ k \ (SeqCTermP \ vf \ (Var \ s) \ (Var \ k) \ t))$   
 $\langle proof \rangle$

**nominal\_termination**  $(eqvt)$   
 $\langle proof \rangle$

**lemma**  
**shows**  $CTermP\_fresh\_iff \ [simp]: a \ \# \ CTermP \ vf \ t \longleftrightarrow a \ \# \ t$  **(is ?thesis1)**  
**and**  $CTermP\_sf \ [iff]: Sigma\_fm \ (CTermP \ vf \ t)$  **(is ?thsf)**  
 $\langle proof \rangle$

**lemma**  $CTermP\_subst \ [simp]: (CTermP \ vf \ i)(j::=w) = CTermP \ vf \ (subst \ j \ w \ i)$   
 $\langle proof \rangle$

**abbreviation**  $TermP :: tm \Rightarrow fm$   
**where**  $TermP \equiv CTermP \ \text{True}$

**abbreviation**  $ConstP :: tm \Rightarrow fm$   
**where**  $ConstP \equiv CTermP \ \text{False}$

### 5.3.2 Correctness properties for constants

**lemma**  $ConstP\_imp\_TermP: \{ConstP \ t\} \vdash TermP \ t$   
 $\langle proof \rangle$

## 5.4 Abstraction over terms

**nominal\_function**  $SeqStTermP :: tm \Rightarrow tm \Rightarrow tm \Rightarrow tm \Rightarrow tm \Rightarrow tm \Rightarrow fm$   
**where**  $\llbracket atom \ l \ \# \ (s,k,v,i,sl,sl',m,n,sm,sm',sn,sn');$   
 $atom \ sl \ \# \ (s,v,i,sl',m,n,sm,sm',sn,sn'); atom \ sl' \ \# \ (s,v,i,m,n,sm,sm',sn,sn');$   
 $atom \ m \ \# \ (s,n,sm,sm',sn,sn'); atom \ n \ \# \ (s,sm,sm',sn,sn');$   
 $atom \ sm \ \# \ (s,sm',sn,sn'); atom \ sm' \ \# \ (s,sn,sn');$   
 $atom \ sn \ \# \ (s,sn'); atom \ sn' \ \# \ s \rrbracket \Longrightarrow$   
 $SeqStTermP \ v \ i \ t \ u \ s \ k =$   
 $VarP \ v \ AND \ LstSeqP \ s \ k \ (HPair \ t \ u) \ AND$   
 $All2 \ l \ (SUCC \ k) \ (Ex \ sl \ (Ex \ sl' \ (HPair \ (Var \ l) \ (HPair \ (Var \ sl) \ (Var \ sl')) \ IN \ s \ AND$   
 $((Var \ sl \ EQ \ v \ AND \ Var \ sl' \ EQ \ i) \ OR$   
 $((IndP \ (Var \ sl) \ OR \ Var \ sl \ NEQ \ v) \ AND \ Var \ sl' \ EQ \ Var \ sl)) \ OR$   
 $Ex \ m \ (Ex \ n \ (Ex \ sm \ (Ex \ sm' \ (Ex \ sn \ (Ex \ sn' \ (Var \ m \ IN \ Var \ l \ AND \ Var \ n \ IN \ Var \ l \ AND$   
 $HPair \ (Var \ m) \ (HPair \ (Var \ sm) \ (Var \ sm')) \ IN \ s \ AND$   
 $HPair \ (Var \ n) \ (HPair \ (Var \ sn) \ (Var \ sn')) \ IN \ s \ AND$   
 $Var \ sl \ EQ \ Q\_Eats \ (Var \ sm) \ (Var \ sn) \ AND$

$Var\ sl'\ EQ\ Q\_Eats\ (Var\ sm'\ (Var\ sn'\ ))))))))$

$\langle proof \rangle$

**nominal\_termination** (*eqvt*)

$\langle proof \rangle$

**lemma**

**shows** *SeqStTermP\_fresh\_iff* [*simp*]:  
 $a \# SeqStTermP\ v\ i\ t\ u\ s\ k \longleftrightarrow a \# v \wedge a \# i \wedge a \# t \wedge a \# u \wedge a \# s \wedge a \# k$  (**is** *?thesis1*)  
**and** *SeqStTermP\_sf* [*iff*]:  
 $\Sigma_{fm}\ (SeqStTermP\ v\ i\ t\ u\ s\ k)$  (**is** *?thsf*)  
**and** *SeqStTermP\_imp\_OrdP*:  
 $\{ SeqStTermP\ v\ i\ t\ u\ s\ k \} \vdash OrdP\ k$  (**is** *?thord*)  
**and** *SeqStTermP\_imp\_VarP*:  
 $\{ SeqStTermP\ v\ i\ t\ u\ s\ k \} \vdash VarP\ v$  (**is** *?thvar*)  
**and** *SeqStTermP\_imp\_LstSeqP*:  
 $\{ SeqStTermP\ v\ i\ t\ u\ s\ k \} \vdash LstSeqP\ s\ k\ (HPair\ t\ u)$  (**is** *?thlstseq*)

$\langle proof \rangle$

**lemma** *SeqStTermP\_subst* [*simp*]:

$(SeqStTermP\ v\ i\ t\ u\ s\ k)(j::=w) =$   
 $SeqStTermP\ (subst\ j\ w\ v)\ (subst\ j\ w\ i)\ (subst\ j\ w\ t)\ (subst\ j\ w\ u)\ (subst\ j\ w\ s)\ (subst\ j\ w\ k)$

$\langle proof \rangle$

**lemma** *SeqStTermP\_cong*:

$\llbracket H \vdash t\ EQ\ t'; H \vdash u\ EQ\ u'; H \vdash s\ EQ\ s'; H \vdash k\ EQ\ k' \rrbracket$   
 $\implies H \vdash SeqStTermP\ v\ i\ t\ u\ s\ k\ IFF\ SeqStTermP\ v\ i\ t'\ u'\ s'\ k'$

$\langle proof \rangle$

**declare** *SeqStTermP.simps* [*simp del*]

### 5.4.1 Defining the syntax: main predicate

**nominal\_function** *AbstTermP* ::  $tm \Rightarrow tm \Rightarrow tm \Rightarrow tm \Rightarrow fm$

**where**  $\llbracket atom\ s\ \# (v,i,t,u,k); atom\ k\ \# (v,i,t,u) \rrbracket \implies$   
 $AbstTermP\ v\ i\ t\ u =$   
 $OrdP\ i\ AND\ Ex\ s\ (Ex\ k\ (SeqStTermP\ v\ (Q\_Ind\ i)\ t\ u\ (Var\ s)\ (Var\ k)))$

$\langle proof \rangle$

**nominal\_termination** (*eqvt*)

$\langle proof \rangle$

**lemma**

**shows** *AbstTermP\_fresh\_iff* [*simp*]:  
 $a \# AbstTermP\ v\ i\ t\ u \longleftrightarrow a \# v \wedge a \# i \wedge a \# t \wedge a \# u$  (**is** *?thesis1*)  
**and** *AbstTermP\_sf* [*iff*]:  
 $\Sigma_{fm}\ (AbstTermP\ v\ i\ t\ u)$  (**is** *?thsf*)  
**and** *AbstTermP\_imp\_VarP*:  
 $\{ AbstTermP\ v\ i\ t\ u \} \vdash VarP\ v$  (**is** *?thvar*)  
**and** *AbstTermP\_imp\_OrdP*:  
 $\{ AbstTermP\ v\ i\ t\ u \} \vdash OrdP\ i$  (**is** *?thord*)

$\langle proof \rangle$

**lemma** *AbstTermP\_subst* [*simp*]:

$(AbstTermP\ v\ i\ t\ u)(j::=w) = AbstTermP\ (subst\ j\ w\ v)\ (subst\ j\ w\ i)\ (subst\ j\ w\ t)\ (subst\ j\ w\ u)$

$\langle proof \rangle$

**declare** *AbstTermP.simps* [*simp del*]

## 5.5 Substitution over terms

### 5.5.1 Defining the syntax

**nominal\_function** *SubstTermP* ::  $tm \Rightarrow tm \Rightarrow tm \Rightarrow tm \Rightarrow fm$   
**where**  $\llbracket atom\ s\ \#(v,i,t,u,k); atom\ k\ \#(v,i,t,u) \rrbracket \Longrightarrow$   
 $SubstTermP\ v\ i\ t\ u = TermP\ i\ AND\ Ex\ s\ (Ex\ k\ (SeqStTermP\ v\ i\ t\ u\ (Var\ s)\ (Var\ k)))$   
 $\langle proof \rangle$

**nominal\_termination** (*eqvt*)  
 $\langle proof \rangle$

**lemma**

**shows** *SubstTermP\_fresh\_iff* [*simp*]:  
 $a\ \#(SubstTermP\ v\ i\ t\ u) \longleftrightarrow a\ \#v \wedge a\ \#i \wedge a\ \#t \wedge a\ \#u$  (**is** *?thesis1*)  
**and** *SubstTermP\_sf* [*iff*]:  
 $Sigma\_fm\ (SubstTermP\ v\ i\ t\ u)$  (**is** *?thsf*)  
**and** *SubstTermP\_imp\_TermP*:  
 $\{ SubstTermP\ v\ i\ t\ u \} \vdash TermP\ i$  (**is** *?thterm*)  
**and** *SubstTermP\_imp\_VarP*:  
 $\{ SubstTermP\ v\ i\ t\ u \} \vdash VarP\ v$  (**is** *?thvar*)  
 $\langle proof \rangle$

**lemma** *SubstTermP\_subst* [*simp*]:  
 $(SubstTermP\ v\ i\ t\ u)(j::=w) = SubstTermP\ (subst\ j\ w\ v)\ (subst\ j\ w\ i)\ (subst\ j\ w\ t)\ (subst\ j\ w\ u)$   
 $\langle proof \rangle$

**lemma** *SubstTermP\_cong*:  
 $\llbracket H \vdash v\ EQ\ v'; H \vdash i\ EQ\ i'; H \vdash t\ EQ\ t'; H \vdash u\ EQ\ u' \rrbracket$   
 $\Longrightarrow H \vdash SubstTermP\ v\ i\ t\ u\ IFF\ SubstTermP\ v'\ i'\ t'\ u'$   
 $\langle proof \rangle$

**declare** *SubstTermP.simps* [*simp del*]

## 5.6 Abstraction over formulas

### 5.6.1 The predicate *AbstAtomicP*

**nominal\_function** *AbstAtomicP* ::  $tm \Rightarrow tm \Rightarrow tm \Rightarrow tm \Rightarrow fm$   
**where**  $\llbracket atom\ t\ \#(v,i,y,y',t',u,u'); atom\ t'\ \#(v,i,y,y',u,u');$   
 $atom\ u\ \#(v,i,y,y',u'); atom\ u'\ \#(v,i,y,y') \rrbracket \Longrightarrow$   
 $AbstAtomicP\ v\ i\ y\ y' =$   
 $Ex\ t\ (Ex\ u\ (Ex\ t'\ (Ex\ u'$   
 $(AbstTermP\ v\ i\ (Var\ t)\ (Var\ t')\ AND\ AbstTermP\ v\ i\ (Var\ u)\ (Var\ u')\ AND$   
 $((y\ EQ\ Q\_Eq\ (Var\ t)\ (Var\ u)\ AND\ y'\ EQ\ Q\_Eq\ (Var\ t')\ (Var\ u'))\ OR$   
 $(y\ EQ\ Q\_Mem\ (Var\ t)\ (Var\ u)\ AND\ y'\ EQ\ Q\_Mem\ (Var\ t')\ (Var\ u'))))))))$   
 $\langle proof \rangle$

**nominal\_termination** (*eqvt*)  
 $\langle proof \rangle$

**lemma**

**shows** *AbstAtomicP\_fresh\_iff* [*simp*]:  
 $a\ \#(AbstAtomicP\ v\ i\ y\ y') \longleftrightarrow a\ \#v \wedge a\ \#i \wedge a\ \#y \wedge a\ \#y'$  (**is** *?thesis1*)  
**and** *AbstAtomicP\_sf* [*iff*]:  $Sigma\_fm\ (AbstAtomicP\ v\ i\ y\ y')$  (**is** *?thsf*)  
 $\langle proof \rangle$

**lemma** *AbstAtomicP\_subst* [*simp*]:

(*AbstAtomicP* *v tm y y'*)(*i::=w*) = *AbstAtomicP* (*subst i w v*) (*subst i w tm*) (*subst i w y*) (*subst i w y'*)  
 <proof>

declare *AbstAtomicP.simps* [*simp del*]

## 5.6.2 The predicate *AbsMakeForm*

**nominal\_function** *SeqAbstFormP* :: *tm*  $\Rightarrow$  *tm*  $\Rightarrow$  *tm*  $\Rightarrow$  *tm*  $\Rightarrow$  *tm*  $\Rightarrow$  *tm*  $\Rightarrow$  *fm*

**where**  $\llbracket$  *atom l*  $\#$  (*s,k,v,sl,sl',m,n,smi,sm,sm',sni,sn,sn'*);  
*atom sli*  $\#$  (*s,v,sl,sl',m,n,smi,sm,sm',sni,sn,sn'*);  
*atom sl*  $\#$  (*s,v,sl',m,n,smi,sm,sm',sni,sn,sn'*);  
*atom sl'*  $\#$  (*s,v,m,n,smi,sm,sm',sni,sn,sn'*);  
*atom m*  $\#$  (*s,n,smi,sm,sm',sni,sn,sn'*);  
*atom n*  $\#$  (*s,smi,sm,sm',sni,sn,sn'*); *atom smi*  $\#$  (*s,sm,sm',sni,sn,sn'*);  
*atom sm*  $\#$  (*s,sm',sni,sn,sn'*); *atom sm'*  $\#$  (*s,sni,sn,sn'*);  
*atom sni*  $\#$  (*s,sn,sn'*); *atom sn*  $\#$  (*s,sn'*); *atom sn'*  $\#$  (*s*)  $\rrbracket \implies$

*SeqAbstFormP v i x x' s k* =  
*LstSeqP s k* (*HPair i* (*HPair x x'*)) *AND*  
*All2 l* (*SUCC k*) (*Ex sli* (*Ex sl* (*Ex sl'* (*HPair* (*Var l*) (*HPair* (*Var sli*) (*HPair* (*Var sl*) (*Var sl'*)))))  
*IN s AND*

(*AbstAtomicP v* (*Var sli*) (*Var sl*) (*Var sl'*) *OR*  
*OrdP* (*Var sli*) *AND*  
*Ex m* (*Ex n* (*Ex smi* (*Ex sm* (*Ex sm'* (*Ex sni* (*Ex sn* (*Ex sn'*  
 (*Var m IN Var l AND Var n IN Var l AND*  
*HPair* (*Var m*) (*HPair* (*Var smi*) (*HPair* (*Var sm*) (*Var sm'*)))) *IN s AND*  
*HPair* (*Var n*) (*HPair* (*Var sni*) (*HPair* (*Var sn*) (*Var sn'*)))) *IN s AND*  
 ((*Var sli EQ Var smi AND Var sli EQ Var sni AND*  
*Var sl EQ Q\_Disj* (*Var sm*) (*Var sn*) *AND*  
*Var sl' EQ Q\_Disj* (*Var sm'*) (*Var sn'*) *OR*  
 (*Var sli EQ Var smi AND*  
*Var sl EQ Q\_Neg* (*Var sm*) *AND Var sl' EQ Q\_Neg* (*Var sm'*) *OR*  
 (*SUCC* (*Var sli*) *EQ Var smi AND*  
*Var sl EQ Q\_Ex* (*Var sm*) *AND Var sl' EQ Q\_Ex* (*Var sm'*))))))))))))))

<proof>

**nominal\_termination** (*eqvt*)

<proof>

**lemma**

**shows** *SeqAbstFormP\_fresh\_iff* [*simp*]:

*a*  $\#$  *SeqAbstFormP v i x x' s k*  $\longleftrightarrow$  *a*  $\#$  *v*  $\wedge$  *a*  $\#$  *i*  $\wedge$  *a*  $\#$  *x*  $\wedge$  *a*  $\#$  *x'*  $\wedge$  *a*  $\#$  *s*  $\wedge$  *a*  $\#$  *k* (**is** *?thesis1*)

**and** *SeqAbstFormP\_sf* [*iff*]:

*Sigma\_fm* (*SeqAbstFormP v i x x' s k*) (**is** *?thsf*)

**and** *SeqAbstFormP\_imp\_OrdP*:

{ *SeqAbstFormP v u x x' s k* }  $\vdash$  *OrdP k* (**is** *?thOrd*)

**and** *SeqAbstFormP\_imp\_LstSeqP*:

{ *SeqAbstFormP v u x x' s k* }  $\vdash$  *LstSeqP s k* (*HPair u* (*HPair x x'*)) (**is** *?thLstSeq*)

<proof>

**lemma** *SeqAbstFormP\_subst* [*simp*]:

(*SeqAbstFormP v u x x' s k*)(*i::=t*) =

*SeqAbstFormP* (*subst i t v*) (*subst i t u*) (*subst i t x*) (*subst i t x'*) (*subst i t s*) (*subst i t k*)

<proof>

declare *SeqAbstFormP.simps* [*simp del*]

### 5.6.3 Defining the syntax: the main `AbstForm` predicate

**nominal\_function** `AbstFormP` ::  $tm \Rightarrow tm \Rightarrow tm \Rightarrow tm \Rightarrow fm$   
**where**  $\llbracket atom\ s\ \# (v, i, x, x', k);$   
 $atom\ k\ \# (v, i, x, x') \rrbracket \Longrightarrow$   
 $AbstFormP\ v\ i\ x\ x' = VarP\ v\ AND\ OrdP\ i\ AND\ Ex\ s\ (Ex\ k\ (SeqAbstFormP\ v\ i\ x\ x'\ (Var\ s)\ (Var\ k)))$   
 $\langle proof \rangle$

**nominal\_termination** (*eqvt*)  
 $\langle proof \rangle$

**lemma**  
**shows** `AbstFormP_fresh_iff` [*simp*]:  
 $a\ \# AbstFormP\ v\ i\ x\ x' \longleftrightarrow a\ \# v \wedge a\ \# i \wedge a\ \# x \wedge a\ \# x' \text{ (is ?thesis1)}$   
**and** `AbstFormP_sf` [*iff*]:  
 $Sigma\_fm\ (AbstFormP\ v\ i\ x\ x') \text{ (is ?thsf)}$   
 $\langle proof \rangle$

**lemma** `AbstFormP_subst` [*simp*]:  
 $(AbstFormP\ v\ i\ x\ x')(j::=t) = AbstFormP\ (subst\ j\ t\ v)\ (subst\ j\ t\ i)\ (subst\ j\ t\ x)\ (subst\ j\ t\ x')$   
 $\langle proof \rangle$

**declare** `AbstFormP.simps` [*simp del*]

## 5.7 Substitution over formulas

### 5.7.1 The predicate `SubstAtomicP`

**nominal\_function** `SubstAtomicP` ::  $tm \Rightarrow tm \Rightarrow tm \Rightarrow tm \Rightarrow fm$   
**where**  $\llbracket atom\ t\ \# (v, tm, y, y', t', u, u');$   
 $atom\ t'\ \# (v, tm, y, y', u, u');$   
 $atom\ u\ \# (v, tm, y, y', u');$   
 $atom\ u'\ \# (v, tm, y, y') \rrbracket \Longrightarrow$   
 $SubstAtomicP\ v\ tm\ y\ y' =$   
 $Ex\ t\ (Ex\ u\ (Ex\ t'\ (Ex\ u'$   
 $(SubstTermP\ v\ tm\ (Var\ t)\ (Var\ t')\ AND\ SubstTermP\ v\ tm\ (Var\ u)\ (Var\ u')\ AND$   
 $((y\ EQ\ Q\_Eq\ (Var\ t)\ (Var\ u)\ AND\ y'\ EQ\ Q\_Eq\ (Var\ t')\ (Var\ u'))\ OR$   
 $(y\ EQ\ Q\_Mem\ (Var\ t)\ (Var\ u)\ AND\ y'\ EQ\ Q\_Mem\ (Var\ t')\ (Var\ u'))))))))$   
 $\langle proof \rangle$

**nominal\_termination** (*eqvt*)  
 $\langle proof \rangle$

**lemma**  
**shows** `SubstAtomicP_fresh_iff` [*simp*]:  
 $a\ \# SubstAtomicP\ v\ tm\ y\ y' \longleftrightarrow a\ \# v \wedge a\ \# tm \wedge a\ \# y \wedge a\ \# y' \text{ (is ?thesis1)}$   
**and** `SubstAtomicP_sf` [*iff*]:  $Sigma\_fm\ (SubstAtomicP\ v\ tm\ y\ y') \text{ (is ?thsf)}$   
 $\langle proof \rangle$

**lemma** `SubstAtomicP_subst` [*simp*]:  
 $(SubstAtomicP\ v\ tm\ y\ y')(i::=w) = SubstAtomicP\ (subst\ i\ w\ v)\ (subst\ i\ w\ tm)\ (subst\ i\ w\ y)\ (subst\ i\ w\ y')$   
 $\langle proof \rangle$

**lemma** `SubstAtomicP_cong`:  
 $\llbracket H \vdash v\ EQ\ v'; H \vdash tm\ EQ\ tm'; H \vdash x\ EQ\ x'; H \vdash y\ EQ\ y' \rrbracket$   
 $\Longrightarrow H \vdash SubstAtomicP\ v\ tm\ x\ y\ IFF\ SubstAtomicP\ v'\ tm'\ x'\ y'$   
 $\langle proof \rangle$

## 5.7.2 The predicate *SubstMakeForm*

**nominal\_function** *SeqSubstFormP* ::  $tm \Rightarrow tm \Rightarrow tm \Rightarrow tm \Rightarrow tm \Rightarrow tm \Rightarrow fm$   
**where**  $\llbracket atom\ s \# (s,k,v,u,sl,sl',m,n,sm,sm',sn,sn') ;$   
 $atom\ sl \# (s,v,u,sl',m,n,sm,sm',sn,sn') ;$   
 $atom\ sl' \# (s,v,u,m,n,sm,sm',sn,sn') ;$   
 $atom\ m \# (s,n,sm,sm',sn,sn') ; atom\ n \# (s,sm,sm',sn,sn') ;$   
 $atom\ sm \# (s,sm',sn,sn') ; atom\ sm' \# (s,sn,sn') ;$   
 $atom\ sn \# (s,sn') ; atom\ sn' \# s \rrbracket \implies$   
*SeqSubstFormP*  $v\ u\ x\ x'\ s\ k =$   
 $LstSeqP\ s\ k\ (HPair\ x\ x')\ AND$   
 $All2\ l\ (SUCC\ k)\ (Ex\ sl\ (Ex\ sl'\ (HPair\ (Var\ l)\ (HPair\ (Var\ sl)\ (Var\ sl'))\ IN\ s\ AND$   
 $(SubstAtomicP\ v\ u\ (Var\ sl)\ (Var\ sl')\ OR$   
 $Ex\ m\ (Ex\ n\ (Ex\ sm\ (Ex\ sm'\ (Ex\ sn\ (Ex\ sn'\ (Var\ m\ IN\ Var\ l\ AND\ Var\ n\ IN\ Var\ l\ AND$   
 $HPair\ (Var\ m)\ (HPair\ (Var\ sm)\ (Var\ sm'))\ IN\ s\ AND$   
 $HPair\ (Var\ n)\ (HPair\ (Var\ sn)\ (Var\ sn'))\ IN\ s\ AND$   
 $((Var\ sl\ EQ\ Q\_Disj\ (Var\ sm)\ (Var\ sn)\ AND$   
 $Var\ sl'\ EQ\ Q\_Disj\ (Var\ sm')\ (Var\ sn'))\ OR$   
 $(Var\ sl\ EQ\ Q\_Neg\ (Var\ sm)\ AND\ Var\ sl'\ EQ\ Q\_Neg\ (Var\ sm'))\ OR$   
 $(Var\ sl\ EQ\ Q\_Ex\ (Var\ sm)\ AND\ Var\ sl'\ EQ\ Q\_Ex\ (Var\ sm'))))))))))))$   
 $\langle proof \rangle$

**nominal\_termination** (*eqvt*)  
 $\langle proof \rangle$

**lemma**  
**shows** *SeqSubstFormP\_fresh\_iff* [*simp*]:  
 $a \# SeqSubstFormP\ v\ u\ x\ x'\ s\ k \longleftrightarrow a \# v \wedge a \# u \wedge a \# x \wedge a \# x' \wedge a \# s \wedge a \# k$  (**is** *?thesis1*)  
**and** *SeqSubstFormP\_sf* [*iff*]:  
 $Sigma\_fm\ (SeqSubstFormP\ v\ u\ x\ x'\ s\ k)$  (**is** *?thsf*)  
**and** *SeqSubstFormP\_imp\_OrdP*:  
 $\{ SeqSubstFormP\ v\ u\ x\ x'\ s\ k \} \vdash OrdP\ k$  (**is** *?thOrd*)  
**and** *SeqSubstFormP\_imp\_LstSeqP*:  
 $\{ SeqSubstFormP\ v\ u\ x\ x'\ s\ k \} \vdash LstSeqP\ s\ k\ (HPair\ x\ x')$  (**is** *?thLstSeq*)  
 $\langle proof \rangle$

**lemma** *SeqSubstFormP\_subst* [*simp*]:  
 $(SeqSubstFormP\ v\ u\ x\ x'\ s\ k)(i::=t) =$   
 $SeqSubstFormP\ (subst\ i\ t\ v)\ (subst\ i\ t\ u)\ (subst\ i\ t\ x)\ (subst\ i\ t\ x')\ (subst\ i\ t\ s)\ (subst\ i\ t\ k)$   
 $\langle proof \rangle$

**lemma** *SeqSubstFormP\_cong*:  
 $\llbracket H \vdash t\ EQ\ t' ; H \vdash u\ EQ\ u' ; H \vdash s\ EQ\ s' ; H \vdash k\ EQ\ k \rrbracket$   
 $\implies H \vdash SeqSubstFormP\ v\ i\ t\ u\ s\ k\ IFF\ SeqSubstFormP\ v\ i\ t'\ u'\ s'\ k'$   
 $\langle proof \rangle$

**declare** *SeqSubstFormP.simps* [*simp del*]

## 5.7.3 Defining the syntax: the main *SubstForm* predicate

**nominal\_function** *SubstFormP* ::  $tm \Rightarrow tm \Rightarrow tm \Rightarrow tm \Rightarrow fm$   
**where**  $\llbracket atom\ s \# (v,i,x,x',k) ; atom\ k \# (v,i,x,x') \rrbracket \implies$   
 $SubstFormP\ v\ i\ x\ x' =$   
 $VarP\ v\ AND\ TermP\ i\ AND\ Ex\ s\ (Ex\ k\ (SeqSubstFormP\ v\ i\ x\ x'\ (Var\ s)\ (Var\ k)))$   
 $\langle proof \rangle$

**nominal\_termination** (*eqvt*)  
 $\langle proof \rangle$

**lemma**

**shows** *SubstFormP\_fresh\_iff* [simp]:

$a \# \text{SubstFormP } v \ i \ x \ x' \longleftrightarrow a \# v \wedge a \# i \wedge a \# x \wedge a \# x' \text{ (is ?thesis1)}$

**and** *SubstFormP\_sf* [iff]:

$\text{Sigma\_fm } (\text{SubstFormP } v \ i \ x \ x') \text{ (is ?thsf)}$

*<proof>*

**lemma** *SubstFormP\_subst* [simp]:

$(\text{SubstFormP } v \ i \ x \ x')(j::=t) = \text{SubstFormP } (\text{subst } j \ t \ v) \ (\text{subst } j \ t \ i) \ (\text{subst } j \ t \ x) \ (\text{subst } j \ t \ x')$

*<proof>*

**lemma** *SubstFormP\_cong*:

$\llbracket H \vdash v \text{ EQ } v'; H \vdash i \text{ EQ } i'; H \vdash t \text{ EQ } t'; H \vdash u \text{ EQ } u' \rrbracket$

$\implies H \vdash \text{SubstFormP } v \ i \ t \ u \text{ IFF } \text{SubstFormP } v' \ i' \ t' \ u'$

*<proof>*

**lemma** *ground\_SubstFormP* [simp]:  $\text{ground\_fm } (\text{SubstFormP } v \ y \ x \ x') \longleftrightarrow \text{ground } v \wedge \text{ground } y \wedge \text{ground } x \wedge \text{ground } x'$

*<proof>*

**declare** *SubstFormP.simps* [simp del]

## 5.8 The predicate *AtomicP*

**nominal\_function** *AtomicP* :: *tm*  $\Rightarrow$  *fm*

**where**  $\llbracket \text{atom } t \# (u, y); \text{atom } u \# y \rrbracket \implies$

$\text{AtomicP } y = \text{Ex } t \ (\text{Ex } u \ (\text{TermP } (\text{Var } t) \text{ AND } \text{TermP } (\text{Var } u) \text{ AND}$   
 $(y \text{ EQ } Q\_Eq \ (\text{Var } t) \ (\text{Var } u) \text{ OR}$   
 $y \text{ EQ } Q\_Mem \ (\text{Var } t) \ (\text{Var } u))))$

*<proof>*

**nominal\_termination** (*eqvt*)

*<proof>*

**lemma**

**shows** *AtomicP\_fresh\_iff* [simp]:  $a \# \text{AtomicP } y \longleftrightarrow a \# y \text{ (is ?thesis1)}$

**and** *AtomicP\_sf* [iff]:  $\text{Sigma\_fm } (\text{AtomicP } y) \text{ (is ?thsf)}$

*<proof>*

**lemma** *AtomicP\_subst* [simp]:  $(\text{AtomicP } t)(j::=w) = \text{AtomicP } (\text{subst } j \ w \ t)$

*<proof>*

## 5.9 The predicate *MakeForm*

**nominal\_function** *MakeFormP* :: *tm*  $\Rightarrow$  *tm*  $\Rightarrow$  *tm*  $\Rightarrow$  *fm*

**where**  $\llbracket \text{atom } v \# (y, u, w, au); \text{atom } au \# (y, u, w) \rrbracket \implies$

$\text{MakeFormP } y \ u \ w =$

$y \text{ EQ } Q\_Disj \ u \ w \text{ OR } y \text{ EQ } Q\_Neg \ u \text{ OR}$

$\text{Ex } v \ (\text{Ex } au \ (\text{AbstFormP } (\text{Var } v) \ \text{Zero } u \ (\text{Var } au) \text{ AND } y \text{ EQ } Q\_Ex \ (\text{Var } au)))$

*<proof>*

**nominal\_termination** (*eqvt*)

*<proof>*

**lemma**

**shows** *MakeFormP\_fresh\_iff* [simp]:

$a \# \text{MakeFormP } y \ u \ w \longleftrightarrow a \# y \wedge a \# u \wedge a \# w \text{ (is ?thesis1)}$

**and** *MakeFormP\_sf* [iff]:  
 $\text{Sigma\_fm } (\text{MakeFormP } y \ u \ w) \ (\text{is } ?\text{thsf})$   
 ⟨proof⟩

**declare** *MakeFormP.simps* [simp del]

**lemma** *MakeFormP\_subst* [simp]:  $(\text{MakeFormP } y \ u \ t)(j::=w) = \text{MakeFormP } (\text{subst } j \ w \ y) \ (\text{subst } j \ w \ u)$   
 ⟨proof⟩

## 5.10 The predicate *SeqFormP*

**nominal\_function** *SeqFormP* ::  $tm \Rightarrow tm \Rightarrow tm \Rightarrow fm$   
**where**  $\llbracket \text{atom } l \ \# \ (s,k,t,sl,m,n,sm,sn); \text{atom } sl \ \# \ (s,k,t,m,n,sm,sn);$   
 $\text{atom } m \ \# \ (s,k,t,n,sm,sn); \text{atom } n \ \# \ (s,k,t,sm,sn);$   
 $\text{atom } sm \ \# \ (s,k,t,sn); \text{atom } sn \ \# \ (s,k,t) \rrbracket \Longrightarrow$   
 $\text{SeqFormP } s \ k \ t =$   
 $\text{LstSeqP } s \ k \ t \ \text{AND}$   
 $\text{All2 } n \ (\text{SUCC } k) \ (\text{Ex } sn \ (\text{HPair } (\text{Var } n) \ (\text{Var } sn) \ \text{IN } s \ \text{AND } (\text{AtomicP } (\text{Var } sn) \ \text{OR}$   
 $\text{Ex } m \ (\text{Ex } l \ (\text{Ex } sm \ (\text{Ex } sl \ (\text{Var } m \ \text{IN } \text{Var } n \ \text{AND } \text{Var } l \ \text{IN } \text{Var } n \ \text{AND}$   
 $\text{HPair } (\text{Var } m) \ (\text{Var } sm) \ \text{IN } s \ \text{AND } \text{HPair } (\text{Var } l) \ (\text{Var } sl) \ \text{IN } s \ \text{AND}$   
 $\text{MakeFormP } (\text{Var } sn) \ (\text{Var } sm) \ (\text{Var } sl))))))$   
 ⟨proof⟩

**nominal\_termination** (*eqvt*)  
 ⟨proof⟩

**lemma**  
**shows** *SeqFormP\_fresh\_iff* [simp]:  
 $a \ \# \ \text{SeqFormP } s \ k \ t \longleftrightarrow a \ \# \ s \wedge a \ \# \ k \wedge a \ \# \ t \ (\text{is } ?\text{thesis1})$   
**and** *SeqFormP\_sf* [iff]:  $\text{Sigma\_fm } (\text{SeqFormP } s \ k \ t) \ (\text{is } ?\text{thsf})$   
**and** *SeqFormP\_imp\_OrdP*:  
 $\{ \text{SeqFormP } s \ k \ t \} \vdash \text{OrdP } k \ (\text{is } ?\text{thOrd})$   
**and** *SeqFormP\_imp\_LstSeqP*:  
 $\{ \text{SeqFormP } s \ k \ t \} \vdash \text{LstSeqP } s \ k \ t \ (\text{is } ?\text{thLstSeq})$   
 ⟨proof⟩

**lemma** *SeqFormP\_subst* [simp]:  
 $(\text{SeqFormP } s \ k \ t)(j::=w) = \text{SeqFormP } (\text{subst } j \ w \ s) \ (\text{subst } j \ w \ k) \ (\text{subst } j \ w \ t)$   
 ⟨proof⟩

## 5.11 The predicate *FormP*

### 5.11.1 Definition

**nominal\_function** *FormP* ::  $tm \Rightarrow fm$   
**where**  $\llbracket \text{atom } k \ \# \ (s,y); \text{atom } s \ \# \ y \rrbracket \Longrightarrow$   
 $\text{FormP } y = \text{Ex } k \ (\text{Ex } s \ (\text{SeqFormP } (\text{Var } s) \ (\text{Var } k) \ y))$   
 ⟨proof⟩

**nominal\_termination** (*eqvt*)  
 ⟨proof⟩

**lemma**  
**shows** *FormP\_fresh\_iff* [simp]:  $a \ \# \ \text{FormP } y \longleftrightarrow a \ \# \ y \ (\text{is } ?\text{thesis1})$   
**and** *FormP\_sf* [iff]:  $\text{Sigma\_fm } (\text{FormP } y) \ (\text{is } ?\text{thsf})$   
 ⟨proof⟩

**lemma** *FormP\_subst* [simp]:  $(FormP\ y)(j::=w) = FormP\ (subst\ j\ w\ y)$   
⟨proof⟩

### 5.11.2 The predicate *VarNonOccFormP* (Derived from *SubstFormP*)

**nominal\_function** *VarNonOccFormP* ::  $tm \Rightarrow tm \Rightarrow fm$   
**where** *VarNonOccFormP*  $v\ x = FormP\ x\ AND\ SubstFormP\ v\ Zero\ x\ x$   
⟨proof⟩

**nominal\_termination** (*eqvt*)  
⟨proof⟩

**lemma**  
**shows** *VarNonOccFormP\_fresh\_iff* [simp]:  $a \# VarNonOccFormP\ v\ y \longleftrightarrow a \# v \wedge a \# y$  (**is** *?thesis1*)  
**and** *VarNonOccFormP\_sf* [iff]: *Sigma\_fm* (*VarNonOccFormP*  $v\ y$ ) (**is** *?thsf*)  
⟨proof⟩

**declare** *VarNonOccFormP.simps* [simp del]

**end**

## Chapter 6

# Formalizing Provability

```
theory Pf_Predicates
imports Coding_Predicates
begin
```

### 6.1 Section 4 Predicates (Leading up to Pf)

#### 6.1.1 The predicate *SentP*, for the Sentential (Boolean) Axioms

```
nominal_function SentP :: tm ⇒ fm
where [[atom y ‡ (z,w,x); atom z ‡ (w,x); atom w ‡ x]] ⇒
  SentP x = Ex y (Ex z (Ex w (FormP (Var y) AND FormP (Var z) AND FormP (Var w) AND
    ( (x EQ Q_Imp (Var y) (Var y)) OR
      (x EQ Q_Imp (Var y) (Q_Disj (Var y) (Var z)) OR
        (x EQ Q_Imp (Q_Disj (Var y) (Var y)) (Var y)) OR
          (x EQ Q_Imp (Q_Disj (Var y) (Q_Disj (Var z) (Var w)))
            (Q_Disj (Q_Disj (Var y) (Var z)) (Var w))) OR
            (x EQ Q_Imp (Q_Disj (Var y) (Var z))
              (Q_Imp (Q_Disj (Q_Neg (Var y)) (Var w)) (Q_Disj (Var z) (Var w))))))))))
  ⟨proof⟩

nominal_termination (eqvt)
  ⟨proof⟩
```

lemma

```
shows SentP_fresh_iff [simp]: a ‡ SentP x ↔ a ‡ x          (is ?thesis1)
and SentP_sf [iff]:      Sigma_fm (SentP x)                (is ?thsf)
  ⟨proof⟩
```

#### 6.1.2 The predicate *Equality\_axP*, for the Equality Axioms

```
function Equality_axP :: tm ⇒ fm
where Equality_axP x =
  x EQ «refl_ax» OR x EQ «eq_cong_ax» OR x EQ «mem_cong_ax» OR x EQ «eats_cong_ax»
  ⟨proof⟩

termination
  ⟨proof⟩
```

#### 6.1.3 The predicate *HF\_axP*, for the HF Axioms

```
function HF_axP :: tm ⇒ fm
where HF_axP x = x EQ «HF1» OR x EQ «HF2»
```

*<proof>*

**termination**

*<proof>*

**lemma** *HF\_axP\_sf [iff]: Sigma\_fm (HF\_axP t)*

*<proof>*

## 6.1.4 The specialisation axioms

**Defining the syntax**

**nominal\_function** *Special\_axP :: tm  $\Rightarrow$  fm where*

*[[atom v # (p,sx,y,ax,x); atom x # (p,sx,y,ax);  
atom ax # (p,sx,y); atom y # (p,sx); atom sx # p]]  $\implies$   
Special\_axP p = Ex v (Ex x (Ex ax (Ex y (Ex sx  
(FormP (Var x) AND VarP (Var v) AND TermP (Var y) AND  
AbstFormP (Var v) Zero (Var x) (Var ax) AND  
SubstFormP (Var v) (Var y) (Var x) (Var sx) AND  
p EQ Q\_Imp (Var sx) (Q\_Ex (Var ax))))))))))*

*<proof>*

**nominal\_termination** (*eqvt*)

*<proof>*

**lemma**

**shows** *Special\_axP\_fresh\_iff [simp]: a # Special\_axP p  $\longleftrightarrow$  a # p (is ?thesis1)*

**and** *Special\_axP\_sf [iff]: Sigma\_fm (Special\_axP p) (is ?thesis3)*

*<proof>*

## 6.1.5 The induction axioms

**Defining the syntax**

**nominal\_function** *Induction\_axP :: tm  $\Rightarrow$  fm where*

*[[atom ax # (p,v,w,x,x0,xw,xevw,allw,allvw);  
atom allvw # (p,v,w,x,x0,xw,xevw,allw); atom allw # (p,v,w,x,x0,xw,xevw);  
atom xevw # (p,v,w,x,x0,xw); atom xw # (p,v,w,x,x0);  
atom x0 # (p,v,w,x); atom x # (p,v,w);  
atom w # (p,v); atom v # p]]  $\implies$   
Induction\_axP p = Ex v (Ex w (Ex x (Ex x0 (Ex xw (Ex xevw (Ex allw (Ex allvw (Ex ax  
((Var v NEQ Var w) AND VarNonOccFormP (Var w) (Var x) AND  
SubstFormP (Var v) Zero (Var x) (Var x0) AND  
SubstFormP (Var v) (Var w) (Var x) (Var xw) AND  
SubstFormP (Var v) (Q\_Eats (Var v) (Var w)) (Var x) (Var xevw) AND  
AbstFormP (Var w) Zero (Q\_Imp (Var x) (Q\_Imp (Var xw) (Var xevw))) (Var allw) AND  
AbstFormP (Var v) Zero (Q\_All (Var allw)) (Var allvw) AND  
AbstFormP (Var v) Zero (Var x) (Var ax) AND  
p EQ Q\_Imp (Var x0) (Q\_Imp (Q\_All (Var allvw)) (Q\_All (Var ax))))))))))))))*

*<proof>*

**nominal\_termination** (*eqvt*)

*<proof>*

**lemma**

**shows** *Induction\_axP\_fresh\_iff [simp]: a # Induction\_axP p  $\longleftrightarrow$  a # p (is ?thesis1)*

**and** *Induction\_axP\_sf [iff]: Sigma\_fm (Induction\_axP p) (is ?thesis3)*

*<proof>*

### 6.1.6 The predicate $AxiomP$ , for any Axioms

**definition**  $AxiomP :: tm \Rightarrow fm$

**where**  $AxiomP x \equiv x EQ \langle extra\_axiom \rangle OR SentP x OR Equality\_axP x OR HF\_axP x OR Special\_axP x OR Induction\_axP x$

**lemma**  $AxiomP\_I$ :

$\{\} \vdash AxiomP \langle extra\_axiom \rangle$   
 $\{\} \vdash SentP x \implies \{\} \vdash AxiomP x$   
 $\{\} \vdash Equality\_axP x \implies \{\} \vdash AxiomP x$   
 $\{\} \vdash HF\_axP x \implies \{\} \vdash AxiomP x$   
 $\{\} \vdash Special\_axP x \implies \{\} \vdash AxiomP x$   
 $\{\} \vdash Induction\_axP x \implies \{\} \vdash AxiomP x$   
 $\langle proof \rangle$

**lemma**  $AxiomP\_eqvt [eqvt]: (p \cdot AxiomP x) = AxiomP (p \cdot x)$

$\langle proof \rangle$

**lemma**  $AxiomP\_fresh\_iff [simp]: a \# AxiomP x \longleftrightarrow a \# x$

$\langle proof \rangle$

**lemma**  $AxiomP\_sf [iff]: Sigma\_fm (AxiomP t)$

$\langle proof \rangle$

### 6.1.7 The predicate $ModPonP$ , for the inference rule Modus Ponens

**definition**  $ModPonP :: tm \Rightarrow tm \Rightarrow tm \Rightarrow fm$

**where**  $ModPonP x y z = (y EQ Q\_Imp x z)$

**lemma**  $ModPonP\_eqvt [eqvt]: (p \cdot ModPonP x y z) = ModPonP (p \cdot x) (p \cdot y) (p \cdot z)$

$\langle proof \rangle$

**lemma**  $ModPonP\_fresh\_iff [simp]: a \# ModPonP x y z \longleftrightarrow a \# x \wedge a \# y \wedge a \# z$

$\langle proof \rangle$

**lemma**  $ModPonP\_sf [iff]: Sigma\_fm (ModPonP t u v)$

$\langle proof \rangle$

**lemma**  $ModPonP\_subst [simp]:$

$(ModPonP t u v)(i::=w) = ModPonP (subst i w t) (subst i w u) (subst i w v)$

$\langle proof \rangle$

### 6.1.8 The predicate $ExistsP$ , for the existential rule

**Definition**

**nominal\_function**  $ExistsP :: tm \Rightarrow tm \Rightarrow fm$  **where**

$\llbracket atom x \# (p, q, v, y, x'); atom x' \# (p, q, v, y);$   
 $atom y \# (p, q, v); atom v \# (p, q) \rrbracket \implies$   
 $ExistsP p q = Ex x (Ex x' (Ex y (Ex v (FormP (Var x) AND$   
 $VarNonOccFormP (Var v) (Var y) AND$   
 $AbstFormP (Var v) Zero (Var x) (Var x') AND$   
 $p EQ Q\_Imp (Var x) (Var y) AND$   
 $q EQ Q\_Imp (Q\_Ex (Var x')) (Var y))))))$

$\langle proof \rangle$

**nominal\_termination** ( $eqvt$ )

$\langle proof \rangle$

**lemma**

**shows**  $ExistsP\_fresh\_iff$  [simp]:  $a \# ExistsP\ p\ q \longleftrightarrow a \# p \wedge a \# q$  (is ?thesis1)  
**and**  $ExistsP\_sf$  [iff]:  $Sigma\_fm\ (ExistsP\ p\ q)$  (is ?thesis3)  
<proof>

**lemma**  $ExistsP\_subst$  [simp]:  $(ExistsP\ p\ q)(j::=w) = ExistsP\ (subst\ j\ w\ p)\ (subst\ j\ w\ q)$   
<proof>

### 6.1.9 The predicate $SubstP$ , for the substitution rule

Although the substitution rule is derivable in the calculus, the derivation is too complicated to reproduce within the proof function. It is much easier to provide it as an immediate inference step, justifying its soundness in terms of other inference rules.

**Definition**

**nominal\_function**  $SubstP :: tm \Rightarrow tm \Rightarrow fm$  **where**

$\llbracket atom\ u\ \# (p,q,v); atom\ v\ \# (p,q) \rrbracket \Longrightarrow$   
 $SubstP\ p\ q = Ex\ v\ (Ex\ u\ (SubstFormP\ (Var\ v)\ (Var\ u)\ p\ q))$   
<proof>

**nominal\_termination** (eqvt)  
<proof>

**lemma**

**shows**  $SubstP\_fresh\_iff$  [simp]:  $a \# SubstP\ p\ q \longleftrightarrow a \# p \wedge a \# q$  (is ?thesis1)  
**and**  $SubstP\_sf$  [iff]:  $Sigma\_fm\ (SubstP\ p\ q)$  (is ?thesis3)  
<proof>

**lemma**  $SubstP\_subst$  [simp]:  $(SubstP\ p\ q)(j::=w) = SubstP\ (subst\ j\ w\ p)\ (subst\ j\ w\ q)$   
<proof>

### 6.1.10 The predicate $PrfP$

**nominal\_function**  $PrfP :: tm \Rightarrow tm \Rightarrow tm \Rightarrow fm$

**where**  $\llbracket atom\ l\ \# (s,sl,m,n,sm,sn); atom\ sl\ \# (s,m,n,sm,sn);$   
 $atom\ m\ \# (s,n,sm,sn); atom\ n\ \# (s,k,sm,sn);$   
 $atom\ sm\ \# (s,sn); atom\ sn\ \# (s) \rrbracket \Longrightarrow$

$PrfP\ s\ k\ t =$   
 $LstSeqP\ s\ k\ t\ AND$   
 $All2\ n\ (SUCC\ k)\ (Ex\ sn\ (HPair\ (Var\ n)\ (Var\ sn)\ IN\ s\ AND\ (AxiomP\ (Var\ sn)\ OR$   
 $Ex\ m\ (Ex\ l\ (Ex\ sm\ (Ex\ sl\ (Var\ m\ IN\ Var\ n\ AND\ Var\ l\ IN\ Var\ n\ AND$   
 $HPair\ (Var\ m)\ (Var\ sm)\ IN\ s\ AND\ HPair\ (Var\ l)\ (Var\ sl)\ IN\ s\ AND$   
 $(ModPonP\ (Var\ sm)\ (Var\ sl)\ (Var\ sn)\ OR$   
 $ExistsP\ (Var\ sm)\ (Var\ sn)\ OR$   
 $SubstP\ (Var\ sm)\ (Var\ sn))))))))))$

<proof>

**nominal\_termination** (eqvt)  
<proof>

**lemma**

**shows**  $PrfP\_fresh\_iff$  [simp]:  $a \# PrfP\ s\ k\ t \longleftrightarrow a \# s \wedge a \# k \wedge a \# t$  (is ?thesis1)  
**and**  $PrfP\_imp\_OrdP$  [simp]:  $\{PrfP\ s\ k\ t\} \vdash OrdP\ k$  (is ?thord)  
**and**  $PrfP\_imp\_LstSeqP$  [simp]:  $\{PrfP\ s\ k\ t\} \vdash LstSeqP\ s\ k\ t$  (is ?thlstseq)  
**and**  $PrfP\_sf$  [iff]:  $Sigma\_fm\ (PrfP\ s\ k\ t)$  (is ?thsf)  
<proof>

**lemma** *PrfP\_subst* [*simp*]:  
 $(PrfP\ t\ u\ v)(j::=w) = PrfP\ (subst\ j\ w\ t)\ (subst\ j\ w\ u)\ (subst\ j\ w\ v)$   
 ⟨*proof*⟩

### 6.1.11 The predicate *PfP*

**nominal\_function** *PfP* ::  $tm \Rightarrow fm$   
**where**  $\llbracket atom\ k\ \#\ (s,y); atom\ s\ \#\ y \rrbracket \Longrightarrow$   
 $PfP\ y = Ex\ k\ (Ex\ s\ (PrfP\ (Var\ s)\ (Var\ k)\ y))$   
 ⟨*proof*⟩

**nominal\_termination** (*eqvt*)  
 ⟨*proof*⟩

**lemma**  
**shows** *PfP\_fresh\_iff* [*simp*]:  $a\ \#\ PfP\ y \longleftrightarrow a\ \#\ y$  (is *?thesis1*)  
**and** *PfP\_sf* [*iff*]:  $Sigma\_fm\ (PfP\ y)$  (is *?thsf*)  
 ⟨*proof*⟩

**lemma** *PfP\_subst* [*simp*]:  $(PfP\ t)(j::=w) = PfP\ (subst\ j\ w\ t)$   
 ⟨*proof*⟩

**lemma** *ground\_PfP* [*simp*]:  $ground\_fm\ (PfP\ y) = ground\ y$   
 ⟨*proof*⟩

**end**

## Chapter 7

# Syntactic Preliminaries for the Second Incompleteness Theorem

```
theory II_Prelims
imports Pf_Predicates
begin
```

```
declare IndP.simps [simp del]
```

```
lemma OrdP_ORD_OF [intro]:  $H \vdash \text{OrdP } (\text{ORD\_OF } n)$ 
⟨proof⟩
```

```
lemma VarP_Var [intro]:  $H \vdash \text{VarP } \langle \text{Var } i \rangle$ 
⟨proof⟩
```

```
lemma VarP_neq_IndP:  $\{t \text{ EQ } v, \text{VarP } v, \text{IndP } t\} \vdash \text{Fls}$ 
⟨proof⟩
```

```
lemma Mem_HFun_Sigma_OrdP:  $\{\text{HPair } t \text{ u IN } f, \text{HFun\_Sigma } f\} \vdash \text{OrdP } t$ 
⟨proof⟩
```

### 7.1 NotInDom

```
nominal_function NotInDom ::  $tm \Rightarrow tm \Rightarrow fm$ 
  where  $\text{atom } z \# (t, r) \Longrightarrow \text{NotInDom } t \text{ r} = \text{All } z (\text{Neg } (\text{HPair } t (\text{Var } z) \text{ IN } r))$ 
⟨proof⟩
```

```
nominal_termination (eqvt)
⟨proof⟩
```

```
lemma NotInDom_fresh_iff [simp]:  $a \# \text{NotInDom } t \text{ r} \longleftrightarrow a \# (t, r)$ 
⟨proof⟩
```

```
lemma subst_fm_NotInDom [simp]:  $(\text{NotInDom } t \text{ r})(i::=x) = \text{NotInDom } (\text{subst } i \text{ x } t) (\text{subst } i \text{ x } r)$ 
⟨proof⟩
```

```
lemma NotInDom_cong:  $H \vdash t \text{ EQ } t' \Longrightarrow H \vdash r \text{ EQ } r' \Longrightarrow H \vdash \text{NotInDom } t \text{ r} \text{ IFF } \text{NotInDom } t' \text{ r}'$ 
⟨proof⟩
```

```
lemma NotInDom_Zero:  $H \vdash \text{NotInDom } t \text{ Zero}$ 
⟨proof⟩
```

**lemma** *NotInDom\_Fls*:  $\{HPair\ d\ d'\ IN\ r,\ NotInDom\ d\ r\} \vdash A$   
 ⟨proof⟩

**lemma** *NotInDom\_Contra*:  $H \vdash NotInDom\ d\ r \implies H \vdash HPair\ x\ y\ IN\ r \implies insert\ (x\ EQ\ d)\ H \vdash A$   
 ⟨proof⟩

## 7.2 Restriction of a Sequence to a Domain

**nominal\_function** *RestrictedP* ::  $tm \Rightarrow tm \Rightarrow tm \Rightarrow fm$   
**where**  $\llbracket atom\ x\ \# (y,f,k,g); atom\ y\ \# (f,k,g) \rrbracket \implies$   
 $RestrictedP\ f\ k\ g =$   
 $g\ SUBS\ f\ AND$   
 $All\ x\ (All\ y\ (HPair\ (Var\ x)\ (Var\ y)\ IN\ g\ IFF$   
 $(Var\ x)\ IN\ k\ AND\ HPair\ (Var\ x)\ (Var\ y)\ IN\ f))$   
 ⟨proof⟩

**nominal\_termination** (*eqvt*)  
 ⟨proof⟩

**lemma** *RestrictedP\_fresh\_iff* [*simp*]:  $a\ \# RestrictedP\ f\ k\ g \longleftrightarrow a\ \# f \wedge a\ \# k \wedge a\ \# g$   
 ⟨proof⟩

**lemma** *subst\_fm\_RestrictedP* [*simp*]:  
 $(RestrictedP\ f\ k\ g)(i::=u) = RestrictedP\ (subst\ i\ u\ f)\ (subst\ i\ u\ k)\ (subst\ i\ u\ g)$   
 ⟨proof⟩

**lemma** *RestrictedP\_cong*:  
 $\llbracket H \vdash f\ EQ\ f'; H \vdash k\ EQ\ A'; H \vdash g\ EQ\ g' \rrbracket$   
 $\implies H \vdash RestrictedP\ f\ k\ g\ IFF\ RestrictedP\ f'\ A'\ g'$   
 ⟨proof⟩

**lemma** *RestrictedP\_Zero*:  $H \vdash RestrictedP\ Zero\ k\ Zero$   
 ⟨proof⟩

**lemma** *RestrictedP\_Mem*:  $\{ RestrictedP\ s\ k\ s', HPair\ a\ b\ IN\ s,\ a\ IN\ k \} \vdash HPair\ a\ b\ IN\ s'$   
 ⟨proof⟩

**lemma** *RestrictedP\_imp\_Subset*:  $\{ RestrictedP\ s\ k\ s' \} \vdash s'\ SUBS\ s$   
 ⟨proof⟩

**lemma** *RestrictedP\_Mem2*:  
 $\{ RestrictedP\ s\ k\ s', HPair\ a\ b\ IN\ s' \} \vdash HPair\ a\ b\ IN\ s\ AND\ a\ IN\ k$   
 ⟨proof⟩

**lemma** *RestrictedP\_Mem\_D*:  $H \vdash RestrictedP\ s\ k\ t \implies H \vdash a\ IN\ t \implies insert\ (a\ IN\ s)\ H \vdash A \implies H \vdash A$   
 ⟨proof⟩

**lemma** *RestrictedP\_Eats*:  
 $\{ RestrictedP\ s\ k\ s', a\ IN\ k \} \vdash RestrictedP\ (Eats\ s\ (HPair\ a\ b))\ k\ (Eats\ s'\ (HPair\ a\ b))$  ⟨proof⟩

**lemma** *exists\_RestrictedP*:  
**assumes**  $s: atom\ s\ \# (f,k)$   
**shows**  $H \vdash Ex\ s\ (RestrictedP\ f\ k\ (Var\ s))$  ⟨proof⟩

**lemma** *cut\_RestrictedP*:  
**assumes**  $s: atom\ s\ \# (f,k,A)$  **and**  $\forall C \in H. atom\ s\ \# C$   
**shows**  $insert\ (RestrictedP\ f\ k\ (Var\ s))\ H \vdash A \implies H \vdash A$   
 ⟨proof⟩

**lemma** *RestrictedP\_NotInDom*: { *RestrictedP s k s'*, *Neg (j IN k)* }  $\vdash$  *NotInDom j s'*  
 ⟨*proof*⟩

**declare** *RestrictedP.simps* [*simp del*]

## 7.3 Applications to LstSeqP

**lemma** *HFun\_Sigma\_Eats*:

**assumes**  $H \vdash \text{HFun\_Sigma } r \ H \vdash \text{NotInDom } d \ r \ H \vdash \text{OrdP } d$   
**shows**  $H \vdash \text{HFun\_Sigma } (\text{Eats } r \ (\text{HPair } d \ d'))$  ⟨*proof*⟩

**lemma** *HFun\_Sigma\_single* [*iff*]:  $H \vdash \text{OrdP } d \implies H \vdash \text{HFun\_Sigma } (\text{Eats } \text{Zero} \ (\text{HPair } d \ d'))$   
 ⟨*proof*⟩

**lemma** *LstSeqP\_single* [*iff*]:  $H \vdash \text{LstSeqP } (\text{Eats } \text{Zero} \ (\text{HPair } \text{Zero} \ x)) \ \text{Zero } x$   
 ⟨*proof*⟩

**lemma** *NotInDom\_LstSeqP\_Eats*:

{ *NotInDom (SUCC k) s*, *LstSeqP s k y* }  $\vdash$  *LstSeqP (Eats s (HPair (SUCC k) z)) (SUCC k) z*  
 ⟨*proof*⟩

**lemma** *RestrictedP\_HDomain\_Incl*: { *HDomain\_Incl s k*, *RestrictedP s k s'* }  $\vdash$  *HDomain\_Incl s' k*  
 ⟨*proof*⟩

**lemma** *RestrictedP\_HFun\_Sigma*: { *HFun\_Sigma s*, *RestrictedP s k s'* }  $\vdash$  *HFun\_Sigma s'*  
 ⟨*proof*⟩

**lemma** *RestrictedP\_LstSeqP*:

{ *RestrictedP s (SUCC k) s'*, *LstSeqP s k y* }  $\vdash$  *LstSeqP s' k y*  
 ⟨*proof*⟩

**lemma** *RestrictedP\_LstSeqP\_Eats*:

{ *RestrictedP s (SUCC k) s'*, *LstSeqP s k y* }  
 $\vdash$  *LstSeqP (Eats s' (HPair (SUCC k) z)) (SUCC k) z*

⟨*proof*⟩

## 7.4 Ordinal Addition

### 7.4.1 Predicate form, defined on sequences

**nominal\_function** *SeqHaddP* ::  $tm \Rightarrow tm \Rightarrow tm \Rightarrow tm \Rightarrow fm$

**where**  $\llbracket \text{atom } l \ \# \ (sl, s, k, j); \ \text{atom } sl \ \# \ (s, j) \rrbracket \implies$

*SeqHaddP s j k y = LstSeqP s k y AND*

*HPair Zero j IN s AND*

*All2 l k (Ex sl (HPair (Var l) (Var sl) IN s AND*

*HPair (SUCC (Var l)) (SUCC (Var sl)) IN s)*

⟨*proof*⟩

**nominal\_termination** (*eqvt*)

⟨*proof*⟩

**lemma** *SeqHaddP\_fresh\_iff* [*simp*]:  $a \ \# \ \text{SeqHaddP } s \ j \ k \ y \longleftrightarrow a \ \# \ s \wedge a \ \# \ j \wedge a \ \# \ k \wedge a \ \# \ y$   
 ⟨*proof*⟩

**lemma** *SeqHaddP\_subst* [*simp*]:

$(\text{SeqHaddP } s \ j \ k \ y)(i::=t) = \text{SeqHaddP } (\text{subst } i \ t \ s) \ (\text{subst } i \ t \ j) \ (\text{subst } i \ t \ k) \ (\text{subst } i \ t \ y)$

⟨*proof*⟩

**declare** *SeqHaddP.simps* [*simp del*]

**nominal\_function** *HaddP* :: *tm*  $\Rightarrow$  *tm*  $\Rightarrow$  *tm*  $\Rightarrow$  *fm*  
**where**  $\llbracket \text{atom } s \# (x,y,z) \rrbracket \Longrightarrow$   
*HaddP* *x y z* = *Ex s* (*SeqHaddP* (*Var s*) *x y z*)  
 $\langle \text{proof} \rangle$

**nominal\_termination** (*eqvt*)  
 $\langle \text{proof} \rangle$

**lemma** *HaddP\_fresh\_iff* [*simp*]:  $a \# \text{HaddP } x \ y \ z \longleftrightarrow a \# x \wedge a \# y \wedge a \# z$   
 $\langle \text{proof} \rangle$

**lemma** *HaddP\_subst* [*simp*]:  $(\text{HaddP } x \ y \ z)(i::=t) = \text{HaddP } (\text{subst } i \ t \ x) (\text{subst } i \ t \ y) (\text{subst } i \ t \ z)$   
 $\langle \text{proof} \rangle$

**lemma** *HaddP\_cong*:  $\llbracket H \vdash t \ EQ \ t'; H \vdash u \ EQ \ u'; H \vdash v \ EQ \ v' \rrbracket \Longrightarrow H \vdash \text{HaddP } t \ u \ v \ IFF \ \text{HaddP } t' \ u' \ v'$   
 $\langle \text{proof} \rangle$

**declare** *HaddP.simps* [*simp del*]

**lemma** *HaddP\_Zero2*:  $H \vdash \text{HaddP } x \ \text{Zero } x$   
 $\langle \text{proof} \rangle$

**lemma** *HaddP\_imp\_OrdP*:  $\{\text{HaddP } x \ y \ z\} \vdash \text{OrdP } y$   
 $\langle \text{proof} \rangle$

**lemma** *HaddP\_SUCC2*:  $\{\text{HaddP } x \ y \ z\} \vdash \text{HaddP } x \ (\text{SUCC } y) (\text{SUCC } z)$   $\langle \text{proof} \rangle$

## 7.4.2 Proving that these relations are functions

**lemma** *SeqHaddP\_Zero\_E*:  $\{\text{SeqHaddP } s \ w \ \text{Zero } z\} \vdash w \ EQ \ z$   
 $\langle \text{proof} \rangle$

**lemma** *SeqHaddP\_SUCC\_lemma*:  
**assumes**  $y': \text{atom } y' \# (s,j,k,y)$   
**shows**  $\{\text{SeqHaddP } s \ j \ (\text{SUCC } k) \ y\} \vdash \text{Ex } y' (\text{SeqHaddP } s \ j \ k \ (\text{Var } y') \ \text{AND } y \ EQ \ \text{SUCC } (\text{Var } y'))$   
 $\langle \text{proof} \rangle$

**lemma** *SeqHaddP\_SUCC*:  
**assumes**  $H \vdash \text{SeqHaddP } s \ j \ (\text{SUCC } k) \ y \ \text{atom } y' \# (s,j,k,y)$   
**shows**  $H \vdash \text{Ex } y' (\text{SeqHaddP } s \ j \ k \ (\text{Var } y') \ \text{AND } y \ EQ \ \text{SUCC } (\text{Var } y'))$   
 $\langle \text{proof} \rangle$

**lemma** *SeqHaddP\_unique*:  $\{\text{OrdP } x, \text{SeqHaddP } s \ w \ x \ y, \text{SeqHaddP } s' \ w \ x \ y'\} \vdash y' \ EQ \ y$   $\langle \text{proof} \rangle$

**lemma** *HaddP\_unique*:  $\{\text{HaddP } w \ x \ y, \text{HaddP } w \ x \ y'\} \vdash y' \ EQ \ y$   
 $\langle \text{proof} \rangle$

**lemma** *HaddP\_Zero1*: **assumes**  $H \vdash \text{OrdP } x$  **shows**  $H \vdash \text{HaddP } \text{Zero } x \ x$   
 $\langle \text{proof} \rangle$

**lemma** *HaddP\_Zero\_D1*:  $\text{insert } (\text{HaddP } \text{Zero } x \ y) \ H \vdash x \ EQ \ y$   
 $\langle \text{proof} \rangle$

**lemma** *HaddP\_Zero\_D2*:  $\text{insert } (\text{HaddP } x \ \text{Zero } y) \ H \vdash x \ EQ \ y$   
 $\langle \text{proof} \rangle$

**lemma** *HaddP\_SUCC\_Ex2*:

**assumes**  $H \vdash \text{HaddP } x \text{ (SUCC } y) z \text{ atom } z' \# (x,y,z)$   
**shows**  $H \vdash \text{Ex } z' \text{ (HaddP } x \text{ y (Var } z') \text{ AND } z \text{ EQ SUCC (Var } z'))$

*<proof>*

**lemma** *HaddP\_SUCC1*:  $\{ \text{HaddP } x \text{ y } z \} \vdash \text{HaddP (SUCC } x) \text{ y (SUCC } z)$  *<proof>*

**lemma** *HaddP\_commute*:  $\{ \text{HaddP } x \text{ y } z, \text{OrdP } x \} \vdash \text{HaddP } y \text{ x } z$  *<proof>*

**lemma** *HaddP\_SUCC\_Ex1*:

**assumes**  $\text{atom } i \# (x,y,z)$   
**shows**  $\text{insert (SUCC } x) \text{ y } z \text{ (insert (OrdP } x) \text{ H)}$   
 $\vdash \text{Ex } i \text{ (HaddP } x \text{ y (Var } i) \text{ AND } z \text{ EQ SUCC (Var } i))$

*<proof>*

**lemma** *HaddP\_inv2*:  $\{ \text{HaddP } x \text{ y } z, \text{HaddP } x \text{ y}' z, \text{OrdP } x \} \vdash y' \text{ EQ } y$  *<proof>*

**lemma** *Mem\_imp\_subtract*: *<proof>*

**lemma** *HaddP\_OrdP*:

**assumes**  $H \vdash \text{HaddP } x \text{ y } z \text{ H} \vdash \text{OrdP } x$  **shows**  $H \vdash \text{OrdP } z$  *<proof>*

**lemma** *HaddP\_Mem\_cancel\_left*:

**assumes**  $H \vdash \text{HaddP } x \text{ y}' z' \text{ H} \vdash \text{HaddP } x \text{ y } z \text{ H} \vdash \text{OrdP } x$   
**shows**  $H \vdash z' \text{ IN } z \text{ IFF } y' \text{ IN } y$  *<proof>*

**lemma** *HaddP\_Mem\_cancel\_right\_Mem*:

**assumes**  $H \vdash \text{HaddP } x' \text{ y } z' \text{ H} \vdash \text{HaddP } x \text{ y } z \text{ H} \vdash x' \text{ IN } x \text{ H} \vdash \text{OrdP } x$   
**shows**  $H \vdash z' \text{ IN } z$

*<proof>*

**lemma** *HaddP\_Mem\_cases*:

**assumes**  $H \vdash \text{HaddP } k1 \text{ k2 } k \text{ H} \vdash \text{OrdP } k1$   
 $\text{insert } (x \text{ IN } k1) \text{ H} \vdash A$   
 $\text{insert } (\text{Var } i \text{ IN } k2) \text{ (insert (HaddP } k1 \text{ (Var } i) x) \text{ H)} \vdash A$   
**and**  $i: \text{atom } (i::\text{name}) \# (k1,k2,k,x,A)$  **and**  $\forall C \in H. \text{atom } i \# C$   
**shows**  $\text{insert } (x \text{ IN } k) \text{ H} \vdash A$  *<proof>*

**lemma** *HaddP\_Mem\_contra*:

**assumes**  $H \vdash \text{HaddP } x \text{ y } z \text{ H} \vdash z \text{ IN } x \text{ H} \vdash \text{OrdP } x$   
**shows**  $H \vdash A$

*<proof>*

**lemma** *exists\_HaddP*:

**assumes**  $H \vdash \text{OrdP } y \text{ atom } j \# (x,y)$   
**shows**  $H \vdash \text{Ex } j \text{ (HaddP } x \text{ y (Var } j))$

*<proof>*

**lemma** *HaddP\_Mem\_I*:

**assumes**  $H \vdash \text{HaddP } x \text{ y } z \text{ H} \vdash \text{OrdP } x$  **shows**  $H \vdash x \text{ IN SUCC } z$

*<proof>*

## 7.5 A Shifted Sequence

**nominal\_function** *ShiftP* ::  $tm \Rightarrow tm \Rightarrow tm \Rightarrow tm \Rightarrow fm$

**where**  $\llbracket \text{atom } x \# (x',y,z,f,del,k); \text{atom } x' \# (y,z,f,del,k); \text{atom } y \# (z,f,del,k); \text{atom } z \# (f,del,g,k) \rrbracket \Longrightarrow$

*ShiftP f k del g =*

*All z (Var z IN g IFF*

*(Ex x (Ex x' (Ex y ((Var z) EQ HPair (Var x') (Var y) AND*

*HaddP del (Var x) (Var x') AND*

*HPair (Var x) (Var y) IN f AND Var x IN k))))))*

*<proof>*

**nominal\_termination** (*eqvt*)

*<proof>*

**lemma** *ShiftP\_fresh\_iff* [simp]:  $a \# \text{ShiftP } f \ k \ \text{del } g \longleftrightarrow a \# f \wedge a \# k \wedge a \# \text{del} \wedge a \# g$   
 ⟨proof⟩

**lemma** *subst\_fm\_ShiftP* [simp]:  
 $(\text{ShiftP } f \ k \ \text{del } g)(i::=u) = \text{ShiftP } (\text{subst } i \ u \ f) (\text{subst } i \ u \ k) (\text{subst } i \ u \ \text{del}) (\text{subst } i \ u \ g)$   
 ⟨proof⟩

**lemma** *ShiftP\_Zero*:  $\{\} \vdash \text{ShiftP } \text{Zero } k \ d \ \text{Zero}$   
 ⟨proof⟩

**lemma** *ShiftP\_Mem1*:  
 $\{\text{ShiftP } f \ k \ \text{del } g, \text{HPair } a \ b \ \text{IN } f, \text{HaddP } \text{del } a \ a', a \ \text{IN } k\} \vdash \text{HPair } a' \ b \ \text{IN } g$   
 ⟨proof⟩

**lemma** *ShiftP\_Mem2*:  
**assumes**  $\text{atom } u \# (f, k, \text{del}, a, b)$   
**shows**  $\{\text{ShiftP } f \ k \ \text{del } g, \text{HPair } a \ b \ \text{IN } g\} \vdash \text{Ex } u ((\text{Var } u) \ \text{IN } k \ \text{AND } \text{HaddP } \text{del } (\text{Var } u) \ a \ \text{AND } \text{HPair } (\text{Var } u) \ b \ \text{IN } f)$   
 ⟨proof⟩

**lemma** *ShiftP\_Mem\_D*:  
**assumes**  $H \vdash \text{ShiftP } f \ k \ \text{del } g \ H \vdash a \ \text{IN } g$   
 $\text{atom } x \# (x', y, a, f, \text{del}, k) \ \text{atom } x' \# (y, a, f, \text{del}, k) \ \text{atom } y \# (a, f, \text{del}, k)$   
**shows**  $H \vdash (\text{Ex } x (\text{Ex } x' (\text{Ex } y (a \ \text{EQ} \ \text{HPair } (\text{Var } x') (\text{Var } y) \ \text{AND} \ \text{HaddP } \text{del } (\text{Var } x) (\text{Var } x') \ \text{AND} \ \text{HPair } (\text{Var } x) (\text{Var } y) \ \text{IN } f \ \text{AND } \text{Var } x \ \text{IN } k))))$   
 (is  $\_ \vdash ?\text{concl}$ )  
 ⟨proof⟩

**lemma** *ShiftP\_Eats\_Eats*:  
 $\{\text{ShiftP } f \ k \ \text{del } g, \text{HaddP } \text{del } a \ a', a \ \text{IN } k\}$   
 $\vdash \text{ShiftP } (\text{Eats } f (\text{HPair } a \ b)) \ k \ \text{del } (\text{Eats } g (\text{HPair } a' \ b))$  ⟨proof⟩

**lemma** *ShiftP\_Eats\_Neg*:  
**assumes**  $\text{atom } u \# (u', v, f, k, \text{del}, g, c) \ \text{atom } u' \# (v, f, k, \text{del}, g, c) \ \text{atom } v \# (f, k, \text{del}, g, c)$   
**shows**  
 $\{\text{ShiftP } f \ k \ \text{del } g,$   
 $\text{Neg } (\text{Ex } u (\text{Ex } u' (\text{Ex } v (c \ \text{EQ} \ \text{HPair } (\text{Var } u) (\text{Var } v) \ \text{AND } \text{Var } u \ \text{IN } k \ \text{AND } \text{HaddP } \text{del } (\text{Var } u) (\text{Var } u'))))\}$   
 $\vdash \text{ShiftP } (\text{Eats } f \ c) \ k \ \text{del } g$  ⟨proof⟩

**lemma** *exists\_ShiftP*:  
**assumes**  $t: \text{atom } t \# (s, k, \text{del})$   
**shows**  $H \vdash \text{Ex } t (\text{ShiftP } s \ k \ \text{del } (\text{Var } t))$  ⟨proof⟩

## 7.6 Union of Two Sets

**nominal\_function** *UnionP* ::  $tm \Rightarrow tm \Rightarrow tm \Rightarrow fm$   
**where**  $\text{atom } i \# (x, y, z) \Longrightarrow \text{UnionP } x \ y \ z = \text{All } i (\text{Var } i \ \text{IN } z \ \text{IFF } (\text{Var } i \ \text{IN } x \ \text{OR } \text{Var } i \ \text{IN } y))$   
 ⟨proof⟩

**nominal\_termination** (*eqvt*)  
 ⟨proof⟩

**lemma** *UnionP\_fresh\_iff* [simp]:  $a \# \text{UnionP } x \ y \ z \longleftrightarrow a \# x \wedge a \# y \wedge a \# z$   
 ⟨proof⟩

**lemma** *subst\_fm\_UnionP* [simp]:  
 $(\text{UnionP } x \ y \ z)(i::=u) = \text{UnionP } (\text{subst } i \ u \ x) (\text{subst } i \ u \ y) (\text{subst } i \ u \ z)$

*<proof>*

**lemma** *Union\_Zero1*:  $H \vdash \text{UnionP Zero } x \ x$

*<proof>*

**lemma** *Union\_Eats*:  $\{\text{UnionP } x \ y \ z\} \vdash \text{UnionP (Eats } x \ a) \ y \ (\text{Eats } z \ a)$

*<proof>*

**lemma** *exists\_Union\_lemma*:

**assumes**  $z: \text{atom } z \ \# \ (i,y)$  **and**  $i: \text{atom } i \ \# \ y$

**shows**  $\{\} \vdash \text{Ex } z \ (\text{UnionP (Var } i) \ y \ (\text{Var } z))$

*<proof>*

**lemma** *exists\_UnionP*:

**assumes**  $z: \text{atom } z \ \# \ (x,y)$  **shows**  $H \vdash \text{Ex } z \ (\text{UnionP } x \ y \ (\text{Var } z))$

*<proof>*

**lemma** *UnionP\_Mem1*:  $\{\text{UnionP } x \ y \ z, \ a \ \text{IN } x\} \vdash \ a \ \text{IN } z$

*<proof>*

**lemma** *UnionP\_Mem2*:  $\{\text{UnionP } x \ y \ z, \ a \ \text{IN } y\} \vdash \ a \ \text{IN } z$

*<proof>*

**lemma** *UnionP\_Mem*:  $\{\text{UnionP } x \ y \ z, \ a \ \text{IN } z\} \vdash \ a \ \text{IN } x \ \text{OR } a \ \text{IN } y$

*<proof>*

**lemma** *UnionP\_Mem\_E*:

**assumes**  $H \vdash \text{UnionP } x \ y \ z$

**and**  $\text{insert } (a \ \text{IN } x) \ H \vdash \ A$

**and**  $\text{insert } (a \ \text{IN } y) \ H \vdash \ A$

**shows**  $\text{insert } (a \ \text{IN } z) \ H \vdash \ A$

*<proof>*

## 7.7 Append on Sequences

**nominal\_function** *SeqAppendP* ::  $tm \Rightarrow tm \Rightarrow tm \Rightarrow tm \Rightarrow tm \Rightarrow fm$

**where**  $\llbracket \text{atom } g1 \ \# \ (g2, f1, k1, f2, k2, g); \ \text{atom } g2 \ \# \ (f1, k1, f2, k2, g) \rrbracket \Longrightarrow$

$\text{SeqAppendP } f1 \ k1 \ f2 \ k2 \ g =$

$(\text{Ex } g1 \ (\text{Ex } g2 \ (\text{RestrictedP } f1 \ k1 \ (\text{Var } g1) \ \text{AND}$

$\text{ShiftP } f2 \ k2 \ k1 \ (\text{Var } g2) \ \text{AND}$

$\text{UnionP } (\text{Var } g1) \ (\text{Var } g2) \ g))$

*<proof>*

**nominal\_termination** (*eqvt*)

*<proof>*

**lemma** *SeqAppendP\_fresh\_iff* [*simp*]:

$a \ \# \ \text{SeqAppendP } f1 \ k1 \ f2 \ k2 \ g \longleftrightarrow a \ \# \ f1 \wedge a \ \# \ k1 \wedge a \ \# \ f2 \wedge a \ \# \ k2 \wedge a \ \# \ g$

*<proof>*

**lemma** *subst\_fm\_SeqAppendP* [*simp*]:

$(\text{SeqAppendP } f1 \ k1 \ f2 \ k2 \ g)(i::=u) =$

$\text{SeqAppendP } (\text{subst } i \ u \ f1) \ (\text{subst } i \ u \ k1) \ (\text{subst } i \ u \ f2) \ (\text{subst } i \ u \ k2) \ (\text{subst } i \ u \ g)$

*<proof>*

**lemma** *exists\_SeqAppendP*:

**assumes**  $\text{atom } g \ \# \ (f1, k1, f2, k2)$

**shows**  $H \vdash \text{Ex } g \ (\text{SeqAppendP } f1 \ k1 \ f2 \ k2 \ (\text{Var } g))$

*<proof>*

**lemma** *SeqAppendP\_Mem1*:  $\{SeqAppendP\ f1\ k1\ f2\ k2\ g,\ HPair\ x\ y\ IN\ f1,\ x\ IN\ k1\} \vdash HPair\ x\ y\ IN\ g$   
*<proof>*

**lemma** *SeqAppendP\_Mem2*:  $\{SeqAppendP\ f1\ k1\ f2\ k2\ g,\ HaddP\ k1\ x\ x',\ x\ IN\ k2,\ HPair\ x\ y\ IN\ f2\} \vdash HPair\ x'\ y\ IN\ g$   
*<proof>*

**lemma** *SeqAppendP\_Mem\_E*:

**assumes**  $H \vdash SeqAppendP\ f1\ k1\ f2\ k2\ g$   
**and**  $insert\ (HPair\ x\ y\ IN\ f1)\ (insert\ (x\ IN\ k1)\ H) \vdash A$   
**and**  $insert\ (HPair\ (Var\ u)\ y\ IN\ f2)\ (insert\ (HaddP\ k1\ (Var\ u)\ x)\ (insert\ (Var\ u\ IN\ k2)\ H)) \vdash A$   
**and**  $u: atom\ u \# (f1,k1,f2,k2,x,y,g,A) \forall C \in H. atom\ u \# C$   
**shows**  $insert\ (HPair\ x\ y\ IN\ g)\ H \vdash A$  *<proof>*

## 7.8 LstSeqP and SeqAppendP

**lemma** *HDomain\_Incl\_SeqAppendP*: — The And eliminates the need to prove *cut5*  
 $\{SeqAppendP\ f1\ k1\ f2\ k2\ g,\ HDomain\_Incl\ f1\ k1\ AND\ HDomain\_Incl\ f2\ k2,\ HaddP\ k1\ k2\ k,\ OrdP\ k1\} \vdash HDomain\_Incl\ g\ k$  *<proof>*

**declare** *SeqAppendP.simps* [*simp del*]

**lemma** *HFun\_Sigma\_SeqAppendP*:

$\{SeqAppendP\ f1\ k1\ f2\ k2\ g,\ HFun\_Sigma\ f1,\ HFun\_Sigma\ f2,\ OrdP\ k1\} \vdash HFun\_Sigma\ g$  *<proof>*

**lemma** *LstSeqP\_SeqAppendP*:

**assumes**  $H \vdash SeqAppendP\ f1\ (SUCC\ k1)\ f2\ (SUCC\ k2)\ g$   
 $H \vdash LstSeqP\ f1\ k1\ y1\ H \vdash LstSeqP\ f2\ k2\ y2\ H \vdash HaddP\ k1\ k2\ k$   
**shows**  $H \vdash LstSeqP\ g\ (SUCC\ k)\ y2$

*<proof>*

**lemma** *SeqAppendP\_NotInDom*:  $\{SeqAppendP\ f1\ k1\ f2\ k2\ g,\ HaddP\ k1\ k2\ k,\ OrdP\ k1\} \vdash NotInDom\ k\ g$   
*<proof>*

**lemma** *LstSeqP\_SeqAppendP\_Eats*:

**assumes**  $H \vdash SeqAppendP\ f1\ (SUCC\ k1)\ f2\ (SUCC\ k2)\ g$   
 $H \vdash LstSeqP\ f1\ k1\ y1\ H \vdash LstSeqP\ f2\ k2\ y2\ H \vdash HaddP\ k1\ k2\ k$   
**shows**  $H \vdash LstSeqP\ (Eats\ g\ (HPair\ (SUCC\ (SUCC\ k))\ z))\ (SUCC\ (SUCC\ k))\ z$

*<proof>*

## 7.9 Substitution and Abstraction on Terms

### 7.9.1 Atomic cases

**lemma** *SeqStTermP\_Var\_same*:

**assumes**  $atom\ s \# (k,v,i)\ atom\ k \# (v,i)$   
**shows**  $\{VarP\ v\} \vdash Ex\ s\ (Ex\ k\ (SeqStTermP\ v\ i\ v\ i\ (Var\ s)\ (Var\ k)))$

*<proof>*

**lemma** *SeqStTermP\_Var\_diff*:

**assumes**  $atom\ s \# (k,v,w,i)\ atom\ k \# (v,w,i)$   
**shows**  $\{VarP\ v,\ VarP\ w,\ Neg\ (v\ EQ\ w)\} \vdash Ex\ s\ (Ex\ k\ (SeqStTermP\ v\ i\ w\ w\ (Var\ s)\ (Var\ k)))$

*<proof>*

**lemma** *SeqStTermP\_Zero*:

**assumes**  $atom\ s \# (k,v,i)\ atom\ k \# (v,i)$   
**shows**  $\{VarP\ v\} \vdash Ex\ s\ (Ex\ k\ (SeqStTermP\ v\ i\ Zero\ Zero\ (Var\ s)\ (Var\ k)))$  *<proof>*

**corollary** *SubstTermP\_Zero*:  $\{TermP\ t\} \vdash SubstTermP \llbracket Var\ v \rrbracket t\ Zero\ Zero$   
 $\langle proof \rangle$

**corollary** *SubstTermP\_Var\_same*:  $\{VarP\ v, TermP\ t\} \vdash SubstTermP\ v\ t\ v\ t$   
 $\langle proof \rangle$

**corollary** *SubstTermP\_Var\_diff*:  $\{VarP\ v, VarP\ w, Neg\ (v\ EQ\ w), TermP\ t\} \vdash SubstTermP\ v\ t\ w\ w$   
 $\langle proof \rangle$

**lemma** *SeqStTermP\_Ind*:

**assumes** *atom s*  $\# (k, v, t, i)$  *atom k*  $\# (v, t, i)$

**shows**  $\{VarP\ v, IndP\ t\} \vdash Ex\ s\ (Ex\ k\ (SeqStTermP\ v\ i\ t\ t\ (Var\ s)\ (Var\ k)))$

$\langle proof \rangle$

**corollary** *SubstTermP\_Ind*:  $\{VarP\ v, IndP\ w, TermP\ t\} \vdash SubstTermP\ v\ t\ w\ w$   
 $\langle proof \rangle$

## 7.9.2 Non-atomic cases

**lemma** *SeqStTermP\_Eats*:

**assumes** *sk*: *atom s*  $\# (k, s1, s2, k1, k2, t1, t2, u1, u2, v, i)$

*atom k*  $\# (t1, t2, u1, u2, v, i)$

**shows**  $\{SeqStTermP\ v\ i\ t1\ u1\ s1\ k1, SeqStTermP\ v\ i\ t2\ u2\ s2\ k2\}$

$\vdash Ex\ s\ (Ex\ k\ (SeqStTermP\ v\ i\ (Q\_Eats\ t1\ t2)\ (Q\_Eats\ u1\ u2)\ (Var\ s)\ (Var\ k)))$   $\langle proof \rangle$

**theorem** *SubstTermP\_Eats*:

$\{SubstTermP\ v\ i\ t1\ u1, SubstTermP\ v\ i\ t2\ u2\} \vdash SubstTermP\ v\ i\ (Q\_Eats\ t1\ t2)\ (Q\_Eats\ u1\ u2)$

$\langle proof \rangle$

## 7.9.3 Substitution over a constant

**lemma** *SeqConstP\_lemma*:

**assumes** *atom m*  $\# (s, k, c, n, sm, sn)$  *atom n*  $\# (s, k, c, sm, sn)$

*atom sm*  $\# (s, k, c, sn)$  *atom sn*  $\# (s, k, c)$

**shows**  $\{SeqConstP\ s\ k\ c\}$

$\vdash c\ EQ\ Zero\ OR$

$Ex\ m\ (Ex\ n\ (Ex\ sm\ (Ex\ sn\ (Var\ m\ IN\ k\ AND\ Var\ n\ IN\ k\ AND$

$SeqConstP\ s\ (Var\ m)\ (Var\ sm)\ AND$

$SeqConstP\ s\ (Var\ n)\ (Var\ sn)\ AND$

$c\ EQ\ Q\_Eats\ (Var\ sm)\ (Var\ sn))))$   $\langle proof \rangle$

**lemma** *SeqConstP\_imp\_SubstTermP*:  $\{SeqConstP\ s\ k\ c, TermP\ t\} \vdash SubstTermP \llbracket Var\ w \rrbracket t\ c\ c$   $\langle proof \rangle$

**theorem** *SubstTermP\_Const*:  $\{ConstP\ c, TermP\ t\} \vdash SubstTermP \llbracket Var\ w \rrbracket t\ c\ c$

$\langle proof \rangle$

## 7.10 Substitution on Formulas

### 7.10.1 Membership

**lemma** *SubstAtomicP\_Mem*:

$\{SubstTermP\ v\ i\ x\ x', SubstTermP\ v\ i\ y\ y'\} \vdash SubstAtomicP\ v\ i\ (Q\_Mem\ x\ y)\ (Q\_Mem\ x'\ y')$

$\langle proof \rangle$

**lemma** *SeqSubstFormP\_Mem*:

**assumes** *atom s*  $\# (k, x, y, x', y', v, i)$  *atom k*  $\# (x, y, x', y', v, i)$

**shows**  $\{SubstTermP\ v\ i\ x\ x', SubstTermP\ v\ i\ y\ y'\}$

$\vdash Ex\ s\ (Ex\ k\ (SeqSubstFormP\ v\ i\ (Q\_Mem\ x\ y)\ (Q\_Mem\ x'\ y')\ (Var\ s)\ (Var\ k)))$

$\langle proof \rangle$

**lemma** *SubstFormP\_Mem*:

$\{SubstTermP\ v\ i\ x\ x',\ SubstTermP\ v\ i\ y\ y'\} \vdash SubstFormP\ v\ i\ (Q\_Mem\ x\ y)\ (Q\_Mem\ x'\ y')$   
 $\langle proof \rangle$

### 7.10.2 Equality

**lemma** *SubstAtomicP\_Eq*:

$\{SubstTermP\ v\ i\ x\ x',\ SubstTermP\ v\ i\ y\ y'\} \vdash SubstAtomicP\ v\ i\ (Q\_Eq\ x\ y)\ (Q\_Eq\ x'\ y')$   
 $\langle proof \rangle$

**lemma** *SeqSubstFormP\_Eq*:

**assumes**  $atom\ s\ \# (k, x, y, x', y', v, i)\ atom\ k\ \# (x, y, x', y', v, i)$

**shows**  $\{SubstTermP\ v\ i\ x\ x',\ SubstTermP\ v\ i\ y\ y'\}$

$\vdash Ex\ s\ (Ex\ k\ (SeqSubstFormP\ v\ i\ (Q\_Eq\ x\ y)\ (Q\_Eq\ x'\ y')\ (Var\ s)\ (Var\ k)))$

$\langle proof \rangle$

**lemma** *SubstFormP\_Eq*:

$\{SubstTermP\ v\ i\ x\ x',\ SubstTermP\ v\ i\ y\ y'\} \vdash SubstFormP\ v\ i\ (Q\_Eq\ x\ y)\ (Q\_Eq\ x'\ y')$   
 $\langle proof \rangle$

### 7.10.3 Negation

**lemma** *SeqSubstFormP\_Neg*:

**assumes**  $atom\ s\ \# (k, s1, k1, x, x', v, i)\ atom\ k\ \# (s1, k1, x, x', v, i)$

**shows**  $\{SeqSubstFormP\ v\ i\ x\ x'\ s1\ k1,\ TermP\ i,\ VarP\ v\}$

$\vdash Ex\ s\ (Ex\ k\ (SeqSubstFormP\ v\ i\ (Q\_Neg\ x)\ (Q\_Neg\ x')\ (Var\ s)\ (Var\ k))) \langle proof \rangle$

**theorem** *SubstFormP\_Neg*:  $\{SubstFormP\ v\ i\ x\ x'\} \vdash SubstFormP\ v\ i\ (Q\_Neg\ x)\ (Q\_Neg\ x')$   
 $\langle proof \rangle$

### 7.10.4 Disjunction

**lemma** *SeqSubstFormP\_Disj*:

**assumes**  $atom\ s\ \# (k, s1, s2, k1, k2, x, y, x', y', v, i)\ atom\ k\ \# (s1, s2, k1, k2, x, y, x', y', v, i)$

**shows**  $\{SeqSubstFormP\ v\ i\ x\ x'\ s1\ k1,\ SeqSubstFormP\ v\ i\ y\ y'\ s2\ k2,\ TermP\ i,\ VarP\ v\}$

$\vdash Ex\ s\ (Ex\ k\ (SeqSubstFormP\ v\ i\ (Q\_Disj\ x\ y)\ (Q\_Disj\ x'\ y')\ (Var\ s)\ (Var\ k))) \langle proof \rangle$

**theorem** *SubstFormP\_Disj*:

$\{SubstFormP\ v\ i\ x\ x',\ SubstFormP\ v\ i\ y\ y'\} \vdash SubstFormP\ v\ i\ (Q\_Disj\ x\ y)\ (Q\_Disj\ x'\ y')$   
 $\langle proof \rangle$

### 7.10.5 Existential

**lemma** *SeqSubstFormP\_Ex*:

**assumes**  $atom\ s\ \# (k, s1, k1, x, x', v, i)\ atom\ k\ \# (s1, k1, x, x', v, i)$

**shows**  $\{SeqSubstFormP\ v\ i\ x\ x'\ s1\ k1,\ TermP\ i,\ VarP\ v\}$

$\vdash Ex\ s\ (Ex\ k\ (SeqSubstFormP\ v\ i\ (Q\_Ex\ x)\ (Q\_Ex\ x')\ (Var\ s)\ (Var\ k))) \langle proof \rangle$

**theorem** *SubstFormP\_Ex*:  $\{SubstFormP\ v\ i\ x\ x'\} \vdash SubstFormP\ v\ i\ (Q\_Ex\ x)\ (Q\_Ex\ x')$   
 $\langle proof \rangle$

## 7.11 Constant Terms

**lemma** *ConstP\_Zero*:  $\{\} \vdash ConstP\ Zero$

$\langle proof \rangle$

**lemma** *SeqConstP\_Eats*:

**assumes**  $atom\ s\ \# (k, s1, s2, k1, k2, t1, t2)\ atom\ k\ \# (s1, s2, k1, k2, t1, t2)$

**shows**  $\{SeqConstP\ s1\ k1\ t1,\ SeqConstP\ s2\ k2\ t2\}$

$\vdash Ex\ s\ (Ex\ k\ (SeqConstP\ (Var\ s)\ (Var\ k)\ (Q\_Eats\ t1\ t2))) \langle proof \rangle$

**theorem** *ConstP\_Eats*:  $\{ConstP\ t1,\ ConstP\ t2\} \vdash ConstP\ (Q\_Eats\ t1\ t2)$

*<proof>*

**lemma** *TermP\_Zero*: {} ⊢ *TermP Zero*

*<proof>*

**lemma** *TermP\_Var*: {} ⊢ *TermP «Var x»*

*<proof>*

**lemma** *SeqTermP\_Eats*:

**assumes** *atom s* ‡ (*k,s1,s2,k1,k2,t1,t2*) *atom k* ‡ (*s1,s2,k1,k2,t1,t2*)

**shows** {*SeqTermP s1 k1 t1*, *SeqTermP s2 k2 t2*}

⊢ *Ex s (Ex k (SeqTermP (Var s) (Var k) (Q\_Eats t1 t2)))* *<proof>*

**theorem** *TermP\_Eats*: {*TermP t1*, *TermP t2*} ⊢ *TermP (Q\_Eats t1 t2)*

*<proof>*

## 7.12 Proofs

**lemma** *PrfP\_inference*:

**assumes** *atom s* ‡ (*k,s1,s2,k1,k2,α1,α2,β*) *atom k* ‡ (*s1,s2,k1,k2,α1,α2,β*)

**shows** {*PrfP s1 k1 α1*, *PrfP s2 k2 α2*, *ModPonP α1 α2 β OR ExistsP α1 β OR SubstP α1 β*}

⊢ *Ex k (Ex s (PrfP (Var s) (Var k) β))* *<proof>*

**corollary** *PfP\_inference*: {*PfP α1*, *PfP α2*, *ModPonP α1 α2 β OR ExistsP α1 β OR SubstP α1 β*}

⊢ *PfP β*

*<proof>*

**theorem** *PfP\_implies\_SubstForm\_PfP*:

**assumes** *H* ⊢ *PfP y* *H* ⊢ *SubstFormP x t y z*

**shows** *H* ⊢ *PfP z*

*<proof>*

**theorem** *PfP\_implies\_ModPon\_PfP*: [*H* ⊢ *PfP (Q\_Imp x y)*; *H* ⊢ *PfP x*] ⇒ *H* ⊢ *PfP y*

*<proof>*

**corollary** *PfP\_implies\_ModPon\_PfP\_quot*: [*H* ⊢ *PfP «α IMP β»*; *H* ⊢ *PfP «α»*] ⇒ *H* ⊢ *PfP «β»*

*<proof>*

**lemma** *TermP\_quot*:

**fixes** *α* :: *tm*

**shows** {} ⊢ *TermP «α»*

*<proof>*

**lemma** *TermP\_quot\_dbtm*:

**fixes** *α* :: *tm*

**assumes** *wf\_dbtm u*

**shows** {} ⊢ *TermP (quot\_dbtm u)*

*<proof>*

## 7.13 Formulas

## 7.14 Abstraction on Formulas

### 7.14.1 Membership

**lemma** *AbstAtomicP\_Mem*:

{*AbstTermP v i x x'*, *AbstTermP v i y y'*} ⊢ *AbstAtomicP v i (Q\_Mem x y) (Q\_Mem x' y')*

*<proof>*

**lemma** *SeqAbstFormP\_Mem*:

**assumes**  $atom\ s \# (k, x, y, x', y', v, i)$   $atom\ k \# (x, y, x', y', v, i)$

**shows**  $\{AbstTermP\ v\ i\ x\ x', AbstTermP\ v\ i\ y\ y'\}$

$\vdash Ex\ s\ (Ex\ k\ (SeqAbstFormP\ v\ i\ (Q\_Mem\ x\ y)\ (Q\_Mem\ x'\ y')\ (Var\ s)\ (Var\ k)))$

*<proof>*

**lemma** *AbstFormP\_Mem*:

$\{AbstTermP\ v\ i\ x\ x', AbstTermP\ v\ i\ y\ y'\} \vdash AbstFormP\ v\ i\ (Q\_Mem\ x\ y)\ (Q\_Mem\ x'\ y')$

*<proof>*

## 7.14.2 Equality

**lemma** *AbstAtomicP\_Eq*:

$\{AbstTermP\ v\ i\ x\ x', AbstTermP\ v\ i\ y\ y'\} \vdash AbstAtomicP\ v\ i\ (Q\_Eq\ x\ y)\ (Q\_Eq\ x'\ y')$

*<proof>*

**lemma** *SeqAbstFormP\_Eq*:

**assumes**  $sk: atom\ s \# (k, x, y, x', y', v, i)$   $atom\ k \# (x, y, x', y', v, i)$

**shows**  $\{AbstTermP\ v\ i\ x\ x', AbstTermP\ v\ i\ y\ y'\}$

$\vdash Ex\ s\ (Ex\ k\ (SeqAbstFormP\ v\ i\ (Q\_Eq\ x\ y)\ (Q\_Eq\ x'\ y')\ (Var\ s)\ (Var\ k)))$

*<proof>*

**lemma** *AbstFormP\_Eq*:

$\{AbstTermP\ v\ i\ x\ x', AbstTermP\ v\ i\ y\ y'\} \vdash AbstFormP\ v\ i\ (Q\_Eq\ x\ y)\ (Q\_Eq\ x'\ y')$

*<proof>*

## 7.14.3 Negation

**lemma** *SeqAbstFormP\_Neg*:

**assumes**  $atom\ s \# (k, s1, k1, x, x', v, i)$   $atom\ k \# (s1, k1, x, x', v, i)$

**shows**  $\{SeqAbstFormP\ v\ i\ x\ x'\ s1\ k1, OrdP\ i, VarP\ v\}$

$\vdash Ex\ s\ (Ex\ k\ (SeqAbstFormP\ v\ i\ (Q\_Neg\ x)\ (Q\_Neg\ x')\ (Var\ s)\ (Var\ k)))$  *<proof>*

**theorem** *AbstFormP\_Neg*:  $\{AbstFormP\ v\ i\ x\ x'\} \vdash AbstFormP\ v\ i\ (Q\_Neg\ x)\ (Q\_Neg\ x')$

*<proof>*

## 7.14.4 Disjunction

**lemma** *SeqAbstFormP\_Disj*:

**assumes**  $atom\ s \# (k, s1, s2, k1, k2, x, y, x', y', v, i)$   $atom\ k \# (s1, s2, k1, k2, x, y, x', y', v, i)$

**shows**  $\{SeqAbstFormP\ v\ i\ x\ x'\ s1\ k1,$

$SeqAbstFormP\ v\ i\ y\ y'\ s2\ k2, OrdP\ i, VarP\ v\}$

$\vdash Ex\ s\ (Ex\ k\ (SeqAbstFormP\ v\ i\ (Q\_Disj\ x\ y)\ (Q\_Disj\ x'\ y')\ (Var\ s)\ (Var\ k)))$  *<proof>*

**theorem** *AbstFormP\_Disj*:

$\{AbstFormP\ v\ i\ x\ x', AbstFormP\ v\ i\ y\ y'\} \vdash AbstFormP\ v\ i\ (Q\_Disj\ x\ y)\ (Q\_Disj\ x'\ y')$

*<proof>*

## 7.14.5 Existential

**lemma** *SeqAbstFormP\_Ex*:

**assumes**  $atom\ s \# (k, s1, k1, x, x', v, i)$   $atom\ k \# (s1, k1, x, x', v, i)$

**shows**  $\{SeqAbstFormP\ v\ (SUCC\ i)\ x\ x'\ s1\ k1, OrdP\ i, VarP\ v\}$

$\vdash Ex\ s\ (Ex\ k\ (SeqAbstFormP\ v\ i\ (Q\_Ex\ x)\ (Q\_Ex\ x')\ (Var\ s)\ (Var\ k)))$  *<proof>*

**theorem** *AbstFormP\_Ex*:  $\{AbstFormP\ v\ (SUCC\ i)\ x\ x'\} \vdash AbstFormP\ v\ i\ (Q\_Ex\ x)\ (Q\_Ex\ x')$

*<proof>*

**corollary** *AbstTermP\_Zero*:  $\{OrdP\ t\} \vdash AbstTermP\ \llbracket Var\ v \rrbracket\ t\ Zero\ Zero$

*<proof>*

**corollary** *AbstTermP\_Var\_same*:  $\{VarP\ v, OrdP\ t\} \vdash AbstTermP\ v\ t\ v\ (Q\_Ind\ t)$   
 ⟨proof⟩

**corollary** *AbstTermP\_Var\_diff*:  $\{VarP\ v, VarP\ w, Neg\ (v\ EQ\ w), OrdP\ t\} \vdash AbstTermP\ v\ t\ w\ w$   
 ⟨proof⟩

**theorem** *AbstTermP\_Eats*:

$\{AbstTermP\ v\ i\ t1\ u1, AbstTermP\ v\ i\ t2\ u2\} \vdash AbstTermP\ v\ i\ (Q\_Eats\ t1\ t2)\ (Q\_Eats\ u1\ u2)$   
 ⟨proof⟩

**corollary** *AbstTermP\_Ind*:  $\{VarP\ v, IndP\ w, OrdP\ t\} \vdash AbstTermP\ v\ t\ w\ w$   
 ⟨proof⟩

**lemma** *ORD\_OF\_EQ\_diff*:  $x \neq y \implies \{ORD\_OF\ x\ EQ\ ORD\_OF\ y\} \vdash Fls$   
 ⟨proof⟩

**lemma** *quot\_Var\_EQ\_diff*:  $i \neq x \implies \{\langle Var\ i \rangle\ EQ\ \langle Var\ x \rangle\} \vdash Fls$   
 ⟨proof⟩

**lemma** *AbstTermP\_dbtm*:  $\{\} \vdash AbstTermP\ \langle Var\ i \rangle\ (ORD\_OF\ n)\ (quot\_dbtm\ u)\ (quot\_dbtm\ (abst\_dbtm\ i\ n\ u))$   
 ⟨proof⟩

**lemma** *AbstFormP\_dbfm*:  $\{\} \vdash AbstFormP\ \langle Var\ i \rangle\ (ORD\_OF\ n)\ (quot\_dbfm\ db)\ (quot\_dbfm\ (abst\_dbfm\ i\ n\ db))$   
 ⟨proof⟩

**lemmas** *AbstFormP = AbstFormP\_dbfm*[**where** *db=trans\_fm* [] **A and** *n = 0 for A,*  
*simplified, folded quot\_fm\_def, unfolded abst\_trans\_fm*]

**lemma** *SubstTermP\_trivial\_dbtm*:

*atom\ i\ \# u*  $\implies \{\} \vdash SubstTermP\ \langle Var\ i \rangle\ Zero\ (quot\_dbtm\ u)\ (quot\_dbtm\ u)$   
 ⟨proof⟩

**lemma** *SubstTermP\_dbtm*: *wf\_dbtm\ t*  $\implies$   
 $\{\} \vdash SubstTermP\ \langle Var\ i \rangle\ (quot\_dbtm\ t)\ (quot\_dbtm\ u)\ (quot\_dbtm\ (subst\_dbtm\ t\ i\ u))$   
 ⟨proof⟩

**lemma** *SubstFormP\_trivial\_dbfm*:

**fixes** *X :: fm*  
**assumes** *atom\ i\ \# db*  
**shows**  $\{\} \vdash SubstFormP\ \langle Var\ i \rangle\ Zero\ (quot\_dbfm\ db)\ (quot\_dbfm\ db)$   
 ⟨proof⟩

**lemma** *SubstFormP\_dbfm*:

**assumes** *wf\_dbtm\ t*  
**shows**  $\{\} \vdash SubstFormP\ \langle Var\ i \rangle\ (quot\_dbtm\ t)\ (quot\_dbfm\ db)\ (quot\_dbfm\ (subst\_dbfm\ t\ i\ db))$   
 ⟨proof⟩

**lemmas** *SubstFormP\_trivial = SubstFormP\_trivial\_dbfm*[**where** *db=trans\_fm* [] **A for A,  
*simplified, folded quot\_tm\_def quot\_fm\_def quot\_subst\_eq*]**

**lemmas** *SubstFormP = SubstFormP\_dbfm*[*OF wf\_dbtm\_trans\_tm, where db=trans\_fm* [] **A for A,**  
*simplified, folded quot\_tm\_def quot\_fm\_def quot\_subst\_eq*]

**lemmas** *SubstFormP\_Zero = SubstFormP\_dbfm*[*OF wf\_dbtm.Zero, where db=trans\_fm* [] **A for A,**  
*simplified, folded trans\_tm.simps[of []], folded quot\_tm\_def quot\_fm\_def quot\_subst\_eq*]

**lemma** *AtomicP\_Mem*:

$\{TermP\ x, TermP\ y\} \vdash AtomicP\ (Q\_Mem\ x\ y)$   
*<proof>*

**lemma** *AtomicP\_Eq*:

$\{TermP\ x, TermP\ y\} \vdash AtomicP\ (Q\_Eq\ x\ y)$   
*<proof>*

**lemma** *SeqFormP\_Mem*:

**assumes**  $atom\ s\ \# (k, x, y)\ atom\ k\ \# (x, y)$   
**shows**  $\{TermP\ x, TermP\ y\} \vdash Ex\ k\ (Ex\ s\ (SeqFormP\ (Var\ s)\ (Var\ k)\ (Q\_Mem\ x\ y)))$   
*<proof>*

**lemma** *SeqFormP\_Eq*:

**assumes**  $atom\ s\ \# (k, x, y)\ atom\ k\ \# (x, y)$   
**shows**  $\{TermP\ x, TermP\ y\} \vdash Ex\ k\ (Ex\ s\ (SeqFormP\ (Var\ s)\ (Var\ k)\ (Q\_Eq\ x\ y)))$   
*<proof>*

**lemma** *FormP\_Mem*:

$\{TermP\ x, TermP\ y\} \vdash FormP\ (Q\_Mem\ x\ y)$   
*<proof>*

**lemma** *FormP\_Eq*:

$\{TermP\ x, TermP\ y\} \vdash FormP\ (Q\_Eq\ x\ y)$   
*<proof>*

## 7.14.6 MakeForm

**lemma** *MakeFormP\_Neg*:  $\{\} \vdash MakeFormP\ (Q\_Neg\ x)\ x\ y$   
*<proof>*

**lemma** *MakeFormP\_Disj*:  $\{\} \vdash MakeFormP\ (Q\_Disj\ x\ y)\ x\ y$   
*<proof>*

**lemma** *MakeFormP\_Ex*:  $\{AbstFormP\ v\ Zero\ t\ x\} \vdash MakeFormP\ (Q\_Ex\ x)\ t\ y$   
*<proof>*

## 7.14.7 Negation

**lemma** *SeqFormP\_Neg*:

**assumes**  $atom\ s\ \# (k, s1, k1, x)\ atom\ k\ \# (s1, k1, x)$   
**shows**  $\{SeqFormP\ s1\ k1\ x\} \vdash Ex\ k\ (Ex\ s\ (SeqFormP\ (Var\ s)\ (Var\ k)\ (Q\_Neg\ x)))$  *<proof>*  
**theorem** *FormP\_Neg*:  $\{FormP\ x\} \vdash FormP\ (Q\_Neg\ x)$   
*<proof>*

## 7.14.8 Disjunction

**lemma** *SeqFormP\_Disj*:

**assumes**  $atom\ s\ \# (k, s1, s2, k1, k2, x, y)\ atom\ k\ \# (s1, s2, k1, k2, x, y)$   
**shows**  $\{SeqFormP\ s1\ k1\ x, SeqFormP\ s2\ k2\ y\}$   
 $\vdash Ex\ k\ (Ex\ s\ (SeqFormP\ (Var\ s)\ (Var\ k)\ (Q\_Disj\ x\ y)))$  *<proof>*

**theorem** *FormP\_Disj*:

$\{FormP\ x, FormP\ y\} \vdash FormP\ (Q\_Disj\ x\ y)$   
*<proof>*

## 7.14.9 Existential

**lemma** *SeqFormP\_Ex*:

**assumes**  $atom\ s\ \# (k, s1, k1, x, y, v)\ atom\ k\ \# (s1, k1, x, y, v)$

**shows**  $\{SeqFormP\ s1\ k1\ x, AbstFormP\ v\ Zero\ x\ y, VarP\ v\} \vdash Ex\ k\ (Ex\ s\ (SeqFormP\ (Var\ s)\ (Var\ k)\ (Q\_Ex\ y)))$   
 $\langle proof \rangle$

**theorem** *FormP\_Ex*:  $\{FormP\ t, AbstFormP\ \langle Var\ i \rangle\ Zero\ t\ x\} \vdash FormP\ (Q\_Ex\ x)$   
 $\langle proof \rangle$

**lemma** *FormP\_quot\_dbfm*:  
**fixes**  $A :: dbfm$   
**shows**  $wf\_dbfm\ A \implies \{\} \vdash FormP\ (quot\_dbfm\ A)$   
 $\langle proof \rangle$

**lemma** *FormP\_quot*:  
**fixes**  $A :: fm$   
**shows**  $\{\} \vdash FormP\ \langle A \rangle$   
 $\langle proof \rangle$

**lemma** *PfP\_I*:  
**assumes**  $\{\} \vdash PrfP\ S\ K\ A$   
**shows**  $\{\} \vdash PfP\ A$   
 $\langle proof \rangle$

**lemmas** *PfP\_Single\_I* = *PfP\_I*[of *Eats Zero (HPair Zero «A») Zero for A*]

**lemma** *PfP\_extra*:  $\{\} \vdash PfP\ \langle extra\_axiom \rangle$   
 $\langle proof \rangle$

**lemma** *SentP\_I*:  
**assumes**  $A \in boolean\_axioms$   
**shows**  $\{\} \vdash SentP\ \langle A \rangle$   
 $\langle proof \rangle$

**lemma** *SentP\_subst [simp]*:  $(SentP\ A)(j::=w) = SentP\ (subst\ j\ w\ A)$   
 $\langle proof \rangle$

**theorem** *proved\_imp\_proved\_PfP*:  
**assumes**  $\{\} \vdash \alpha$   
**shows**  $\{\} \vdash PfP\ \langle \alpha \rangle$   
 $\langle proof \rangle$

**end**

# Chapter 8

## Pseudo-Coding: Section 7 Material

```
theory Pseudo_Coding
imports II_Prelims
begin
```

### 8.1 General Lemmas

```
lemma Collect_disj_Un: {f i |i. P i ∨ Q i} = {f i |i. P i} ∪ {f i |i. Q i}
⟨proof⟩
```

```
abbreviation Q_Subset :: tm ⇒ tm ⇒ tm
where Q_Subset t u ≡ (Q_All (Q_Imp (Q_Mem (Q_Ind Zero) t) (Q_Mem (Q_Ind Zero) u)))
```

```
lemma NEQ_quot_tm: i ≠ j ⇒ {} ⊢ «Var i» NEQ «Var j»
⟨proof⟩
```

```
lemma EQ_quot_tm_Fls: i ≠ j ⇒ insert («Var i» EQ «Var j») H ⊢ Fls
⟨proof⟩
```

```
lemma perm_commute: a # p ⇒ a' # p ⇒ (a = a') + p = p + (a = a')
⟨proof⟩
```

```
lemma perm_self_inverseI: [¬p = q; a # p; a' # p] ⇒ - ((a = a') + p) = (a = a') + q
⟨proof⟩
```

```
lemma fresh_image:
fixes f :: 'a ⇒ 'b::fs shows finite A ⇒ i # f ' A ↔ (∀x∈A. i # f x)
⟨proof⟩
```

```
lemma atom_in_atom_image [simp]: atom j ∈ atom ' V ↔ j ∈ V
⟨proof⟩
```

```
lemma fresh_star_empty [simp]: {} #* bs
⟨proof⟩
```

```
declare fresh_star_insert [simp]
```

```
lemma fresh_star_finite_insert:
fixes S :: ('a::fs) set shows finite S ⇒ a #* insert x S ↔ a #* x ∧ a #* S
⟨proof⟩
```

**lemma** *fresh\_finite\_Diff\_single* [simp]:  
**fixes**  $V :: \text{name set}$  **shows**  $\text{finite } V \implies a \# (V - \{j\}) \longleftrightarrow (a \# j \longrightarrow a \# V)$   
 ⟨proof⟩

**lemma** *fresh\_image\_atom* [simp]:  $\text{finite } A \implies i \# \text{atom } A \longleftrightarrow i \# A$   
 ⟨proof⟩

**lemma** *atom\_fresh\_star\_atom\_set\_conv*:  $\llbracket \text{atom } i \# \text{bs}; \text{finite bs} \rrbracket \implies \text{bs} \#* i$   
 ⟨proof⟩

**lemma** *notin\_V*:  
**assumes**  $p: \text{atom } i \# p$  **and**  $V: \text{finite } V \text{ atom } (p \cdot V) \#* V$   
**shows**  $i \notin V \text{ and } i \notin p \cdot V$   
 ⟨proof⟩

## 8.2 Simultaneous Substitution

**definition** *ssubst* ::  $\text{tm} \Rightarrow \text{name set} \Rightarrow (\text{name} \Rightarrow \text{tm}) \Rightarrow \text{tm}$   
**where**  $\text{ssubst } t \ V \ F = \text{Finite\_Set.fold } (\lambda i. \text{subst } i \ (F \ i)) \ t \ V$

**definition** *make\_F* ::  $\text{name set} \Rightarrow \text{perm} \Rightarrow \text{name} \Rightarrow \text{tm}$   
**where**  $\text{make\_F } \text{Vs } p \equiv \lambda i. \text{if } i \in \text{Vs} \text{ then } \text{Var } (p \cdot i) \text{ else } \text{Var } i$

**lemma** *ssubst\_empty* [simp]:  $\text{ssubst } t \ \{\} \ F = t$   
 ⟨proof⟩

Renaming a finite set of variables. Based on the theorem *at\_set\_avoiding*

**locale** *quote\_perm* =  
**fixes**  $p :: \text{perm}$  **and**  $\text{Vs} :: \text{name set}$  **and**  $F :: \text{name} \Rightarrow \text{tm}$   
**assumes**  $p: \text{atom } (p \cdot \text{Vs}) \#* \text{Vs}$   
**and**  $\text{pinv}: -p = p$   
**and**  $\text{Vs}: \text{finite } \text{Vs}$   
**defines**  $F \equiv \text{make\_F } \text{Vs } p$   
**begin**

**lemma** *F\_unfold*:  $F \ i = (\text{if } i \in \text{Vs} \text{ then } \text{Var } (p \cdot i) \text{ else } \text{Var } i)$   
 ⟨proof⟩

**lemma** *finite\_V* [simp]:  $V \subseteq \text{Vs} \implies \text{finite } V$   
 ⟨proof⟩

**lemma** *perm\_exits\_Vs*:  $i \in \text{Vs} \implies (p \cdot i) \notin \text{Vs}$   
 ⟨proof⟩

**lemma** *atom\_fresh\_perm*:  $\llbracket x \in \text{Vs}; y \in \text{Vs} \rrbracket \implies \text{atom } x \# p \cdot y$   
 ⟨proof⟩

**lemma** *fresh\_pj*:  $\llbracket a \# p; j \in \text{Vs} \rrbracket \implies a \# p \cdot j$   
 ⟨proof⟩

**lemma** *fresh\_Vs*:  $a \# p \implies a \# \text{Vs}$   
 ⟨proof⟩

**lemma** *fresh\_pVs*:  $a \# p \implies a \# p \cdot \text{Vs}$   
 ⟨proof⟩

**lemma** *assumes*  $V \subseteq \text{Vs} \ a \# p$

**shows** *fresh\_pV* [simp]:  $a \# p \cdot V$  **and** *fresh\_V* [simp]:  $a \# V$   
 ⟨proof⟩

**lemma** *qp\_insert*:  
**fixes**  $i::name$  **and**  $i'::name$   
**assumes**  $atom\ i \# p\ atom\ i' \# (i,p)$   
**shows** *quote\_perm*  $((atom\ i = atom\ i') + p)$  (*insert i Vs*)  
 ⟨proof⟩

**lemma** *subst\_F\_left\_commute*:  $subst\ x\ (F\ x)\ (subst\ y\ (F\ y)\ t) = subst\ y\ (F\ y)\ (subst\ x\ (F\ x)\ t)$   
 ⟨proof⟩

**lemma**  
**assumes**  $finite\ V\ i \notin V$   
**shows** *ssubst\_insert*:  $ssubst\ t\ (insert\ i\ V)\ F = subst\ i\ (F\ i)\ (ssubst\ t\ V\ F)$  (**is** *?thesis1*)  
**and** *ssubst\_insert2*:  $ssubst\ t\ (insert\ i\ V)\ F = ssubst\ (subst\ i\ (F\ i)\ t)\ V\ F$  (**is** *?thesis2*)  
 ⟨proof⟩

**lemma** *ssubst\_insert\_if*:  
 $finite\ V \implies$   
 $ssubst\ t\ (insert\ i\ V)\ F = (if\ i \in V\ then\ ssubst\ t\ V\ F$   
 $else\ subst\ i\ (F\ i)\ (ssubst\ t\ V\ F))$   
 ⟨proof⟩

**lemma** *ssubst\_single* [simp]:  $ssubst\ t\ \{i\}\ F = subst\ i\ (F\ i)\ t$   
 ⟨proof⟩

**lemma** *ssubst\_Var\_if* [simp]:  
**assumes**  $finite\ V$   
**shows**  $ssubst\ (Var\ i)\ V\ F = (if\ i \in V\ then\ F\ i\ else\ Var\ i)$   
 ⟨proof⟩

**lemma** *ssubst\_Zero* [simp]:  $finite\ V \implies ssubst\ Zero\ V\ F = Zero$   
 ⟨proof⟩

**lemma** *ssubst\_Eats* [simp]:  $finite\ V \implies ssubst\ (Eats\ t\ u)\ V\ F = Eats\ (ssubst\ t\ V\ F)\ (ssubst\ u\ V\ F)$   
 ⟨proof⟩

**lemma** *ssubst\_SUCC* [simp]:  $finite\ V \implies ssubst\ (SUCC\ t)\ V\ F = SUCC\ (ssubst\ t\ V\ F)$   
 ⟨proof⟩

**lemma** *ssubst\_ORD\_OF* [simp]:  $finite\ V \implies ssubst\ (ORD\_OF\ n)\ V\ F = ORD\_OF\ n$   
 ⟨proof⟩

**lemma** *ssubst\_HPair* [simp]:  
 $finite\ V \implies ssubst\ (HPair\ t\ u)\ V\ F = HPair\ (ssubst\ t\ V\ F)\ (ssubst\ u\ V\ F)$   
 ⟨proof⟩

**lemma** *ssubst\_HTuple* [simp]:  $finite\ V \implies ssubst\ (HTuple\ n)\ V\ F = (HTuple\ n)$   
 ⟨proof⟩

**lemma** *ssubst\_Subset*:  
**assumes**  $finite\ V$  **shows**  $ssubst\ [t\ SUBS\ u]\ V\ V\ F = Q\_Subset\ (ssubst\ [t]\ V\ V\ F)\ (ssubst\ [u]\ V\ V\ F)$   
 ⟨proof⟩

**lemma** *fresh\_ssubst*:  
**assumes**  $finite\ V\ a \# p \cdot V\ a \# t$   
**shows**  $a \# ssubst\ t\ V\ F$

*<proof>*

**lemma** *fresh\_ssubst'*:

**assumes** *finite V atom i # t atom (p · i) # t*

**shows** *atom i # ssubst t V F*

*<proof>*

**lemma** *ssubst\_vquot\_Ex*:

$\llbracket \text{finite } V; \text{ atom } i \# p \cdot V \rrbracket$

$\implies \text{ssubst } [Ex\ i\ A](\text{insert } i\ V)\ (\text{insert } i\ V)\ F = \text{ssubst } [Ex\ i\ A]\ V\ V\ F$

*<proof>*

**lemma** *ground\_ssubst\_eq*:  $\llbracket \text{finite } V; \text{ supp } t = \{\} \rrbracket \implies \text{ssubst } t\ V\ F = t$

*<proof>*

**lemma** *ssubst\_quot\_tm* [*simp*]:

**fixes** *t::tm* **shows** *finite V*  $\implies \text{ssubst } \langle t \rangle V\ F = \langle t \rangle$

*<proof>*

**lemma** *ssubst\_quot\_fm* [*simp*]:

**fixes** *A::fm* **shows** *finite V*  $\implies \text{ssubst } \langle A \rangle V\ F = \langle A \rangle$

*<proof>*

**lemma** *atom\_in\_p\_Vs*:  $\llbracket i \in p \cdot V; V \subseteq Vs \rrbracket \implies i \in p \cdot Vs$

*<proof>*

## 8.3 The Main Theorems of Section 7

**lemma** *SubstTermP\_vquot\_dbtm*:

**assumes** *w: w*  $\in Vs - V$  **and** *V: V*  $\subseteq Vs$  *V' = p · V*

**and** *s: supp dbtm*  $\subseteq \text{atom } 'Vs$

**shows**

*insert (ConstP (F w)) {ConstP (F i) | i. i*  $\in V$

$\vdash \text{SubstTermP } \langle \text{Var } w \rangle (F w)$

$(\text{ssubst } (\text{vquot\_dbtm } V\ \text{dbtm})\ V\ F)$

$(\text{subst } w\ (F w)\ (\text{ssubst } (\text{vquot\_dbtm } (\text{insert } w\ V)\ \text{dbtm})\ V\ F))$

*<proof>*

**lemma** *SubstFormP\_vquot\_dbfm*:

**assumes** *w: w*  $\in Vs - V$  **and** *V: V*  $\subseteq Vs$  *V' = p · V*

**and** *s: supp dbfm*  $\subseteq \text{atom } 'Vs$

**shows**

*insert (ConstP (F w)) {ConstP (F i) | i. i*  $\in V$

$\vdash \text{SubstFormP } \langle \text{Var } w \rangle (F w)$

$(\text{ssubst } (\text{vquot\_dbfm } V\ \text{dbfm})\ V\ F)$

$(\text{subst } w\ (F w)\ (\text{ssubst } (\text{vquot\_dbfm } (\text{insert } w\ V)\ \text{dbfm})\ V\ F))$

*<proof>*

Lemmas 7.5 and 7.6

**lemma** *ssubst\_SubstFormP*:

**fixes** *A::fm*

**assumes** *w: w*  $\in Vs - V$  **and** *V: V*  $\subseteq Vs$  *V' = p · V*

**and** *s: supp A*  $\subseteq \text{atom } 'Vs$

**shows**

*insert (ConstP (F w)) {ConstP (F i) | i. i*  $\in V$

$\vdash \text{SubstFormP } \langle \text{Var } w \rangle (F w)$

$(\text{ssubst } [A]\ V\ V\ F)$

$(\text{ssubst } [A](\text{insert } w\ V)\ (\text{insert } w\ V)\ F)$

*<proof>*

Theorem 7.3

**theorem** *PfP\_implies\_PfP\_ssubst:*

**fixes**  $\beta::fm$

**assumes**  $\beta: \{\} \vdash PfP \llbracket \beta \rrbracket$

**and**  $V: V \subseteq Vs$

**and**  $s: supp \beta \subseteq atom \ ' \ Vs$

**shows**  $\{ConstP (F i) \mid i. i \in V\} \vdash PfP (ssubst \llbracket \beta \rrbracket V V F)$

*<proof>*

**end**

**end**

## Chapter 9

# Quotations of the Free Variables

```
theory Quote
imports Pseudo_Coding
begin
```

### 9.1 Sequence version of the “Special p-Function, F\*”

The definition below describes a relation, not a function. This material relates to Section 8, but omits the ordering of the universe.

#### 9.1.1 Defining the syntax: quantified body

```
nominal_function SeqQuoteP :: tm ⇒ tm ⇒ tm ⇒ tm ⇒ fm
where [[atom l # (s,k,sl,sl',m,n,sm,sm',sn,sn');
      atom sl # (s,sl',m,n,sm,sm',sn,sn'); atom sl' # (s,m,n,sm,sm',sn,sn');
      atom m # (s,n,sm,sm',sn,sn'); atom n # (s,sm,sm',sn,sn');
      atom sm # (s,sm',sn,sn'); atom sm' # (s,sn,sn');
      atom sn # (s,sn'); atom sn' # s]] ⇒⇒
SeqQuoteP t u s k =
  LstSeqP s k (HPair t u) AND
  All2 l (SUCC k) (Ex sl (Ex sl' (HPair (Var l) (HPair (Var sl) (Var sl'))) IN s AND
    ((Var sl EQ Zero AND Var sl' EQ Zero) OR
    Ex m (Ex n (Ex sm (Ex sm' (Ex sn (Ex sn' (Var m IN Var l AND Var n IN Var l AND
      HPair (Var m) (HPair (Var sm) (Var sm'))) IN s AND
      HPair (Var n) (HPair (Var sn) (Var sn'))) IN s AND
      Var sl EQ Eats (Var sm) (Var sn) AND
      Var sl' EQ Q_Eats (Var sm') (Var sn'))))))))))))
```

<proof>

**nominal\_termination** (eqvt)

<proof>

**lemma**

```
shows SeqQuoteP_fresh_iff [simp]:
  a # SeqQuoteP t u s k ⟷ a # t ∧ a # u ∧ a # s ∧ a # k (is ?thesis1)
and SeqQuoteP_sf [iff]:
  Sigma_fm (SeqQuoteP t u s k) (is ?thsf)
and SeqQuoteP_imp_OrdP:
  { SeqQuoteP t u s k } ⊢ OrdP k (is ?thord)
and SeqQuoteP_imp_LstSeqP:
  { SeqQuoteP t u s k } ⊢ LstSeqP s k (HPair t u) (is ?thlstseq)
```

<proof>

**lemma** *SeqQuoteP\_subst* [*simp*]:  
 (*SeqQuoteP* *t u s k*)(*j::=w*) =  
*SeqQuoteP* (*subst j w t*) (*subst j w u*) (*subst j w s*) (*subst j w k*)  
 ⟨*proof*⟩

**declare** *SeqQuoteP.simps* [*simp del*]

## 9.1.2 Correctness properties

**lemma** *SeqQuoteP\_lemma*:  
**fixes** *m::name* **and** *sm::name* **and** *sm'::name* **and** *n::name* **and** *sn::name* **and** *sn'::name*  
**assumes** *atom m*  $\#$  (*t,u,s,k,n,sm,sm',sn,sn'*) *atom n*  $\#$  (*t,u,s,k,sm,sm',sn,sn'*)  
*atom sm*  $\#$  (*t,u,s,k,sm',sn,sn'*) *atom sm'*  $\#$  (*t,u,s,k,sn,sn'*)  
*atom sn*  $\#$  (*t,u,s,k,sn'*) *atom sn'*  $\#$  (*t,u,s,k*)  
**shows** { *SeqQuoteP t u s k* }  
 $\vdash$  (*t EQ Zero AND u EQ Zero*) OR  
*Ex m* (*Ex n* (*Ex sm* (*Ex sm'* (*Ex sn* (*Ex sn'* (*Var m IN k AND Var n IN k AND*  
*SeqQuoteP* (*Var sm*) (*Var sm'*) *s* (*Var m*) AND  
*SeqQuoteP* (*Var sn*) (*Var sn'*) *s* (*Var n*) AND  
*t EQ Eats* (*Var sm*) (*Var sn*) AND  
*u EQ Q\_Eats* (*Var sm'*) (*Var sn'*)))))))))  
 ⟨*proof*⟩

## 9.2 The “special function” itself

**nominal\_function** *QuoteP* :: *tm*  $\Rightarrow$  *tm*  $\Rightarrow$  *fm*  
**where**  $\llbracket$ *atom s*  $\#$  (*t,u,k*); *atom k*  $\#$  (*t,u*) $\rrbracket \Longrightarrow$   
*QuoteP t u* = *Ex s* (*Ex k* (*SeqQuoteP t u* (*Var s*) (*Var k*)))  
 ⟨*proof*⟩

**nominal\_termination** (*eqvt*)  
 ⟨*proof*⟩

**lemma**  
**shows** *QuoteP\_fresh\_iff* [*simp*]: *a*  $\#$  *QuoteP t u*  $\longleftrightarrow$  *a*  $\#$  *t*  $\wedge$  *a*  $\#$  *u* (**is** *?thesis1*)  
**and** *QuoteP\_sf* [*iff*]: *Sigma\_fm* (*QuoteP t u*) (**is** *?thsf*)  
 ⟨*proof*⟩

**lemma** *QuoteP\_subst* [*simp*]:  
 (*QuoteP t u*)(*j::=w*) = *QuoteP* (*subst j w t*) (*subst j w u*)  
 ⟨*proof*⟩

**declare** *QuoteP.simps* [*simp del*]

### 9.2.1 Correctness properties

**lemma** *QuoteP\_Zero*: { }  $\vdash$  *QuoteP Zero Zero*  
 ⟨*proof*⟩

**lemma** *SeqQuoteP\_Eats*:  
**assumes** *atom s*  $\#$  (*k,s1,s2,k1,k2,t1,t2,u1,u2*) *atom k*  $\#$  (*s1,s2,k1,k2,t1,t2,u1,u2*)  
**shows** {*SeqQuoteP t1 u1 s1 k1*, *SeqQuoteP t2 u2 s2 k2*}  $\vdash$   
*Ex s* (*Ex k* (*SeqQuoteP* (*Eats t1 t2*) (*Q\_Eats u1 u2*) (*Var s*) (*Var k*)))  
 ⟨*proof*⟩

**lemma** *QuoteP\_Eats*:  $\{ \text{QuoteP } t1 \ u1, \text{QuoteP } t2 \ u2 \} \vdash \text{QuoteP } (\text{Eats } t1 \ t2) \ (Q\_Eats \ u1 \ u2)$   
 $\langle \text{proof} \rangle$

**lemma** *exists\_QuoteP*:  
**assumes**  $j: \text{atom } j \ \# \ x$  **shows**  $\{ \} \vdash \text{Ex } j \ (\text{QuoteP } x \ (\text{Var } j))$   
 $\langle \text{proof} \rangle$

**lemma** *QuoteP\_imp\_ConstP*:  $\{ \text{QuoteP } x \ y \} \vdash \text{ConstP } y$   
 $\langle \text{proof} \rangle$

**lemma** *SeqQuoteP\_imp\_QuoteP*:  $\{ \text{SeqQuoteP } t \ u \ s \ k \} \vdash \text{QuoteP } t \ u$   
 $\langle \text{proof} \rangle$

**lemmas** *QuoteP\_I* = *SeqQuoteP\_imp\_QuoteP* [THEN *cut1*]

## 9.3 The Operator *quote\_all*

### 9.3.1 Definition and basic properties

**definition** *quote\_all* ::  $[\text{perm}, \text{name set}] \Rightarrow \text{fm set}$   
**where**  $\text{quote\_all } p \ V = \{ \text{QuoteP } (\text{Var } i) \ (\text{Var } (p \cdot i)) \mid i. i \in V \}$

**lemma** *quote\_all\_empty* [*simp*]:  $\text{quote\_all } p \ \{ \} = \{ \}$   
 $\langle \text{proof} \rangle$

**lemma** *quote\_all\_insert* [*simp*]:  
 $\text{quote\_all } p \ (\text{insert } i \ V) = \text{insert } (\text{QuoteP } (\text{Var } i) \ (\text{Var } (p \cdot i))) \ (\text{quote\_all } p \ V)$   
 $\langle \text{proof} \rangle$

**lemma** *finite\_quote\_all* [*simp*]:  $\text{finite } V \Longrightarrow \text{finite } (\text{quote\_all } p \ V)$   
 $\langle \text{proof} \rangle$

**lemma** *fresh\_quote\_all* [*simp*]:  $\text{finite } V \Longrightarrow i \ \# \ \text{quote\_all } p \ V \longleftrightarrow i \ \# \ V \wedge i \ \# \ p \cdot V$   
 $\langle \text{proof} \rangle$

**lemma** *fresh\_quote\_all\_mem*:  $\llbracket A \in \text{quote\_all } p \ V; \text{finite } V; i \ \# \ V; i \ \# \ p \cdot V \rrbracket \Longrightarrow i \ \# \ A$   
 $\langle \text{proof} \rangle$

**lemma** *quote\_all\_perm\_eq*:  
**assumes**  $\text{finite } V \ \text{atom } i \ \# \ (p, V) \ \text{atom } i' \ \# \ (p, V)$   
**shows**  $\text{quote\_all } ((\text{atom } i \Rightarrow \text{atom } i') + p) \ V = \text{quote\_all } p \ V$   
 $\langle \text{proof} \rangle$

### 9.3.2 Transferring theorems to the level of derivability

**context** *quote\_perm*  
**begin**

**lemma** *QuoteP\_imp\_ConstP\_F\_hyps*:  
**assumes**  $Us \subseteq Vs \ \{ \text{ConstP } (F \ i) \mid i. i \in Us \} \vdash A$  **shows**  $\text{quote\_all } p \ Us \vdash A$   
 $\langle \text{proof} \rangle$

Lemma 8.3

**theorem** *quote\_all\_PfP\_ssubst*:  
**assumes**  $\beta: \{ \} \vdash \beta$   
**and**  $V: V \subseteq Vs$   
**and**  $s: \text{supp } \beta \subseteq \text{atom } ' Vs$   
**shows**  $\text{quote\_all } p \ V \vdash \text{PfP } (\text{ssubst } [\beta] V \ V \ F)$

*<proof>*

Lemma 8.4

**corollary** *quote\_all\_MonPon\_PfP\_ssubst:*

**assumes**  $A: \{\} \vdash \alpha \text{ IMP } \beta$

**and**  $V: V \subseteq Vs$

**and**  $s: \text{supp } \alpha \subseteq \text{atom } ' Vs \text{ supp } \beta \subseteq \text{atom } ' Vs$

**shows**  $\text{quote\_all } p \ V \vdash \text{PfP } (\text{ssubst } [\alpha] \ V \ V \ F) \ \text{IMP} \ \text{PfP } (\text{ssubst } [\beta] \ V \ V \ F)$

*<proof>*

Lemma 8.4b

**corollary** *quote\_all\_MonPon2\_PfP\_ssubst:*

**assumes**  $A: \{\} \vdash \alpha1 \ \text{IMP} \ \alpha2 \ \text{IMP} \ \beta$

**and**  $V: V \subseteq Vs$

**and**  $s: \text{supp } \alpha1 \subseteq \text{atom } ' Vs \ \text{supp } \alpha2 \subseteq \text{atom } ' Vs \ \text{supp } \beta \subseteq \text{atom } ' Vs$

**shows**  $\text{quote\_all } p \ V \vdash \text{PfP } (\text{ssubst } [\alpha1] \ V \ V \ F) \ \text{IMP} \ \text{PfP } (\text{ssubst } [\alpha2] \ V \ V \ F) \ \text{IMP} \ \text{PfP } (\text{ssubst } [\beta] \ V \ V \ F)$

*<proof>*

**lemma** *quote\_all\_Disj\_I1\_PfP\_ssubst:*

**assumes**  $V \subseteq Vs \ \text{supp } \alpha \subseteq \text{atom } ' Vs \ \text{supp } \beta \subseteq \text{atom } ' Vs$

**and** *prems:*  $H \vdash \text{PfP } (\text{ssubst } [\alpha] \ V \ V \ F) \ \text{quote\_all } p \ V \subseteq H$

**shows**  $H \vdash \text{PfP } (\text{ssubst } [\alpha \ \text{OR} \ \beta] \ V \ V \ F)$

*<proof>*

**lemma** *quote\_all\_Disj\_I2\_PfP\_ssubst:*

**assumes**  $V \subseteq Vs \ \text{supp } \alpha \subseteq \text{atom } ' Vs \ \text{supp } \beta \subseteq \text{atom } ' Vs$

**and** *prems:*  $H \vdash \text{PfP } (\text{ssubst } [\beta] \ V \ V \ F) \ \text{quote\_all } p \ V \subseteq H$

**shows**  $H \vdash \text{PfP } (\text{ssubst } [\alpha \ \text{OR} \ \beta] \ V \ V \ F)$

*<proof>*

**lemma** *quote\_all\_Conj\_I\_PfP\_ssubst:*

**assumes**  $V \subseteq Vs \ \text{supp } \alpha \subseteq \text{atom } ' Vs \ \text{supp } \beta \subseteq \text{atom } ' Vs$

**and** *prems:*  $H \vdash \text{PfP } (\text{ssubst } [\alpha] \ V \ V \ F) \ H \vdash \text{PfP } (\text{ssubst } [\beta] \ V \ V \ F) \ \text{quote\_all } p \ V \subseteq H$

**shows**  $H \vdash \text{PfP } (\text{ssubst } [\alpha \ \text{AND} \ \beta] \ V \ V \ F)$

*<proof>*

**lemma** *quote\_all\_Contra\_PfP\_ssubst:*

**assumes**  $V \subseteq Vs \ \text{supp } \alpha \subseteq \text{atom } ' Vs$

**shows**  $\text{quote\_all } p \ V$

$\vdash \text{PfP } (\text{ssubst } [\alpha] \ V \ V \ F) \ \text{IMP} \ \text{PfP } (\text{ssubst } [\text{Neg } \alpha] \ V \ V \ F) \ \text{IMP} \ \text{PfP } (\text{ssubst } [\text{Fls}] \ V \ V \ F)$

*<proof>*

**lemma** *fresh\_ssubst\_dbtm:*  $\llbracket \text{atom } i \ \# \ p \cdot V; \ V \subseteq Vs \rrbracket \implies \text{atom } i \ \# \ \text{ssubst } (\text{vquot\_dbtm } V \ t) \ V \ F$

*<proof>*

**lemma** *fresh\_ssubst\_dbfm:*  $\llbracket \text{atom } i \ \# \ p \cdot V; \ V \subseteq Vs \rrbracket \implies \text{atom } i \ \# \ \text{ssubst } (\text{vquot\_dbfm } V \ A) \ V \ F$

*<proof>*

**lemma** *fresh\_ssubst\_fm:*

**fixes**  $A::\text{fm}$  **shows**  $\llbracket \text{atom } i \ \# \ p \cdot V; \ V \subseteq Vs \rrbracket \implies \text{atom } i \ \# \ \text{ssubst } ([A] \ V) \ V \ F$

*<proof>*

**end**

## 9.4 Star Property. Equality and Membership: Lemmas 9.3 and 9.4

**lemma** *SeqQuoteP\_Mem\_imp\_QMem\_and\_Subset*:  
**assumes**  $atom\ i \# (j, j', i', si, ki, sj, kj)$   $atom\ i' \# (j, j', si, ki, sj, kj)$   
 $atom\ j \# (j', si, ki, sj, kj)$   $atom\ j' \# (si, ki, sj, kj)$   
 $atom\ si \# (ki, sj, kj)$   $atom\ sj \# (ki, kj)$   
**shows**  $\{SeqQuoteP\ (Var\ i)\ (Var\ i')\ (Var\ si)\ ki,\ SeqQuoteP\ (Var\ j)\ (Var\ j')\ (Var\ sj)\ kj\}$   
 $\vdash (Var\ i\ IN\ Var\ j\ IMP\ Pfp\ (Q\_Mem\ (Var\ i')\ (Var\ j')))\ AND$   
 $(Var\ i\ SUBS\ Var\ j\ IMP\ Pfp\ (Q\_Subset\ (Var\ i')\ (Var\ j')))$   
*<proof>*

**lemma**  
**assumes**  $atom\ i \# (j, j', i')$   $atom\ i' \# (j, j')$   $atom\ j \# (j')$   
**shows** *QuoteP\_Mem\_imp\_QMem*:  
 $\{QuoteP\ (Var\ i)\ (Var\ i'),\ QuoteP\ (Var\ j)\ (Var\ j'),\ Var\ i\ IN\ Var\ j\}$   
 $\vdash Pfp\ (Q\_Mem\ (Var\ i')\ (Var\ j'))\ \ (is\ ?thesis1)$   
**and** *QuoteP\_Mem\_imp\_QSubset*:  
 $\{QuoteP\ (Var\ i)\ (Var\ i'),\ QuoteP\ (Var\ j)\ (Var\ j'),\ Var\ i\ SUBS\ Var\ j\}$   
 $\vdash Pfp\ (Q\_Subset\ (Var\ i')\ (Var\ j'))\ \ (is\ ?thesis2)$   
*<proof>*

## 9.5 Star Property. Universal Quantifier: Lemma 9.7

**lemma** (**in** *quote\_perm*) *SeqQuoteP\_Mem\_imp\_All2*:  
**assumes** *IH*:  $insert\ (QuoteP\ (Var\ i)\ (Var\ i'))\ (quote\_all\ p\ Vs)$   
 $\vdash \alpha\ IMP\ Pfp\ (ssubst\ [\alpha]\ (insert\ i\ Vs)\ (insert\ i\ Vs)\ Fi)$   
**and** *sp*:  $supp\ \alpha - \{atom\ i\} \subseteq atom\ 'Vs$   
**and** *j*:  $j \in Vs$  **and** *j'*:  $p \cdot j = j'$   
**and** *pi*:  $pi = (atom\ i \equiv atom\ i') + p$   
**and** *Fi*:  $Fi = make\_F\ (insert\ i\ Vs)\ pi$   
**and** *atoms*:  $atom\ i \# (j, j', s, k, p)$   $atom\ i' \# (i, p, \alpha)$   
 $atom\ j \# (j', s, k, \alpha)$   $atom\ j' \# (s, k, \alpha)$   
 $atom\ s \# (k, \alpha)$   $atom\ k \# (\alpha, p)$   
**shows**  $insert\ (SeqQuoteP\ (Var\ j)\ (Var\ j')\ (Var\ s)\ (Var\ k))\ (quote\_all\ p\ (Vs - \{j\}))$   
 $\vdash All2\ i\ (Var\ j)\ \alpha\ IMP\ Pfp\ (ssubst\ [All2\ i\ (Var\ j)\ \alpha]\ Vs\ Vs\ F)$   
*<proof>*

**lemma** (**in** *quote\_perm*) *quote\_all\_Mem\_imp\_All2*:  
**assumes** *IH*:  $insert\ (QuoteP\ (Var\ i)\ (Var\ i'))\ (quote\_all\ p\ Vs)$   
 $\vdash \alpha\ IMP\ Pfp\ (ssubst\ [\alpha]\ (insert\ i\ Vs)\ (insert\ i\ Vs)\ Fi)$   
**and** *supp*:  $All2\ i\ (Var\ j)\ \alpha \subseteq atom\ 'Vs$   
**and** *j*:  $atom\ j \# (i, \alpha)$  **and** *i*:  $atom\ i \# p$  **and** *i'*:  $atom\ i' \# (i, p, \alpha)$   
**and** *pi*:  $pi = (atom\ i \equiv atom\ i') + p$   
**and** *Fi*:  $Fi = make\_F\ (insert\ i\ Vs)\ pi$   
**shows**  $insert\ (All2\ i\ (Var\ j)\ \alpha)\ (quote\_all\ p\ Vs) \vdash Pfp\ (ssubst\ [All2\ i\ (Var\ j)\ \alpha]\ Vs\ Vs\ F)$   
*<proof>*

## 9.6 The Derivability Condition, Theorem 9.1

**lemma** *SpecI*:  $H \vdash A\ IMP\ Ex\ i\ A$   
*<proof>*

**lemma** *star*:  
**fixes**  $p :: perm$  **and**  $F :: name \Rightarrow tm$   
**assumes**  $C: ss\_fm\ \alpha$

**and**  $p: \text{atom } ' (p \cdot V) \#* V -p = p$   
**and**  $V: \text{finite } V \text{ supp } \alpha \subseteq \text{atom } ' V$   
**and**  $F: F = \text{make\_F } V p$   
**shows**  $\text{insert } \alpha (\text{quote\_all } p V) \vdash \text{PfP } (\text{ssubst } [\alpha] V V F)$   
 $\langle \text{proof} \rangle$

**theorem** *Provability*:  
**assumes**  $\text{Sigma\_fm } \alpha \text{ ground\_fm } \alpha$   
**shows**  $\{\alpha\} \vdash \text{PfP } \langle \alpha \rangle$   
 $\langle \text{proof} \rangle$

**end**

## Chapter 10

# Uniqueness Results: Syntactic Relations are Functions

```
theory Functions
imports Coding_Predicates
begin
```

### 10.0.1 SeqStTermP

```
lemma not_IndP_VarP: {IndP x, VarP x} ⊢ A
⟨proof⟩
```

It IS a pair, but not just any pair.

```
lemma IndP_HPaiRE: insert (IndP (HPair (HPair Zero (HPair Zero Zero)) x)) H ⊢ A
⟨proof⟩
```

```
lemma atom_HPaiRE:
  assumes H ⊢ x EQ HPair (HPair Zero (HPair Zero Zero)) y
  shows insert (IndP x OR x NEQ v) H ⊢ A
⟨proof⟩
```

```
lemma SeqStTermP_lemma:
  assumes atom m # (v,i,t,u,s,k,n,sm,sm',sn,sn') atom n # (v,i,t,u,s,k,sm,sm',sn,sn')
  atom sm # (v,i,t,u,s,k,sm',sn,sn') atom sm' # (v,i,t,u,s,k,sn,sn')
  atom sn # (v,i,t,u,s,k,sn') atom sn' # (v,i,t,u,s,k)
  shows { SeqStTermP v i t u s k }
  ⊢ ((t EQ v AND u EQ i) OR
    ((IndP t OR t NEQ v) AND u EQ t)) OR
    (Ex m (Ex n (Ex sm (Ex sm' (Ex sn (Ex sn' (Var m IN k AND Var n IN k AND
      SeqStTermP v i (Var sm) (Var sm') s (Var m) AND
      SeqStTermP v i (Var sn) (Var sn') s (Var n) AND
      t EQ Q_Eats (Var sm) (Var sn) AND
      u EQ Q_Eats (Var sm') (Var sn'))))))))
⟨proof⟩
```

```
lemma SeqStTermP_unique: {SeqStTermP v a t u s kk, SeqStTermP v a t u' s' kk'} ⊢ u' EQ u
⟨proof⟩
```

```
theorem SubstTermP_unique: {SubstTermP v tm t u, SubstTermP v tm t u'} ⊢ u' EQ u
⟨proof⟩
```

### 10.0.2 *SubstAtomicP*

**lemma** *SubstTermP\_eq*:

$\llbracket H \vdash \text{SubstTermP } v \text{ tm } x \ z; \text{ insert } (\text{SubstTermP } v \text{ tm } y \ z) \ H \vdash A \rrbracket \implies \text{insert } (x \text{ EQ } y) \ H \vdash A$   
 <proof>

**lemma** *SubstAtomicP\_unique*:  $\{ \text{SubstAtomicP } v \text{ tm } x \ y, \text{ SubstAtomicP } v \text{ tm } x \ y' \} \vdash y' \text{ EQ } y$   
 <proof>

### 10.0.3 *SeqSubstFormP*

**lemma** *SeqSubstFormP\_lemma*:

**assumes**  $\text{atom } m \# (v, u, x, y, s, k, n, sm, sm', sn, sn')$   $\text{atom } n \# (v, u, x, y, s, k, sm, sm', sn, sn')$   
 $\text{atom } sm \# (v, u, x, y, s, k, sm', sn, sn')$   $\text{atom } sm' \# (v, u, x, y, s, k, sn, sn')$   
 $\text{atom } sn \# (v, u, x, y, s, k, sn')$   $\text{atom } sn' \# (v, u, x, y, s, k)$   
**shows**  $\{ \text{SeqSubstFormP } v \ u \ x \ y \ s \ k \}$   
 $\vdash \text{SubstAtomicP } v \ u \ x \ y \ \text{OR}$   
 $\text{Ex } m \ (\text{Ex } n \ (\text{Ex } sm \ (\text{Ex } sm' \ (\text{Ex } sn \ (\text{Ex } sn' \ (\text{Var } m \ \text{IN } k \ \text{AND } \text{Var } n \ \text{IN } k \ \text{AND}$   
 $\text{SeqSubstFormP } v \ u \ (\text{Var } sm) \ (\text{Var } sm') \ s \ (\text{Var } m) \ \text{AND}$   
 $\text{SeqSubstFormP } v \ u \ (\text{Var } sn) \ (\text{Var } sn') \ s \ (\text{Var } n) \ \text{AND}$   
 $((x \ \text{EQ} \ Q\_Disj \ (\text{Var } sm) \ (\text{Var } sn) \ \text{AND} \ y \ \text{EQ} \ Q\_Disj \ (\text{Var } sm') \ (\text{Var } sn')) \ \text{OR}$   
 $(x \ \text{EQ} \ Q\_Neg \ (\text{Var } sm) \ \text{AND} \ y \ \text{EQ} \ Q\_Neg \ (\text{Var } sm')) \ \text{OR}$   
 $(x \ \text{EQ} \ Q\_Ex \ (\text{Var } sm) \ \text{AND} \ y \ \text{EQ} \ Q\_Ex \ (\text{Var } sm')))))))))))$

<proof>

**lemma**

**shows**  $\text{Neg\_SubstAtomicP\_Fls}$ :  $\{ y \ \text{EQ} \ Q\_Neg \ z, \text{ SubstAtomicP } v \ \text{tm } y \ y' \} \vdash \text{Fls}$  (is ?thesis1)  
**and**  $\text{Disj\_SubstAtomicP\_Fls}$ :  $\{ y \ \text{EQ} \ Q\_Disj \ z \ w, \text{ SubstAtomicP } v \ \text{tm } y \ y' \} \vdash \text{Fls}$  (is ?thesis2)  
**and**  $\text{Ex\_SubstAtomicP\_Fls}$ :  $\{ y \ \text{EQ} \ Q\_Ex \ z, \text{ SubstAtomicP } v \ \text{tm } y \ y' \} \vdash \text{Fls}$  (is ?thesis3)

<proof>

**lemma** *SeqSubstFormP\_eq*:

$\llbracket H \vdash \text{SeqSubstFormP } v \ \text{tm } x \ z \ s \ k; \text{ insert } (\text{SeqSubstFormP } v \ \text{tm } y \ z \ s \ k) \ H \vdash A \rrbracket$   
 $\implies \text{insert } (x \ \text{EQ} \ y) \ H \vdash A$   
 <proof>

**lemma** *SeqSubstFormP\_unique*:  $\{ \text{SeqSubstFormP } v \ a \ x \ y \ s \ k, \text{ SeqSubstFormP } v \ a \ x \ y' \ s' \ k' \} \vdash y' \ \text{EQ} \ y$   
 <proof>

### 10.0.4 *SubstFormP*

**theorem** *SubstFormP\_unique*:  $\{ \text{SubstFormP } v \ \text{tm } x \ y, \text{ SubstFormP } v \ \text{tm } x \ y' \} \vdash y' \ \text{EQ} \ y$   
 <proof>

**end**

# Chapter 11

## Section 6 Material and Gödel's First Incompleteness Theorem

```
theory Goedel_I
imports Pf_Predicates Functions II_Prelims
begin
```

### 11.1 The Function W and Lemma 6.1

#### 11.1.1 Predicate form, defined on sequences

```
nominal_function SeqWRP :: tm  $\Rightarrow$  tm  $\Rightarrow$  tm  $\Rightarrow$  fm
  where  $\llbracket$ atom l  $\#$  (s,k,sl); atom sl  $\#$  (s) $\rrbracket \Longrightarrow$ 
    SeqWRP s k y = LstSeqP s k y AND
    HPair Zero Zero IN s AND
    All2 l k (Ex sl (HPair (Var l) (Var sl) IN s AND
    HPair (SUCC (Var l)) (Q_Succ (Var sl)) IN s))
```

*<proof>*

```
nominal_termination (eqvt)
```

*<proof>*

```
lemma
```

```
shows SeqWRP_fresh_iff [simp]: a  $\#$  SeqWRP s k y  $\longleftrightarrow$  a  $\#$  s  $\wedge$  a  $\#$  k  $\wedge$  a  $\#$  y (is ?thesis1)
and SeqWRP_sf [iff]: Sigma_fm (SeqWRP s k y) (is ?thsf)
and SeqWRP_imp_OrdP: {SeqWRP s k t}  $\vdash$  OrdP k (is ?thOrd)
and SeqWRP_LstSeqP: {SeqWRP s k t}  $\vdash$  LstSeqP s k t (is ?thlstseq)
```

*<proof>*

```
lemma SeqWRP_subst [simp]:
```

```
(SeqWRP s k y)(i::=t) = SeqWRP (subst i t s) (subst i t k) (subst i t y)
```

*<proof>*

```
lemma SeqWRP_cong:
```

```
assumes H  $\vdash$  s EQ s' and H  $\vdash$  k EQ k' and H  $\vdash$  y EQ y'
shows H  $\vdash$  SeqWRP s k y IFF SeqWRP s' k' y'
```

*<proof>*

```
declare SeqWRP.simps [simp del]
```

### 11.1.2 Predicate form of W

**nominal\_function**  $WRP :: tm \Rightarrow tm \Rightarrow fm$

**where**  $\llbracket atom\ s\ \#(x,y) \rrbracket \Longrightarrow$

$WRP\ x\ y = Ex\ s\ (SeqWRP\ (Var\ s)\ x\ y)$

$\langle proof \rangle$

**nominal\_termination** ( $eqvt$ )

$\langle proof \rangle$

**lemma**

**shows**  $WRP\_fresh\_iff\ [simp]: a\ \#(WRP\ x\ y) \longleftrightarrow a\ \#x \wedge a\ \#y$  (**is**  $?thesis1$ )

**and**  $sigma\_fm\_WRP\ [simp]: Sigma\_fm\ (WRP\ x\ y)$  (**is**  $?thsf$ )

$\langle proof \rangle$

**lemma**  $WRP\_subst\ [simp]: (WRP\ x\ y)(i::=t) = WRP\ (subst\ i\ t\ x)\ (subst\ i\ t\ y)$

$\langle proof \rangle$

**lemma**  $WRP\_cong: H \vdash t\ EQ\ t' \Longrightarrow H \vdash u\ EQ\ u' \Longrightarrow H \vdash WRP\ t\ u\ IFF\ WRP\ t'\ u'$

$\langle proof \rangle$

**declare**  $WRP.simps\ [simp\ del]$

**lemma**  $ground\_WRP\ [simp]: ground\_fm\ (WRP\ x\ y) \longleftrightarrow ground\ x \wedge ground\ y$

$\langle proof \rangle$

**lemma**  $SeqWRP\_Zero: \{\} \vdash SyntaxN.Ex\ s\ (SeqWRP\ (Var\ s)\ Zero\ Zero)$

$\langle proof \rangle$

**lemma**  $WRP\_Zero: \{\} \vdash WRP\ Zero\ Zero$

$\langle proof \rangle$

**lemma**  $SeqWRP\_HPair\_Zero\_Zero: \{SeqWRP\ s\ k\ y\} \vdash HPair\ Zero\ Zero\ IN\ s$

$\langle proof \rangle$

**lemma**  $SeqWRP\_Succ:$

**assumes**  $atom\ s\ \#(s1,k1,y)$

**shows**  $\{SeqWRP\ s1\ k1\ y\} \vdash SyntaxN.Ex\ s\ (SeqWRP\ (Var\ s)\ (SUCC\ k1)\ (Q\_Succ\ y))$

$\langle proof \rangle$

**lemma**  $WRP\_Succ: \{OrdP\ i,\ WRP\ i\ y\} \vdash WRP\ (SUCC\ i)\ (Q\_Succ\ y)$

$\langle proof \rangle$

**lemma**  $WRP: \{\} \vdash WRP\ (ORD\_OF\ i)\ \llbracket ORD\_OF\ i \rrbracket$

$\langle proof \rangle$

**lemma**  $prove\_WRP: \{\} \vdash WRP\ \llbracket Var\ x \rrbracket \llbracket \llbracket Var\ x \rrbracket \rrbracket$

$\langle proof \rangle$

### 11.1.3 Proving that these relations are functions

**lemma**  $SeqWRP\_Zero\_E:$

**assumes**  $insert\ (y\ EQ\ Zero)\ H \vdash A\ H \vdash k\ EQ\ Zero$

**shows**  $insert\ (SeqWRP\ s\ k\ y)\ H \vdash A$

$\langle proof \rangle$

**lemma**  $SeqWRP\_SUCC\_lemma:$

**assumes**  $y': atom\ y' \#(s,k,y)$

**shows**  $\{SeqWRP\ s\ (SUCC\ k)\ y\} \vdash Ex\ y' (SeqWRP\ s\ k\ (Var\ y')\ AND\ y\ EQ\ Q\_Succ\ (Var\ y'))$

*<proof>*

**lemma** *SeqWRP\_SUCC\_E*:

**assumes**  $y'$ : *atom*  $y' \# (s, k, y)$  **and**  $k'$ :  $H \vdash k' \text{ EQ } (\text{SUCC } k)$

**shows**  $\text{insert } (\text{SeqWRP } s \ k' \ y) \ H \vdash \text{Ex } y' \ (\text{SeqWRP } s \ k \ (\text{Var } y') \ \text{AND } y \ \text{EQ } Q\_Succ \ (\text{Var } y'))$   
*<proof>*

**lemma** *SeqWRP\_unique*:  $\{ \text{OrdP } x, \text{SeqWRP } s \ x \ y, \text{SeqWRP } s' \ x \ y' \} \vdash y' \ \text{EQ } y$

*<proof>*

**theorem** *WRP\_unique*:  $\{ \text{OrdP } x, \text{WRP } x \ y, \text{WRP } x \ y' \} \vdash y' \ \text{EQ } y$

*<proof>*

## 11.2 The Function HF and Lemma 6.2

### 11.2.1 Defining the syntax: quantified body

**nominal\_function** *SeqHRP* ::  $tm \Rightarrow tm \Rightarrow tm \Rightarrow tm \Rightarrow fm$

**where**  $\llbracket \text{atom } l \# (s, k, sl, sl', m, n, sm, sm', sn, sn') \rrbracket$ ;

$\text{atom } sl \# (s, sl', m, n, sm, sm', sn, sn')$ ;

$\text{atom } sl' \# (s, m, n, sm, sm', sn, sn')$ ;

$\text{atom } m \# (s, n, sm, sm', sn, sn')$ ;

$\text{atom } n \# (s, sm, sm', sn, sn')$ ;

$\text{atom } sm \# (s, sm', sn, sn')$ ;

$\text{atom } sm' \# (s, sn, sn')$ ;

$\text{atom } sn \# (s, sn')$ ;

$\text{atom } sn' \# (s)$   $\implies$

*SeqHRP*  $x \ x' \ s \ k =$

$\text{LstSeqP } s \ k \ (\text{HPair } x \ x') \ \text{AND}$

$\text{All2 } l \ (\text{SUCC } k) \ (\text{Ex } sl \ (\text{Ex } sl' \ (\text{HPair } (\text{Var } l) \ (\text{HPair } (\text{Var } sl) \ (\text{Var } sl')) \ \text{IN } s \ \text{AND}$

$((\text{OrdP } (\text{Var } sl) \ \text{AND } \text{WRP } (\text{Var } sl) \ (\text{Var } sl')) \ \text{OR}$

$\text{Ex } m \ (\text{Ex } n \ (\text{Ex } sm \ (\text{Ex } sm' \ (\text{Ex } sn \ (\text{Ex } sn' \ (\text{Var } m \ \text{IN } \text{Var } l \ \text{AND } \text{Var } n \ \text{IN } \text{Var } l \ \text{AND}$

$\text{HPair } (\text{Var } m) \ (\text{HPair } (\text{Var } sm) \ (\text{Var } sm')) \ \text{IN } s \ \text{AND}$

$\text{HPair } (\text{Var } n) \ (\text{HPair } (\text{Var } sn) \ (\text{Var } sn')) \ \text{IN } s \ \text{AND}$

$\text{Var } sl \ \text{EQ } \text{HPair } (\text{Var } sm) \ (\text{Var } sn) \ \text{AND}$

$\text{Var } sl' \ \text{EQ } Q\_HPair \ (\text{Var } sm') \ (\text{Var } sn'))))))))))))$

*<proof>*

**nominal\_termination** (*eqvt*)

*<proof>*

**lemma**

**shows** *SeqHRP\_fresh\_iff* [*simp*]:

$a \# \text{SeqHRP } x \ x' \ s \ k \longleftrightarrow a \# x \wedge a \# x' \wedge a \# s \wedge a \# k$  (**is** *?thesis1*)

**and** *SeqHRP\_sf* [*iff*]:  $\text{Sigma\_fm } (\text{SeqHRP } x \ x' \ s \ k)$  (**is** *?thsf*)

**and** *SeqHRP\_imp\_OrdP*:  $\{ \text{SeqHRP } x \ y \ s \ k \} \vdash \text{OrdP } k$  (**is** *?thord*)

**and** *SeqHRP\_imp\_LstSeqP*:  $\{ \text{SeqHRP } x \ y \ s \ k \} \vdash \text{LstSeqP } s \ k \ (\text{HPair } x \ y)$  (**is** *?thlstseq*)

*<proof>*

**lemma** *SeqHRP\_subst* [*simp*]:

$(\text{SeqHRP } x \ x' \ s \ k)(i::=t) = \text{SeqHRP } (\text{subst } i \ t \ x) \ (\text{subst } i \ t \ x') \ (\text{subst } i \ t \ s) \ (\text{subst } i \ t \ k)$

*<proof>*

**lemma** *SeqHRP\_cong*:

**assumes**  $H \vdash x \ \text{EQ } x'$  **and**  $H \vdash y \ \text{EQ } y'$   $H \vdash s \ \text{EQ } s'$  **and**  $H \vdash k \ \text{EQ } k'$

**shows**  $H \vdash \text{SeqHRP } x \ y \ s \ k \ \text{IFF } \text{SeqHRP } x' \ y' \ s' \ k'$

*<proof>*

## 11.2.2 Defining the syntax: main predicate

**nominal\_function**  $HRP :: tm \Rightarrow tm \Rightarrow fm$   
**where**  $\llbracket atom\ s\ \#(x,x',k); atom\ k\ \#(x,x') \rrbracket \Longrightarrow$   
 $HRP\ x\ x' = Ex\ s\ (Ex\ k\ (SeqHRP\ x\ x'\ (Var\ s)\ (Var\ k)))$   
 $\langle proof \rangle$

**nominal\_termination** ( $eqvt$ )  
 $\langle proof \rangle$

**lemma**  
**shows**  $HRP\_fresh\_iff\ [simp]: a\ \#(HRP\ x\ x') \longleftrightarrow a\ \#x \wedge a\ \#x'$  (**is**  $?thesis1$ )  
**and**  $HRP\_sf\ [iff]: Sigma\_fm\ (HRP\ x\ x')$  (**is**  $?thsf$ )  
 $\langle proof \rangle$

**lemma**  $HRP\_subst\ [simp]: (HRP\ x\ x')(i::=t) = HRP\ (subst\ i\ t\ x)\ (subst\ i\ t\ x')$   
 $\langle proof \rangle$

## 11.2.3 Proving that these relations are functions

**lemma**  $SeqHRP\_lemma:$   
**assumes**  $atom\ m\ \#(x,x',s,k,n,sm,sm',sn,sn')$   $atom\ n\ \#(x,x',s,k,sm,sm',sn,sn')$   
 $atom\ sm\ \#(x,x',s,k,sm',sn,sn')$   $atom\ sm'\ \#(x,x',s,k,sn,sn')$   
 $atom\ sn\ \#(x,x',s,k,sn')$   $atom\ sn'\ \#(x,x',s,k)$   
**shows**  $\{ SeqHRP\ x\ x'\ s\ k \}$   
 $\vdash (OrdP\ x\ AND\ WRP\ x\ x')\ OR$   
 $Ex\ m\ (Ex\ n\ (Ex\ sm\ (Ex\ sm'\ (Ex\ sn\ (Ex\ sn'\ (Var\ m\ IN\ k\ AND\ Var\ n\ IN\ k\ AND$   
 $SeqHRP\ (Var\ sm)\ (Var\ sm')\ s\ (Var\ m)\ AND$   
 $SeqHRP\ (Var\ sn)\ (Var\ sn')\ s\ (Var\ n)\ AND$   
 $x\ EQ\ HPair\ (Var\ sm)\ (Var\ sn)\ AND$   
 $x'\ EQ\ Q\_HPair\ (Var\ sm')\ (Var\ sn'))))))))$   
 $\langle proof \rangle$

**lemma**  $SeqHRP\_unique: \{SeqHRP\ x\ y\ s\ u, SeqHRP\ x\ y'\ s'\ u'\} \vdash y' EQ y$   
 $\langle proof \rangle$

**theorem**  $HRP\_unique: \{HRP\ x\ y, HRP\ x\ y'\} \vdash y' EQ y$   
 $\langle proof \rangle$

**lemma**  $HRP\_ORD\_OF: \{ \} \vdash HRP\ (ORD\_OF\ i)\ \langle ORD\_OF\ i \rangle$   
 $\langle proof \rangle$

**lemma**  $SeqHRP\_HPair:$   
**assumes**  $atom\ s\ \#(k,s1,s2,k1,k2,x,y,x',y')$   $atom\ k\ \#(s1,s2,k1,k2,x,y,x',y')$   
**shows**  $\{SeqHRP\ x\ x'\ s1\ k1,$   
 $SeqHRP\ y\ y'\ s2\ k2\}$   
 $\vdash Ex\ s\ (Ex\ k\ (SeqHRP\ (HPair\ x\ y)\ (Q\_HPair\ x'\ y')\ (Var\ s)\ (Var\ k)))$   $\langle proof \rangle$   
**lemma**  $HRP\_HPair: \{HRP\ x\ x', HRP\ y\ y'\} \vdash HRP\ (HPair\ x\ y)\ (Q\_HPair\ x'\ y')$   
 $\langle proof \rangle$

**lemma**  $HRP\_HPair\_quot: \{HRP\ x\ \langle x \rangle, HRP\ y\ \langle y \rangle\} \vdash HRP\ (HPair\ x\ y)\ \langle HPair\ x\ y \rangle$   
 $\langle proof \rangle$

**lemma**  $prove\_HRP\_coding\_tm: fixes\ t::tm\ shows\ coding\_tm\ t \Longrightarrow \{ \} \vdash HRP\ t\ \langle t \rangle$   
 $\langle proof \rangle$

**lemmas**  $prove\_HRP = prove\_HRP\_coding\_tm[OF\ quot\_fm\_coding]$

### 11.3 The Function K and Lemma 6.3

```
nominal_function KRP :: tm ⇒ tm ⇒ tm ⇒ fm
  where atom y ‡ (v, x, x') ⇒
    KRP v x x' = Ex y (HRP x (Var y) AND SubstFormP v (Var y) x x')
  ⟨proof⟩

nominal_termination (eqvt)
  ⟨proof⟩

lemma KRP_fresh_iff [simp]: a ‡ KRP v x x' ↔ a ‡ v ∧ a ‡ x ∧ a ‡ x'
  ⟨proof⟩

lemma KRP_subst [simp]: (KRP v x x')(i::=t) = KRP (subst i t v) (subst i t x) (subst i t x')
  ⟨proof⟩

declare KRP.simps [simp del]

lemma prove_SubstFormP: {} ⊢ SubstFormP «Var i» «A» «A» «A(i::=A)»
  ⟨proof⟩

lemma prove_KRP: {} ⊢ KRP «Var i» «A» «A(i::=A)»
  ⟨proof⟩

lemma KRP_unique: {KRP v x y, KRP v x y'} ⊢ y' EQ y
  ⟨proof⟩

lemma KRP_subst_fm: {KRP «Var i» «β» (Var j)} ⊢ Var j EQ «β(i::=β)»
  ⟨proof⟩

end
```

# Chapter 12

## The Instantiation

**definition**  $Fvars\ t = \{a :: name. \neg\ atom\ a\ \#\ t\}$

**lemma**  $Fvars\_tm\_simps[simp]$ :

$Fvars\ Zero = \{\}$   
 $Fvars\ (Var\ a) = \{a\}$   
 $Fvars\ (Eats\ x\ y) = Fvars\ x \cup Fvars\ y$   
 $\langle proof \rangle$

**lemma**  $finite\_Fvars\_tm[simp]$ :

**fixes**  $t :: tm$   
**shows**  $finite\ (Fvars\ t)$   
 $\langle proof \rangle$

**lemma**  $Fvars\_fm\_simps[simp]$ :

$Fvars\ (x\ IN\ y) = Fvars\ x \cup Fvars\ y$   
 $Fvars\ (x\ EQ\ y) = Fvars\ x \cup Fvars\ y$   
 $Fvars\ (A\ OR\ B) = Fvars\ A \cup Fvars\ B$   
 $Fvars\ (A\ AND\ B) = Fvars\ A \cup Fvars\ B$   
 $Fvars\ (A\ IMP\ B) = Fvars\ A \cup Fvars\ B$   
 $Fvars\ Fls = \{\}$   
 $Fvars\ (Neg\ A) = Fvars\ A$   
 $Fvars\ (Ex\ a\ A) = Fvars\ A - \{a\}$   
 $Fvars\ (All\ a\ A) = Fvars\ A - \{a\}$   
 $\langle proof \rangle$

**lemma**  $finite\_Fvars\_fm[simp]$ :

**fixes**  $A :: fm$   
**shows**  $finite\ (Fvars\ A)$   
 $\langle proof \rangle$

**lemma**  $subst\_tm\_subst\_tm[simp]$ :

$x \neq y \implies atom\ x\ \#\ u \implies subst\ y\ u\ (subst\ x\ t\ v) = subst\ x\ (subst\ y\ u\ t)\ (subst\ y\ u\ v)$   
 $\langle proof \rangle$

**lemma**  $subst\_fm\_subst\_fm[simp]$ :

$x \neq y \implies atom\ x\ \#\ u \implies (A(x::=t))(y::=u) = (A(y::=u))(x::=subst\ y\ u\ t)$   
 $\langle proof \rangle$

**lemma**  $Fvars\_ground\_aux: Fvars\ t \subseteq B \implies ground\_aux\ t\ (atom\ 'B)$

$\langle proof \rangle$

**lemma** *ground\_Fvars*:  $ground\ t \longleftrightarrow Fvars\ t = \{\}$   
*<proof>*

**lemma** *Fvars\_ground\_fm\_aux*:  $Fvars\ A \subseteq B \implies ground\_fm\_aux\ A\ (atom\ 'B)$   
*<proof>*

**lemma** *ground\_fm\_Fvars*:  $ground\_fm\ A \longleftrightarrow Fvars\ A = \{\}$   
*<proof>*

**interpretation** *Generic\_Syntax* **where**

*var* = *UNIV* :: *name set*  
**and** *trm* = *UNIV* :: *tm set*  
**and** *fmla* = *UNIV* :: *fm set*  
**and** *Var* = *Var*  
**and** *FvarsT* = *Fvars*  
**and** *substT* =  $\lambda t\ u\ x.\ subst\ x\ u\ t$   
**and** *Fvars* = *Fvars*  
**and** *subst* =  $\lambda A\ u\ x.\ subst\_fm\ A\ x\ u$   
*<proof>*

**lemma** *coding\_tm\_Fvars\_empty[simp]*:  $coding\_tm\ t \implies Fvars\ t = \{\}$   
*<proof>*

**lemma** *Fvars\_empty\_ground[simp]*:  $Fvars\ t = \{\} \implies ground\ t$   
*<proof>*

**interpretation** *Syntax\_with\_Numerals* **where**

*var* = *UNIV* :: *name set*  
**and** *trm* = *UNIV* :: *tm set*  
**and** *fmla* = *UNIV* :: *fm set*  
**and** *num* =  $\{t.\ ground\ t\}$   
**and** *Var* = *Var*  
**and** *FvarsT* = *Fvars*  
**and** *substT* =  $\lambda t\ u\ x.\ subst\ x\ u\ t$   
**and** *Fvars* = *Fvars*  
**and** *subst* =  $\lambda A\ u\ x.\ subst\_fm\ A\ x\ u$   
*<proof>*

**declare** *FvarsT\_num[simp del]*

**interpretation** *Deduct2\_with\_False* **where**

*var* = *UNIV* :: *name set*  
**and** *trm* = *UNIV* :: *tm set*  
**and** *fmla* = *UNIV* :: *fm set*  
**and** *num* =  $\{t.\ ground\ t\}$   
**and** *Var* = *Var*  
**and** *FvarsT* = *Fvars*  
**and** *substT* =  $\lambda t\ u\ x.\ subst\ x\ u\ t$   
**and** *Fvars* = *Fvars*  
**and** *subst* =  $\lambda A\ u\ x.\ subst\_fm\ A\ x\ u$   
**and** *eql* = (*EQ*)  
**and** *cnj* = (*AND*)  
**and** *imp* = (*IMP*)  
**and** *all* = *All*  
**and** *exi* = *Ex*  
**and** *fls* = *Fls*  
**and** *prv* =  $(\vdash)\ \{\}$   
**and** *bprv* =  $(\vdash)\ \{\}$

*<proof>*

**interpretation HBL1 where**

*var = UNIV :: name set*  
**and** *trm = UNIV :: tm set*  
**and** *fmla = UNIV :: fm set*  
**and** *num = {t. ground t}*  
**and** *Var = Var*  
**and** *FvarsT = Fvars*  
**and** *substT =  $\lambda t u x. subst\ x\ u\ t$*   
**and** *Fvars = Fvars*  
**and** *subst =  $\lambda A u x. subst\_fm\ A\ x\ u$*   
**and** *eql = (EQ)*  
**and** *cnj = (AND)*  
**and** *imp = (IMP)*  
**and** *all = All*  
**and** *exi = Ex*  
**and** *prv = ( $\vdash$ ) {}*  
**and** *bprv = ( $\vdash$ ) {}*  
**and** *enc = quot*  
**and** *P = PfP (Var xx)*  
*<proof>*

**interpretation Goedel\_Form where**

*var = UNIV :: name set*  
**and** *trm = UNIV :: tm set*  
**and** *fmla = UNIV :: fm set*  
**and** *num = {t. ground t}*  
**and** *Var = Var*  
**and** *FvarsT = Fvars*  
**and** *substT =  $\lambda t u x. subst\ x\ u\ t$*   
**and** *Fvars = Fvars*  
**and** *subst =  $\lambda A u x. subst\_fm\ A\ x\ u$*   
**and** *eql = (EQ)*  
**and** *cnj = (AND)*  
**and** *imp = (IMP)*  
**and** *all = All*  
**and** *exi = Ex*  
**and** *fls = Fls*  
**and** *prv = ( $\vdash$ ) {}*  
**and** *bprv = ( $\vdash$ ) {}*  
**and** *enc = quot*  
**and** *S = KRP (quot (Var xx)) (Var xx) (Var yy)*  
**and** *P = PfP (Var xx)*  
*<proof>*

**interpretation g2: Goedel\_Second\_Assumptions where**

*var = UNIV :: name set*  
**and** *trm = UNIV :: tm set*  
**and** *fmla = UNIV :: fm set*  
**and** *num = {t. ground t}*  
**and** *Var = Var*  
**and** *FvarsT = Fvars*  
**and** *substT =  $\lambda t u x. subst\ x\ u\ t$*   
**and** *Fvars = Fvars*  
**and** *subst =  $\lambda A u x. subst\_fm\ A\ x\ u$*   
**and** *eql = (EQ)*  
**and** *cnj = (AND)*

```

and imp = (IMP)
and all = All
and exi = Ex
and fls = Fls
and prv = ( $\vdash$ ) {}
and bprv = ( $\vdash$ ) {}
and enc = quot
and S = KRP (quot (Var xx)) (Var xx) (Var yy)
and P = PfP (Var xx)
<proof>

```

```

theorem  $\neg$  {}  $\vdash$  Fls  $\implies$   $\neg$  {}  $\vdash$  neg (PfP (quot Fls))
<proof>

```