

# Robinson Arithmetic

Andrei Popescu      Dmitriy Traytel

September 13, 2023

## Abstract

We instantiate our syntax-independent logic infrastructure developed in a [separate AFP entry](#) to the FOL theory of Robinson arithmetic (also known as Q). The latter was formalised using Nominal Isabelle by adapting [Larry Paulson's formalization of the Hereditarily Finite Set theory](#).

## Contents

<b>1</b>	<b>Terms and Formulas</b>	<b>1</b>
1.1	The datatypes	1
1.2	Substitution	1
1.3	Semantics	3
1.4	Derived logical connectives	5
1.4.1	Conjunction	5
1.4.2	If and only if	5
1.4.3	False	6
<b>2</b>	<b>Axioms and Theorems</b>	<b>6</b>
2.1	Logical axioms	6
2.2	Concrete variables	6
2.3	Equality axioms	7
2.4	The Q (Robinson-arithmetic-specific) axioms	7
2.5	The proof system	7
2.6	Derived rules of inference	8
2.7	The deduction theorem	10
2.8	Cut rules	11
<b>3</b>	<b>Miscellaneous Logical Rules</b>	<b>12</b>
3.1	Quantifier reasoning	15
3.2	Congruence rules	16
<b>4</b>	<b>Equality Reasoning</b>	<b>16</b>
4.1	The congruence property for $(EQ)$ , and other basic properties of equality	16
4.2	The congruence properties for $suc$ , $pls$ and $tms$	18
4.3	Substitution for equalities	20
4.4	Congruence rules for predicates	20
4.5	The formula $fls$	21
<b>5</b>	<b>Instantiation of Syntax-Independent Logic Infrastructure</b>	<b>23</b>
5.1	Preliminaries	23
5.2	Instantiation of the generic syntax and deduction relation	25
5.3	Instantiation of the arithmetic-enriched generic syntax and deduction relation	27
5.4	Instantiation of the abstract notion of standard model and truth	28

# 1 Terms and Formulas

nat is a pure permutation type

**instance** *nat* :: pure by standard

**atom\_decl** name

**declare** *fresh\_set\_empty* [simp]

**lemma** *supp\_name* [simp]: **fixes** *i::name* **shows**  $\text{supp } i = \{\text{atom } i\}$   
**by** (rule *supp\_at\_base*)

## 1.1 The datatypes

**nominal\_datatype** *trm* = *zer* | *Var name* | *suc trm* | *pls trm trm* | *tms trm trm*

**nominal\_datatype** *fmla* =  
  *eql trm trm* (infixr EQ 150)  
  | *dsj fmla fmla* (infixr OR 130)  
  | *neg fmla*  
  | *exi x::name f::fmla binds x in f*

*eql* are atomic formulas; *dsj*, *neg*, *exi* are non-atomic

**declare** *trm.supp* [simp] *fmla.supp* [simp]

## 1.2 Substitution

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

**where**

*subst i x zer* = *zer*  
| *subst i x (Var k)* = (if  $i=k$  then *x* else *Var k*)  
| *subst i x (suc t)* = *suc (subst i x t)*  
| *subst i x (pls t u)* = *pls (subst i x t) (subst i x u)*  
| *subst i x (tms t u)* = *tms (subst i x t) (subst i x u)*

**by** (auto simp: *eqvt\_def subst\_graph\_aux\_def*) (*metis trm.strong\_exhaust*)

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

**by** *lexicographic\_order*

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

$j \# \text{subst } i \ x \ t \longleftrightarrow (\text{atom } i \# t \wedge j \# t) \vee (j \# x \wedge (j \# t \vee j = \text{atom } i))$

**by** (*induct t rule: trm.induct*) (*auto simp: fresh\_at\_base*)

**lemma** *forget\_subst\_trm* [simp]:  $\text{atom } a \# \text{trm} \implies \text{subst } a \ x \ \text{trm} = \text{trm}$

**by** (*induct trm rule: trm.induct*) (*simp\_all add: fresh\_at\_base*)

**lemma** *subst\_trm\_id* [simp]:  $\text{subst } a \ (\text{Var } a) \ \text{trm} = \text{trm}$

**by** (*induct trm rule: trm.induct*) *simp\_all*

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

$\text{atom } j \# \text{trm} \implies \text{subst } j \ u \ (\text{subst } i \ t \ \text{trm}) = \text{subst } i \ (\text{subst } j \ u \ t) \ \text{trm}$

**by** (*induct trm rule: trm.induct*) (*auto simp: fresh\_Pair*)

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

$\text{atom } j \# t \implies \text{atom } i \# u \implies i \neq j \implies \text{subst } j \ u \ (\text{subst } i \ t \ \text{trm}) = \text{subst } i \ t \ (\text{subst } j \ u \ \text{trm})$

**by** (*induct trm rule: trm.induct*) *auto*

**lemma** *repeat\_subst\_trm* [simp]:  $\text{subst } i \ u \ (\text{subst } i \ t \ \text{trm}) = \text{subst } i \ (\text{subst } i \ u \ t) \ \text{trm}$

by (induct trm rule: trm.induct) auto

**nominal\_function** subst\_fmula :: fmula  $\Rightarrow$  name  $\Rightarrow$  trm  $\Rightarrow$  fmula ( $\_'$ ( $\_::=\_'$ ) [1000, 0, 0] 200)

where

eql: (eql t u)( $i::=x$ ) = eql (subst i x t) (subst i x u)  
| dsj: (dsj A B)( $i::=x$ ) = dsj (A( $i::=x$ )) (B( $i::=x$ ))  
| neg: (neg A)( $i::=x$ ) = neg (A( $i::=x$ ))  
| exi: atom j  $\#$  (i, x)  $\Longrightarrow$  (exi j A)( $i::=x$ ) = exi j (A( $i::=x$ ))

subgoal by (simp add: eqvt\_def subst\_fmula\_graph\_aux\_def)

subgoal by auto

subgoal by (metis fmula.strong\_exhaust fresh\_star\_insert old.prod.exhaust)

subgoal by auto

subgoal by auto

subgoal by auto

subgoal by auto

subgoal by auto

subgoal by auto

subgoal by auto

subgoal by auto

subgoal by auto

subgoal

by (simp add: eqvt\_at\_def fresh\_star\_def fresh\_Pair fresh\_at\_base)  
(metis flip\_at\_base\_simps(3) flip\_fresh\_fresh) .

**nominal\_termination** (eqvt)

by lexicographic\_order

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

by (nominal\_induct A avoiding: i x rule: fmula.strong\_induct) auto

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

by (nominal\_induct A avoiding: a x rule: fmula.strong\_induct) (auto simp: fresh\_at\_base)

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

by (nominal\_induct A avoiding: a rule: fmula.strong\_induct) (auto simp: fresh\_at\_base)

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

$j \#$  (A( $i::=x$ ))  $\longleftrightarrow$  (atom i  $\#$  A  $\wedge$  j  $\#$  A)  $\vee$  (j  $\#$  x  $\wedge$  (j  $\#$  A  $\vee$  j = atom i))

by (nominal\_induct A avoiding: i x rule: fmula.strong\_induct) (auto simp: fresh\_at\_base)

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

atom j  $\#$  A  $\Longrightarrow$  (A( $i::=t$ ))(j $::=u$ ) = A(i  $::=$  subst j u t)

by (nominal\_induct A avoiding: i j t u rule: fmula.strong\_induct) (auto simp: fresh\_at\_base)

**lemma** repeat\_subst\_fmula [simp]: (A( $i::=t$ ))(i $::=u$ ) = A(i  $::=$  subst i u t)

by (nominal\_induct A avoiding: i t u rule: fmula.strong\_induct) auto

**lemma** subst\_fmula\_exi\_with\_renaming:

atom i'  $\#$  (A, i, j, t)  $\Longrightarrow$  (exi i A)(j  $::=$  t) = exi i' (((i  $\leftrightarrow$  i')  $\cdot$  A)(j  $::=$  t))

by (rule subst [of exi i' ((i  $\leftrightarrow$  i')  $\cdot$  A) exi i A])

(auto simp: Abs1\_eq\_iff flip\_def swap\_commute)

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

**lemma** flip\_subst\_trm: atom y  $\#$  t  $\Longrightarrow$  (x  $\leftrightarrow$  y)  $\cdot$  t = subst x (Var y) t

apply(nominal\_induct t avoiding: x y rule: trm.strong\_induct)

by auto

**lemma** *flip\_subst\_fm*l: *atom*  $y \# \varphi \implies (x \leftrightarrow y) \cdot \varphi = \varphi(x ::= \text{Var } y)$   
**apply**(*nominal\_induct*  $\varphi$  *avoiding*:  $x \ y$  *rule*: *fmla.strong\_induct*)  
**apply** (*auto simp*: *flip\_subst\_trm*)  
**using** *fresh\_at\_base(2)* **by** *blast*

**lemma** *exi\_ren\_subst\_fresh*: *atom*  $y \# \varphi \implies \text{exi } x \ \varphi = \text{exi } y \ (\varphi(x ::= \text{Var } y))$   
**using** *flip\_subst\_fm*l **by** *auto*

### 1.3 Semantics

**definition** *e0* :: (*name*, *nat*) *finfun* — the null environment  
**where**  $e0 \equiv \text{finfun\_const } 0$

**nominal\_function** *eval\_trm* :: (*name*, *nat*) *finfun*  $\Rightarrow$  *trm*  $\Rightarrow$  *nat*  
**where**  
 $\text{eval\_trm } e \ \text{zer} = 0$   
 $|\ \text{eval\_trm } e \ (\text{Var } k) = \text{finfun\_apply } e \ k$   
 $|\ \text{eval\_trm } e \ (\text{suc } t) = \text{Suc } (\text{eval\_trm } e \ t)$   
 $|\ \text{eval\_trm } e \ (\text{pls } t \ u) = \text{eval\_trm } e \ t + \text{eval\_trm } e \ u$   
 $|\ \text{eval\_trm } e \ (\text{tms } t \ u) = \text{eval\_trm } e \ t * \text{eval\_trm } e \ u$   
**by** (*auto simp*: *eqvt\_def eval\_trm\_graph\_aux\_def*) (*metis trm.strong\_exhaust*)

**nominal\_termination** (*eqvt*)  
**by** *lexicographic\_order*

**nominal\_function** *eval\_fm*l :: (*name*, *nat*) *finfun*  $\Rightarrow$  *fmla*  $\Rightarrow$  *bool*  
**where**  
 $\text{eval\_fm}l \ e \ (t \ \text{EQ } u) \longleftrightarrow \text{eval\_trm } e \ t = \text{eval\_trm } e \ u$   
 $|\ \text{eval\_fm}l \ e \ (A \ \text{OR } B) \longleftrightarrow \text{eval\_fm}l \ e \ A \ \vee \ \text{eval\_fm}l \ e \ B$   
 $|\ \text{eval\_fm}l \ e \ (\text{neg } A) \longleftrightarrow (\sim \text{eval\_fm}l \ e \ A)$   
 $|\ \text{atom } k \ \# \ e \implies \text{eval\_fm}l \ e \ (\text{exi } k \ A) \longleftrightarrow (\exists x. \text{eval\_fm}l \ (\text{finfun\_update } e \ k \ x) \ A)$   
**supply** [*simp* *del*: *defined\_all*]  
**apply**(*simp add*: *eqvt\_def eval\_fm\_l\_graph\_aux\_def*)  
**apply**(*auto del*: *iffI*) [*I1*]  
**apply**(*rule\_tac*  $y=b$  **and**  $c=(a)$  **in** *fmla.strong\_exhaust*)  
**apply**(*auto simp*: *fresh\_star\_def*)[*4*]  
**using** [*simp* *del*: *alpha\_lst*] **apply** *clarsimp*  
**apply**(*erule\_tac*  $c=(ea)$  **in** *Abs\_lst1\_fcb2'*)  
**apply**(*rule pure\_fresh*)  
**apply**(*simp add*: *fresh\_star\_def*)  
**apply** (*simp\_all add*: *eqvt\_at\_def*)  
**apply** (*simp\_all add*: *perm\_supp\_eq*)  
**done**

**nominal\_termination** (*eqvt*)  
**by** *lexicographic\_order*

**lemma** *eval\_trm\_rename*:  
**assumes** *atom*  $k' \ \# \ t$   
**shows**  $\text{eval\_trm } (\text{finfun\_update } e \ k \ x) \ t =$   
 $\text{eval\_trm } (\text{finfun\_update } e \ k' \ x) \ ((k' \leftrightarrow k) \cdot t)$   
**using** *assms*  
**by** (*induct*  $t$  *rule*: *trm.induct*) (*auto simp*: *permute\_flip\_at*)

**lemma** *eval\_fm\_l\_rename*:  
**assumes** *atom*  $k' \ \# \ A$   
**shows**  $\text{eval\_fm}l \ (\text{finfun\_update } e \ k \ x) \ A = \text{eval\_fm}l \ (\text{finfun\_update } e \ k' \ x) \ ((k' \leftrightarrow k) \cdot A)$   
**using** *assms*

**apply** (*nominal\_induct*  $A$  *avoiding*:  $e$   $k$   $k'$   $x$  *rule*:  $fmla.strong\_induct$ )  
**apply** (*simp\_all* *add*:  $eval\_trm\_rename[symmetric]$ , *metis*)  
**apply** (*simp* *add*:  $fresh\_finfun\_update$   $fresh\_at\_base$   $finfun\_update\_twist$ )  
**done**

**lemma** *better\_ex\_eval\_fmula*[*simp*]:  
 $eval\_fmula\ e\ (exi\ k\ A) \longleftrightarrow (\exists x. eval\_fmula\ (finfun\_update\ e\ k\ x)\ A)$   
**proof** –  
**obtain**  $k'::name$  **where**  $k'$ : *atom*  $k' \# (k, e, A)$   
**by** (*rule* *obtain\_fresh*)  
**then have**  $eq: exi\ k'\ ((k' \leftrightarrow k) \cdot A) = exi\ k\ A$   
**by** (*simp* *add*:  $Abs1\_eq\_iff\_flip\_def$ )  
**have**  $eval\_fmula\ e\ (exi\ k'\ ((k' \leftrightarrow k) \cdot A)) = (\exists x. eval\_fmula\ (finfun\_update\ e\ k'\ x)\ ((k' \leftrightarrow k) \cdot A))$   
**using**  $k'$  **by** *simp*  
**also have**  $\dots = (\exists x. eval\_fmula\ (finfun\_update\ e\ k\ x)\ A)$   
**by** (*metis*  $eval\_fmula\_rename\ k'\ fresh\_Pair$ )  
**finally show** *?thesis*  
**by** (*metis* *eq*)  
**qed**

**lemma** *forget\_eval\_trm* [*simp*]: *atom*  $i \# t \implies$   
 $eval\_trm\ (finfun\_update\ e\ i\ x)\ t = eval\_trm\ e\ t$   
**by** (*induct*  $t$  *rule*:  $trm.induct$ ) (*simp\_all* *add*:  $fresh\_at\_base$ )

**lemma** *forget\_eval\_fmula* [*simp*]:  
*atom*  $k \# A \implies eval\_fmula\ (finfun\_update\ e\ k\ x)\ A = eval\_fmula\ e\ A$   
**by** (*nominal\_induct*  $A$  *avoiding*:  $k$   $e$  *rule*:  $fmla.strong\_induct$ )  
(*simp\_all* *add*:  $fresh\_at\_base$   $finfun\_update\_twist$ )

**lemma** *eval\_subst\_trm*:  $eval\_trm\ e\ (subst\ i\ t\ u) =$   
 $eval\_trm\ (finfun\_update\ e\ i\ (eval\_trm\ e\ t))\ u$   
**by** (*induct*  $u$  *rule*:  $trm.induct$ ) (*auto*)

**lemma** *eval\_subst\_fmula*:  $eval\_fmula\ e\ (fmula(i::= t)) =$   
 $eval\_fmula\ (finfun\_update\ e\ i\ (eval\_trm\ e\ t))\ fmula$   
**by** (*nominal\_induct*  $fmula$  *avoiding*:  $i\ t\ e$  *rule*:  $fmla.strong\_induct$ )  
(*simp\_all* *add*:  $eval\_subst\_trm\ finfun\_update\_twist\ fresh\_at\_base$ )

## 1.4 Derived logical connectives

**abbreviation**  $imp :: fmla \Rightarrow fmla \Rightarrow fmla$  (**infixr** *IMP* 125)  
**where**  $imp\ A\ B \equiv dsj\ (neg\ A)\ B$

**abbreviation**  $all :: name \Rightarrow fmla \Rightarrow fmla$   
**where**  $all\ i\ A \equiv neg\ (exi\ i\ (neg\ A))$

### 1.4.1 Conjunction

**definition**  $cnj :: fmla \Rightarrow fmla \Rightarrow fmla$  (**infixr** *AND* 135)  
**where**  $cnj\ A\ B \equiv neg\ (dsj\ (neg\ A)\ (neg\ B))$

**lemma** *cnj\_eqvt* [*eqvt*]:  $p \cdot (A\ AND\ B) = (p \cdot A)\ AND\ (p \cdot B)$   
**by** (*simp* *add*: *cnj\_def*)

**lemma** *fresh\_cnj* [*simp*]:  $a \# A\ AND\ B \longleftrightarrow (a \# A \wedge a \# B)$   
**by** (*auto* *simp*: *cnj\_def*)

**lemma** *supp\_cnj* [*simp*]:  $supp\ (A\ AND\ B) = supp\ A \cup supp\ B$   
**by** (*auto* *simp*: *cnj\_def*)

**lemma** *size\_cnj* [*simp*]:  $\text{size } (A \text{ AND } B) = \text{size } A + \text{size } B + 4$   
**by** (*simp add: cnj\_def*)

**lemma** *cnj\_injective\_iff* [*iff*]:  $(A \text{ AND } B) = (A' \text{ AND } B') \longleftrightarrow (A = A' \wedge B = B')$   
**by** (*auto simp: cnj\_def*)

**lemma** *subst\_fmula\_cnj* [*simp*]:  $(A \text{ AND } B)(i::=x) = (A(i::=x)) \text{ AND } (B(i::=x))$   
**by** (*auto simp: cnj\_def*)

**lemma** *eval\_fmula\_cnj* [*simp*]:  $\text{eval\_fmula } e \text{ (cnj } A \ B) \longleftrightarrow (\text{eval\_fmula } e \ A \wedge \text{eval\_fmula } e \ B)$   
**by** (*auto simp: cnj\_def*)

### 1.4.2 If and only if

**definition** *Iff* ::  $\text{fmula} \Rightarrow \text{fmula} \Rightarrow \text{fmula}$  (**infixr** *IFF* 125)  
**where**  $\text{Iff } A \ B = \text{cnj } (\text{imp } A \ B) \ (\text{imp } B \ A)$

**lemma** *Iff\_eqvt* [*eqvt*]:  $p \cdot (A \text{ IFF } B) = (p \cdot A) \text{ IFF } (p \cdot B)$   
**by** (*simp add: Iff\_def*)

**lemma** *fresh\_Iff* [*simp*]:  $a \# A \text{ IFF } B \longleftrightarrow (a \# A \wedge a \# B)$   
**by** (*auto simp: cnj\_def Iff\_def*)

**lemma** *size\_Iff* [*simp*]:  $\text{size } (A \text{ IFF } B) = 2 * (\text{size } A + \text{size } B) + 8$   
**by** (*simp add: Iff\_def*)

**lemma** *Iff\_injective\_iff* [*iff*]:  $(A \text{ IFF } B) = (A' \text{ IFF } B') \longleftrightarrow (A = A' \wedge B = B')$   
**by** (*auto simp: Iff\_def*)

**lemma** *subst\_fmula\_Iff* [*simp*]:  $(A \text{ IFF } B)(i::=x) = (A(i::=x)) \text{ IFF } (B(i::=x))$   
**by** (*auto simp: Iff\_def*)

**lemma** *eval\_fmula\_Iff* [*simp*]:  $\text{eval\_fmula } e \text{ (Iff } A \ B) \longleftrightarrow (\text{eval\_fmula } e \ A \longleftrightarrow \text{eval\_fmula } e \ B)$   
**by** (*auto simp: Iff\_def*)

### 1.4.3 False

**definition** *fls* **where**  $\text{fls} \equiv \text{neg } (\text{zer } \text{EQ } \text{zer})$

**lemma** *fls\_eqvt* [*eqvt*]:  $(p \cdot \text{fls}) = \text{fls}$   
**by** (*simp add: fls\_def*)

**lemma** *fls\_fresh* [*simp*]:  $a \# \text{fls}$   
**by** (*simp add: fls\_def*)

## 2 Axioms and Theorems

### 2.1 Logical axioms

**inductive\_set** *boolean\_axioms* :: *fmula set*

**where**

- Ident*:  $A \text{ IMP } A \in \text{boolean\_axioms}$
- dsjI1*:  $A \text{ IMP } (A \text{ OR } B) \in \text{boolean\_axioms}$
- dsjCont*:  $(A \text{ OR } A) \text{ IMP } A \in \text{boolean\_axioms}$
- dsjAssoc*:  $(A \text{ OR } (B \text{ OR } C)) \text{ IMP } ((A \text{ OR } B) \text{ OR } C) \in \text{boolean\_axioms}$
- dsjcnj*:  $(C \text{ OR } A) \text{ IMP } ((\text{neg } C) \text{ OR } B) \text{ IMP } (A \text{ OR } B) \in \text{boolean\_axioms}$

**lemma** *boolean\_axioms\_hold*:  $A \in \text{boolean\_axioms} \implies \text{eval\_fmla } e \ A$   
**by** (*induct rule*: *boolean\_axioms.induct*, *auto*)

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

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

**lemma** *special\_axioms\_hold*:  $A \in \text{special\_axioms} \implies \text{eval\_fmla } e \ A$   
**by** (*induct rule*: *special\_axioms.induct*, *auto*) (*metis eval\_subst\_fmla*)

**lemma** *twist\_forget\_eval\_fmla* [*simp*]:

*atom j*  $\# (i, A)$

$\implies \text{eval\_fmla } (\text{finfun\_update } (\text{finfun\_update } (\text{finfun\_update } e \ i \ x) \ j \ y) \ i \ z) \ A =$   
 $\text{eval\_fmla } (\text{finfun\_update } e \ i \ z) \ A$

**by** (*metis finfun\_update\_twice finfun\_update\_twist forget\_eval\_fmla fresh\_Pair*)

## 2.2 Concrete variables

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

**abbreviation**

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

**abbreviation**

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

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

**abbreviation**

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

**abbreviation**

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

**abbreviation**

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

## 2.3 Equality axioms

**definition** *refl\_ax* :: *fmla* **where**

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

**lemma** *refl\_ax\_holds*:  $\text{eval\_fmla } e \ \text{refl\_ax}$

**by** (*auto simp*: *refl\_ax\_def*)

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

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

**lemma** *eq\_cong\_ax\_holds*:  $\text{eval\_fmla } e \ \text{eq\_cong\_ax}$

**by** (*auto simp*: *cnj\_def eq\_cong\_ax\_def*)

**definition** *syc\_cong\_ax* :: *fmla* **where**

$\text{syc\_cong\_ax} = ((\text{Var } X1 \ \text{EQ} \ \text{Var } X2)) \ \text{IMP}$   
 $((\text{suc } (\text{Var } X1)) \ \text{EQ} \ (\text{suc } (\text{Var } X2)))$

**lemma** *syc\_cong\_ax\_holds*:  $\text{eval\_fmla } e \ \text{syc\_cong\_ax}$

**by** (*auto simp*: *cnj\_def syc\_cong\_ax\_def*)

**definition** *pls\_cong\_ax* :: *fmla* **where**

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

$((pls (Var X1) (Var X3)) EQ (pls (Var X2) (Var X4)))$

**lemma** *pls\_cong\_ax\_holds*: *eval\_fm1a e pls\_cong\_ax*  
**by** (*auto simp: cnj\_def pls\_cong\_ax\_def*)

**definition** *tms\_cong\_ax* :: *fm1a* **where**  
*tms\_cong\_ax* =  $((Var X1 EQ Var X2) AND (Var X3 EQ Var X4)) IMP$   
 $((tms (Var X1) (Var X3)) EQ (tms (Var X2) (Var X4)))$

**lemma** *tms\_cong\_ax\_holds*: *eval\_fm1a e tms\_cong\_ax*  
**by** (*auto simp: cnj\_def tms\_cong\_ax\_def*)

**definition** *equality\_axioms* :: *fm1a set* **where**  
*equality\_axioms* = {*refl\_ax*, *eq\_cong\_ax*, *syc\_cong\_ax*, *pls\_cong\_ax*, *tms\_cong\_ax*}

**lemma** *equality\_axioms\_hold*:  $A \in \text{equality\_axioms} \implies \text{eval\_fm1a } e \ A$   
**by** (*auto simp: equality\_axioms\_def refl\_ax\_holds eq\_cong\_ax\_holds*  
*syc\_cong\_ax\_holds pls\_cong\_ax\_holds tms\_cong\_ax\_holds*)

## 2.4 The Q (Robinson-arithmetic-specific) axioms

**definition** *Q\_axioms*  $\equiv$   
 $\{A \mid A \ X1 \ X2.$   
 $X1 \neq X2 \wedge$   
 $(A = \text{neg } (\text{zer } EQ \ \text{suc } (Var X1))) \vee$   
 $A = \text{suc } (Var X1) \ EQ \ \text{suc } (Var X2) \ IMP \ Var X1 \ EQ \ Var X2 \vee$   
 $A = Var X2 \ EQ \ \text{zer} \ OR \ \text{exi } X1 \ (Var X2 \ EQ \ \text{suc } (Var X1)) \vee$   
 $A = \text{pls } (Var X1) \ \text{zer} \ EQ \ Var X1 \vee$   
 $A = \text{pls } (Var X1) \ (\text{suc } (Var X2)) \ EQ \ \text{suc } (\text{pls } (Var X1) \ (Var X2)) \vee$   
 $A = \text{tms } (Var X1) \ \text{zer} \ EQ \ \text{zer} \vee$   
 $A = \text{tms } (Var X1) \ (\text{suc } (Var X2)) \ EQ \ \text{pls } (\text{tms } (Var X1) \ (Var X2)) \ (Var X1)\}$

## 2.5 The proof system

**inductive** *nprv* :: *fm1a set*  $\Rightarrow$  *fm1a*  $\Rightarrow$  *bool* (**infixl**  $\vdash$  55)  
**where**  
*Hyp*:  $A \in H \implies H \vdash A$   
| *Q*:  $A \in Q\_axioms \implies H \vdash A$   
| *Bool*:  $A \in \text{boolean\_axioms} \implies H \vdash A$   
| *eq1*:  $A \in \text{equality\_axioms} \implies H \vdash A$   
| *Spec*:  $A \in \text{special\_axioms} \implies H \vdash A$   
| *MP*:  $H \vdash A \ IMP \ B \implies H' \vdash A \implies H \cup H' \vdash B$   
| *exists*:  $H \vdash A \ IMP \ B \implies \text{atom } i \ \# \ B \implies \forall C \in H. \text{atom } i \ \# \ C \implies H \vdash (\text{exi } i \ A) \ IMP \ B$

## 2.6 Derived rules of inference

**lemma** *contraction*:  $\text{insert } A \ (\text{insert } A \ H) \vdash B \implies \text{insert } A \ H \vdash B$   
**by** (*metis insert\_absorb2*)

**lemma** *thin\_Un*:  $H \vdash A \implies H \cup H' \vdash A$   
**by** (*metis Bool MP boolean\_axioms.Ident sup\_commute*)

**lemma** *thin*:  $H \vdash A \implies H \subseteq H' \implies H' \vdash A$   
**by** (*metis Un\_absorb1 thin\_Un*)

**lemma** *thin0*:  $\{\} \vdash A \implies H \vdash A$   
**by** (*metis sup\_bot\_left thin\_Un*)

**lemma** *thin1*:  $H \vdash B \implies \text{insert } A \ H \vdash B$

by (*metis subset\_insertI thin*)

**lemma thin2:**  $insert\ A1\ H \vdash B \implies insert\ A1\ (insert\ A2\ H) \vdash B$   
by (*blast intro: thin*)

**lemma thin3:**  $insert\ A1\ (insert\ A2\ H) \vdash B \implies insert\ A1\ (insert\ A2\ (insert\ A3\ H)) \vdash B$   
by (*blast intro: thin*)

**lemma thin4:**  
 $insert\ A1\ (insert\ A2\ (insert\ A3\ H)) \vdash B$   
 $\implies insert\ A1\ (insert\ A2\ (insert\ A3\ (insert\ A4\ H))) \vdash B$   
by (*blast intro: thin*)

**lemma rotate2:**  $insert\ A2\ (insert\ A1\ H) \vdash B \implies insert\ A1\ (insert\ A2\ H) \vdash B$   
by (*blast intro: thin*)

**lemma rotate3:**  $insert\ A3\ (insert\ A1\ (insert\ A2\ H)) \vdash B \implies insert\ A1\ (insert\ A2\ (insert\ A3\ H)) \vdash B$   
by (*blast intro: thin*)

**lemma rotate4:**  
 $insert\ A4\ (insert\ A1\ (insert\ A2\ (insert\ A3\ H))) \vdash B$   
 $\implies insert\ A1\ (insert\ A2\ (insert\ A3\ (insert\ A4\ H))) \vdash B$   
by (*blast intro: thin*)

**lemma rotate5:**  
 $insert\ A5\ (insert\ A1\ (insert\ A2\ (insert\ A3\ (insert\ A4\ H)))) \vdash B$   
 $\implies insert\ A1\ (insert\ A2\ (insert\ A3\ (insert\ A4\ (insert\ A5\ H)))) \vdash B$   
by (*blast intro: thin*)

**lemma rotate6:**  
 $insert\ A6\ (insert\ A1\ (insert\ A2\ (insert\ A3\ (insert\ A4\ (insert\ A5\ H)))))) \vdash B$   
 $\implies insert\ A1\ (insert\ A2\ (insert\ A3\ (insert\ A4\ (insert\ A5\ (insert\ A6\ H)))))) \vdash B$   
by (*blast intro: thin*)

**lemma rotate7:**  
 $insert\ A7\ (insert\ A1\ (insert\ A2\ (insert\ A3\ (insert\ A4\ (insert\ A5\ (insert\ A6\ H)))))) \vdash B$   
 $\implies insert\ A1\ (insert\ A2\ (insert\ A3\ (insert\ A4\ (insert\ A5\ (insert\ A6\ (insert\ A7\ H)))))) \vdash B$   
by (*blast intro: thin*)

**lemma rotate8:**  
 $insert\ A8\ (insert\ A1\ (insert\ A2\ (insert\ A3\ (insert\ A4\ (insert\ A5\ (insert\ A6\ (insert\ A7\ H)))))) \vdash B$   
 $\implies insert\ A1\ (insert\ A2\ (insert\ A3\ (insert\ A4\ (insert\ A5\ (insert\ A6\ (insert\ A7\ (insert\ A8\ H)))))) \vdash B$   
by (*blast intro: thin*)

**lemma rotate9:**  
 $insert\ A9\ (insert\ A1\ (insert\ A2\ (insert\ A3\ (insert\ A4\ (insert\ A5\ (insert\ A6\ (insert\ A7\ (insert\ A8\ H)))))) \vdash B$   
 $\implies insert\ A1\ (insert\ A2\ (insert\ A3\ (insert\ A4\ (insert\ A5\ (insert\ A6\ (insert\ A7\ (insert\ A8\ (insert\ A9\ H)))))) \vdash B$   
by (*blast intro: thin*)

**lemma rotate10:**  
 $insert\ A10\ (insert\ A1\ (insert\ A2\ (insert\ A3\ (insert\ A4\ (insert\ A5\ (insert\ A6\ (insert\ A7\ (insert\ A8\ (insert\ A9\ H)))))) \vdash B$   
 $\implies insert\ A1\ (insert\ A2\ (insert\ A3\ (insert\ A4\ (insert\ A5\ (insert\ A6\ (insert\ A7\ (insert\ A8\ (insert\ A9\ (insert\ A10\ H)))))) \vdash B$   
by (*blast intro: thin*)

**lemma rotate11:**

*insert A11 (insert A1 (insert A2 (insert A3 (insert A4 (insert A5 (insert A6 (insert A7 (insert A8 (insert A9 (insert A10 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 H))))))))))*  $\vdash B$   
**by** (*blast intro: thin*)

**lemma rotate12:**

*insert A12 (insert A1 (insert A2 (insert A3 (insert A4 (insert A5 (insert A6 (insert A7 (insert A8 (insert A9 (insert A10 (insert A11 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 H))))))))))*  $\vdash B$   
**by** (*blast intro: thin*)

**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))))))))))*  $\vdash B$   
**by** (*blast intro: thin*)

**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))))))))))*  $\vdash B$   
**by** (*blast intro: thin*)

**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))))))))))*  $\vdash B$   
**by** (*blast intro: thin*)

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

**by** (*metis MP Un\_absorb*)

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

**by** (*metis MP\_same le\_sup\_iff thin*)

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

**by** (*metis MP\_same thin0*)

**lemma dsj\_commute:**  $H \vdash B \text{ OR } A \implies H \vdash A \text{ OR } B$

**using** *dsjcnj [of B A B] Ident [of B]*

**by** (*metis Bool MP\_same*)

**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** –

**have**  $H' \cup H \vdash (\text{neg } A) \text{ OR } (C \text{ OR } (\text{neg } A))$

**by** (*metis Bool MP MP\_same boolean\_axioms.dsjcnj dsj\_commute dsjAssoc assms*)

**thus** *?thesis*

**by** (*metis Bool dsj\_commute Un\_commute MP\_same dsjAssoc dsjCont dsjI1*)

**qed**

**lemma Assume:** *insert A H*  $\vdash A$

```

by (metis Hyp insertI1)

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:  $H \vdash B \implies H \vdash A \text{ IMP } B$ 
by (metis Bool dsj_commute MP_same boolean_axioms.dsjI1)

lemma dsjAssoc1:  $H \vdash A \text{ OR } (B \text{ OR } C) \implies H \vdash (A \text{ OR } B) \text{ OR } C$ 
by (metis Bool MP_same boolean_axioms.dsjAssoc)

lemma dsjAssoc2:  $H \vdash (A \text{ OR } B) \text{ OR } C \implies H \vdash A \text{ OR } (B \text{ OR } C)$ 
by (metis dsjAssoc1 dsj_commute)

lemma dsj_commute_imp:  $H \vdash (B \text{ OR } A) \text{ IMP } (A \text{ OR } B)$ 
using dsjcnj [of B A B] Ident [of B]
by (metis Bool dsjAssoc2 dsj_commute MP_same)

lemma dsj_Semicong_1:  $H \vdash A \text{ OR } C \implies H \vdash A \text{ IMP } B \implies H \vdash B \text{ OR } C$ 
using dsjcnj [of A C B]
by (metis Bool dsj_commute MP_same)

lemma imp_imp_commute:  $H \vdash B \text{ IMP } (A \text{ IMP } C) \implies H \vdash A \text{ IMP } (B \text{ IMP } C)$ 
by (metis dsjAssoc1 dsjAssoc2 dsj_Semicong_1 dsj_commute_imp)

```

## 2.7 The deduction theorem

```

lemma deduction_Diff: assumes  $H \vdash B$  shows  $H - \{C\} \vdash C \text{ IMP } B$ 
using assms
proof (induct)
  case (Hyp A H) thus ?case
  by (metis Bool imp_triv_I boolean_axioms.Ident nprv.Hyp member_remove remove_def)
next
  case (Q H) thus ?case
  by (metis imp_triv_I nprv.Q)
next
  case (Bool A H) thus ?case
  by (metis imp_triv_I nprv.Bool)
next
  case (eql A H) thus ?case
  by (metis imp_triv_I nprv.eql)
next
  case (Spec A H) thus ?case
  by (metis imp_triv_I nprv.Spec)
next
  case (MP H A B H')
  hence  $(H - \{C\}) \cup (H' - \{C\}) \vdash \text{imp } C B$ 
  by (simp add: S)
  thus ?case
  by (metis Un_Diff)
next
  case (exists H A B i) show ?case
proof (cases C  $\in$  H)
  case True

```

```

hence atom i # C using exists by auto
moreover have H - {C} ⊢ A IMP C IMP B using exists
  by (metis imp_imp_commute)
ultimately have H - {C} ⊢ (exi i A) IMP C IMP B using exists
  using nprv.eq1
  by (simp add: nprv.exists)
thus ?thesis
  by (metis imp_imp_commute)
next
case False
hence H - {C} = H by auto
thus ?thesis using exists
  by (metis imp_triv_I nprv.exists)
qed
qed

```

```

theorem imp_I [intro!]: insert A H ⊢ B ⇒ H ⊢ A IMP B
  by (metis Diff_insert_absorb imp_triv_I deduction_Diff insert_absorb)

```

```

lemma anti_deduction: H ⊢ A IMP B ⇒ insert A H ⊢ B
  by (metis Assume MP_same thin1)

```

## 2.8 Cut rules

```

lemma cut: H ⊢ A ⇒ insert A H' ⊢ B ⇒ H ∪ H' ⊢ B
  by (metis MP Un_commute imp_I)

```

```

lemma cut_same: H ⊢ A ⇒ insert A H ⊢ B ⇒ H ⊢ B
  by (metis Un_absorb cut)

```

```

lemma cut_thin: HA ⊢ A ⇒ insert A HB ⊢ B ⇒ HA ∪ HB ⊆ H ⇒ H ⊢ B
  by (metis thin cut)

```

```

lemma cut0: {} ⊢ A ⇒ insert A H ⊢ B ⇒ H ⊢ B
  by (metis cut_same thin0)

```

```

lemma cut1: {A} ⊢ B ⇒ H ⊢ A ⇒ H ⊢ B
  by (metis cut sup_bot_right)

```

```

lemma rcut1: {A} ⊢ B ⇒ insert B H ⊢ C ⇒ insert A H ⊢ C
  by (metis Assume cut1 cut_same rotate2 thin1)

```

```

lemma cut2: [[A,B] ⊢ C; H ⊢ A; H ⊢ B] ⇒ H ⊢ C
  by (metis Un_empty_right Un_insert_right cut cut_same)

```

```

lemma rcut2: {A,B} ⊢ C ⇒ insert C H ⊢ D ⇒ H ⊢ B ⇒ insert A H ⊢ D
  by (metis Assume cut2 cut_same insert_commute thin1)

```

```

lemma cut3: [[A,B,C] ⊢ D; H ⊢ A; H ⊢ B; H ⊢ C] ⇒ H ⊢ D
  by (metis MP_same cut2 imp_I)

```

```

lemma cut4: [[A,B,C,D] ⊢ E; H ⊢ A; H ⊢ B; H ⊢ C; H ⊢ D] ⇒ H ⊢ E
  by (metis MP_same cut3 [of B C D] imp_I)

```

## 3 Miscellaneous Logical Rules

```

lemma dsj_I1: H ⊢ A ⇒ H ⊢ A OR B
  by (metis Bool MP_same boolean_axioms.dsjI1)

```

**lemma dsj\_I2:**  $H \vdash B \implies H \vdash A \text{ OR } B$   
**by** (*metis dsj\_commute dsj\_I1*)

**lemma Peirce:**  $H \vdash (\text{neg } A) \text{ IMP } A \implies H \vdash A$   
**using** *dsjcnj [of neg A A A] dsjCont [of A]*  
**by** (*metis Bool MP\_same boolean\_axioms.Ident*)

**lemma Contra:** *insert (neg A) H*  $\vdash A \implies H \vdash A$   
**by** (*metis Peirce imp\_I*)

**lemma imp\_neg\_I:**  $H \vdash A \text{ IMP } B \implies H \vdash A \text{ IMP } (\text{neg } B) \implies H \vdash \text{neg } A$   
**by** (*metis dsjcnj [of B neg A neg A] dsjCont Bool dsj\_commute MP\_same*)

**lemma negneg\_I:**  $H \vdash A \implies H \vdash \text{neg } (\text{neg } A)$   
**using** *dsjcnj [of neg (neg A) neg A neg (neg A)]*  
**by** (*metis Bool Ident MP\_same*)

**lemma negneg\_D:**  $H \vdash \text{neg } (\text{neg } A) \implies H \vdash A$   
**by** (*metis dsj\_I1 Peirce*)

**lemma neg\_D:**  $H \vdash \text{neg } A \implies H \vdash A \implies H \vdash B$   
**by** (*metis imp\_neg\_I imp\_triv\_I negneg\_D*)

**lemma dsj\_neg\_1:**  $H \vdash A \text{ OR } B \implies H \vdash \text{neg } B \implies H \vdash A$   
**by** (*metis dsj\_I1 dsj\_Semicong\_1 dsj\_commute Peirce*)

**lemma dsj\_neg\_2:**  $H \vdash A \text{ OR } B \implies H \vdash \text{neg } A \implies H \vdash B$   
**by** (*metis dsj\_neg\_1 dsj\_commute*)

**lemma neg\_dsj\_I:**  $H \vdash \text{neg } A \implies H \vdash \text{neg } B \implies H \vdash \text{neg } (A \text{ OR } B)$   
**by** (*metis Bool dsj\_neg\_1 MP\_same boolean\_axioms.Ident dsjAssoc*)

**lemma cnj\_I [intro!]:**  $H \vdash A \implies H \vdash B \implies H \vdash A \text{ AND } B$   
**by** (*metis cnj\_def negneg\_I neg\_dsj\_I*)

**lemma cnj\_E1:**  $H \vdash A \text{ AND } B \implies H \vdash A$   
**by** (*metis cnj\_def Bool dsj\_neg\_1 negneg\_D boolean\_axioms.dsjI1*)

**lemma cnj\_E2:**  $H \vdash A \text{ AND } B \implies H \vdash B$   
**by** (*metis cnj\_def Bool dsj\_I2 dsj\_neg\_2 MP\_same dsjAssoc Ident*)

**lemma cnj\_commute:**  $H \vdash B \text{ AND } A \implies H \vdash A \text{ AND } B$   
**by** (*metis cnj\_E1 cnj\_E2 cnj\_I*)

**lemma cnj\_E:** **assumes** *insert A (insert B H)  $\vdash C$  shows insert (A AND B) H  $\vdash C$*   
**apply** (*rule cut\_same [where A=A], metis cnj\_E1 Hyp insertI1*)  
**by** (*metis (full\_types) AssumeH(2) cnj\_E2 assms cut\_same [where A=B] insert\_commute thin2*)

**lemmas** *cnj\_EH = cnj\_E cnj\_E [THEN rotate2] cnj\_E [THEN rotate3] cnj\_E [THEN rotate4] cnj\_E [THEN rotate5]*  
*cnj\_E [THEN rotate6] cnj\_E [THEN rotate7] cnj\_E [THEN rotate8] cnj\_E [THEN rotate9] cnj\_E [THEN rotate10]*  
**declare** *cnj\_EH [intro!]*

**lemma neg\_I0:** **assumes**  $(\bigwedge B. \text{atom } i \nmid B \implies \text{insert } A \text{ H} \vdash B)$  **shows**  $H \vdash \text{neg } A$   
**by** (*meson fls\_fresh imp\_I imp\_neg\_I assms fmla.fresh(3)*)

**lemma** *neg\_mono*:  $insert\ A\ H \vdash B \implies insert\ (neg\ B)\ H \vdash neg\ A$   
**by** (*rule neg\_I0*) (*metis Hyp neg\_D insert\_commute insertI1 thin1*)

**lemma** *cnj\_mono*:  $insert\ A\ H \vdash B \implies insert\ C\ H \vdash D \implies insert\ (A\ AND\ C)\ H \vdash B\ AND\ D$   
**by** (*metis cnj\_E1 cnj\_E2 cnj\_I Hyp Un\_absorb2 cut insertI1 subset\_insertI*)

**lemma** *dsj\_mono*:

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

**proof** –

{ **fix**  $A\ B\ C\ H$   
**have**  $insert\ (A\ OR\ C)\ H \vdash (A\ IMP\ B)\ IMP\ C\ OR\ B$   
**by** (*metis Bool Hyp MP\_same boolean\_axioms.dsjcnj insertI1*)  
**hence**  $insert\ A\ H \vdash B \implies insert\ (A\ OR\ C)\ H \vdash C\ OR\ B$   
**by** (*metis MP\_same Un\_absorb Un\_insert\_right imp\_I thin\_Un*)

}

**thus** *?thesis*

**by** (*metis cut\_same assms thin2*)

**qed**

**lemma** *dsj\_E*:

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

**by** (*metis A B dsj\_mono negneg\_I Peirce*)

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

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

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

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

**by** (*metis Contra neg\_mono*)

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

**by** (*metis negneg\_D neg\_mono*)

**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** –

**have**  $insert\ (A\ IMP\ B)\ H \vdash B$

**by** (*metis Hyp A thin1 MP\_same insertI1*)

**thus** *?thesis*

**by** (*metis cut [where B=C] Un\_insert\_right sup\_commute sup\_idem B*)

**qed**

**lemma** *imp\_cut*:

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

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

**by** (*metis Contra dsj\_I1 neg\_mono assms rcut1*)

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

by (metis Iff\_def cnj\_I imp\_I)

**lemma** Iff\_MP\_same:  $H \vdash A \text{ IFF } B \implies H \vdash A \implies H \vdash B$   
by (metis Iff\_def cnj\_E1 MP\_same)

**lemma** Iff\_MP2\_same:  $H \vdash A \text{ IFF } B \implies H \vdash B \implies H \vdash A$   
by (metis Iff\_def cnj\_E2 MP\_same)

**lemma** Iff\_refl [intro!]:  $H \vdash A \text{ IFF } A$   
by (metis Hyp Iff\_I insertI1)

**lemma** Iff\_sym:  $H \vdash A \text{ IFF } B \implies H \vdash B \text{ IFF } A$   
by (metis Iff\_def cnj\_commute)

**lemma** Iff\_trans:  $H \vdash A \text{ IFF } B \implies H \vdash B \text{ IFF } C \implies H \vdash A \text{ IFF } C$   
**unfolding** Iff\_def  
by (metis cnj\_E1 cnj\_E2 cnj\_I dsj\_Semicong\_1 dsj\_commute)

**lemma** Iff\_E:  
 $\text{insert } A (\text{insert } B H) \vdash C \implies \text{insert } (\text{neg } A) (\text{insert } (\text{neg } B) H) \vdash C \implies \text{insert } (A \text{ IFF } B) H \vdash C$   
by (simp add: Assume Iff\_def anti\_deduction cnj\_E dsj\_EH(2) dsj\_I1 insert\_commute)

**lemma** Iff\_E1:  
**assumes**  $A: H \vdash A$  **and**  $B: \text{insert } B H \vdash C$  **shows**  $\text{insert } (A \text{ IFF } B) H \vdash C$   
by (metis Iff\_def A B cnj\_E imp\_E insert\_commute thin1)

**lemma** Iff\_E2:  
**assumes**  $A: H \vdash A$  **and**  $B: \text{insert } B H \vdash C$  **shows**  $\text{insert } (B \text{ IFF } A) H \vdash C$   
by (metis Iff\_def A B Bool cnj\_E2 cnj\_mono imp\_E boolean\_axioms.Ident)

**lemma** Iff\_MP\_left:  $H \vdash A \text{ IFF } B \implies \text{insert } A H \vdash C \implies \text{insert } B H \vdash C$   
by (metis Hyp Iff\_E2 cut\_same insertI1 insert\_commute thin1)

**lemma** Iff\_MP\_left':  $H \vdash A \text{ IFF } B \implies \text{insert } B H \vdash C \implies \text{insert } A H \vdash C$   
by (metis Iff\_MP\_left Iff\_sym)

**lemma** Swap:  $\text{insert } (\text{neg } B) H \vdash A \implies \text{insert } (\text{neg } A) H \vdash B$   
by (metis negneg\_D neg\_mono)

**lemma** Cases:  $\text{insert } A H \vdash B \implies \text{insert } (\text{neg } A) H \vdash B \implies H \vdash B$   
by (metis Contra neg\_D neg\_mono)

**lemma** neg\_cnj\_E:  $H \vdash B \implies \text{insert } (\text{neg } A) H \vdash C \implies \text{insert } (\text{neg } (A \text{ AND } B)) H \vdash C$   
by (metis cnj\_I Swap thin1)

**lemma** dsj\_CI:  $\text{insert } (\text{neg } B) H \vdash A \implies H \vdash A \text{ OR } B$   
by (metis Contra dsj\_I1 dsj\_I2 Swap)

**lemma** dsj\_3I:  $\text{insert } (\text{neg } A) (\text{insert } (\text{neg } C) H) \vdash B \implies H \vdash A \text{ OR } B \text{ OR } C$   
by (metis dsj\_CI dsj\_commute insert\_commute)

**lemma** Contrapos1:  $H \vdash A \text{ IMP } B \implies H \vdash \text{neg } B \text{ IMP } \text{neg } A$   
by (metis Bool MP\_same boolean\_axioms.dsjcnj boolean\_axioms.Ident)

**lemma** Contrapos2:  $H \vdash (\text{neg } B) \text{ IMP } (\text{neg } A) \implies H \vdash A \text{ IMP } B$   
by (metis Bool MP\_same boolean\_axioms.dsjcnj boolean\_axioms.Ident)

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

by (metis Hyp Swap thin1)

**lemma** *ContraAssume*:  $neg B \in H \implies insert B H \vdash A$   
by (metis dsj\_I1 Hyp anti\_deduction)

**lemma** *ContraProve*:  $H \vdash B \implies insert (neg B) H \vdash A$   
by (metis Swap thin1)

**lemma** *dsj\_IE1*:  $insert B H \vdash C \implies insert (A OR B) H \vdash A OR C$   
by (metis Assume dsj\_mono)

**lemmas** *dsj\_IE1H* = *dsj\_IE1 dsj\_IE1 [THEN rotate2] dsj\_IE1 [THEN rotate3] dsj\_IE1 [THEN rotate4] dsj\_IE1 [THEN rotate5]*  
*dsj\_IE1 [THEN rotate6] dsj\_IE1 [THEN rotate7] dsj\_IE1 [THEN rotate8]*  
**declare** *dsj\_IE1H* [intro!]

### 3.1 Quantifier reasoning

**lemma** *exi\_I*:  $H \vdash A(i::=x) \implies H \vdash exi i A$   
by (metis MP\_same Spec special\_axioms.intros)

**lemma** *exi\_E*:  
assumes  $insert A H \vdash B$  *atom i*  $\# B \forall C \in H. \text{atom } i \# C$   
shows  $insert (exi i A) H \vdash B$   
by (metis exists imp\_I anti\_deduction assms)

**lemma** *exi\_E\_with\_renaming*:  
assumes  $insert ((i \leftrightarrow i') \cdot A) H \vdash B$  *atom i'*  $\# (A, i, B) \forall C \in H. \text{atom } i' \# C$   
shows  $insert (exi i A) H \vdash B$   
**proof** –  
have  $exi i A = exi i' ((i \leftrightarrow i') \cdot A)$   
using *assms* using *flip\_subst\_fm* by auto  
thus ?thesis  
by (metis exi\_E assms fresh\_Pair)

qed

**lemmas** *exi\_EH* = *exi\_E exi\_E [THEN rotate2] exi\_E [THEN rotate3] exi\_E [THEN rotate4] exi\_E [THEN rotate5]*  
*exi\_E [THEN rotate6] exi\_E [THEN rotate7] exi\_E [THEN rotate8] exi\_E [THEN rotate9]*  
*exi\_E [THEN rotate10]*  
**declare** *exi\_EH* [intro!]

**lemma** *exi\_mono*:  $insert A H \vdash B \implies \forall C \in H. \text{atom } i \# C \implies insert (exi i A) H \vdash (exi i B)$   
by (auto simp add: intro: exi\_I [where x=Var i])

**lemma** *all\_I* [intro!]:  $H \vdash A \implies \forall C \in H. \text{atom } i \# C \implies H \vdash all i A$   
by (auto intro: ContraProve neg\_I0)

**lemma** *all\_D*:  $H \vdash all i A \implies H \vdash A(i::=x)$   
by (metis Assume exi\_I negneg\_D neg\_mono neg\_cut\_same)

**lemma** *all\_E*:  $insert (A(i::=x)) H \vdash B \implies insert (all i A) H \vdash B$   
by (metis exi\_I negneg\_D neg\_mono neg)

**lemma** *all\_E'*:  $H \vdash all i A \implies insert (A(i::=x)) H \vdash B \implies H \vdash B$   
by (metis all\_D cut\_same)

## 3.2 Congruence rules

**lemma** *neg\_cong*:  $H \vdash A \text{ IFF } A' \implies H \vdash \text{neg } A \text{ IFF } \text{neg } A'$   
 by (*metis Iff\_def cnj\_E1 cnj\_E2 cnj\_I Contrapos1*)

**lemma** *dsj\_cong*:  $H \vdash A \text{ IFF } A' \implies H \vdash B \text{ IFF } B' \implies H \vdash A \text{ OR } B \text{ IFF } A' \text{ OR } B'$   
 by (*metis cnj\_E1 cnj\_E2 dsj\_mono Iff\_I Iff\_def anti\_deduction*)

**lemma** *cnj\_cong*:  $H \vdash A \text{ IFF } A' \implies H \vdash B \text{ IFF } B' \implies H \vdash A \text{ AND } B \text{ IFF } A' \text{ AND } B'$   
 by (*metis cnj\_def dsj\_cong neg\_cong*)

**lemma** *imp\_cong*:  $H \vdash A \text{ IFF } A' \implies H \vdash B \text{ IFF } B' \implies H \vdash (A \text{ IMP } B) \text{ IFF } (A' \text{ IMP } B')$   
 by (*metis dsj\_cong neg\_cong*)

**lemma** *Iff\_cong*:  $H \vdash A \text{ IFF } A' \implies H \vdash B \text{ IFF } B' \implies H \vdash (A \text{ IFF } B) \text{ IFF } (A' \text{ IFF } B')$   
 by (*metis Iff\_def cnj\_cong imp\_cong*)

**lemma** *exi\_cong*:  $H \vdash A \text{ IFF } A' \implies \forall C \in H. \text{atom } i \# C \implies H \vdash (\text{exi } i \ A) \text{ IFF } (\text{exi } i \ A')$   
 apply (*rule Iff\_I*)  
 apply (*metis exi\_mono Hyp Iff\_MP\_same Un\_absorb Un\_insert\_right insertI1 thin\_Un*)  
 apply (*metis exi\_mono Hyp Iff\_MP2\_same Un\_absorb Un\_insert\_right insertI1 thin\_Un*)  
 done

**lemma** *all\_cong*:  $H \vdash A \text{ IFF } A' \implies \forall C \in H. \text{atom } i \# C \implies H \vdash (\text{all } i \ A) \text{ IFF } (\text{all } i \ A')$   
 by (*metis exi\_cong neg\_cong*)

**lemma** *Subst*:  $H \vdash A \implies \forall B \in H. \text{atom } i \# B \implies H \vdash A (i::=x)$   
 by (*metis all\_D all\_I*)

## 4 Equality Reasoning

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

**lemma** *eql\_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** –

**obtain** *v2::name and v3::name and v4::name*  
**where** *v2: atom v2 # (t,X1,X3,X4)*  
**and** *v3: atom v3 # (t,t',X1,v2,X4)*  
**and** *v4: atom v4 # (t,t',u,X1,v2,v3)*  
 by (*metis obtain\_fresh*)  
**have**  $\{\} \vdash (\text{Var } X1 \text{ EQ } \text{Var } X2 \text{ AND } \text{Var } X3 \text{ EQ } \text{Var } X4) \text{ IMP } (\text{Var } X1 \text{ EQ } \text{Var } X3 \text{ IMP } \text{Var } X2 \text{ EQ } \text{Var } X4)$   
 by (*rule eql*) (*simp add: eq\_cong\_ax\_def equality\_axioms\_def*)  
**hence**  $\{\} \vdash (\text{Var } X1 \text{ EQ } \text{Var } X2 \text{ AND } \text{Var } X3 \text{ EQ } \text{Var } X4) \text{ IMP } (\text{Var } X1 \text{ EQ } \text{Var } X3 \text{ IMP } \text{Var } X2 \text{ EQ } \text{Var } X4)$   
 by (*drule\_tac i=X1 and x=Var X1 in Subst*) *simp\_all*  
**hence**  $\{\} \vdash (\text{Var } X1 \text{ EQ } \text{Var } v2 \text{ AND } \text{Var } X3 \text{ EQ } \text{Var } X4) \text{ IMP } (\text{Var } X1 \text{ EQ } \text{Var } X3 \text{ IMP } \text{Var } v2 \text{ EQ } \text{Var } X4)$   
 by (*drule\_tac i=X2 and x=Var v2 in Subst*) *simp\_all*  
**hence**  $\{\} \vdash (\text{Var } X1 \text{ EQ } \text{Var } v2 \text{ AND } \text{Var } v3 \text{ EQ } \text{Var } X4) \text{ IMP } (\text{Var } X1 \text{ EQ } \text{Var } v3 \text{ IMP } \text{Var } v2 \text{ EQ } \text{Var } X4)$   
 using *v2*  
 by (*drule\_tac i=X3 and x=Var v3 in Subst*) *simp\_all*  
**hence**  $\{\} \vdash (\text{Var } X1 \text{ EQ } \text{Var } v2 \text{ AND } \text{Var } v3 \text{ EQ } \text{Var } v4) \text{ IMP } (\text{Var } X1 \text{ EQ } \text{Var } v3 \text{ IMP } \text{Var } v2 \text{ EQ } \text{Var } v4)$   
 using *v2 v3*  
 by (*drule\_tac i=X4 and x=Var v4 in Subst*) *simp\_all*

**hence**  $\{\} \vdash (t \text{ EQ } \text{Var } v2 \text{ AND } \text{Var } v3 \text{ EQ } \text{Var } v4) \text{ IMP } (t \text{ EQ } \text{Var } v3 \text{ IMP } \text{Var } v2 \text{ EQ } \text{Var } v4)$   
**using**  $v2 \ v3 \ v4$   
**by** (*drule\_tac*  $i=X1$  **and**  $x=t$  **in** *Subst*) *simp\_all*  
**hence**  $\{\} \vdash (t \text{ EQ } t' \text{ AND } \text{Var } v3 \text{ EQ } \text{Var } v4) \text{ IMP } (t \text{ EQ } \text{Var } v3 \text{ IMP } t' \text{ EQ } \text{Var } v4)$   
**using**  $v2 \ v3 \ v4$   
**by** (*drule\_tac*  $i=v2$  **and**  $x=t'$  **in** *Subst*) *simp\_all*  
**hence**  $\{\} \vdash (t \text{ EQ } t' \text{ AND } u \text{ EQ } \text{Var } v4) \text{ IMP } (t \text{ EQ } u \text{ IMP } t' \text{ EQ } \text{Var } v4)$   
**using**  $v3 \ v4$   
**by** (*drule\_tac*  $i=v3$  **and**  $x=u$  **in** *Subst*) *simp\_all*  
**thus** *?thesis*  
**using**  $v4$   
**by** (*drule\_tac*  $i=v4$  **and**  $x=u'$  **in** *Subst*) *simp\_all*  
**qed**

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

**proof** –  
**have**  $\{\} \vdash \text{Var } X1 \text{ EQ } \text{Var } X1$   
**by** (*rule* *eql*) (*simp* *add*: *equality\_axioms\_def* *refl\_ax\_def*)  
**hence**  $\{\} \vdash t \text{ EQ } t$   
**by** (*drule\_tac*  $i=X1$  **and**  $x=t$  **in** *Subst*) *simp\_all*  
**thus** *?thesis*  
**by** (*metis* *empty\_subsetI* *thin*)  
**qed**

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** –  
**have**  $\{\} \vdash (t \text{ EQ } u \text{ AND } t \text{ EQ } t) \text{ IMP } (t \text{ EQ } t \text{ IMP } u \text{ EQ } t)$   
**by** (*rule* *eql\_cong1*)  
**moreover** **have**  $\{t \text{ EQ } u\} \vdash t \text{ EQ } u \text{ AND } t \text{ EQ } t$   
**by** (*metis* *Assume* *cnj\_I* *Refl*)  
**ultimately** **have**  $\{t \text{ EQ } u\} \vdash u \text{ EQ } t$   
**by** (*metis* *MP\_same* *MP* *Refl* *sup\_bot\_left*)  
**thus**  $H \vdash u \text{ EQ } t$  **by** (*metis* *assms* *cut1*)  
**qed**

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

**by** (*metis* *Assume* *Sym* *Un\_empty\_left* *Un\_insert\_left* *cut*)

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

**proof** –  
**have**  $\bigwedge H. H \vdash (x \text{ EQ } x \text{ AND } y \text{ EQ } z) \text{ IMP } (x \text{ EQ } y \text{ IMP } x \text{ EQ } z)$   
**by** (*metis* *eql\_cong1* *bot\_least* *thin*)  
**moreover** **have**  $\{x \text{ EQ } y, y \text{ EQ } z\} \vdash x \text{ EQ } x \text{ AND } y \text{ EQ } z$   
**by** (*metis* *Assume* *cnj\_I* *Refl* *thin1*)  
**ultimately** **have**  $\{x \text{ EQ } y, y \text{ EQ } z\} \vdash x \text{ EQ } z$   
**by** (*metis* *Hyp* *MP\_same* *insertI1*)  
**thus** *?thesis*  
**by** (*metis* *assms* *cut2*)  
**qed**

**lemma** *eql\_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** –  
**{** **fix**  $t \ t' \ u \ u'$   
**assume**  $H \vdash t \text{ EQ } t'$   $H \vdash u \text{ EQ } u'$   
**moreover** **have**  $\{t \text{ EQ } t', u \text{ EQ } u'\} \vdash t \text{ EQ } u \text{ IMP } t' \text{ EQ } u'$  **using** *eql\_cong1*  
**by** (*metis* *Assume* *cnj\_I* *MP\_null* *insert\_commute*)  
**}**

```

ultimately have  $H \vdash t \text{ EQ } u \text{ IMP } t' \text{ EQ } u'$ 
  by (metis cut2)
}
thus ?thesis
  by (metis Iff_def conj_I assms Sym)
qed

```

```

lemma eql_Trans_E:  $H \vdash x \text{ EQ } u \implies \text{insert } (t \text{ EQ } u) H \vdash A \implies \text{insert } (x \text{ EQ } t) H \vdash A$ 
  by (metis Assume Sym_L Trans cut_same thin1 thin2)

```

## 4.2 The congruence properties for *suc*, *pls* and *tms*

```

lemma suc_cong1:  $\{\} \vdash (t \text{ EQ } t') \text{ IMP } (\text{suc } t \text{ EQ } \text{suc } t')$ 

```

proof –

```

obtain  $v2::\text{name}$  and  $v3::\text{name}$  and  $v4::\text{name}$ 
  where  $v2: \text{atom } v2 \# (t, X1, X3, X4)$ 
    and  $v3: \text{atom } v3 \# (t', X1, v2, X4)$ 
    and  $v4: \text{atom } v4 \# (t, t', X1, v2, v3)$ 
  by (metis obtain_fresh)
have  $\{\} \vdash (\text{Var } X1 \text{ EQ } \text{Var } X2) \text{ IMP } (\text{suc } (\text{Var } X1) \text{ EQ } \text{suc } (\text{Var } X2))$ 
  by (metis syc_cong_ax_def equality_axioms_def insert_iff eql)
hence  $\{\} \vdash (\text{Var } X1 \text{ EQ } \text{Var } v2) \text{ IMP } (\text{suc } (\text{Var } X1) \text{ EQ } \text{suc } (\text{Var } v2))$ 
  by (drule_tac  $i=X2$  and  $x=\text{Var } v2$  in Subst) simp_all
hence  $\{\} \vdash (t \text{ EQ } \text{Var } v2) \text{ IMP } (\text{suc } t \text{ EQ } \text{suc } (\text{Var } v2))$ 
  using  $v2 \ v3 \ v4$ 
  by (drule_tac  $i=X1$  and  $x=t$  in Subst) simp_all
hence  $\{\} \vdash (t \text{ EQ } t') \text{ IMP } (\text{suc } t \text{ EQ } \text{suc } t')$ 
  using  $v2 \ v3 \ v4$ 
  by (drule_tac  $i=v2$  and  $x=t'$  in Subst) simp_all
thus ?thesis
  using  $v4$ 
  by (drule_tac  $i=v4$  in Subst) simp_all
qed

```

```

lemma suc_cong:  $\llbracket H \vdash t \text{ EQ } t' \rrbracket \implies H \vdash \text{suc } t \text{ EQ } \text{suc } t'$ 

```

```

  by (metis anti_deduction suc_cong1 cut1)

```

```

lemma pls_cong1:  $\{\} \vdash (t \text{ EQ } t' \text{ AND } u \text{ EQ } u') \text{ IMP } (\text{pls } t \ u \text{ EQ } \text{pls } t' \ u')$ 

```

proof –

```

obtain  $v2::\text{name}$  and  $v3::\text{name}$  and  $v4::\text{name}$ 
  where  $v2: \text{atom } v2 \# (t, X1, X3, X4)$ 
    and  $v3: \text{atom } v3 \# (t', X1, v2, X4)$ 
    and  $v4: \text{atom } v4 \# (t, t', u, X1, v2, v3)$ 
  by (metis obtain_fresh)
have  $\{\} \vdash (\text{Var } X1 \text{ EQ } \text{Var } X2 \text{ AND } \text{Var } X3 \text{ EQ } \text{Var } X4) \text{ IMP } (\text{pls } (\text{Var } X1) (\text{Var } X3) \text{ EQ } \text{pls } (\text{Var } X2) (\text{Var } X4))$ 
  by (metis pls_cong_ax_def equality_axioms_def insert_iff eql)
hence  $\{\} \vdash (\text{Var } X1 \text{ EQ } \text{Var } v2 \text{ AND } \text{Var } X3 \text{ EQ } \text{Var } X4) \text{ IMP } (\text{pls } (\text{Var } X1) (\text{Var } X3) \text{ EQ } \text{pls } (\text{Var } v2) (\text{Var } X4))$ 
  by (drule_tac  $i=X2$  and  $x=\text{Var } v2$  in Subst) simp_all
hence  $\{\} \vdash (\text{Var } X1 \text{ EQ } \text{Var } v2 \text{ AND } \text{Var } v3 \text{ EQ } \text{Var } X4) \text{ IMP } (\text{pls } (\text{Var } X1) (\text{Var } v3) \text{ EQ } \text{pls } (\text{Var } v2) (\text{Var } X4))$ 
  using  $v2$ 
  by (drule_tac  $i=X3$  and  $x=\text{Var } v3$  in Subst) simp_all
hence  $\{\} \vdash (\text{Var } X1 \text{ EQ } \text{Var } v2 \text{ AND } \text{Var } v3 \text{ EQ } \text{Var } v4) \text{ IMP } (\text{pls } (\text{Var } X1) (\text{Var } v3) \text{ EQ } \text{pls } (\text{Var } v2) (\text{Var } v4))$ 
  using  $v2 \ v3$ 
  by (drule_tac  $i=X4$  and  $x=\text{Var } v4$  in Subst) simp_all

```

**hence**  $\{\} \vdash (t \text{ EQ } \text{Var } v2 \text{ AND } \text{Var } v3 \text{ EQ } \text{Var } v4) \text{ IMP } (\text{pls } t \text{ (Var } v3) \text{ EQ } \text{pls } (\text{Var } v2) \text{ (Var } v4))$   
**using**  $v2 \ v3 \ v4$   
**by**  $(\text{drule\_tac } i=X1 \text{ and } x=t \text{ in } \text{Subst}) \text{ simp\_all}$   
**hence**  $\{\} \vdash (t \text{ EQ } t' \text{ AND } \text{Var } v3 \text{ EQ } \text{Var } v4) \text{ IMP } (\text{pls } t \text{ (Var } v3) \text{ EQ } \text{pls } t' \text{ (Var } v4))$   
**using**  $v2 \ v3 \ v4$   
**by**  $(\text{drule\_tac } i=v2 \text{ and } x=t' \text{ in } \text{Subst}) \text{ simp\_all}$   
**hence**  $\{\} \vdash (t \text{ EQ } t' \text{ AND } u \text{ EQ } \text{Var } v4) \text{ IMP } (\text{pls } t \ u \text{ EQ } \text{pls } t' \text{ (Var } v4))$   
**using**  $v3 \ v4$   
**by**  $(\text{drule\_tac } i=v3 \text{ and } x=u \text{ in } \text{Subst}) \text{ simp\_all}$   
**thus** *?thesis*  
**using**  $v4$   
**by**  $(\text{drule\_tac } i=v4 \text{ and } x=u' \text{ in } \text{Subst}) \text{ simp\_all}$   
**qed**

**lemma** *pls\_cong*:  $\llbracket H \vdash t \text{ EQ } t'; H \vdash u \text{ EQ } u' \rrbracket \implies H \vdash \text{pls } t \ u \text{ EQ } \text{pls } t' \ u'$   
**by**  $(\text{metis } \text{cnj\_I } \text{anti\_deduction } \text{pls\_cong1 } \text{cut1})$

**lemma** *tms\_cong1*:  $\{\} \vdash (t \text{ EQ } t' \text{ AND } u \text{ EQ } u') \text{ IMP } (\text{tms } t \ u \text{ EQ } \text{tms } t' \ u')$

**proof** –

**obtain**  $v2::\text{name}$  **and**  $v3::\text{name}$  **and**  $v4::\text{name}$   
**where**  $v2: \text{atom } v2 \ \#\ (t, X1, X3, X4)$   
**and**  $v3: \text{atom } v3 \ \#\ (t, t', X1, v2, X4)$   
**and**  $v4: \text{atom } v4 \ \#\ (t, t', u, X1, v2, v3)$   
**by**  $(\text{metis } \text{obtain\_fresh})$   
**have**  $\{\} \vdash (\text{Var } X1 \text{ EQ } \text{Var } X2 \text{ AND } \text{Var } X3 \text{ EQ } \text{Var } X4) \text{ IMP } (\text{tms } (\text{Var } X1) \text{ (Var } X3) \text{ EQ } \text{tms } (\text{Var } X2) \text{ (Var } X4))$   
**by**  $(\text{metis } \text{tms\_cong\_ax\_def } \text{equality\_axioms\_def } \text{insert\_iff } \text{eql})$   
**hence**  $\{\} \vdash (\text{Var } X1 \text{ EQ } \text{Var } v2 \text{ AND } \text{Var } X3 \text{ EQ } \text{Var } X4) \text{ IMP } (\text{tms } (\text{Var } X1) \text{ (Var } X3) \text{ EQ } \text{tms } (\text{Var } v2) \text{ (Var } X4))$   
**by**  $(\text{drule\_tac } i=X2 \text{ and } x=\text{Var } v2 \text{ in } \text{Subst}) \text{ simp\_all}$   
**hence**  $\{\} \vdash (\text{Var } X1 \text{ EQ } \text{Var } v2 \text{ AND } \text{Var } v3 \text{ EQ } \text{Var } X4) \text{ IMP } (\text{tms } (\text{Var } X1) \text{ (Var } v3) \text{ EQ } \text{tms } (\text{Var } v2) \text{ (Var } X4))$   
**using**  $v2$   
**by**  $(\text{drule\_tac } i=X3 \text{ and } x=\text{Var } v3 \text{ in } \text{Subst}) \text{ simp\_all}$   
**hence**  $\{\} \vdash (\text{Var } X1 \text{ EQ } \text{Var } v2 \text{ AND } \text{Var } v3 \text{ EQ } \text{Var } v4) \text{ IMP } (\text{tms } (\text{Var } X1) \text{ (Var } v3) \text{ EQ } \text{tms } (\text{Var } v2) \text{ (Var } v4))$   
**using**  $v2 \ v3$   
**by**  $(\text{drule\_tac } i=X4 \text{ and } x=\text{Var } v4 \text{ in } \text{Subst}) \text{ simp\_all}$   
**hence**  $\{\} \vdash (t \text{ EQ } \text{Var } v2 \text{ AND } \text{Var } v3 \text{ EQ } \text{Var } v4) \text{ IMP } (\text{tms } t \text{ (Var } v3) \text{ EQ } \text{tms } (\text{Var } v2) \text{ (Var } v4))$   
**using**  $v2 \ v3 \ v4$   
**by**  $(\text{drule\_tac } i=X1 \text{ and } x=t \text{ in } \text{Subst}) \text{ simp\_all}$   
**hence**  $\{\} \vdash (t \text{ EQ } t' \text{ AND } \text{Var } v3 \text{ EQ } \text{Var } v4) \text{ IMP } (\text{tms } t \text{ (Var } v3) \text{ EQ } \text{tms } t' \text{ (Var } v4))$   
**using**  $v2 \ v3 \ v4$   
**by**  $(\text{drule\_tac } i=v2 \text{ and } x=t' \text{ in } \text{Subst}) \text{ simp\_all}$   
**hence**  $\{\} \vdash (t \text{ EQ } t' \text{ AND } u \text{ EQ } \text{Var } v4) \text{ IMP } (\text{tms } t \ u \text{ EQ } \text{tms } t' \text{ (Var } v4))$   
**using**  $v3 \ v4$   
**by**  $(\text{drule\_tac } i=v3 \text{ and } x=u \text{ in } \text{Subst}) \text{ simp\_all}$   
**thus** *?thesis*  
**using**  $v4$   
**by**  $(\text{drule\_tac } i=v4 \text{ and } x=u' \text{ in } \text{Subst}) \text{ simp\_all}$   
**qed**

**lemma** *tms\_cong*:  $\llbracket H \vdash t \text{ EQ } t'; H \vdash u \text{ EQ } u' \rrbracket \implies H \vdash \text{tms } t \ u \text{ EQ } \text{tms } t' \ u'$   
**by**  $(\text{metis } \text{cnj\_I } \text{anti\_deduction } \text{tms\_cong1 } \text{cut1})$

### 4.3 Substitution for equalities

**lemma** *eql\_subst\_trm\_Iff*:  $\{t \text{ EQ } u\} \vdash \text{subst } i \ t \ \text{trm} \text{ EQ } \text{subst } i \ u \ \text{trm}$

by (induct trm rule: trm.induct) (auto simp: suc\_cong pls\_cong tms\_cong)

**lemma** *eql\_subst\_fmula\_Iff*: insert (t EQ u) H  $\vdash$  A(i::=t) IFF A(i::=u)

**proof** –

have { t EQ u }  $\vdash$  A(i::=t) IFF A(i::=u)

by (nominal\_induct A avoiding: i t u rule: fmula.strong\_induct)

(auto simp: dsj\_cong neg\_cong exi\_cong eql\_cong eql\_subst\_trm\_Iff)

**thus** ?thesis

by (metis Assume cut1)

**qed**

**lemma** *Var\_eql\_subst\_Iff*: insert (Var i EQ t) H  $\vdash$  A(i::=t) IFF A

by (metis eql\_subst\_fmula\_Iff Iff\_sym subst\_fmula\_id)

**lemma** *Var\_eql\_imp\_subst\_Iff*: H  $\vdash$  Var i EQ t  $\implies$  H  $\vdash$  A(i::=t) IFF A

by (metis Var\_eql\_subst\_Iff cut\_same)

## 4.4 Congruence rules for predicates

**lemma** *P1\_cong*:

fixes tms :: trm list

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

shows H  $\vdash$  P x IFF P x'

**proof** –

**obtain** i::name **where** i: atom i  $\#$  tms

by (metis obtain\_fresh)

have insert (x EQ x') H  $\vdash$  (P (Var i))(i::=x) IFF (P (Var i))(i::=x')

by (rule eql\_subst\_fmula\_Iff)

**thus** ?thesis using assms i

by (metis cut\_same subst.simps(2))

**qed**

**lemma** *P2\_cong*:

fixes tms :: trm list

assumes sub:  $\bigwedge i t u x. \text{atom } i \# \text{tms} \implies (P t u)(i::=x) = P (\text{subst } i x t) (\text{subst } i x u)$

and eq: H  $\vdash$  x EQ x' H  $\vdash$  y EQ y'

shows H  $\vdash$  P x y IFF P x' y'

**proof** –

have yy': { y EQ y' }  $\vdash$  P x' y IFF P x' y'

by (rule P1\_cong [where tms=[y,x']@tms]) (auto simp: fresh\_Cons sub)

have { x EQ x' }  $\vdash$  P x y IFF P x' y

by (rule P1\_cong [where tms=[y,x']@tms]) (auto simp: fresh\_Cons sub)

hence {x EQ x', y EQ y'}  $\vdash$  P x y IFF P x' y'

by (metis Assume Iff\_trans cut1 rotate2 yy')

**thus** ?thesis

by (metis cut2 eq)

**qed**

**lemma** *P3\_cong*:

fixes tms :: trm list

assumes sub:  $\bigwedge i t u v x. \text{atom } i \# \text{tms} \implies$

(P t u v)(i::=x) = P (subst i x t) (subst i x u) (subst i x v)

and eq: H  $\vdash$  x EQ x' H  $\vdash$  y EQ y' H  $\vdash$  z EQ z'

shows H  $\vdash$  P x y z IFF P x' y' z'

**proof** –

**obtain** i::name **where** i: atom i  $\#$  (z,z',y,y',x,x')

by (metis obtain\_fresh)

have tl: { y EQ y', z EQ z' }  $\vdash$  P x' y z IFF P x' y' z'

```

    by (rule P2_cong [where tms=[z,z',y,y',x,x']@tms]) (auto simp: fresh_Cons sub)
  have hd: { x EQ x' } ⊢ P x y z IFF P x' y z
    by (rule P1_cong [where tms=[z,y,x']@tms]) (auto simp: fresh_Cons sub)
  have {x EQ x', y EQ y', z EQ z'} ⊢ P x y z IFF P x' y' z'
    by (metis Assume thin1 hd [THEN cut1] tl Iff_trans)
  thus ?thesis
    by (rule cut3) (rule eq)+
qed

lemma P4_cong:
  fixes tms :: trm list
  assumes sub:  $\bigwedge i t1 t2 t3 t4 x. \text{atom } i \# \text{tms} \implies$ 
    (P t1 t2 t3 t4)(i:=x) = P (subst i x t1) (subst i x t2) (subst i x t3) (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 -
  obtain i::name where i:  $\text{atom } i \# (x4,x4',x3,x3',x2,x2',x1,x1')$ 
    by (metis obtain_fresh)
  have tl: { x2 EQ x2', x3 EQ x3', x4 EQ x4' } ⊢ P x1' x2 x3 x4 IFF P x1' x2' x3' x4'
    by (rule P3_cong [where tms=[x4,x4',x3,x3',x2,x2',x1,x1']@tms]) (auto simp: fresh_Cons sub)
  have hd: { x1 EQ x1' } ⊢ P x1 x2 x3 x4 IFF P x1' x2 x3 x4
    by (auto simp: fresh_Cons sub intro!: P1_cong [where tms=[x4,x3,x2,x1']@tms])
  have {x1 EQ x1', x2 EQ x2', x3 EQ x3', x4 EQ x4'} ⊢ P x1 x2 x3 x4 IFF P x1' x2' x3' x4'
    by (metis Assume thin1 hd [THEN cut1] tl Iff_trans)
  thus ?thesis
    by (rule cut4) (rule eq)+
qed

```

## 4.5 The formula *fls*

```

lemma neg_I [intro!]: insert A H ⊢ fls  $\implies$  H ⊢ neg A
  unfolding fls_def
  by (meson neg_D neg_I0 Refl)

```

```

lemma neg_E [intro!]: H ⊢ A  $\implies$  insert (neg A) H ⊢ fls
  by (rule ContraProve)

```

```

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!]

```

```

lemma neg_imp_I [intro!]: H ⊢ A  $\implies$  insert B H ⊢ fls  $\implies$  H ⊢ neg (A IMP B)
  by (metis negneg_I neg_dsj_I neg_I)

```

```

lemma neg_imp_E [intro!]: insert (neg B) (insert A H) ⊢ C  $\implies$  insert (neg (A IMP B)) H ⊢ C
  apply (rule cut_same [where A=A])
  apply (metis Assume dsj_I1 negneg_D neg_mono)
  apply (metis Swap imp_I rotate2 thin1)
  done

```

```

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
  by (simp add: ContraProve fls_def)

```

```

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)
  by (metis fls_E neg_I)

```

```

lemma exFalso: H  $\vdash$  fls  $\implies$  H  $\vdash$  A
  by (metis neg_D truth_provable)

```

Soundness of the provability relation

```

theorem nprv_sound: assumes H  $\vdash$  A shows ( $\forall B \in H. \text{eval\_fmla } e \ B$ )  $\implies$  eval_fmla e A
using assms

```

```

proof (induct arbitrary: e)
  case (Hyp A H) thus ?case
    by auto
next
  case (Q H) thus ?case
    unfolding Q_axioms_def
    using not0_implies_Suc by fastforce
next
  case (Bool A H) thus ?case
    by (metis boolean_axioms_hold)
next
  case (eql A H) thus ?case
    by (metis equality_axioms_hold)
next
  case (Spec A H) thus ?case
    by (metis special_axioms_hold)
next
  case (MP H A B H') thus ?case
    by auto
next
  case (exiists H A B i e) thus ?case
    by auto (metis forget_eval_fmla)
qed

```

## 5 Instantiation of Syntax-Independent Logic Infrastructure

### 5.1 Preliminaries

```

inductive_set num :: trm set where
  zer[intro!,simp]: zer  $\in$  num
  suc[simp]: t  $\in$  num  $\implies$  suc t  $\in$  num

```

```

definition ground_aux :: trm  $\Rightarrow$  atom set  $\Rightarrow$  bool

```

**where**  $ground\_aux\ t\ S \equiv (supp\ t \subseteq S)$

**abbreviation**  $ground :: trm \Rightarrow bool$   
**where**  $ground\ t \equiv ground\_aux\ t\ \{\}$

**definition**  $ground\_fmla\_aux :: fmla \Rightarrow atom\ set \Rightarrow bool$   
**where**  $ground\_fmla\_aux\ A\ S \equiv (supp\ A \subseteq S)$

**abbreviation**  $ground\_fmla :: fmla \Rightarrow bool$   
**where**  $ground\_fmla\ A \equiv ground\_fmla\_aux\ A\ \{\}$

**lemma**  $ground\_aux\_simps[simp]$ :

$ground\_aux\ zer\ S = True$

$ground\_aux\ (Var\ k)\ S = (if\ atom\ k \in S\ then\ True\ else\ False)$

$ground\_aux\ (suc\ t)\ S = (ground\_aux\ t\ S)$

$ground\_aux\ (pls\ t\ u)\ S = (ground\_aux\ t\ S \wedge ground\_aux\ u\ S)$

$ground\_aux\ (tms\ t\ u)\ S = (ground\_aux\ t\ S \wedge ground\_aux\ u\ S)$

**unfolding**  $ground\_aux\_def$

**by**  $(simp\_all\ add:\ supp\_at\_base)$

**lemma**  $ground\_fmla\_aux\_simps[simp]$ :

$ground\_fmla\_aux\ fls\ S = True$

$ground\_fmla\_aux\ (t\ EQ\ u)\ S = (ground\_aux\ t\ S \wedge ground\_aux\ u\ S)$

$ground\_fmla\_aux\ (A\ OR\ B)\ S = (ground\_fmla\_aux\ A\ S \wedge ground\_fmla\_aux\ B\ S)$

$ground\_fmla\_aux\ (A\ AND\ B)\ S = (ground\_fmla\_aux\ A\ S \wedge ground\_fmla\_aux\ B\ S)$

$ground\_fmla\_aux\ (A\ IFF\ B)\ S = (ground\_fmla\_aux\ A\ S \wedge ground\_fmla\_aux\ B\ S)$

$ground\_fmla\_aux\ (neg\ A)\ S = (ground\_fmla\_aux\ A\ S)$

$ground\_fmla\_aux\ (exi\ x\ A)\ S = (ground\_fmla\_aux\ A\ (S \cup \{atom\ x\}))$

**by**  $(auto\ simp:\ ground\_fmla\_aux\_def\ ground\_aux\_def\ supp\_conv\_fresh)$

**lemma**  $ground\_fresh[simp]$ :

$ground\ t \implies atom\ i \# t$

$ground\_fmla\ A \implies atom\ i \# A$

**unfolding**  $ground\_aux\_def\ ground\_fmla\_aux\_def\ fresh\_def$

**by**  $simp\_all$

**definition**  $Fvars\ t = \{a :: name. \neg\ atom\ a \# t\}$

**lemma**  $Fvars\_trm\_simps[simp]$ :

$Fvars\ zer = \{\}$

$Fvars\ (Var\ a) = \{a\}$

$Fvars\ (suc\ x) = Fvars\ x$

$Fvars\ (pls\ x\ y) = Fvars\ x \cup Fvars\ y$

$Fvars\ (tms\ x\ y) = Fvars\ x \cup Fvars\ y$

**by**  $(auto\ simp:\ Fvars\_def\ fresh\_at\_base(2))$

**lemma**  $finite\_Fvars\_trm[simp]$ :

**fixes**  $t :: trm$

**shows**  $finite\ (Fvars\ t)$

**by**  $(induct\ t\ rule:\ trm.induct)\ auto$

**lemma**  $Fvars\_fmla\_simps[simp]$ :

$Fvars\ (x\ EQ\ y) = Fvars\ x \cup Fvars\ y$

$Fvars\ (A\ OR\ B) = Fvars\ A \cup Fvars\ B$

$Fvars\ (A\ AND\ B) = Fvars\ A \cup Fvars\ B$

$Fvars\ (A\ IMP\ B) = Fvars\ A \cup Fvars\ B$

$Fvars\ fls = \{\}$

$Fvars (neg A) = Fvars A$   
 $Fvars (exi a A) = Fvars A - \{a\}$   
 $Fvars (all a A) = Fvars A - \{a\}$   
**by** (*auto simp: Fvars\_def fresh\_at\_base(2)*)

**lemma** *finite\_Fvars\_fmula[simp]*:  
**fixes**  $A :: fmla$   
**shows** *finite* ( $Fvars A$ )  
**by** (*induct A rule: fmla.induct*) *auto*

**lemma** *subst\_trm\_subst\_trm[simp]*:  
 $x \neq y \implies atom\ x \# u \implies subst\ y\ u\ (subst\ x\ t\ v) = subst\ x\ (subst\ y\ u\ t)\ (subst\ y\ u\ v)$   
**by** (*induct v rule: trm.induct*) *auto*

**lemma** *subst\_fmula\_subst\_fmula[simp]*:  
 $x \neq y \implies atom\ x \# u \implies (A(x::=t))(y::=u) = (A(y::=u))(x::=subst\ y\ u\ t)$   
**by** (*nominal\_induct A avoiding: x t y u rule: fmla.strong\_induct*) *auto*

**lemma** *Fvars\_empty\_ground[simp]*:  $Fvars\ t = \{\} \implies ground\ t$   
**by** (*induct t rule: trm.induct*) *auto*

**lemma** *Fvars\_ground\_aux*:  $Fvars\ t \subseteq B \implies ground\_aux\ t\ (atom\ 'B)$   
**by** (*induct t rule: trm.induct*) *auto*

**lemma** *ground\_Fvars*:  $ground\ t \longleftrightarrow Fvars\ t = \{\}$   
**apply** (*rule iffI*)  
**subgoal** **by** (*auto simp only: Fvars\_def ground\_fresh*) []  
**by** *auto*

**lemma** *Fvars\_ground\_fmula\_aux*:  $Fvars\ A \subseteq B \implies ground\_fmula\_aux\ A\ (atom\ 'B)$   
**apply** (*induct A arbitrary: B rule: fmla.induct*)  
**subgoal** **by** (*auto simp: Diff\_subset\_conv Fvars\_ground\_aux*)  
**subgoal** **by** (*auto simp: Diff\_subset\_conv Fvars\_ground\_aux*)  
**subgoal** **by** (*auto simp: Diff\_subset\_conv Fvars\_ground\_aux*)  
**subgoal** **by** (*metis Diff\_subset\_conv Fvars\_fmula\_simps(7) Un\_insert\_left Un\_insert\_right ground\_fmula\_aux\_simps(7) image\_insert sup\_bot.left\_neutral sup\_bot.right\_neutral*) .

**lemma** *ground\_fmula\_Fvars*:  $ground\_fmula\ A \longleftrightarrow Fvars\ A = \{\}$   
**apply** (*rule iffI*)  
**subgoal** **by** (*auto simp only: Fvars\_def ground\_fresh*)  
**by** (*auto intro: Fvars\_ground\_fmula\_aux[of A \{\}, simplified]*)

**lemma** *obtain\_const\_trm*:  
**obtains**  $t$  **where**  $eval\_trm\ e\ t = x\ t \in num$   
**apply** (*induct x*)  
**using**  $eval\_trm.simps(1)$   $eval\_trm.simps(3)$   $num.suc$  **by** *blast+*

**lemma** *ex\_eval\_fmula\_iff\_exists\_num*:  
 $eval\_fmula\ e\ (exi\ k\ A) \longleftrightarrow (\exists t. eval\_fmula\ e\ (A(k::=t)) \wedge t \in num)$   
**by** (*auto simp: eval\_subst\_fmula*) (*metis obtain\_const\_trm*)

**lemma** *exi\_ren*:  $y \notin Fvars\ \varphi \implies exi\ x\ \varphi = exi\ y\ (\varphi(x::=Var\ y))$   
**using** *exi\_ren\_subst\_fresh Fvars\_def* **by** *blast*

**lemma** *all\_ren*:  $y \notin Fvars\ \varphi \implies all\ x\ \varphi = all\ y\ (\varphi(x::=Var\ y))$   
**by** (*simp add: exi\_ren*)

**lemma** *Fvars\_num*[simp]:  $t \in \text{num} \implies \text{Fvars } t = \{\}$   
**by** (*induct t rule: trm.induct*) (*auto elim: num.cases*)

## 5.2 Instantiation of the generic syntax and deduction relation

**interpretation** *Generic\_Syntax* **where**

```

  var = UNIV :: name set
  and trm = UNIV :: trm set
  and fmla = UNIV :: fmla set
  and Var = Var
  and FvarsT = Fvars
  and substT =  $\lambda t u x. \text{subst } x u t$ 
  and Fvars = Fvars
  and subst =  $\lambda A u x. \text{subst\_fmla } A x u$ 
  apply unfold_locales
  subgoal by simp
  subgoal by simp
  subgoal by simp
  subgoal by simp
  subgoal by simp
  subgoal by simp
  subgoal by simp
  subgoal by simp
  subgoal for t by (induct t rule: trm.induct) auto
  subgoal by simp
  subgoal by simp
  subgoal by simp
  subgoal unfolding Fvars_def fresh_subst_fmla_if by auto
  subgoal unfolding Fvars_def by auto
  subgoal unfolding Fvars_def by simp
  subgoal by simp
  subgoal unfolding Fvars_def by simp .

```

**interpretation** *Syntax\_with\_Numerals* **where**

```

  var = UNIV :: name set
  and trm = UNIV :: trm set
  and fmla = UNIV :: fmla set
  and num = num
  and Var = Var
  and FvarsT = Fvars
  and substT =  $\lambda t u x. \text{subst } x u t$ 
  and Fvars = Fvars
  and subst =  $\lambda A u x. \text{subst\_fmla } A x u$ 
  apply unfold_locales
  subgoal by (auto intro!: exI[of _ zer])
  subgoal by simp
  subgoal by (simp add: ground_Fvars) .

```

**interpretation** *Deduct\_with\_False* **where**

```

  var = UNIV :: name set
  and trm = UNIV :: trm set
  and fmla = UNIV :: fmla set
  and num = num
  and Var = Var
  and FvarsT = Fvars
  and substT =  $\lambda t u x. \text{subst } x u t$ 
  and Fvars = Fvars
  and subst =  $\lambda A u x. \text{subst\_fmla } A x u$ 

```

```

and eql = eql and cnj = cnj and imp = imp and all = all
and exi = exi and fls = fls
and prv = (⊢) {}
apply unfold_locales
subgoal by simp
subgoal by simp
subgoal by simp
subgoal by simp
subgoal by simp
subgoal by simp
subgoal by simp
subgoal by simp
subgoal by simp
subgoal by simp
subgoal by simp
subgoal by simp
subgoal by simp
subgoal by simp
subgoal by simp
subgoal by simp
subgoal by simp
subgoal by simp
subgoal by simp
subgoal by simp
subgoal using MP_null by blast
subgoal by blast
subgoal for A B C
  apply (rule imp_I)+
  apply (rule MP_same[of B])
  apply (rule MP_same[of C])
  apply (auto intro: neg_D) .
subgoal by blast
subgoal by blast
subgoal by blast
subgoal unfolding Fvars_def by (auto intro: MP_null)
subgoal unfolding Fvars_def by (auto intro: MP_null)
subgoal by (auto intro: all_D)
subgoal by (auto intro: exi_I)
subgoal by simp
subgoal by (metis cnj_E2 Iff_def imp_I Var_eql_subst_Iff)
subgoal by blast .

```

**interpretation** *Deduct\_with\_False\_Disj* **where**

```

  var = UNIV :: name set
and trm = UNIV :: trm set
and fmla = UNIV :: fmla set
and num = num
and Var = Var
and FvarsT = Fvars
and substT =  $\lambda t u x. \text{subst } x u t$ 
and Fvars = Fvars
and subst =  $\lambda A u x. \text{subst\_fmla } A x u$ 
and eql = eql and cnj = cnj and dsj = dsj and imp = imp
and all = all and exi = exi and fls = fls
and prv = (⊢) {}
apply unfold_locales
subgoal by simp
subgoal by simp
subgoal by simp
subgoal by (auto intro: dsj_I1)

```

**subgoal by** (*auto intro: dsj\_I2*)  
**subgoal by** (*auto intro: ContraAssume*) .

### 5.3 Instantiation of the arithmetic-enriched generic syntax and deduction relation

**interpretation** *Syntax\_Arith\_aux* **where**

*var* = *UNIV* :: *name set*  
**and** *trm* = *UNIV* :: *trm set*  
**and** *fmla* = *UNIV* :: *fmla set*  
**and** *num* = *num*  
**and** *Var* = *Var*  
**and** *FvarsT* = *Fvars*  
**and** *substT* =  $\lambda t u x. \text{subst } x u t$   
**and** *Fvars* = *Fvars*  
**and** *subst* =  $\lambda A u x. \text{subst\_fmla } A x u$   
**and** *eql* = *eql* **and** *cnj* = *cnj* **and** *imp* = *imp* **and** *all* = *all*  
**and** *exi* = *exi* **and** *dsj* = *dsj* **and** *fls* = *fls*  
**and** *zer* = *zer* **and** *suc* = *suc* **and** *pls* = *pls* **and** *tms* = *tms*  
**by** *unfold\_locales* (*auto simp: exi\_ren all\_ren*)

**lemma** *num\_range\_Num*: *num* = *range Num*

**proof** –

{**fix** *t* **assume** *t* ∈ *num*  
**then have**  $\exists n. t = \text{Num } n$   
**apply**(*induct t rule: trm.induct*)  
**subgoal by** (*auto intro: exI[of \_ 0]*)  
**subgoal by** (*auto elim: num.cases*)  
**subgoal by** (*metis Num.simps(2) num.cases trm.distinct(3) trm.eq\_iff(3)*)  
**by** (*auto elim: num.cases*)  
}  
**moreover**  
{**fix** *n* **have** *Num n* ∈ *num*  
**by** (*induct n*) *auto*  
}  
**ultimately show** *?thesis* **by** *auto*

**qed**

**lemma** [*simp*]:  $\{\} \vdash \text{neg } (\text{zer } EQ \text{ suc } (\text{Var } xx))$

**proof** –

**have**  $0: \{\} \vdash \text{Robinson\_Arithmetic.neg } (\text{zer } EQ \text{ suc } (\text{Var } xx))$   
**by** (*intro nprv.Q*) (*auto intro!: exI[of \_ zz] simp: Q\_axioms\_def*)  
**show** *?thesis* **unfolding** *neg\_def*  
**by** (*simp add: 0 dsj\_I1*)

**qed**

**lemma** [*simp*]:  $\{\} \vdash \text{Var } yy \text{ EQ } \text{zer } OR \text{ exi } xx \text{ (Var } yy \text{ EQ } \text{ suc } (\text{Var } xx))$

**by** (*intro nprv.Q*) (*auto intro!: exI[of \_ zz] simp: Q\_axioms\_def*)

**lemma** [*simp*]:  $\{\} \vdash \text{pls } (\text{Var } xx) \text{ zer } EQ \text{ Var } xx$

**by** (*intro nprv.Q*) (*auto intro!: exI[of \_ zz] simp: Q\_axioms\_def*)

**lemma** [*simp*]:  $\{\} \vdash \text{tms } (\text{Var } xx) \text{ zer } EQ \text{ zer}$

**by** (*intro nprv.Q*) (*auto intro!: exI[of \_ zz] simp: Q\_axioms\_def*)

**interpretation** *S*: *Syntax\_Arith* **where**

*var* = *UNIV* :: *name set*  
**and** *trm* = *UNIV* :: *trm set*

```

and fmla = UNIV :: fmla set
and num = num
and Var = Var
and FvarsT = Fvars
and substT =  $\lambda t u x. \text{subst } x u t$ 
and Fvars = Fvars
and subst =  $\lambda A u x. \text{subst\_fmla } A x u$ 
and eql = eql and cnj = cnj and imp = imp and all = all
and exi = exi and dsj = dsj and fls = fls and zer = zer
and suc = suc and pls = pls and tms = tms
using num_range_Num by unfold_locales auto

```

**interpretation** *Deduct\_Q* **where**

```

  var = UNIV :: name set
and trm = UNIV :: trm set
and fmla = UNIV :: fmla set
and num = num
and Var = Var
and FvarsT = Fvars
and substT =  $\lambda t u x. \text{subst } x u t$ 
and Fvars = Fvars
and subst =  $\lambda A u x. \text{subst\_fmla } A x u$ 
and eql = eql and cnj = cnj and imp = imp and all = all
and exi = exi and dsj = dsj and fls = fls and zer = zer
and suc = suc and pls = pls and tms = tms
and prv = ( $\vdash$ ) {}
by unfold_locales (auto simp add: Q Q_axioms_def)

```

## 5.4 Instantiation of the abstract notion of standard model and truth

**interpretation** *Minimal\_Truth\_Soundness* **where**

```

  var = UNIV :: name set
and trm = UNIV :: trm set
and fmla = UNIV :: fmla set
and num = num
and Var = Var
and FvarsT = Fvars
and substT =  $\lambda t u x. \text{subst } x u t$ 
and Fvars = Fvars
and subst =  $\lambda A u x. \text{subst\_fmla } A x u$ 
and eql = eql and cnj = cnj and dsj = dsj and imp = imp
and all = all and exi = exi and fls = fls
and prv = ( $\vdash$ ) {}
and isTrue = eval_fmla e0
apply unfold_locales
subgoal by (auto simp: fls_def)
subgoal by simp
subgoal by (auto simp only: ex_eval_fmla_iff_exists_num eval_fmla.simps subst_fmla.simps)
subgoal by (auto simp only: ex_eval_fmla_iff_exists_num)
subgoal by (simp add: neg_def)
subgoal by (auto dest: nprv_sound) .

```