

Robinson Arithmetic

Andrei Popescu Dmitriy Traytel

May 15, 2021

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

1	Terms and Formulas	2
1.1	The datatypes	2
1.2	Substitution	2
1.3	Semantics	3
1.4	Derived logical connectives	4
1.4.1	Conjunction	5
1.4.2	If and only if	5
1.4.3	False	5
2	Axioms and Theorems	6
2.1	Logical axioms	6
2.2	Concrete variables	6
2.3	Equality axioms	6
2.4	The Q (Robinson-arithmetic-specific) axioms	7
2.5	The proof system	7
2.6	Derived rules of inference	7
2.7	The deduction theorem	10
2.8	Cut rules	10
3	Miscellaneous Logical Rules	11
3.1	Quantifier reasoning	14
3.2	Congruence rules	14
4	Equality Reasoning	15
4.1	The congruence property for (EQ) , and other basic properties of equality	15
4.2	The congruence properties for <i>suc</i> , <i>pls</i> and <i>tms</i>	15
4.3	Substitution for equalities	16
4.4	Congruence rules for predicates	16
4.5	The formula <i>fls</i>	16
5	Instantiation of Syntax-Independent Logic Infrastructure	17
5.1	Preliminaries	17
5.2	Instantiation of the generic syntax and deduction relation	19
5.3	Instantiation of the arithmetic-enriched generic syntax and deduction relation	20
5.4	Instantiation of the abstract notion of standard model and truth	21

1 Terms and Formulas

nat is a pure permutation type

instance *nat* :: *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 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)*

<proof>

nominal_termination (*eqvt*)

<proof>

lemma *fresh_subst_if* [*simp*]:

j # subst i x t \longleftrightarrow (*atom i # t* \wedge *j # t*) \vee (*j # x* \wedge (*j # t* \vee *j = atom i*))
<proof>

lemma *forget_subst_trm* [*simp*]: *atom a # trm* \implies *subst a x trm* = *trm*

<proof>

lemma *subst_trm_id* [*simp*]: *subst a (Var a) trm* = *trm*

<proof>

lemma *subst_trm_commute* [*simp*]:

atom j # trm \implies *subst j u (subst i t trm)* = *subst i (subst j u t) trm*
<proof>

lemma *subst_trm_commute2* [*simp*]:

atom j # t \implies *atom i # u* \implies *i* \neq *j* \implies *subst j u (subst i t trm)* = *subst i t (subst j u trm)*
<proof>

lemma *repeat_subst_trm* [*simp*]: *subst i u (subst i t trm)* = *subst i (subst i u t) trm*

$\langle \text{proof} \rangle$

nominal_function *subst_fm1a* :: *fmla* \Rightarrow *name* \Rightarrow *trm* \Rightarrow *fmla* ($_'$ ($_ ::= _'$) [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*))

$\langle \text{proof} \rangle$

nominal_termination (*eqvt*)

$\langle \text{proof} \rangle$

lemma *size_subst_fm1a* [*simp*]: *size* (*A*(*i*::=*x*)) = *size A*

$\langle \text{proof} \rangle$

lemma *forget_subst_fm1a* [*simp*]: *atom a* $\#$ *A* \Longrightarrow *A*(*a*::=*x*) = *A*

$\langle \text{proof} \rangle$

lemma *subst_fm1a_id* [*simp*]: *A*(*a*::=*Var a*) = *A*

$\langle \text{proof} \rangle$

lemma *fresh_subst_fm1a_if* [*simp*]:

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

$\langle \text{proof} \rangle$

lemma *subst_fm1a_commute* [*simp*]:

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

$\langle \text{proof} \rangle$

lemma *repeat_subst_fm1a* [*simp*]: (*A*(*i*::=*t*))(*i*::=*u*) = *A*(*i* ::= *subst i u t*)

$\langle \text{proof} \rangle$

lemma *subst_fm1a_exi_with_renaming*:

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

$\langle \text{proof} \rangle$

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*

$\langle \text{proof} \rangle$

lemma *flip_subst_fm1a*: *atom y* $\#$ φ \Longrightarrow (*x* \leftrightarrow *y*) \cdot φ = φ (*x*::=*Var y*)

$\langle \text{proof} \rangle$

lemma *exi_ren_subst_fresh*: *atom y* $\#$ φ \Longrightarrow *exi x* φ = *exi y* (φ (*x*::=*Var y*))

$\langle \text{proof} \rangle$

1.3 Semantics

definition *e0* :: (*name*, *nat*) *finfun* — the null environment

where *e0* \equiv *finfun_const* 0

nominal_function *eval_trm* :: (*name*, *nat*) *finfun* \Rightarrow *trm* \Rightarrow *nat*

where

eval_trm e zer = 0
| *eval_trm e* (*Var k*) = *finfun_apply* *e k*
| *eval_trm e* (*suc t*) = *Suc* (*eval_trm e t*)

| $eval_trm\ e\ (pls\ t\ u) = eval_trm\ e\ t + eval_trm\ e\ u$
| $eval_trm\ e\ (tms\ t\ u) = eval_trm\ e\ t * eval_trm\ e\ u$
⟨proof⟩

nominal_termination (eqvt)
⟨proof⟩

nominal_function $eval_fmla :: (name, nat) \text{ finfun} \Rightarrow fmla \Rightarrow bool$

where

$eval_fmla\ e\ (t\ EQ\ u) \longleftrightarrow eval_trm\ e\ t = eval_trm\ e\ u$

| $eval_fmla\ e\ (A\ OR\ B) \longleftrightarrow eval_fmla\ e\ A \vee eval_fmla\ e\ B$

| $eval_fmla\ e\ (neg\ A) \longleftrightarrow (\sim\ eval_fmla\ e\ A)$

| $atom\ k\ \# \ e \Longrightarrow eval_fmla\ e\ (exi\ k\ A) \longleftrightarrow (\exists x. eval_fmla\ (finfun_update\ e\ k\ x)\ A)$

⟨proof⟩

nominal_termination (eqvt)
⟨proof⟩

lemma $eval_trm_rename$:

assumes $atom\ k' \# t$

shows $eval_trm\ (finfun_update\ e\ k\ x)\ t =$
 $eval_trm\ (finfun_update\ e\ k'\ x)\ ((k' \leftrightarrow k) \cdot t)$

⟨proof⟩

lemma $eval_fmla_rename$:

assumes $atom\ k' \# A$

shows $eval_fmla\ (finfun_update\ e\ k\ x)\ A = eval_fmla\ (finfun_update\ e\ k'\ x)\ ((k' \leftrightarrow k) \cdot A)$

⟨proof⟩

lemma $better_ex_eval_fmla[simp]$:

$eval_fmla\ e\ (exi\ k\ A) \longleftrightarrow (\exists x. eval_fmla\ (finfun_update\ e\ k\ x)\ A)$

⟨proof⟩

lemma $forget_eval_trm\ [simp]$: $atom\ i \# t \Longrightarrow$

$eval_trm\ (finfun_update\ e\ i\ x)\ t = eval_trm\ e\ t$

⟨proof⟩

lemma $forget_eval_fmla\ [simp]$:

$atom\ k \# A \Longrightarrow eval_fmla\ (finfun_update\ e\ k\ x)\ A = eval_fmla\ e\ A$

⟨proof⟩

lemma $eval_subst_trm$: $eval_trm\ e\ (subst\ i\ t\ u) =$

$eval_trm\ (finfun_update\ e\ i\ (eval_trm\ e\ t))\ u$

⟨proof⟩

lemma $eval_subst_fmla$: $eval_fmla\ e\ (fmla(i ::= t)) =$

$eval_fmla\ (finfun_update\ e\ i\ (eval_trm\ e\ t))\ fmla$

⟨proof⟩

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*]: $p \cdot (A\ AND\ B) = (p \cdot A)\ AND\ (p \cdot B)$
(*proof*)

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

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

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

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

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

lemma $eval_fmla_cnj$ [*simp*]: $eval_fmla\ e\ (cnj\ A\ B) \longleftrightarrow (eval_fmla\ e\ A \wedge eval_fmla\ e\ B)$
(*proof*)

1.4.2 If and only if

definition $Iff :: fmla \Rightarrow fmla \Rightarrow fmla$ (**infixr** *IFF* 125)
where $Iff\ A\ B = cnj\ (imp\ A\ B)\ (imp\ B\ A)$

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 $subst_fmla_Iff$ [*simp*]: $(A\ IFF\ B)(i::=x) = (A(i::=x))\ IFF\ (B(i::=x))$
(*proof*)

lemma $eval_fmla_Iff$ [*simp*]: $eval_fmla\ e\ (Iff\ A\ B) \longleftrightarrow (eval_fmla\ e\ A \longleftrightarrow eval_fmla\ e\ B)$
(*proof*)

1.4.3 False

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

lemma fls_eqvt [*eqvt*]: $(p \cdot fls) = fls$
(*proof*)

lemma fls_fresh [*simp*]: $a \# fls$
(*proof*)

2 Axioms and Theorems

2.1 Logical axioms

inductive_set *boolean_axioms* :: *fmla 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*: $A \in \text{boolean_axioms} \implies \text{eval_fmla } e \ A$
(*proof*)

inductive_set *special_axioms* :: *fmla set* **where**

I: $A(i ::= x) \text{ IMP } (\text{exi } i \ A) \in \text{special_axioms}$

lemma *special_axioms_hold*: $A \in \text{special_axioms} \implies \text{eval_fmla } e \ A$
(*proof*)

lemma *twist_forget_eval_fmla* [*simp*]:

atom j $\# (i, A)$
 $\implies \text{eval_fmla } (\text{finfun_update } (\text{finfun_update } (\text{finfun_update } e \ i \ x) \ j \ y) \ i \ z) \ A =$
 $\text{eval_fmla } (\text{finfun_update } e \ i \ z) \ A$
(*proof*)

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**

refl_ax = $\text{Var } X1 \ \text{EQ} \ \text{Var } X1$

lemma *refl_ax_holds*: $\text{eval_fmla } e \ \text{refl_ax}$
(*proof*)

definition *eq_cong_ax* :: *fmla* **where**

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*: *eval_fm1a e eq_cong_ax*
 ⟨*proof*⟩

definition *sync_cong_ax* :: *fm1a* **where**
sync_cong_ax = ((*Var X1 EQ Var X2*) *IMP*
 ((*suc (Var X1)*) *EQ (suc (Var X2))*))

lemma *sync_cong_ax_holds*: *eval_fm1a e sync_cong_ax*
 ⟨*proof*⟩

definition *pls_cong_ax* :: *fm1a* **where**
pls_cong_ax = ((*Var X1 EQ Var X2*) *AND (Var X3 EQ Var X4)*) *IMP*
 ((*pls (Var X1) (Var X3)*) *EQ (pls (Var X2) (Var X4))*)

lemma *pls_cong_ax_holds*: *eval_fm1a e pls_cong_ax*
 ⟨*proof*⟩

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*
 ⟨*proof*⟩

definition *equality_axioms* :: *fm1a set* **where**
equality_axioms = {*refl_ax*, *eq_cong_ax*, *sync_cong_ax*, *pls_cong_ax*, *tms_cong_ax*}

lemma *equality_axioms_hold*: $A \in \text{equality_axioms} \implies \text{eval_fm1a } e \ A$
 ⟨*proof*⟩

2.4 The Q (Robinson-arithmetic-specific) axioms

definition *Q_axioms* ≡
 {*A* | *A X1 X2*.
X1 ≠ *X2* ∧
 (*A = neg (zer EQ suc (Var X1))*) ∨
A = suc (Var X1) EQ suc (Var X2) IMP Var X1 EQ Var X2 ∨
A = Var X2 EQ zer OR exi X1 (Var X2 EQ suc (Var X1)) ∨
A = pls (Var X1) zer EQ Var X1 ∨
A = pls (Var X1) (suc (Var X2)) EQ suc (pls (Var X1) (Var X2)) ∨
A = tms (Var X1) zer EQ zer ∨
A = tms (Var X1) (suc (Var X2)) EQ pls (tms (Var X1) (Var X2)) (Var X1))}
 }
 }
 }

2.5 The proof system

inductive *nprv* :: *fm1a set* ⇒ *fm1a* ⇒ *bool* (**infixl** † 55)

where

- Hyp*: $A \in H \implies H \vdash A$
- | *Q*: $A \in Q_axioms \implies H \vdash A$
- | *Bool*: $A \in \text{boolean_axioms} \implies H \vdash A$
- | *eql*: $A \in \text{equality_axioms} \implies H \vdash A$
- | *Spec*: $A \in \text{special_axioms} \implies H \vdash A$
- | *MP*: $H \vdash A \text{ IMP } B \implies H' \vdash A \implies H \cup H' \vdash B$
- | *exists*: $H \vdash A \text{ IMP } B \implies \text{atom } i \ \# \ B \implies \forall C \in H. \text{atom } i \ \# \ C \implies H \vdash (\text{exi } i \ A) \text{ IMP } B$

2.6 Derived rules of inference

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

lemma thin_Un: $H \vdash A \implies H \cup H' \vdash A$
<proof>

lemma thin: $H \vdash A \implies H \subseteq H' \implies H' \vdash A$
<proof>

lemma thin0: $\{\} \vdash A \implies H \vdash A$
<proof>

lemma thin1: $H \vdash B \implies \text{insert } A \ H \vdash B$
<proof>

lemma thin2: $\text{insert } A1 \ H \vdash B \implies \text{insert } A1 \ (\text{insert } A2 \ H) \vdash B$
<proof>

lemma thin3: $\text{insert } A1 \ (\text{insert } A2 \ H) \vdash B \implies \text{insert } A1 \ (\text{insert } A2 \ (\text{insert } A3 \ H)) \vdash B$
<proof>

lemma thin4:
 $\text{insert } A1 \ (\text{insert } A2 \ (\text{insert } A3 \ H)) \vdash B$
 $\implies \text{insert } A1 \ (\text{insert } A2 \ (\text{insert } A3 \ (\text{insert } A4 \ H))) \vdash B$
<proof>

lemma rotate2: $\text{insert } A2 \ (\text{insert } A1 \ H) \vdash B \implies \text{insert } A1 \ (\text{insert } A2 \ H) \vdash B$
<proof>

lemma rotate3: $\text{insert } A3 \ (\text{insert } A1 \ (\text{insert } A2 \ H)) \vdash B \implies \text{insert } A1 \ (\text{insert } A2 \ (\text{insert } A3 \ H)) \vdash B$
<proof>

lemma rotate4:
 $\text{insert } A4 \ (\text{insert } A1 \ (\text{insert } A2 \ (\text{insert } A3 \ H))) \vdash B$
 $\implies \text{insert } A1 \ (\text{insert } A2 \ (\text{insert } A3 \ (\text{insert } A4 \ H))) \vdash B$
<proof>

lemma rotate5:
 $\text{insert } A5 \ (\text{insert } A1 \ (\text{insert } A2 \ (\text{insert } A3 \ (\text{insert } A4 \ H)))) \vdash B$
 $\implies \text{insert } A1 \ (\text{insert } A2 \ (\text{insert } A3 \ (\text{insert } A4 \ (\text{insert } A5 \ H)))) \vdash B$
<proof>

lemma rotate6:
 $\text{insert } A6 \ (\text{insert } A1 \ (\text{insert } A2 \ (\text{insert } A3 \ (\text{insert } A4 \ (\text{insert } A5 \ H)))))) \vdash B$
 $\implies \text{insert } A1 \ (\text{insert } A2 \ (\text{insert } A3 \ (\text{insert } A4 \ (\text{insert } A5 \ (\text{insert } A6 \ H)))))) \vdash B$
<proof>

lemma rotate7:
 $\text{insert } A7 \ (\text{insert } A1 \ (\text{insert } A2 \ (\text{insert } A3 \ (\text{insert } A4 \ (\text{insert } A5 \ (\text{insert } A6 \ H)))))) \vdash B$
 $\implies \text{insert } A1 \ (\text{insert } A2 \ (\text{insert } A3 \ (\text{insert } A4 \ (\text{insert } A5 \ (\text{insert } A6 \ (\text{insert } A7 \ H)))))) \vdash B$
<proof>

lemma rotate8:
 $\text{insert } A8 \ (\text{insert } A1 \ (\text{insert } A2 \ (\text{insert } A3 \ (\text{insert } A4 \ (\text{insert } A5 \ (\text{insert } A6 \ (\text{insert } A7 \ H)))))) \vdash B$
 $\implies \text{insert } A1 \ (\text{insert } A2 \ (\text{insert } A3 \ (\text{insert } A4 \ (\text{insert } A5 \ (\text{insert } A6 \ (\text{insert } A7 \ (\text{insert } A8 \ H)))))) \vdash B$
<proof>

lemma rotate9:
 $\text{insert } A9 \ (\text{insert } A1 \ (\text{insert } A2 \ (\text{insert } A3 \ (\text{insert } A4 \ (\text{insert } A5 \ (\text{insert } A6 \ (\text{insert } A7 \ (\text{insert } A8 \ H)))))) \vdash B$

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 *dsjAssoc1*: $H \vdash A \text{ OR } (B \text{ OR } C) \implies H \vdash (A \text{ OR } B) \text{ OR } C$
 ⟨*proof*⟩

lemma *dsjAssoc2*: $H \vdash (A \text{ OR } B) \text{ OR } C \implies H \vdash A \text{ OR } (B \text{ OR } C)$
 ⟨*proof*⟩

lemma *dsj_commute_imp*: $H \vdash (B \text{ OR } A) \text{ IMP } (A \text{ OR } B)$
 ⟨*proof*⟩

lemma *dsj_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*⟩

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*⟩

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*: $\llbracket \{A,B\} \vdash C; H \vdash A; H \vdash B \rrbracket \implies H \vdash C$
(*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*: $\llbracket \{A,B,C\} \vdash D; H \vdash A; H \vdash B; H \vdash C \rrbracket \implies H \vdash D$
(*proof*)

lemma *cut4*: $\llbracket \{A,B,C,D\} \vdash E; H \vdash A; H \vdash B; H \vdash C; H \vdash D \rrbracket \implies H \vdash E$
(*proof*)

3 Miscellaneous Logical Rules

lemma *dsj_I1*: $H \vdash A \implies H \vdash A \text{ OR } B$
(*proof*)

lemma *dsj_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 *dsj_neg_1*: $H \vdash A \text{ OR } B \implies H \vdash \text{neg } B \implies H \vdash A$
(*proof*)

lemma *dsj_neg_2*: $H \vdash A \text{ OR } B \implies H \vdash \text{neg } A \implies H \vdash B$
(*proof*)

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

lemma *cnj_I* [*intro!*]: $H \vdash A \implies H \vdash B \implies H \vdash A \text{ AND } B$
(*proof*)

lemma *cnj_E1*: $H \vdash A \text{ AND } B \implies H \vdash A$
(*proof*)

lemma *cnj_E2*: $H \vdash A \text{ AND } B \implies H \vdash B$

<proof>

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

lemma *cnj_E*: **assumes** $\text{insert } A (\text{insert } B H) \vdash C$ **shows** $\text{insert } (A \text{ AND } B) H \vdash C$
<proof>

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** $(\bigwedge B. \text{atom } i \nmid B \implies \text{insert } A H \vdash B)$ **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 *cnj_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 *dsj_mono*:
assumes $\text{insert } A H \vdash B \text{ insert } C H \vdash D$ **shows** $\text{insert } (A \text{ OR } C) H \vdash B \text{ OR } D$
<proof>

lemma *dsj_E*:
assumes $A: \text{insert } A H \vdash C$ **and** $B: \text{insert } B H \vdash C$ **shows** $\text{insert } (A \text{ OR } B) H \vdash C$
<proof>

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'*: $\text{insert } A H \vdash \text{neg } A \implies H \vdash \text{neg } A$
<proof>

lemma *negneg_E* [intro!]: $\text{insert } A H \vdash B \implies \text{insert } (\text{neg } (\text{neg } A)) H \vdash B$
<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** $B: \text{insert } B H \vdash C$ **shows** $\text{insert } (A \text{ IMP } B) H \vdash C$
<proof>

lemma *imp_cut*:
assumes $\text{insert } C H \vdash A \text{ IMP } B \{A\} \vdash C$
shows $H \vdash A \text{ IMP } B$

<proof>

lemma *Iff_I [intro!]: insert A H ⊢ B ⇒ insert B H ⊢ A ⇒ H ⊢ A IFF B*
<proof>

lemma *Iff_MP_same: H ⊢ A IFF B ⇒ H ⊢ A ⇒ H ⊢ B*
<proof>

lemma *Iff_MP2_same: H ⊢ A IFF B ⇒ H ⊢ B ⇒ H ⊢ A*
<proof>

lemma *Iff_refl [intro!]: H ⊢ A IFF A*
<proof>

lemma *Iff_sym: H ⊢ A IFF B ⇒ H ⊢ B IFF A*
<proof>

lemma *Iff_trans: H ⊢ A IFF B ⇒ H ⊢ B IFF C ⇒ H ⊢ A IFF C*
<proof>

lemma *Iff_E:*
insert A (insert B H) ⊢ C ⇒ insert (neg A) (insert (neg B) H) ⊢ C ⇒ insert (A IFF B) H ⊢ C
<proof>

lemma *Iff_E1:*
assumes *A: H ⊢ A and B: insert B H ⊢ C shows insert (A IFF B) H ⊢ C*
<proof>

lemma *Iff_E2:*
assumes *A: H ⊢ A and B: insert B H ⊢ C shows insert (B IFF A) H ⊢ C*
<proof>

lemma *Iff_MP_left: H ⊢ A IFF B ⇒ insert A H ⊢ C ⇒ insert B H ⊢ C*
<proof>

lemma *Iff_MP_left': H ⊢ A IFF B ⇒ insert B H ⊢ C ⇒ insert A H ⊢ C*
<proof>

lemma *Swap: insert (neg B) H ⊢ A ⇒ insert (neg A) H ⊢ B*
<proof>

lemma *Cases: insert A H ⊢ B ⇒ insert (neg A) H ⊢ B ⇒ H ⊢ B*
<proof>

lemma *neg_cnj_E: H ⊢ B ⇒ insert (neg A) H ⊢ C ⇒ insert (neg (A AND B)) H ⊢ C*
<proof>

lemma *dsj_CI: insert (neg B) H ⊢ A ⇒ H ⊢ A OR B*
<proof>

lemma *dsj_3I: insert (neg A) (insert (neg C) H) ⊢ B ⇒ H ⊢ A OR B OR C*
<proof>

lemma *Contrapos1: H ⊢ A IMP B ⇒ H ⊢ neg B IMP neg A*
<proof>

lemma *Contrapos2: H ⊢ (neg B) IMP (neg A) ⇒ H ⊢ A IMP B*
<proof>

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

lemma *ContraAssume*: $\text{neg } B \in H \implies \text{insert } B H \vdash A$
 ⟨proof⟩

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

lemma *dsj_IE1*: $\text{insert } B H \vdash C \implies \text{insert } (A \text{ OR } B) H \vdash A \text{ OR } C$
 ⟨proof⟩

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*: $H \vdash A(i::=x) \implies H \vdash \text{exi } i A$
 ⟨proof⟩

lemma *exi_E*:
assumes $\text{insert } A H \vdash B \text{ atom } i \# B \forall C \in H. \text{ atom } i \# C$
shows $\text{insert } (\text{exi } i A) H \vdash B$
 ⟨proof⟩

lemma *exi_E_with_renaming*:
assumes $\text{insert } ((i \leftrightarrow i') \cdot A) H \vdash B \text{ atom } i' \# (A, i, B) \forall C \in H. \text{ atom } i' \# C$
shows $\text{insert } (\text{exi } i A) H \vdash B$
 ⟨proof⟩

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*: $\text{insert } A H \vdash B \implies \forall C \in H. \text{ atom } i \# C \implies \text{insert } (\text{exi } i A) H \vdash (\text{exi } i B)$
 ⟨proof⟩

lemma *all_I* [intro!]: $H \vdash A \implies \forall C \in H. \text{ atom } i \# C \implies H \vdash \text{all } i A$
 ⟨proof⟩

lemma *all_D*: $H \vdash \text{all } i A \implies H \vdash A(i::=x)$
 ⟨proof⟩

lemma *all_E*: $\text{insert } (A(i::=x)) H \vdash B \implies \text{insert } (\text{all } i A) H \vdash B$
 ⟨proof⟩

lemma *all_E'*: $H \vdash \text{all } i A \implies \text{insert } (A(i::=x)) H \vdash B \implies H \vdash B$
 ⟨proof⟩

3.2 Congruence rules

lemma *neg_cong*: $H \vdash A \text{ IFF } A' \implies H \vdash \text{neg } A \text{ IFF } \text{neg } A'$
 ⟨proof⟩

lemma *dsj_cong*: $H \vdash A \text{ IFF } A' \implies H \vdash B \text{ IFF } B' \implies H \vdash A \text{ OR } B \text{ IFF } A' \text{ OR } B'$
 ⟨proof⟩

lemma *cnj_cong*: $H \vdash A \text{ IFF } A' \implies H \vdash B \text{ IFF } B' \implies H \vdash A \text{ AND } B \text{ IFF } A' \text{ AND } B'$
 ⟨proof⟩

lemma *imp_cong*: $H \vdash A \text{ IFF } A' \implies H \vdash B \text{ IFF } B' \implies H \vdash (A \text{ IMP } B) \text{ IFF } (A' \text{ IMP } B')$
 ⟨proof⟩

lemma *Iff_cong*: $H \vdash A \text{ IFF } A' \implies H \vdash B \text{ IFF } B' \implies H \vdash (A \text{ IFF } B) \text{ IFF } (A' \text{ IFF } B')$
 ⟨proof⟩

lemma *exi_cong*: $H \vdash A \text{ IFF } A' \implies \forall C \in H. \text{atom } i \nmid C \implies H \vdash (\text{exi } i A) \text{ IFF } (\text{exi } i A')$
 ⟨proof⟩

lemma *all_cong*: $H \vdash A \text{ IFF } A' \implies \forall C \in H. \text{atom } i \nmid C \implies H \vdash (\text{all } i A) \text{ IFF } (\text{all } i A')$
 ⟨proof⟩

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

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⟩

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 *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⟩

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

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⟩

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

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

<proof>

lemma *pls_cong*: $\llbracket H \vdash t \text{ EQ } t'; H \vdash u \text{ EQ } u' \rrbracket \implies H \vdash \text{pls } t \ u \text{ EQ } \text{pls } t' \ u'$
<proof>

lemma *tms_cong1*: $\{\} \vdash (t \text{ EQ } t' \text{ AND } u \text{ EQ } u') \text{ IMP } (tms \ t \ u \text{ EQ } tms \ t' \ u')$
<proof>

lemma *tms_cong*: $\llbracket H \vdash t \text{ EQ } t'; H \vdash u \text{ EQ } u' \rrbracket \implies H \vdash tms \ t \ u \text{ EQ } tms \ t' \ u'$
<proof>

4.3 Substitution for equalities

lemma *eql_subst_trm_Iff*: $\{t \text{ EQ } u\} \vdash \text{subst } i \ t \ \text{trm} \ \text{EQ} \ \text{subst } i \ u \ \text{trm}$
<proof>

lemma *eql_subst_fm1a_Iff*: $\text{insert } (t \text{ EQ } u) \ H \vdash A(i::=t) \text{ IFF } A(i::=u)$
<proof>

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

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

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 $H \vdash x \ \text{EQ} \ x'$
shows $H \vdash P \ x \text{ IFF } P \ x'$
<proof>

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*: $H \vdash x \ \text{EQ} \ x' \ H \vdash y \ \text{EQ} \ y'$
shows $H \vdash P \ x \ y \text{ IFF } P \ x' \ y'$
<proof>

lemma *P3_cong*:
fixes *tms* :: *trm list*
assumes *sub*: $\bigwedge i \ t \ u \ v \ x. \text{atom } i \ \#\ \text{tms} \implies$
 $(P \ t \ u \ v)(i::=x) = P(\text{subst } i \ x \ t) (\text{subst } i \ x \ u) (\text{subst } i \ x \ v)$
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* :: *trm list*
assumes *sub*: $\bigwedge i \ t1 \ t2 \ t3 \ t4 \ x. \text{atom } i \ \#\ \text{tms} \implies$
 $(P \ t1 \ t2 \ t3 \ t4)(i::=x) = P(\text{subst } i \ x \ t1) (\text{subst } i \ x \ t2) (\text{subst } i \ x \ t3) (\text{subst } i \ x \ t4)$
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>

4.5 The formula *fls*

lemma *neg_I [intro!]*: $\text{insert } A \ H \vdash \text{fls} \implies H \vdash \text{neg } A$

<proof>

lemma *neg_E* [intro!]: $H \vdash A \implies \text{insert } (\text{neg } A) H \vdash \text{fls}$
<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!]

lemma *neg_imp_I* [intro!]: $H \vdash A \implies \text{insert } B H \vdash \text{fls} \implies H \vdash \text{neg } (A \text{ IMP } B)$
<proof>

lemma *neg_imp_E* [intro!]: $\text{insert } (\text{neg } B) (\text{insert } A H) \vdash C \implies \text{insert } (\text{neg } (A \text{ IMP } B)) H \vdash C$
<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!]: $\text{insert } \text{fls } H \vdash A$
<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*: $H \vdash (\text{neg } \text{fls})$
<proof>

lemma *exFalso*: $H \vdash \text{fls} \implies H \vdash A$
<proof>

Soundness of the provability relation

theorem *nprv_sound*: **assumes** $H \vdash A$ **shows** $(\forall B \in H. \text{eval_fmla } e B) \implies \text{eval_fmla } e A$
<proof>

5 Instantiation of Syntax-Independent Logic Infrastructure

5.1 Preliminaries

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

definition *ground_aux* :: *trm* \Rightarrow *atom set* \Rightarrow *bool*
where *ground_aux* *t S* $\equiv (\text{supp } t \subseteq S)$

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 $ground_aux_simps[simp]$:

$ground_aux\ zer\ S = True$
 $ground_aux\ (Var\ k)\ S = (if\ atom\ k \in S\ then\ True\ else\ False)$
 $ground_aux\ (suc\ t)\ S = (ground_aux\ t\ S)$
 $ground_aux\ (pls\ t\ u)\ S = (ground_aux\ t\ S \wedge ground_aux\ u\ S)$
 $ground_aux\ (tms\ t\ u)\ S = (ground_aux\ t\ S \wedge ground_aux\ u\ S)$
 $\langle proof \rangle$

lemma $ground_fmla_aux_simps[simp]$:

$ground_fmla_aux\ fls\ S = True$
 $ground_fmla_aux\ (t\ EQ\ u)\ S = (ground_aux\ t\ S \wedge ground_aux\ u\ S)$
 $ground_fmla_aux\ (A\ OR\ B)\ S = (ground_fmla_aux\ A\ S \wedge ground_fmla_aux\ B\ S)$
 $ground_fmla_aux\ (A\ AND\ B)\ S = (ground_fmla_aux\ A\ S \wedge ground_fmla_aux\ B\ S)$
 $ground_fmla_aux\ (A\ IFF\ B)\ S = (ground_fmla_aux\ A\ S \wedge ground_fmla_aux\ B\ S)$
 $ground_fmla_aux\ (neg\ A)\ S = (ground_fmla_aux\ A\ S)$
 $ground_fmla_aux\ (exi\ x\ A)\ S = (ground_fmla_aux\ A\ (S \cup \{atom\ x\}))$
 $\langle proof \rangle$

lemma $ground_fresh[simp]$:

$ground\ t \Longrightarrow atom\ i\ \# t$
 $ground_fmla\ A \Longrightarrow atom\ i\ \# A$
 $\langle proof \rangle$

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

lemma $Fvars_trm_simps[simp]$:

$Fvars\ zer = \{\}$
 $Fvars\ (Var\ a) = \{a\}$
 $Fvars\ (suc\ x) = Fvars\ x$
 $Fvars\ (pls\ x\ y) = Fvars\ x \cup Fvars\ y$
 $Fvars\ (tms\ x\ y) = Fvars\ x \cup Fvars\ y$
 $\langle proof \rangle$

lemma $finite_Fvars_trm[simp]$:

fixes $t :: trm$
shows $finite\ (Fvars\ t)$
 $\langle proof \rangle$

lemma $Fvars_fmla_simps[simp]$:

$Fvars\ (x\ EQ\ y) = Fvars\ x \cup Fvars\ y$
 $Fvars\ (A\ OR\ B) = Fvars\ A \cup Fvars\ B$
 $Fvars\ (A\ AND\ B) = Fvars\ A \cup Fvars\ B$
 $Fvars\ (A\ IMP\ B) = Fvars\ A \cup Fvars\ B$
 $Fvars\ fls = \{\}$
 $Fvars\ (neg\ A) = Fvars\ A$
 $Fvars\ (exi\ a\ A) = Fvars\ A - \{a\}$
 $Fvars\ (all\ a\ A) = Fvars\ A - \{a\}$

<proof>

lemma *finite_Fvars_fm1a[simp]*:

fixes $A :: \text{fm1a}$

shows *finite* ($Fvars\ A$)

<proof>

lemma *subst_trm_subst_trm[simp]*:

$x \neq y \implies \text{atom } x \# u \implies \text{subst } y\ u\ (\text{subst } x\ t\ v) = \text{subst } x\ (\text{subst } y\ u\ t)\ (\text{subst } y\ u\ v)$

<proof>

lemma *subst_fm1a_subst_fm1a[simp]*:

$x \neq y \implies \text{atom } x \# u \implies (A(x::=t))(y::=u) = (A(y::=u))(x::=\text{subst } y\ u\ t)$

<proof>

lemma *Fvars_empty_ground[simp]*: $Fvars\ t = \{\} \implies \text{ground } t$

<proof>

lemma *Fvars_ground_aux*: $Fvars\ t \subseteq B \implies \text{ground_aux } t\ (\text{atom } 'B)$

<proof>

lemma *ground_Fvars*: $\text{ground } t \longleftrightarrow Fvars\ t = \{\}$

<proof>

lemma *Fvars_ground_fm1a_aux*: $Fvars\ A \subseteq B \implies \text{ground_fm1a_aux } A\ (\text{atom } 'B)$

<proof>

lemma *ground_fm1a_Fvars*: $\text{ground_fm1a } A \longleftrightarrow Fvars\ A = \{\}$

<proof>

lemma *obtain_const_trm*:

obtains t **where** $\text{eval_trm } e\ t = x\ t \in \text{num}$

<proof>

lemma *ex_eval_fm1a_iff_exists_num*:

$\text{eval_fm1a } e\ (\text{exi } k\ A) \longleftrightarrow (\exists t. \text{eval_fm1a } e\ (A(k::=t)) \wedge t \in \text{num})$

<proof>

lemma *exi_ren*: $y \notin Fvars\ \varphi \implies \text{exi } x\ \varphi = \text{exi } y\ (\varphi(x::=\text{Var } y))$

<proof>

lemma *all_ren*: $y \notin Fvars\ \varphi \implies \text{all } x\ \varphi = \text{all } y\ (\varphi(x::=\text{Var } y))$

<proof>

lemma *Fvars_num[simp]*: $t \in \text{num} \implies Fvars\ t = \{\}$

<proof>

5.2 Instantiation of the generic syntax and deduction relation

interpretation *Generic_Syntax* **where**

$\text{var} = \text{UNIV} :: \text{name set}$

and $\text{trm} = \text{UNIV} :: \text{trm set}$

and $\text{fm1a} = \text{UNIV} :: \text{fm1a set}$

and $\text{Var} = \text{Var}$

and $\text{FvarsT} = \text{Fvars}$

and $\text{substT} = \lambda t\ u\ x. \text{subst } x\ u\ t$

and $\text{Fvars} = \text{Fvars}$

and $\text{subst} = \lambda A\ u\ x. \text{subst_fm1a } A\ x\ u$

<proof>

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* = $\lambda A u x. \text{subst_fmla } A x u$
<proof>

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* = (\vdash) {}
<proof>

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* = (\vdash) {}
<proof>

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** $tms = tms$
 $\langle proof \rangle$

lemma num_range_Num : $num = range\ Num$
 $\langle proof \rangle$

lemma $[simp]$: $\{\} \vdash neg\ (zer\ EQ\ suc\ (Var\ xx))$
 $\langle proof \rangle$

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

lemma $[simp]$: $\{\} \vdash pls\ (Var\ xx)\ zer\ EQ\ Var\ xx$
 $\langle proof \rangle$

lemma $[simp]$: $\{\} \vdash tms\ (Var\ xx)\ zer\ EQ\ zer$
 $\langle proof \rangle$

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 $substT = \lambda t\ u\ x. subst\ x\ u\ t$
and $Fvars = Fvars$
and $subst = \lambda A\ u\ x. subst_fmla\ A\ x\ u$
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** $tms = tms$
 $\langle proof \rangle$

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 $substT = \lambda t\ u\ x. subst\ x\ u\ t$
and $Fvars = Fvars$
and $subst = \lambda A\ u\ x. subst_fmla\ A\ x\ u$
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 $prv = (\vdash)\ \{\}$
 $\langle proof \rangle$

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 =  $\lambda t\ u\ x.$  subst x u t
and Fvars = Fvars
and subst =  $\lambda A\ u\ x.$  subst_fm1a 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_fm1a e0
<proof>
```