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 Paulsons formalization of the Hereditarily Finite Set theory.

Contents

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* ⇒ *trm* ⇒ *trm* ⇒ *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*]:
$j \sharp subst\ i\ x\ t \longleftrightarrow (atom\ i \sharp t \wedge j \sharp t) \vee (j \sharp x \wedge (j \sharp t \vee j = atom\ i))$
⟨*proof*⟩

lemma *forget_subst_trm* [*simp*]: $atom\ a \sharp trm \Longrightarrow subst\ a\ x\ trm = trm$
⟨*proof*⟩

lemma *subst_trm_id* [*simp*]: $subst\ a\ (Var\ a)\ trm = trm$
⟨*proof*⟩

lemma *subst_trm_commute* [*simp*]:
$atom\ j \sharp trm \Longrightarrow subst\ j\ u\ (subst\ i\ t\ trm) = subst\ i\ (subst\ j\ u\ t)\ trm$
⟨*proof*⟩

lemma *subst_trm_commute2* [*simp*]:
$atom\ j \sharp t \Longrightarrow atom\ i \sharp u \Longrightarrow i \neq j \Longrightarrow subst\ j\ u\ (subst\ i\ t\ trm) = subst\ i\ t\ (subst\ j\ u\ trm)$
⟨*proof*⟩

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

$\langle proof \rangle$

nominal_function $subst\_fmla :: fmla \Rightarrow name \Rightarrow trm \Rightarrow fmla$ $(\_'(\_::=\_')$ $[1000, 0, 0]$ $200)$
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: $atom\ j \sharp (i, x) \Longrightarrow (exi\ j\ A)(i::=x) = exi\ j\ (A(i::=x))$
$\langle proof \rangle$

nominal_termination $(eqvt)$
$\langle proof \rangle$

lemma $size\_subst\_fmla$ $[simp]$: $size\ (A(i::=x)) = size\ A$
$\langle proof \rangle$

lemma $forget\_subst\_fmla$ $[simp]$: $atom\ a \sharp A \Longrightarrow A(a::=x) = A$
$\langle proof \rangle$

lemma $subst\_fmla\_id$ $[simp]$: $A(a::=Var\ a) = A$
$\langle proof \rangle$

lemma $fresh\_subst\_fmla\_if$ $[simp]$:
$j \sharp (A(i::=x)) \longleftrightarrow (atom\ i \sharp A \land j \sharp A) \lor (j \sharp x \land (j \sharp A \lor j = atom\ i))$
$\langle proof \rangle$

lemma $subst\_fmla\_commute$ $[simp]$:
$atom\ j \sharp A \Longrightarrow (A(i::=t))(j::=u) = A(i ::= subst\ j\ u\ t)$
$\langle proof \rangle$

lemma $repeat\_subst\_fmla$ $[simp]$: $(A(i::=t))(i::=u) = A(i ::= subst\ i\ u\ t)$
$\langle proof \rangle$

lemma $subst\_fmla\_exi\_with\_renaming$:
$atom\ i' \sharp (A, i, j, t) \Longrightarrow (exi\ i\ A)(j ::= t) = exi\ i'\ (((i \leftrightarrow i') \cdot A)(j ::= t))$
$\langle proof \rangle$

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

lemma $flip\_subst\_trm$: $atom\ y \sharp t \Longrightarrow (x \leftrightarrow y) \cdot t = subst\ x\ (Var\ y)\ t$
$\langle proof \rangle$

lemma $flip\_subst\_fmla$: $atom\ y \sharp \varphi \Longrightarrow (x \leftrightarrow y) \cdot \varphi = \varphi(x::=Var\ y)$
$\langle proof \rangle$

lemma $exi\_ren\_subst\_fresh$: $atom\ y \sharp \varphi \Longrightarrow exi\ x\ \varphi = exi\ y\ (\varphi(x::=Var\ y))$
$\langle proof \rangle$

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*) *finfun* ⇒ *fmla* ⇒ *bool*
where
eval_fmla e (*t EQ u*) ⟷ *eval_trm e t* = *eval_trm e u*
| *eval_fmla e* (*A OR B*) ⟷ *eval_fmla e A* ∨ *eval_fmla e B*
| *eval_fmla e* (*neg A*) ⟷ (~ *eval_fmla e A*)
| *atom k* ♯ *e* ⟹ *eval_fmla e* (*exi k A*) ⟷ (∃ *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′* ↔ *k*) · *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′* ↔ *k*) · *A*)
⟨*proof*⟩

lemma *better_ex_eval_fmla*[*simp*]:
eval_fmla e (*exi k A*) ⟷ (∃ *x*. *eval_fmla* (*finfun_update e k x*) *A*)
⟨*proof*⟩

lemma *forget_eval_trm* [*simp*]: *atom i* ♯ *t* ⟹
eval_trm (*finfun_update e i x*) *t* = *eval_trm e t*
⟨*proof*⟩

lemma *forget_eval_fmla* [*simp*]:
atom k ♯ *A* ⟹ *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* ⇒ *fmla* ⇒ *fmla* (**infixr** *IMP 125*)
where *imp A B* ≡ *dsj* (*neg A*) *B*

abbreviation *all* :: *name* ⇒ *fmla* ⇒ *fmla*
where *all i A* ≡ *neg* (*exi i* (*neg A*))

1.4.1 Conjunction

definition *cnj* :: *fmla* ⇒ *fmla* ⇒ *fmla* (**infixr** *AND 135*)
  where *cnj A B* ≡ *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*]: $a \sharp A\ AND\ B \longleftrightarrow (a \sharp A \wedge a \sharp B)$
  ⟨*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*) ⟷ (*eval_fmla e A* ∧ *eval_fmla e B*)
  ⟨*proof*⟩

1.4.2 If and only if

definition *Iff* :: *fmla* ⇒ *fmla* ⇒ *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*]: $a \sharp A\ IFF\ B \longleftrightarrow (a \sharp A \wedge a \sharp B)$
  ⟨*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*) ⟷ (*eval_fmla e A* ⟷ *eval_fmla e B*)
  ⟨*proof*⟩

1.4.3 False

definition *fls* **where** *fls* ≡ *neg* (*zer EQ zer*)

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

lemma *fls_fresh* [*simp*]: $a \sharp fls$
  ⟨*proof*⟩

2 Axioms and Theorems

2.1 Logical axioms

inductive_set *boolean_axioms* :: *fmla set*
where
Ident: *A IMP A* ∈ *boolean_axioms*
| *dsjI1*: *A IMP* (*A OR B*) ∈ *boolean_axioms*
| *dsjCont*: (*A OR A*) *IMP A* ∈ *boolean_axioms*
| *dsjAssoc*: (*A OR* (*B OR C*)) *IMP* ((*A OR B*) *OR C*) ∈ *boolean_axioms*
| *dsjcnj*: (*C OR A*) *IMP* (((*neg C*) *OR B*) *IMP* (*A OR B*)) ∈ *boolean_axioms*

lemma *boolean_axioms_hold*: *A* ∈ *boolean_axioms* ⟹ *eval_fmla e A*
⟨*proof*⟩

inductive_set *special_axioms* :: *fmla set* **where**
I: *A*(*i*::=*x*) *IMP* (*exi i A*) ∈ *special_axioms*

lemma *special_axioms_hold*: *A* ∈ *special_axioms* ⟹ *eval_fmla e A*
⟨*proof*⟩

lemma *twist_forget_eval_fmla* [*simp*]:
atom j ♯ (*i*, *A*)
⟹ *eval_fmla* (*finfun_update* (*finfun_update* (*finfun_update e i x*) *j y*) *i z*) *A* =
eval_fmla (*finfun_update e i z*) *A*
⟨*proof*⟩

2.2 Concrete variables

declare *Abs_name_inject*[*simp*]

abbreviation
X0 ≡ *Abs_name* (*Atom* (*Sort* ''*Theory_Syntax_Q.name*'' []) *0*)

abbreviation
X1 ≡ *Abs_name* (*Atom* (*Sort* ''*Robinson_Arithmetic.name*'' []) (*Suc 0*))
— We prefer *Suc 0* because simplification will transform 1 to that form anyway.

abbreviation
X2 ≡ *Abs_name* (*Atom* (*Sort* ''*Robinson_Arithmetic.name*'' []) *2*)

abbreviation
X3 ≡ *Abs_name* (*Atom* (*Sort* ''*Robinson_Arithmetic.name*'' []) *3*)

abbreviation
X4 ≡ *Abs_name* (*Atom* (*Sort* ''*Robinson_Arithmetic.name*'' []) *4*)

2.3 Equality axioms

definition *refl_ax* :: *fmla* **where**
refl_ax = *Var X1 EQ Var X1*

lemma *refl_ax_holds*: *eval_fmla e refl_ax*
⟨*proof*⟩

definition *eq_cong_ax* :: *fmla* **where**
eq_cong_ax = ((*Var X1 EQ Var X2*) *AND* (*Var X3 EQ Var X4*)) *IMP*
((*Var X1 EQ Var X3*) *IMP* (*Var X2 EQ Var X4*))

lemma *eq_cong_ax_holds*: *eval_fmla e eq_cong_ax*
⟨*proof*⟩

definition *syc_cong_ax* :: *fmla* **where**
syc_cong_ax = ((*Var X1 EQ Var X2*)) *IMP*
((*suc* (*Var X1*)) *EQ* (*suc* (*Var X2*)))

lemma *syc_cong_ax_holds*: *eval_fmla e syc_cong_ax*
⟨*proof*⟩

definition *pls_cong_ax* :: *fmla* **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_fmla e pls_cong_ax*
⟨*proof*⟩

definition *tms_cong_ax* :: *fmla* **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_fmla e tms_cong_ax*
⟨*proof*⟩

definition *equality_axioms* :: *fmla set* **where**
equality_axioms = {*refl_ax*, *eq_cong_ax*, *syc_cong_ax*, *pls_cong_ax*, *tms_cong_ax*}

lemma *equality_axioms_hold*: *A* ∈ *equality_axioms* ⟹ *eval_fmla 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* :: *fmla set* ⇒ *fmla* ⇒ *bool* (**infixl** ⊢ *55*)
where
Hyp: *A* ∈ *H* ⟹ *H* ⊢ *A*
| *Q*: *A* ∈ *Q_axioms* ⟹ *H* ⊢ *A*
| *Bool*: *A* ∈ *boolean_axioms* ⟹ *H* ⊢ *A*
| *eql*: *A* ∈ *equality_axioms* ⟹ *H* ⊢ *A*
| *Spec*: *A* ∈ *special_axioms* ⟹ *H* ⊢ *A*
| *MP*: *H* ⊢ *A IMP B* ⟹ *H'* ⊢ *A* ⟹ *H* ∪ *H'* ⊢ *B*
| *exiists*: *H* ⊢ *A IMP B* ⟹ *atom i* ♯ *B* ⟹ ∀ *C* ∈ *H*. *atom i* ♯ *C* ⟹ *H* ⊢ (*exi i A*) *IMP B*

2.6 Derived rules of inference

lemma *contraction*: *insert A* (*insert A H*) ⊢ *B* ⟹ *insert A H* ⊢ *B*
⟨*proof*⟩

lemma *thin_Un*: $H \vdash A \Longrightarrow H \cup H' \vdash A$
⟨*proof*⟩

lemma *thin*: $H \vdash A \Longrightarrow H \subseteq H' \Longrightarrow H' \vdash A$
⟨*proof*⟩

lemma *thin0*: $\{\} \vdash A \Longrightarrow H \vdash A$
⟨*proof*⟩

lemma *thin1*: $H \vdash B \Longrightarrow$ *insert* A $H \vdash B$
⟨*proof*⟩

lemma *thin2*: *insert A1 H* $\vdash B \Longrightarrow$ *insert A1* (*insert A2 H*) $\vdash B$
⟨*proof*⟩

lemma *thin3*: *insert A1* (*insert A2 H*) $\vdash B \Longrightarrow$ *insert A1* (*insert A2* (*insert A3 H*)) $\vdash B$
⟨*proof*⟩

lemma *thin4*:
insert A1 (*insert A2* (*insert A3 H*)) $\vdash B$
$\Longrightarrow$ *insert A1* (*insert A2* (*insert A3* (*insert A4 H*))) $\vdash B$
⟨*proof*⟩

lemma *rotate2*: *insert A2* (*insert A1 H*) $\vdash B \Longrightarrow$ *insert A1* (*insert A2 H*) $\vdash B$
⟨*proof*⟩

lemma *rotate3*: *insert A3* (*insert A1* (*insert A2 H*)) $\vdash B \Longrightarrow$ *insert A1* (*insert A2* (*insert A3 H*)) $\vdash B$
⟨*proof*⟩

lemma *rotate4*:
insert A4 (*insert A1* (*insert A2* (*insert A3 H*))) $\vdash B$
$\Longrightarrow$ *insert A1* (*insert A2* (*insert A3* (*insert A4 H*))) $\vdash B$
⟨*proof*⟩

lemma *rotate5*:
insert A5 (*insert A1* (*insert A2* (*insert A3* (*insert A4 H*)))) $\vdash B$
$\Longrightarrow$ *insert A1* (*insert A2* (*insert A3* (*insert A4* (*insert A5 H*)))) $\vdash B$
⟨*proof*⟩

lemma *rotate6*:
insert A6 (*insert A1* (*insert A2* (*insert A3* (*insert A4* (*insert A5 H*))))) $\vdash B$
$\Longrightarrow$ *insert A1* (*insert A2* (*insert A3* (*insert A4* (*insert A5* (*insert A6 H*))))) $\vdash B$
⟨*proof*⟩

lemma *rotate7*:
insert A7 (*insert A1* (*insert A2* (*insert A3* (*insert A4* (*insert A5* (*insert A6 H*)))))) $\vdash B$
$\Longrightarrow$ *insert A1* (*insert A2* (*insert A3* (*insert A4* (*insert A5* (*insert A6* (*insert A7 H*)))))) $\vdash B$
⟨*proof*⟩

lemma *rotate8*:
insert A8 (*insert A1* (*insert A2* (*insert A3* (*insert A4* (*insert A5* (*insert A6* (*insert A7 H*))))))) $\vdash B$
$\Longrightarrow$ *insert A1* (*insert A2* (*insert A3* (*insert A4* (*insert A5* (*insert A6* (*insert A7* (*insert A8 H*)))))))
$\vdash B$
⟨*proof*⟩

lemma *rotate9*:
insert A9 (*insert A1* (*insert A2* (*insert A3* (*insert A4* (*insert A5* (*insert A6* (*insert A7* (*insert A8*

$H))))))))) \vdash B$
$\Longrightarrow$ *insert A1* (*insert A2* (*insert A3* (*insert A4* (*insert A5* (*insert A6* (*insert A7* (*insert A8* (*insert A9 H*)))))))) $\vdash B$
⟨*proof*⟩

lemma *rotate10*:
insert A10 (*insert A1* (*insert A2* (*insert A3* (*insert A4* (*insert A5* (*insert A6* (*insert A7* (*insert A8* (*insert A9 H*))))))))) $\vdash B$
$\Longrightarrow$ *insert A1* (*insert A2* (*insert A3* (*insert A4* (*insert A5* (*insert A6* (*insert A7* (*insert A8* (*insert A9* (*insert A10 H*))))))))) $\vdash B$
⟨*proof*⟩

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*)))))))))) $\vdash B$
$\Longrightarrow$ *insert A1* (*insert A2* (*insert A3* (*insert A4* (*insert A5* (*insert A6* (*insert A7* (*insert A8* (*insert A9* (*insert A10* (*insert A11 H*)))))))))) $\vdash B$
⟨*proof*⟩

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*))))))))))) $\vdash B$
$\Longrightarrow$ *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*))))))))))) $\vdash B$
⟨*proof*⟩

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

lemma *rotate14*:
insert A14 (*insert A1* (*insert A2* (*insert A3* (*insert A4* (*insert A5* (*insert A6* (*insert A7* (*insert A8* (*insert A9* (*insert A10* (*insert A11* (*insert A12* (*insert A13 H*))))))))))))) $\vdash B$
$\Longrightarrow$ *insert A1* (*insert A2* (*insert A3* (*insert A4* (*insert A5* (*insert A6* (*insert A7* (*insert A8* (*insert A9* (*insert A10* (*insert A11* (*insert A12* (*insert A13* (*insert A14 H*))))))))))))) $\vdash B$
⟨*proof*⟩

lemma *rotate15*:
insert A15 (*insert A1* (*insert A2* (*insert A3* (*insert A4* (*insert A5* (*insert A6* (*insert A7* (*insert A8* (*insert A9* (*insert A10* (*insert A11* (*insert A12* (*insert A13* (*insert A14 H*)))))))))))))) $\vdash B$
$\Longrightarrow$ *insert A1* (*insert A2* (*insert A3* (*insert A4* (*insert A5* (*insert A6* (*insert A7* (*insert A8* (*insert A9* (*insert A10* (*insert A11* (*insert A12* (*insert A13* (*insert A14* (*insert A15 H*)))))))))))))) $\vdash B$
⟨*proof*⟩

lemma *MP_same*: $H \vdash A$ *IMP* $B \Longrightarrow H \vdash A \Longrightarrow H \vdash B$
⟨*proof*⟩

lemma *MP_thin*: $HA \vdash A$ *IMP* $B \Longrightarrow HB \vdash A \Longrightarrow HA \cup HB \subseteq H \Longrightarrow H \vdash B$
⟨*proof*⟩

lemma *MP_null*: $\{\} \vdash A$ *IMP* $B \Longrightarrow H \vdash A \Longrightarrow H \vdash B$
⟨*proof*⟩

lemma *dsj_commute*: $H \vdash B$ *OR* $A \Longrightarrow H \vdash A$ *OR* B
⟨*proof*⟩

lemma *S*: **assumes** $H \vdash A\ IMP\ (B\ IMP\ C)$ $H' \vdash A\ IMP\ B$ **shows** $H \cup H' \vdash A\ IMP\ C$
⟨*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*: $H \vdash B \Longrightarrow H \vdash A\ IMP\ B$
⟨*proof*⟩

lemma *dsjAssoc1*: $H \vdash A\ OR\ (B\ OR\ C) \Longrightarrow H \vdash (A\ OR\ B)\ OR\ C$
⟨*proof*⟩

lemma *dsjAssoc2*: $H \vdash (A\ OR\ B)\ OR\ C \Longrightarrow H \vdash A\ OR\ (B\ OR\ C)$
⟨*proof*⟩

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

lemma *dsj_Semicong_1*: $H \vdash A\ OR\ C \Longrightarrow H \vdash A\ IMP\ B \Longrightarrow H \vdash B\ OR\ C$
⟨*proof*⟩

lemma *imp_imp_commute*: $H \vdash B\ IMP\ (A\ IMP\ C) \Longrightarrow H \vdash A\ IMP\ (B\ IMP\ C)$
⟨*proof*⟩

2.7 The deduction theorem

lemma *deduction_Diff*: **assumes** $H \vdash B$ **shows** $H - \{C\} \vdash C\ IMP\ B$
⟨*proof*⟩

theorem *imp_I* [*intro*!]: $insert\ A\ H \vdash B \Longrightarrow H \vdash A\ IMP\ B$
⟨*proof*⟩

lemma *anti_deduction*: $H \vdash A\ IMP\ B \Longrightarrow insert\ A\ H \vdash B$
⟨*proof*⟩

2.8 Cut rules

lemma *cut*: $H \vdash A \Longrightarrow insert\ A\ H' \vdash B \Longrightarrow H \cup H' \vdash B$
⟨*proof*⟩

lemma *cut_same*: $H \vdash A \Longrightarrow insert\ A\ H \vdash B \Longrightarrow H \vdash B$
⟨*proof*⟩

lemma *cut_thin*: $HA \vdash A \Longrightarrow insert\ A\ HB \vdash B \Longrightarrow HA \cup HB \subseteq H \Longrightarrow H \vdash B$
⟨*proof*⟩

lemma *cut0*: $\{\} \vdash A \Longrightarrow insert\ A\ H \vdash B \Longrightarrow H \vdash B$
⟨*proof*⟩

lemma *cut1*: $\{A\} \vdash B \Longrightarrow H \vdash A \Longrightarrow H \vdash B$
⟨*proof*⟩

lemma *rcut1*: $\{A\} \vdash B \Longrightarrow insert\ B\ H \vdash C \Longrightarrow insert\ A\ H \vdash C$
⟨*proof*⟩

lemma *cut2*: $[\![\{A,B\} \vdash C;\ H \vdash A;\ H \vdash B]\!] \Longrightarrow H \vdash C$
⟨*proof*⟩

lemma *rcut2*: $\{A,B\} \vdash C \Longrightarrow insert\ C\ H \vdash D \Longrightarrow H \vdash B \Longrightarrow insert\ A\ H \vdash D$
⟨*proof*⟩

lemma *cut3*: $[\![\{A,B,C\} \vdash D;\ H \vdash A;\ H \vdash B;\ H \vdash C]\!] \Longrightarrow H \vdash D$
⟨*proof*⟩

lemma *cut4*: $[\![\{A,B,C,D\} \vdash E;\ H \vdash A;\ H \vdash B;\ H \vdash C;\ H \vdash D]\!] \Longrightarrow H \vdash E$
⟨*proof*⟩

3 Miscellaneous Logical Rules

lemma *dsj_I1*: $H \vdash A \Longrightarrow H \vdash A\ OR\ B$
⟨*proof*⟩

lemma *dsj_I2*: $H \vdash B \Longrightarrow H \vdash A\ OR\ B$
⟨*proof*⟩

lemma *Peirce*: $H \vdash (neg\ A)\ IMP\ A \Longrightarrow H \vdash A$
⟨*proof*⟩

lemma *Contra*: $insert\ (neg\ A)\ H \vdash A \Longrightarrow H \vdash A$
⟨*proof*⟩

lemma *imp_neg_I*: $H \vdash A\ IMP\ B \Longrightarrow H \vdash A\ IMP\ (neg\ B) \Longrightarrow H \vdash neg\ A$
⟨*proof*⟩

lemma *negneg_I*: $H \vdash A \Longrightarrow H \vdash neg\ (neg\ A)$
⟨*proof*⟩

lemma *negneg_D*: $H \vdash neg\ (neg\ A) \Longrightarrow H \vdash A$
⟨*proof*⟩

lemma *neg_D*: $H \vdash neg\ A \Longrightarrow H \vdash A \Longrightarrow H \vdash B$
⟨*proof*⟩

lemma *dsj_neg_1*: $H \vdash A\ OR\ B \Longrightarrow H \vdash neg\ B \Longrightarrow H \vdash A$
⟨*proof*⟩

lemma *dsj_neg_2*: $H \vdash A\ OR\ B \Longrightarrow H \vdash neg\ A \Longrightarrow H \vdash B$
⟨*proof*⟩

lemma *neg_dsj_I*: $H \vdash neg\ A \Longrightarrow H \vdash neg\ B \Longrightarrow H \vdash neg\ (A\ OR\ B)$
⟨*proof*⟩

lemma *cnj_I* [*intro*!]: $H \vdash A \Longrightarrow H \vdash B \Longrightarrow H \vdash A\ AND\ B$
⟨*proof*⟩

lemma *cnj_E1*: $H \vdash A\ AND\ B \Longrightarrow H \vdash A$
⟨*proof*⟩

lemma *cnj_E2*: $H \vdash A\ AND\ B \Longrightarrow H \vdash B$

⟨*proof*⟩

lemma *cnj_commute*: $H \vdash B$ *AND* $A \Longrightarrow H \vdash A$ *AND* B
⟨*proof*⟩

lemma *cnj_E*: **assumes** *insert* A (*insert* B H) $\vdash C$ **shows** *insert* (A *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** ($\bigwedge B$. *atom* i ♯ $B \Longrightarrow$ *insert* A $H \vdash B$) **shows** $H \vdash$ *neg* A
⟨*proof*⟩

lemma *neg_mono*: *insert* A $H \vdash B \Longrightarrow$ *insert* (*neg* B) $H \vdash$ *neg* A
⟨*proof*⟩

lemma *cnj_mono*: *insert* A $H \vdash B \Longrightarrow$ *insert* C $H \vdash D \Longrightarrow$ *insert* (A *AND* C) $H \vdash B$ *AND* D
⟨*proof*⟩

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*⟩

lemma *dsj_E*:
assumes A: *insert* A $H \vdash C$ **and** B: *insert* B $H \vdash C$ **shows** *insert* (A *OR* B) $H \vdash C$
⟨*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'*: *insert* A $H \vdash$ *neg* $A \Longrightarrow H \vdash$ *neg* A
⟨*proof*⟩

lemma *negneg_E* [*intro*!]: *insert* A $H \vdash B \Longrightarrow$ *insert* (*neg* (*neg* A)) $H \vdash B$
⟨*proof*⟩

declare *negneg_E* [*THEN rotate2*, *intro*!]
declare *negneg_E* [*THEN rotate3*, *intro*!]
declare *negneg_E* [*THEN rotate4*, *intro*!]
declare *negneg_E* [*THEN rotate5*, *intro*!]
declare *negneg_E* [*THEN rotate6*, *intro*!]
declare *negneg_E* [*THEN rotate7*, *intro*!]
declare *negneg_E* [*THEN rotate8*, *intro*!]

lemma *imp_E*:
assumes A: $H \vdash A$ **and** B: *insert* B $H \vdash C$ **shows** *insert* (A *IMP* B) $H \vdash C$
⟨*proof*⟩

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

⟨*proof*⟩

lemma *Iff_I* [*intro*!]: *insert A H* ⊢ *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*]: $B \in H \Longrightarrow insert\ (neg\ B)\ H \vdash A$
⟨*proof*⟩

lemma *ContraAssume*: $neg\ B \in H \Longrightarrow insert\ B\ H \vdash A$
⟨*proof*⟩

lemma *ContraProve*: $H \vdash B \Longrightarrow insert\ (neg\ B)\ H \vdash A$
⟨*proof*⟩

lemma *dsj_IE1*: $insert\ B\ H \vdash C \Longrightarrow insert\ (A\ OR\ B)\ H \vdash A\ OR\ C$
⟨*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*: $H \vdash A(i{::=}x) \Longrightarrow H \vdash exi\ i\ A$
⟨*proof*⟩

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

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

lemma *all_I* [*intro*!]: $H \vdash A \Longrightarrow \forall C \in H.\ atom\ i \sharp C \Longrightarrow H \vdash all\ i\ A$
⟨*proof*⟩

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

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

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

3.2 Congruence rules

lemma *neg_cong*: $H \vdash A\ IFF\ A' \Longrightarrow H \vdash neg\ A\ IFF\ neg\ A'$
⟨*proof*⟩

lemma *dsj_cong*: $H \vdash A\ IFF\ A' \Longrightarrow H \vdash B\ IFF\ B' \Longrightarrow H \vdash A\ OR\ B\ IFF\ A'\ OR\ B'$
⟨*proof*⟩

lemma *cnj_cong*: $H \vdash A\ IFF\ A' \Longrightarrow H \vdash B\ IFF\ B' \Longrightarrow H \vdash A\ AND\ B\ IFF\ A'\ AND\ B'$
⟨*proof*⟩

lemma *imp_cong*: $H \vdash A\ IFF\ A' \Longrightarrow H \vdash B\ IFF\ B' \Longrightarrow H \vdash (A\ IMP\ B)\ IFF\ (A'\ IMP\ B')$
⟨*proof*⟩

lemma *Iff_cong*: $H \vdash A\ IFF\ A' \Longrightarrow H \vdash B\ IFF\ B' \Longrightarrow H \vdash (A\ IFF\ B)\ IFF\ (A'\ IFF\ B')$
⟨*proof*⟩

lemma *exi_cong*: $H \vdash A\ IFF\ A' \Longrightarrow \forall\, C \in H.\ atom\ i\ \sharp\ C \Longrightarrow H \vdash (exi\ i\ A)\ IFF\ (exi\ i\ A')$
⟨*proof*⟩

lemma *all_cong*: $H \vdash A\ IFF\ A' \Longrightarrow \forall\, C \in H.\ atom\ i\ \sharp\ C \Longrightarrow H \vdash (all\ i\ A)\ IFF\ (all\ i\ A')$
⟨*proof*⟩

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

4 Eqluality Reasoning

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

lemma *eql_cong1*: $\{\} \vdash (t\ EQ\ t'\ AND\ u\ EQ\ u')\ IMP\ (t\ EQ\ u\ IMP\ t'\ EQ\ u')$
⟨*proof*⟩

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

Apparently necessary in order to prove the congruence property.

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

lemma *Sym_L*: $insert\ (t\ EQ\ u)\ H \vdash A \Longrightarrow insert\ (u\ EQ\ t)\ H \vdash A$
⟨*proof*⟩

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

lemma *eql_cong*:
assumes $H \vdash t\ EQ\ t'$ $H \vdash u\ EQ\ u'$ **shows** $H \vdash t\ EQ\ u\ IFF\ t'\ EQ\ u'$
⟨*proof*⟩

lemma *eql_Trans_E*: $H \vdash x\ EQ\ u \Longrightarrow insert\ (t\ EQ\ u)\ H \vdash A \Longrightarrow insert\ (x\ EQ\ t)\ H \vdash A$
⟨*proof*⟩

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

lemma *suc_cong1*: $\{\} \vdash (t\ EQ\ t')\ IMP\ (suc\ t\ EQ\ suc\ t')$
⟨*proof*⟩

lemma *suc_cong*: $[\![H \vdash t\ EQ\ t']\!] \Longrightarrow H \vdash suc\ t\ EQ\ suc\ t'$
⟨*proof*⟩

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

⟨*proof*⟩

lemma *pls_cong*: $[\![H \vdash t\ EQ\ t'; H \vdash u\ EQ\ u']\!] \Longrightarrow H \vdash pls\ t\ u\ EQ\ pls\ t'\ u'$
⟨*proof*⟩

lemma *tms_cong1*: $\{\} \vdash (t\ EQ\ t'\ AND\ u\ EQ\ u')\ IMP\ (tms\ t\ u\ EQ\ tms\ t'\ u')$
⟨*proof*⟩

lemma *tms_cong*: $[\![H \vdash t\ EQ\ t'; H \vdash u\ EQ\ u']\!] \Longrightarrow H \vdash tms\ t\ u\ EQ\ tms\ t'\ u'$
⟨*proof*⟩

4.3 Substitution for eqlualities

lemma *eql_subst_trm_Iff*: $\{t\ EQ\ u\} \vdash subst\ i\ t\ trm\ EQ\ subst\ i\ u\ trm$
⟨*proof*⟩

lemma *eql_subst_fmla_Iff*: $insert\ (t\ EQ\ u)\ H \vdash A(i{::=}t)\ IFF\ A(i{::=}u)$
⟨*proof*⟩

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

lemma *Var_eql_imp_subst_Iff*: $H \vdash Var\ i\ EQ\ t \Longrightarrow H \vdash A(i{::=}t)\ IFF\ A$
⟨*proof*⟩

4.4 Congruence rules for predicates

lemma *P1_cong*:
fixes $tms :: trm\ list$
assumes $\bigwedge i\ t\ x.\ atom\ i\ \sharp\ tms \Longrightarrow (P\ t)(i{::=}x) = P\ (subst\ i\ x\ t)$ **and** $H \vdash x\ EQ\ x'$
shows $H \vdash P\ x\ IFF\ P\ x'$
⟨*proof*⟩

lemma *P2_cong*:
fixes $tms :: trm\ list$
assumes *sub*: $\bigwedge i\ t\ u\ x.\ atom\ i\ \sharp\ tms \Longrightarrow (P\ t\ u)(i{::=}x) = P\ (subst\ i\ x\ t)\ (subst\ i\ x\ u)$
and *eq*: $H \vdash x\ EQ\ x'\ H \vdash y\ EQ\ y'$
shows $H \vdash P\ x\ y\ IFF\ P\ x'\ y'$
⟨*proof*⟩

lemma *P3_cong*:
fixes $tms :: trm\ list$
assumes *sub*: $\bigwedge i\ t\ u\ v\ x.\ atom\ i\ \sharp\ tms \Longrightarrow$
$(P\ t\ u\ v)(i{::=}x) = P\ (subst\ i\ x\ t)\ (subst\ i\ x\ u)\ (subst\ i\ x\ v)$
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*⟩

lemma *P4_cong*:
fixes $tms :: trm\ list$
assumes *sub*: $\bigwedge i\ t1\ t2\ t3\ t4\ x.\ atom\ i\ \sharp\ tms \Longrightarrow$
$(P\ t1\ t2\ t3\ t4)(i{::=}x) = P\ (subst\ i\ x\ t1)\ (subst\ i\ x\ t2)\ (subst\ i\ x\ t3)\ (subst\ i\ x\ t4)$
and *eq*: $H \vdash x1\ EQ\ x1'\ H \vdash x2\ EQ\ x2'\ H \vdash x3\ EQ\ x3'\ H \vdash x4\ EQ\ x4'$
shows $H \vdash P\ x1\ x2\ x3\ x4\ IFF\ P\ x1'\ x2'\ x3'\ x4'$
⟨*proof*⟩

4.5 The formula *fls*

lemma *neg_I* [*intro*!]: $insert\ A\ H \vdash fls \Longrightarrow H \vdash neg\ A$

⟨*proof*⟩

lemma *neg_E* [*intro*!]: $H \vdash A \Longrightarrow$ *insert* (*neg A*) $H \vdash$ *fls*
⟨*proof*⟩

declare *neg_E* [*THEN rotate2*, *intro*!]
declare *neg_E* [*THEN rotate3*, *intro*!]
declare *neg_E* [*THEN rotate4*, *intro*!]
declare *neg_E* [*THEN rotate5*, *intro*!]
declare *neg_E* [*THEN rotate6*, *intro*!]
declare *neg_E* [*THEN rotate7*, *intro*!]
declare *neg_E* [*THEN rotate8*, *intro*!]

lemma *neg_imp_I* [*intro*!]: $H \vdash A \Longrightarrow$ *insert B* $H \vdash$ *fls* $\Longrightarrow H \vdash$ *neg* (*A IMP B*)
⟨*proof*⟩

lemma *neg_imp_E* [*intro*!]: *insert* (*neg B*) (*insert A H*) $\vdash C \Longrightarrow$ *insert* (*neg* (*A IMP B*)) $H \vdash C$
⟨*proof*⟩

declare *neg_imp_E* [*THEN rotate2*, *intro*!]
declare *neg_imp_E* [*THEN rotate3*, *intro*!]
declare *neg_imp_E* [*THEN rotate4*, *intro*!]
declare *neg_imp_E* [*THEN rotate5*, *intro*!]
declare *neg_imp_E* [*THEN rotate6*, *intro*!]
declare *neg_imp_E* [*THEN rotate7*, *intro*!]
declare *neg_imp_E* [*THEN rotate8*, *intro*!]

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

declare *fls_E* [*THEN rotate2*, *intro*!]
declare *fls_E* [*THEN rotate3*, *intro*!]
declare *fls_E* [*THEN rotate4*, *intro*!]
declare *fls_E* [*THEN rotate5*, *intro*!]
declare *fls_E* [*THEN rotate6*, *intro*!]
declare *fls_E* [*THEN rotate7*, *intro*!]
declare *fls_E* [*THEN rotate8*, *intro*!]

lemma *truth_provable*: $H \vdash$ (*neg fls*)
⟨*proof*⟩

lemma *exFalso*: $H \vdash$ *fls* $\Longrightarrow H \vdash A$
⟨*proof*⟩

Soundness of the provability relation

theorem *nprv_sound*: **assumes** $H \vdash A$ **shows** ($\forall B \in H$. *eval_fmla e B*) $\Longrightarrow$ *eval_fmla e A*
⟨*proof*⟩

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* $\Longrightarrow$ *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* ⇒ *bool*
  where *ground t* ≡ *ground_aux t* {}

definition *ground_fmla_aux* :: *fmla* ⇒ *atom set* ⇒ *bool*
  where *ground_fmla_aux A S* ≡ (*supp A* ⊆ *S*)

abbreviation *ground_fmla* :: *fmla* ⇒ *bool*
  where *ground_fmla A* ≡ *ground_fmla_aux A* {}

lemma *ground_aux_simps*[*simp*]:
  ground_aux zer S = *True*
  ground_aux (*Var k*) *S* = (*if atom k* ∈ *S then True else False*)
  ground_aux (*suc t*) *S* = (*ground_aux t S*)
  ground_aux (*pls t u*) *S* = (*ground_aux t S* ∧ *ground_aux u S*)
  ground_aux (*tms t u*) *S* = (*ground_aux t S* ∧ *ground_aux u S*)
⟨*proof*⟩

lemma *ground_fmla_aux_simps*[*simp*]:
  ground_fmla_aux fls S = *True*
  ground_fmla_aux (*t EQ u*) *S* = (*ground_aux t S* ∧ *ground_aux u S*)
  ground_fmla_aux (*A OR B*) *S* = (*ground_fmla_aux A S* ∧ *ground_fmla_aux B S*)
  ground_fmla_aux (*A AND B*) *S* = (*ground_fmla_aux A S* ∧ *ground_fmla_aux B S*)
  ground_fmla_aux (*A IFF B*) *S* = (*ground_fmla_aux A S* ∧ *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* ∪ {*atom x*}))
⟨*proof*⟩

lemma *ground_fresh*[*simp*]:
  ground t ⟹ *atom i* ♯ *t*
  ground_fmla A ⟹ *atom i* ♯ *A*
⟨*proof*⟩

definition *Fvars t* = {*a* :: *name*. ¬ *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* ∪ *Fvars y*
  Fvars (*tms x y*) = *Fvars x* ∪ *Fvars y*
  ⟨*proof*⟩

lemma *finite_Fvars_trm*[*simp*]:
  fixes *t* :: *trm*
  shows *finite* (*Fvars t*)
  ⟨*proof*⟩

lemma *Fvars_fmla_simps*[*simp*]:
  Fvars (*x EQ y*) = *Fvars x* ∪ *Fvars y*
  Fvars (*A OR B*) = *Fvars A* ∪ *Fvars B*
  Fvars (*A AND B*) = *Fvars A* ∪ *Fvars B*
  Fvars (*A IMP B*) = *Fvars A* ∪ *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_fmla*[*simp*]:
fixes A :: *fmla*
shows *finite* (*Fvars* A)
⟨*proof*⟩

lemma *subst_trm_subst_trm*[*simp*]:
$x \neq y \Longrightarrow$ *atom* $x \sharp u \Longrightarrow$ *subst* y u (*subst* x t v) = *subst* x (*subst* y u t) (*subst* y u v)
⟨*proof*⟩

lemma *subst_fmla_subst_fmla*[*simp*]:
$x \neq y \Longrightarrow$ *atom* $x \sharp u \Longrightarrow (A(x{::=}t))(y{::=}u) = (A(y{::=}u))(x{::=}$*subst* y u $t)$
⟨*proof*⟩

lemma *Fvars_empty_ground*[*simp*]: *Fvars* $t = \{\} \Longrightarrow$ *ground* t
⟨*proof*⟩

lemma *Fvars_ground_aux*: *Fvars* $t \subseteq B \Longrightarrow$ *ground_aux* t (*atom* ‘ B)
⟨*proof*⟩

lemma *ground_Fvars*: *ground* $t \longleftrightarrow$ *Fvars* $t = \{\}$
⟨*proof*⟩

lemma *Fvars_ground_fmla_aux*: *Fvars* $A \subseteq B \Longrightarrow$ *ground_fmla_aux* A (*atom* ‘ B)
⟨*proof*⟩

lemma *ground_fmla_Fvars*: *ground_fmla* $A \longleftrightarrow$ *Fvars* $A = \{\}$
⟨*proof*⟩

lemma *obtain_const_trm*:
obtains t **where** *eval_trm* e $t = x$ $t \in$ *num*
⟨*proof*⟩

lemma *ex_eval_fmla_iff_exists_num*:
eval_fmla e (*exi* k A) $\longleftrightarrow$ ($\exists\, t.$ *eval_fmla* e $(A(k{::=}t)) \wedge t \in$ *num*)
⟨*proof*⟩

lemma *exi_ren*: $y \notin$ *Fvars* $\varphi \Longrightarrow$ *exi* x φ = *exi* y $(\varphi(x{::=}Var\ y))$
⟨*proof*⟩

lemma *all_ren*: $y \notin$ *Fvars* $\varphi \Longrightarrow$ *all* x φ = *all* y $(\varphi(x{::=}Var\ y))$
⟨*proof*⟩

lemma *Fvars_num*[*simp*]: $t \in$ *num* $\Longrightarrow$ *Fvars* $t = \{\}$
⟨*proof*⟩

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* = $\lambda t\ u\ x.$ *subst* x u t
and *Fvars* = *Fvars*
and *subst* = $\lambda A\ u\ x.$ *subst_fmla* 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* = λ*t u x. subst x u t*
and *Fvars* = *Fvars*
and *subst* = λ*A u x. 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* = λ*t u x. subst x u t*
and *Fvars* = *Fvars*
and *subst* = λ*A u x. 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* = λ*t u x. subst x u t*
and *Fvars* = *Fvars*
and *subst* = λ*A u x. 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* = λ*t u x. subst x u t*
and *Fvars* = *Fvars*
and *subst* = λ*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*
⟨*proof*⟩

lemma *num_range_Num*: *num* = *range Num*
⟨*proof*⟩

lemma [*simp*]: {} ⊢ *neg* (*zer EQ suc* (*Var xx*))
⟨*proof*⟩

lemma [*simp*]: {} ⊢ *Var yy EQ zer OR exi xx* (*Var yy EQ suc* (*Var xx*))
⟨*proof*⟩

lemma [*simp*]: {} ⊢ *pls* (*Var xx*) *zer EQ Var xx*
⟨*proof*⟩

lemma [*simp*]: {} ⊢ *tms* (*Var xx*) *zer EQ zer*
⟨*proof*⟩

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* = λ*t u x*. *subst x u t*
and *Fvars* = *Fvars*
and *subst* = λ*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*
⟨*proof*⟩

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* = λ*t u x*. *subst x u t*
and *Fvars* = *Fvars*
and *subst* = λ*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* = (⊢) {}
⟨*proof*⟩

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 $\mathit{FvarsT} = \mathit{Fvars}$
and $\mathit{substT} = \lambda t\ u\ x.\ \mathit{subst}\ x\ u\ t$
and $\mathit{Fvars} = \mathit{Fvars}$
and $\mathit{subst} = \lambda A\ u\ x.\ \mathit{subst\_fmla}\ A\ x\ u$
and $\mathit{eql} = \mathit{eql}$ **and** $\mathit{cnj} = \mathit{cnj}$ **and** $\mathit{dsj} = \mathit{dsj}$ **and** $\mathit{imp} = \mathit{imp}$
and $\mathit{all} = \mathit{all}$ **and** $\mathit{exi} = \mathit{exi}$ **and** $\mathit{fls} = \mathit{fls}$
and $\mathit{prv} = (\vdash)\ \{\}$
and $\mathit{isTrue} = \mathit{eval\_fmla}\ \mathit{e0}$
$\langle \mathit{proof} \rangle$