Soundness and Completeness of an Axiomatic System for First-Order Logic

Asta Halkjær From

May 4, 2022

Abstract

This work is a formalization of the soundness and completeness of an axiomatic system for first-order logic. The proof system is based on System Q1 by Smullyan and the completeness proof follows his textbook "First-Order Logic" (Springer-Verlag 1968) [2]. The completeness proof is in the Henkin style [1] where a consistent set is extended to a maximal consistent set using Lindenbaum's construction and Henkin witnesses are added during the construction to ensure saturation as well. The resulting set is a Hintikka set which, by the model existence theorem, is satisfiable in the Herbrand universe.

Contents

theory *FOL-Axiomatic* **imports** *HOL−Library.Countable* **begin**

1 Syntax

datatype (*params-tm*: *'f*) *tm*
  = *Var nat* (‹#›)
  | *Fun 'f* ‹*'f tm list*› (‹†›)

abbreviation *Const* (‹⋆›) **where** ‹⋆*a* ≡ †*a* []›

datatype (*params-fm*: *'f*, *'p*) *fm*
  = *Falsity* (‹⊥›)
  | *Pre 'p* ‹*'f tm list*› (‹‡›)
  | *Imp* ‹(*'f*, *'p*) *fm*› ‹(*'f*, *'p*) *fm*› (**infixr** ‹⟶› *55*)
  | *Uni* ‹(*'f*, *'p*) *fm*› (‹∀›)

abbreviation *Neg* (‹¬ -› [*70*] *70*) **where** ‹¬ *p* ≡ *p* ⟶ ⊥›

term ‹∀(⊥ ⟶ ‡''*P*'' [†''*f*'' [#*0*]])›

2 Semantics

definition *shift* (‹-⟨-:-⟩›) **where**
  ‹*E*⟨*n*:*x*⟩ *m* ≡ *if m* < *n then E m else if m* = *n then x else E* (*m*−*1*)›

primrec *semantics-tm* (‹⦇-, -⦈›) **where**
  ‹⦇*E*, *F*⦈ (#*n*) = *E n*›
| ‹⦇*E*, *F*⦈ (†*f ts*) = *F f* (*map* ⦇*E*, *F*⦈ *ts*)›

primrec *semantics-fm* (‹⟦-, -, -⟧›) **where**
  ‹⟦-, -, -⟧ ⊥ = *False*›
| ‹⟦*E*, *F*, *G*⟧ (‡*P ts*) = *G P* (*map* ⦇*E*, *F*⦈ *ts*)›
| ‹⟦*E*, *F*, *G*⟧ (*p* ⟶ *q*) = (⟦*E*, *F*, *G*⟧ *p* ⟶ ⟦*E*, *F*, *G*⟧ *q*)›
| ‹⟦*E*, *F*, *G*⟧ (∀ *p*) = (∀ *x*. ⟦*E*⟨*0*:*x*⟩, *F*, *G*⟧ *p*)›

proposition ‹⟦*E*, *F*, *G*⟧ (∀(‡*P* [# *0*]) ⟶ ‡*P* [⋆*a*])›
  ⟨*proof*⟩

3 Operations

3.1 Shift

context fixes *n m* :: *nat* **begin**

lemma *shift-eq* [*simp*]: ‹*n* = *m* ⟹ *E*⟨*n*:*x*⟩ *m* = *x*›
  ⟨*proof*⟩

lemma *shift-gt* [*simp*]: ‹$m < n \Longrightarrow E\langle n{:}x\rangle\ m = E\ m$›
⟨*proof*⟩

lemma *shift-lt* [*simp*]: ‹$n < m \Longrightarrow E\langle n{:}x\rangle\ m = E\ (m{-}1)$›
⟨*proof*⟩

lemma *shift-commute* [*simp*]: ‹$(E\langle n{:}y\rangle\langle 0{:}x\rangle) = (E\langle 0{:}x\rangle\langle n{+}1{:}y\rangle)$›
⟨*proof*⟩

end

3.2 Parameters

abbreviation ‹*params* $S \equiv \bigcup p \in S.$ *params-fm* p›

lemma *upd-params-tm* [*simp*]: ‹$f \notin$ *params-tm* $t \Longrightarrow (\!|E,\ F(f := x)|\!)\ t = (\!|E,\ F|\!)\ t$›
⟨*proof*⟩

lemma *upd-params-fm* [*simp*]: ‹$f \notin$ *params-fm* $p \Longrightarrow [\![E,\ F(f := x),\ G]\!]\ p = [\![E,\ F,\ G]\!]\ p$›
⟨*proof*⟩

lemma *finite-params-tm* [*simp*]: ‹*finite* (*params-tm* t)›
⟨*proof*⟩

lemma *finite-params-fm* [*simp*]: ‹*finite* (*params-fm* p)›
⟨*proof*⟩

3.3 Instantiation

primrec *lift-tm* (‹↑›) **where**
‹↑(#n) = #($n{+}1$)›
| ‹↑(†f ts) = †f (*map* ↑ ts)›

primrec *inst-tm* (‹⟪-′/-⟫›) **where**
‹⟪s/m⟫(#n) = (*if* $n < m$ *then* #n *else if* $n = m$ *then* s *else* #($n{-}1$))›
| ‹⟪s/m⟫(†f ts) = †f (*map* ⟪s/m⟫ ts)›

primrec *inst-fm* (‹⟨-′/-⟩›) **where**
‹⟨-/-⟩⊥ = ⊥›
| ‹⟨s/m⟩(‡P ts) = ‡P (*map* ⟪s/m⟫ ts)›
| ‹⟨s/m⟩($p \longrightarrow q$) = ⟨s/m⟩$p \longrightarrow$ ⟨s/m⟩q›
| ‹⟨s/m⟩(∀ p) = ∀ (⟨↑s/$m{+}1$⟩p)›

lemma *lift-lemma* [*simp*]: ‹(|$E\langle 0{:}x\rangle$, F|) (↑t) = (|E, F|) t›
⟨*proof*⟩

lemma *inst-tm-semantics* [*simp*]: ‹(|E, F|) (⟪s/m⟫t) = (|$E\langle m{:}(\!|E,\ F|\!)\ s\rangle$, F|) t›

⟨*proof*⟩

lemma *inst-fm-semantics* [*simp*]: ‹⟦*E*, *F*, *G*⟧ (⟨*t*/*m*⟩*p*) = ⟦*E*⟨*m*:(⦇*E*, *F*⦈) *t*⟩, *F*, *G*⟧ *p*›
⟨*proof*⟩

3.4 Size

The built-in *size* is not invariant under substitution.

primrec *size-fm* **where**
‹*size-fm* ⊥ = *1*›
| ‹*size-fm* (‡- -) = *1*›
| ‹*size-fm* (*p* ⟶ *q*) = *1* + *size-fm p* + *size-fm q*›
| ‹*size-fm* (∀ *p*) = *1* + *size-fm p*›

lemma *size-inst-fm* [*simp*]: ‹*size-fm* (⟨*t*/*m*⟩*p*) = *size-fm p*›
⟨*proof*⟩

4 Propositional Semantics

primrec *boolean* **where**
‹*boolean* - - ⊥ = *False*›
| ‹*boolean G* - (‡*P ts*) = *G P ts*›
| ‹*boolean G A* (*p* ⟶ *q*) = (*boolean G A p* ⟶ *boolean G A q*)›
| ‹*boolean* - *A* (∀ *p*) = *A* (∀ *p*)›

abbreviation ‹*tautology p* ≡ ∀ *G A*. *boolean G A p*›

proposition ‹*tautology* (∀ (‡*P* [#*0*]) ⟶ ∀ (‡*P* [#*0*]))›
⟨*proof*⟩

lemma *boolean-semantics*: ‹*boolean* (λ*a*. *G a* ∘ *map* (⦇*E*, *F*⦈)) ⟦*E*, *F*, *G*⟧ = ⟦*E*, *F*, *G*⟧›
⟨*proof*⟩

lemma *tautology*: ‹*tautology p* ⟹ ⟦*E*, *F*, *G*⟧ *p*›
⟨*proof*⟩

proposition ‹∃ *p*. (∀ *E F G*. ⟦*E*, *F*, *G*⟧ *p*) ∧ ¬ *tautology p*›
⟨*proof*⟩

5 Calculus

Adapted from System Q1 by Smullyan in First-Order Logic (1968)

inductive *Axiomatic* (‹⊢ -› [*50*] *50*) **where**
TA: ‹*tautology p* ⟹ ⊢ *p*›
| *IA*: ‹⊢ ∀ *p* ⟶ ⟨*t*/*0*⟩*p*›

| *MP*: ‹$\vdash p \longrightarrow q \Longrightarrow \vdash p \Longrightarrow \vdash q$›
| *GR*: ‹$\vdash q \longrightarrow \langle \star a/0 \rangle p \Longrightarrow a \notin \mathit{params}\ \{p, q\} \Longrightarrow \vdash q \longrightarrow \forall p$›

lemmas
TA[*simp*]
MP[*trans*, *dest*]
GR[*intro*]

We simulate assumptions on the lhs of $\vdash$ with a chain of implications on the rhs.

primrec *imply* (**infixr** ‹$\rightsquigarrow$› *56*) **where**
‹$([] \rightsquigarrow q) = q$›
| ‹$(p \# ps \rightsquigarrow q) = (p \longrightarrow ps \rightsquigarrow q)$›

abbreviation *Axiomatic-assms* (‹- $\vdash$ -› [*50*, *50*] *50*) **where**
‹$ps \vdash q \equiv \vdash ps \rightsquigarrow q$›

6 Soundness

theorem *soundness*: ‹$\vdash p \Longrightarrow [\![E, F, G]\!]\ p$›
⟨*proof*⟩

corollary ‹$\neg (\vdash \bot)$›
⟨*proof*⟩

7 Derived Rules

lemma *AS*: ‹$\vdash (p \longrightarrow q \longrightarrow r) \longrightarrow (p \longrightarrow q) \longrightarrow p \longrightarrow r$›
⟨*proof*⟩

lemma *AK*: ‹$\vdash q \longrightarrow p \longrightarrow q$›
⟨*proof*⟩

lemma *Neg*: ‹$\vdash \neg \neg p \longrightarrow p$›
⟨*proof*⟩

lemma *contraposition*:
‹$\vdash (p \longrightarrow q) \longrightarrow \neg q \longrightarrow \neg p$›
‹$\vdash (\neg q \longrightarrow \neg p) \longrightarrow p \longrightarrow q$›
⟨*proof*⟩

lemma *GR'*: ‹$\vdash \neg \langle \star a/0 \rangle p \longrightarrow q \Longrightarrow a \notin \mathit{params}\ \{p, q\} \Longrightarrow \vdash \neg (\forall p) \longrightarrow q$›
⟨*proof*⟩

lemma *Imp3*: ‹$\vdash (p \longrightarrow q \longrightarrow r) \longrightarrow ((s \longrightarrow p) \longrightarrow (s \longrightarrow q) \longrightarrow s \longrightarrow r)$›
⟨*proof*⟩

lemma *imply-ImpE*: ‹$\vdash ps \rightsquigarrow p \longrightarrow ps \rightsquigarrow (p \longrightarrow q) \longrightarrow ps \rightsquigarrow q$›

⟨*proof*⟩

lemma *MP′* [*trans*, *dest*]: ‹*ps* ⊢ *p* ⟶ *q* ⟹ *ps* ⊢ *p* ⟹ *ps* ⊢ *q*›
⟨*proof*⟩

lemma *imply-Cons* [*intro*]: ‹*ps* ⊢ *q* ⟹ *p* # *ps* ⊢ *q*›
⟨*proof*⟩

lemma *imply-head* [*intro*]: ‹*p* # *ps* ⊢ *p*›
⟨*proof*⟩

lemma *add-imply* [*simp*]: ‹⊢ *q* ⟹ *ps* ⊢ *q*›
⟨*proof*⟩

lemma *imply-mem* [*simp*]: ‹*p* ∈ *set ps* ⟹ *ps* ⊢ *p*›
⟨*proof*⟩

lemma *deduct1*: ‹*ps* ⊢ *p* ⟶ *q* ⟹ *p* # *ps* ⊢ *q*›
⟨*proof*⟩

lemma *imply-append* [*iff*]: ‹(*ps* @ *qs* ⇝ *r*) = (*ps* ⇝ *qs* ⇝ *r*)›
⟨*proof*⟩

lemma *imply-swap-append*: ‹*ps* @ *qs* ⊢ *r* ⟹ *qs* @ *ps* ⊢ *r*›
⟨*proof*⟩

lemma *deduct2*: ‹*p* # *ps* ⊢ *q* ⟹ *ps* ⊢ *p* ⟶ *q*›
⟨*proof*⟩

lemmas *deduct* [*iff*] = *deduct1 deduct2*

lemma *cut* [*trans*, *dest*]: ‹*p* # *ps* ⊢ *r* ⟹ *q* # *ps* ⊢ *p* ⟹ *q* # *ps* ⊢ *r*›
⟨*proof*⟩

lemma *Boole*: ‹(¬ *p*) # *ps* ⊢ ⊥ ⟹ *ps* ⊢ *p*›
⟨*proof*⟩

lemma *imply-weaken*: ‹*ps* ⊢ *q* ⟹ *set ps* ⊆ *set ps′* ⟹ *ps′* ⊢ *q*›
⟨*proof*⟩

8 Consistent

definition ‹*consistent S* ≡ ∄ *S′*. *set S′* ⊆ *S* ∧ *S′* ⊢ ⊥›

lemma *UN-finite-bound*:
assumes ‹*finite A*› **and** ‹*A* ⊆ (⋃ *n*. *f n*)›
shows ‹∃ *m* :: *nat*. *A* ⊆ (⋃ *n* ≤ *m*. *f n*)›
⟨*proof*⟩

lemma *split-list*:
assumes ‹$x \in set\ A$›
shows ‹$set\ (x\ \#\ removeAll\ x\ A) = set\ A \wedge x \notin set\ (removeAll\ x\ A)$›
⟨*proof*⟩

lemma *imply-params-fm*: ‹$params\text{-}fm\ (ps \rightsquigarrow q) = params\text{-}fm\ q \cup (\bigcup p \in set\ ps.\ params\text{-}fm\ p)$›
⟨*proof*⟩

lemma *inconsistent-fm*:
assumes ‹$consistent\ S$› **and** ‹$\neg\ consistent\ (\{p\} \cup S)$›
obtains S' **where** ‹$set\ S' \subseteq S$› **and** ‹$p\ \#\ S' \vdash \bot$›
⟨*proof*⟩

lemma *consistent-add-witness*:
assumes ‹$consistent\ S$› **and** ‹$\neg\ (\forall\ p) \in S$› **and** ‹$a \notin params\ S$›
shows ‹$consistent\ (\{\neg\ \langle \star a/0\rangle p\} \cup S)$›
⟨*proof*⟩

lemma *consistent-add-instance*:
assumes ‹$consistent\ S$› **and** ‹$\forall\ p \in S$›
shows ‹$consistent\ (\{\langle t/0\rangle p\} \cup S)$›
⟨*proof*⟩

9 Extension

fun *witness* **where**
‹$witness\ used\ (\neg\ (\forall\ p)) = \{\neg\ \langle \star(SOME\ a.\ a \notin used)/0\rangle p\}$›
| ‹$witness\ \text{-}\ \text{-} = \{\}$›

primrec *extend* **where**
‹$extend\ S\ f\ 0 = S$›
| ‹$extend\ S\ f\ (Suc\ n) =$
($let\ Sn = extend\ S\ f\ n\ in$
$if\ consistent\ (\{f\ n\} \cup Sn)$
$then\ witness\ (params\ (\{f\ n\} \cup Sn))\ (f\ n) \cup \{f\ n\} \cup Sn$
$else\ Sn$)›

definition ‹$Extend\ S\ f \equiv \bigcup n.\ extend\ S\ f\ n$›

lemma *extend-subset*: ‹$S \subseteq extend\ S\ f\ n$›
⟨*proof*⟩

lemma *Extend-subset*: ‹$S \subseteq Extend\ S\ f$›
⟨*proof*⟩

lemma *extend-bound*: ‹$(\bigcup n \leq m.\ extend\ S\ f\ n) = extend\ S\ f\ m$›
⟨*proof*⟩

lemma *finite-params-witness* [*simp*]: ‹*finite* (*params* (*witness used p*))›
⟨*proof*⟩

lemma *finite-params-extend* [*simp*]: ‹*finite* (*params* (*extend S f n*) − *params S*)›
⟨*proof*⟩

lemma *Set-Diff-Un*: ‹*X* − (*Y* ∪ *Z*) = *X* − *Y* − *Z*›
⟨*proof*⟩

lemma *infinite-params-extend*:
assumes ‹*infinite* (*UNIV* − *params S*)›
shows ‹*infinite* (*UNIV* − *params* (*extend S f n*))›
⟨*proof*⟩

lemma *consistent-witness*:
assumes ‹*consistent S*› **and** ‹*p* ∈ *S*› **and** ‹*params S* ⊆ *used*›
and ‹*infinite* (*UNIV* − *used*)›
shows ‹*consistent* (*witness used p* ∪ *S*)›
⟨*proof*⟩

lemma *consistent-extend*:
assumes ‹*consistent S*› **and** ‹*infinite* (*UNIV* − *params S*)›
shows ‹*consistent* (*extend S f n*)›
⟨*proof*⟩

lemma *consistent-Extend*:
assumes ‹*consistent S*› **and** ‹*infinite* (*UNIV* − *params S*)›
shows ‹*consistent* (*Extend S f*)›
⟨*proof*⟩

10 Maximal

definition ‹*maximal S* ≡ ∀ *p*. *p* ∉ *S* ⟶ ¬ *consistent* ({*p*} ∪ *S*)›

lemma *maximal-exactly-one*:
assumes ‹*consistent S*› **and** ‹*maximal S*›
shows ‹*p* ∈ *S* ⟷ (¬ *p*) ∉ *S*›
⟨*proof*⟩

lemma *maximal-Extend*:
assumes ‹*surj f*›
shows ‹*maximal* (*Extend S f*)›
⟨*proof*⟩

11 Saturation

definition ‹*saturated S* ≡ ∀ *p*. ¬ (∀ *p*) ∈ *S* ⟶ (∃ *a*. (¬ ⟨⋆*a*/*0*⟩*p*) ∈ *S*)›

lemma *saturated-Extend*:
assumes ‹*consistent* (*Extend S f*)› **and** ‹*surj f*›
shows ‹*saturated* (*Extend S f*)›
⟨*proof*⟩

12 Hintikka

locale *Hintikka* =
fixes *H* :: ‹('*f*, '*p*) *fm set*›
assumes
FlsH: ‹⊥ ∉ *H*› **and**
ImpH: ‹(*p* ⟶ *q*) ∈ *H* ⟷ (*p* ∈ *H* ⟶ *q* ∈ *H*)› **and**
UniH: ‹(∀ *p* ∈ *H*) ⟷ (∀ *t*. ⟨*t*/0⟩*p* ∈ *H*)›

12.1 Model Existence

abbreviation *hmodel* (‹⟦-⟧›) **where** ‹⟦*H*⟧ ≡ ⟦#, †, λ*P ts*. ‡*P ts* ∈ *H*⟧›

lemma *semantics-tm-id* [*simp*]: ‹(|#, †|) *t* = *t*›
⟨*proof*⟩

lemma *semantics-tm-id-map* [*simp*]: ‹*map* (|#, †|) *ts* = *ts*›
⟨*proof*⟩

theorem *Hintikka-model*:
assumes ‹*Hintikka H*›
shows ‹*p* ∈ *H* ⟷ ⟦*H*⟧ *p*›
⟨*proof*⟩

12.2 Maximal Consistent Sets are Hintikka Sets

lemma *deriv-iff-MCS*:
assumes ‹*consistent S*› **and** ‹*maximal S*›
shows ‹(∃ *ps*. *set ps* ⊆ *S* ∧ *ps* ⊢ *p*) ⟷ *p* ∈ *S*›
⟨*proof*⟩

lemma *Hintikka-Extend*:
assumes ‹*consistent H*› **and** ‹*maximal H*› **and** ‹*saturated H*›
shows ‹*Hintikka H*›
⟨*proof*⟩

13 Countable Formulas

instance *tm* :: (*countable*) *countable*
⟨*proof*⟩

instance *fm* :: (*countable*, *countable*) *countable*
⟨*proof*⟩

14 Completeness

lemma *infinite-Diff-fin-Un*: ‹*infinite* $(X - Y) \Longrightarrow$ *finite* $Z \Longrightarrow$ *infinite* $(X - (Z \cup Y))$›
⟨*proof*⟩

theorem *strong-completeness*:
fixes p :: ‹$('f$ :: *countable*, $'p$ :: *countable*$)$ *fm*›
assumes ‹$\forall (E$ :: - $\Rightarrow$ $'f$ *tm*$)$ F G. $(\forall q \in X.$ $[\![E, F, G]\!]$ $q) \longrightarrow$ $[\![E, F, G]\!]$ p›
and ‹*infinite* $(UNIV -$ *params* $X)$›
shows ‹$\exists$ *ps*. *set* $ps \subseteq X \wedge ps \vdash p$›
⟨*proof*⟩

theorem *completeness*:
fixes p :: ‹(nat, nat) *fm*›
assumes ‹$\forall (E$ :: *nat* $\Rightarrow$ *nat tm*$)$ F G. $[\![E, F, G]\!]$ p›
shows ‹$\vdash p$›
⟨*proof*⟩

15 Main Result

abbreviation *valid* :: ‹(nat, nat) *fm* $\Rightarrow$ *bool*› **where**
‹*valid* $p \equiv \forall (E$ :: *nat* $\Rightarrow$ *nat tm*$)$ F G. $[\![E, F, G]\!]$ p›

theorem *main*: ‹*valid* $p \longleftrightarrow (\vdash p)$›
⟨*proof*⟩

end

theory *FOL-Axiomatic-Variant* **imports** *HOL−Library.Countable* **begin**

16 Syntax

datatype $'f$ *tm*
= *Var nat* (‹#›)
| *Fun* $'f$ ‹$'f$ *tm list*› (‹†›)

datatype $('f, 'p)$ *fm*
= *Falsity* (‹⊥›)
| *Pre* $'p$ ‹$'f$ *tm list*› (‹‡›)
| *Imp* ‹$('f, 'p)$ *fm*› ‹$('f, 'p)$ *fm*› (**infixr** ‹⟶› *55*)
| *Uni* ‹$('f, 'p)$ *fm*› (‹∀ ›)

abbreviation *Neg* (‹¬ -› [*70*] *70*) **where** ‹¬ $p \equiv p \longrightarrow \bot$›

term ‹$\forall (\bot \longrightarrow$ ‡$''P''$ [†$''f''$ [#*0*]]›

17 Semantics

definition *shift* :: ‹(*nat* ⇒ *'a*) ⇒ *nat* ⇒ *'a* ⇒ *nat* ⇒ *'a*›
  (‹-⟨-:-⟩› [*90*, *0*, *0*] *91*) **where**
  ‹*E*⟨*n*:*x*⟩ = (λ*m*. *if m* < *n then E m else if m* = *n then x else E* (*m*−*1*))›

primrec *semantics-tm* (‹(|-, -|)›) **where**
  ‹(|*E*, *F*|) (#*n*) = *E n*›
| ‹(|*E*, *F*|) (†*f ts*) = *F f* (*map* (|*E*, *F*|) *ts*)›

primrec *semantics-fm* (‹⟦-, -, -⟧›) **where**
  ‹⟦-, -, -⟧ ⊥ = *False*›
| ‹⟦*E*, *F*, *G*⟧ (‡*P ts*) = *G P* (*map* (|*E*, *F*|) *ts*)›
| ‹⟦*E*, *F*, *G*⟧ (*p* ⟶ *q*) = (⟦*E*, *F*, *G*⟧ *p* ⟶ ⟦*E*, *F*, *G*⟧ *q*)›
| ‹⟦*E*, *F*, *G*⟧ (∀ *p*) = (∀ *x*. ⟦*E*⟨*0*:*x*⟩, *F*, *G*⟧ *p*)›

proposition ‹⟦*E*, *F*, *G*⟧ (∀ (‡*P* [# *0*]) ⟶ ‡*P* [† *a* []])›
  ⟨*proof*⟩

18 Operations

18.1 Shift

lemma *shift-eq* [*simp*]: ‹*n* = *m* ⟹ (*E*⟨*n*:*x*⟩) *m* = *x*›
  ⟨*proof*⟩

lemma *shift-gt* [*simp*]: ‹*m* < *n* ⟹ (*E*⟨*n*:*x*⟩) *m* = *E m*›
  ⟨*proof*⟩

lemma *shift-lt* [*simp*]: ‹*n* < *m* ⟹ (*E*⟨*n*:*x*⟩) *m* = *E* (*m*−*1*)›
  ⟨*proof*⟩

lemma *shift-commute* [*simp*]: ‹*E*⟨*n*:*y*⟩⟨*0*:*x*⟩ = *E*⟨*0*:*x*⟩⟨*n*+*1*:*y*⟩›
⟨*proof*⟩

18.2 Variables

primrec *vars-tm* **where**
  ‹*vars-tm* (#*n*) = [*n*]›
| ‹*vars-tm* (†- *ts*) = *concat* (*map vars-tm ts*)›

primrec *vars-fm* **where**
  ‹*vars-fm* ⊥ = []›
| ‹*vars-fm* (‡- *ts*) = *concat* (*map vars-tm ts*)›
| ‹*vars-fm* (*p* ⟶ *q*) = *vars-fm p* @ *vars-fm q*›
| ‹*vars-fm* (∀ *p*) = *vars-fm p*›

abbreviation ‹*vars S* ≡ ⋃ *p* ∈ *S*. *set* (*vars-fm p*)›

primrec *max-list* :: ‹*nat list* ⇒ *nat*› **where**
  ‹*max-list* [] = *0*›
| ‹*max-list* (*x* # *xs*) = *max x* (*max-list xs*)›

lemma *max-list-append*: ‹*max-list* (*xs* @ *ys*) = *max* (*max-list xs*) (*max-list ys*)›
  ⟨*proof*⟩

lemma *upd-vars-tm* [*simp*]: ‹*n* ∉ *set* (*vars-tm t*) ⟹ (|*E*(*n* := *x*), *F*|) *t* = (|*E*, *F*|) *t*›
  ⟨*proof*⟩

lemma *shift-upd-commute*: ‹*m* ≤ *n* ⟹ (*E*(*n* := *x*)⟨*m*:*y*⟩) = ((*E*⟨*m*:*y*⟩)(*n* + *1* := *x*))›
  ⟨*proof*⟩

lemma *max-list-concat*: ‹*xs* ∈ *set xss* ⟹ *max-list xs* ≤ *max-list* (*concat xss*)›
  ⟨*proof*⟩

lemma *max-list-in*: ‹*max-list xs* < *n* ⟹ *n* ∉ *set xs*›
  ⟨*proof*⟩

lemma *upd-vars-fm* [*simp*]: ‹*max-list* (*vars-fm p*) < *n* ⟹ ⟦*E*(*n* := *x*), *F*, *G*⟧ *p* = ⟦*E*, *F*, *G*⟧ *p*›
⟨*proof*⟩

abbreviation ‹*max-var p* ≡ *max-list* (*vars-fm p*)›

18.3 Instantiation

primrec *lift-tm* (‹↑›) **where**
  ‹↑(#*n*) = #(*n*+*1*)›
| ‹↑(†*f ts*) = †*f* (*map* ↑ *ts*)›

primrec *inst-tm* (‹-'⟪-'/-'⟫› [*90*, *0*, *0*] *91*) **where**
  ‹(#*n*)⟪*s*/*m*⟫ = (*if n* < *m then* #*n else if n* = *m then s else* #(*n*−*1*))›
| ‹(†*f ts*)⟪*s*/*m*⟫ = †*f* (*map* (λ*t*. *t*⟪*s*/*m*⟫) *ts*)›

primrec *inst-fm* (‹-'⟨-'/-'⟩› [*90*, *0*, *0*] *91*) **where**
  ‹⊥⟨-/-⟩ = ⊥›
| ‹(‡*P ts*)⟨*s*/*m*⟩ = ‡*P* (*map* (λ*t*. *t*⟪*s*/*m*⟫) *ts*)›
| ‹(*p* ⟶ *q*)⟨*s*/*m*⟩ = (*p*⟨*s*/*m*⟩ ⟶ *q*⟨*s*/*m*⟩)›
| ‹(∀ *p*)⟨*s*/*m*⟩ = ∀ (*p*⟨↑*s*/*m*+*1*⟩)›

lemma *lift-lemma* [*simp*]: ‹(|*E*⟨*0*:*x*⟩, *F*|) (↑*t*) = (|*E*, *F*|) *t*›
  ⟨*proof*⟩

lemma *inst-tm-semantics* [*simp*]: ‹(|*E*, *F*|) (*t*⟪*s*/*m*⟫) = (|*E*⟨*m*:(|*E*, *F*|) *s*⟩, *F*|) *t*›
  ⟨*proof*⟩

lemma *inst-fm-semantics* [*simp*]: ‹$[\![E,\ F,\ G]\!]\ (p\langle t/m\rangle) = [\![E\langle m{:}(\!|E,\ F|\!)\ t\rangle,\ F,\ G]\!]\ p$›
⟨*proof*⟩

18.4 Size

The built-in *size* is not invariant under substitution.

primrec *size-fm* **where**
‹*size-fm* $\bot = 1$›
| ‹*size-fm* $(\ddagger\text{-}\ \text{-}) = 1$›
| ‹*size-fm* $(p \longrightarrow q) = 1 + \textit{size-fm}\ p + \textit{size-fm}\ q$›
| ‹*size-fm* $(\forall\, p) = 1 + \textit{size-fm}\ p$›

lemma *size-inst-fm* [*simp*]:
‹*size-fm* $(p\langle t/m\rangle) = \textit{size-fm}\ p$›
⟨*proof*⟩

19 Propositional Semantics

primrec *boolean* **where**
‹*boolean* - - $\bot$ = *False*›
| ‹*boolean* *G* - $(\ddagger P\ ts) = G\ P\ ts$›
| ‹*boolean* *G* *A* $(p \longrightarrow q) = (\textit{boolean}\ G\ A\ p \longrightarrow \textit{boolean}\ G\ A\ q)$›
| ‹*boolean* - *A* $(\forall\, p) = A\ (\forall\, p)$›

abbreviation ‹*tautology* $p \equiv \forall\, G\ A.\ \textit{boolean}\ G\ A\ p$›

proposition ‹*tautology* $(\forall\, (\ddagger P\ [\#0]) \longrightarrow \forall\, (\ddagger P\ [\#0]))$›
⟨*proof*⟩

lemma *boolean-semantics*: ‹*boolean* $(\lambda a.\ G\ a \circ \textit{map}\ (\!|E,\ F|\!))\ [\![E,\ F,\ G]\!] = [\![E,\ F,\ G]\!]$›
⟨*proof*⟩

lemma *tautology*: ‹*tautology* $p \Longrightarrow [\![E,\ F,\ G]\!]\ p$›
⟨*proof*⟩

proposition ‹$\exists\, p.\ (\forall\, E\ F\ G.\ [\![E,\ F,\ G]\!]\ p) \wedge \neg\ \textit{tautology}\ p$›
⟨*proof*⟩

20 Calculus

Adapted from System Q1 by Smullyan in First-Order Logic (1968)

inductive *Axiomatic* (‹$\vdash$ -› [*50*] *50*) **where**
TA: ‹*tautology* $p \Longrightarrow \vdash p$›
| *IA*: ‹$\vdash \forall\, p \longrightarrow p\langle t/0\rangle$›
| *MP*: ‹$\vdash p \longrightarrow q \Longrightarrow \vdash p \Longrightarrow \vdash q$›

| *GR*: ‹$\vdash q \longrightarrow p\langle\#n/0\rangle \Longrightarrow$ *max-var* $p < n \Longrightarrow$ *max-var* $q < n \Longrightarrow \vdash q \longrightarrow \forall\, p$›

lemmas
TA[*simp*]
MP[*trans*, *dest*]
GR[*intro*]

We simulate assumptions on the lhs of $\vdash$ with a chain of implications on the rhs.

primrec *imply* (**infixr** ‹$\rightsquigarrow$› *56*) **where**
‹$([] \rightsquigarrow q) = q$›
| ‹$(p \# ps \rightsquigarrow q) = (p \longrightarrow ps \rightsquigarrow q)$›

abbreviation *Axiomatic-assms* (‹- $\vdash$ -› [*50*, *50*] *50*) **where**
‹$ps \vdash q \equiv \vdash ps \rightsquigarrow q$›

21 Soundness

theorem *soundness*: ‹$\vdash p \Longrightarrow [\![E, F, G]\!]\ p$›
⟨*proof*⟩

corollary ‹$\neg (\vdash \bot)$›
⟨*proof*⟩

22 Derived Rules

lemma *AS*: ‹$\vdash (p \longrightarrow q \longrightarrow r) \longrightarrow (p \longrightarrow q) \longrightarrow p \longrightarrow r$›
⟨*proof*⟩

lemma *AK*: ‹$\vdash q \longrightarrow p \longrightarrow q$›
⟨*proof*⟩

lemma *Neg*: ‹$\vdash \neg \neg p \longrightarrow p$›
⟨*proof*⟩

lemma *contraposition*:
‹$\vdash (p \longrightarrow q) \longrightarrow \neg q \longrightarrow \neg p$›
‹$\vdash (\neg q \longrightarrow \neg p) \longrightarrow p \longrightarrow q$›
⟨*proof*⟩

lemma *GR'*: ‹$\vdash \neg p\langle\#n/0\rangle \longrightarrow q \Longrightarrow$ *max-var* $p < n \Longrightarrow$ *max-var* $q < n \Longrightarrow \vdash \neg \forall\, p \longrightarrow q$›
⟨*proof*⟩

lemma *Imp3*: ‹$\vdash (p \longrightarrow q \longrightarrow r) \longrightarrow ((s \longrightarrow p) \longrightarrow (s \longrightarrow q) \longrightarrow s \longrightarrow r)$›
⟨*proof*⟩

lemma *imply-ImpE*: ‹$\vdash ps \rightsquigarrow p \longrightarrow ps \rightsquigarrow (p \longrightarrow q) \longrightarrow ps \rightsquigarrow q$›

⟨*proof*⟩

lemma *MP′* [*trans*, *dest*]: ‹$ps \vdash p \longrightarrow q \Longrightarrow ps \vdash p \Longrightarrow ps \vdash q$›
⟨*proof*⟩

lemma *imply-Cons* [*intro*]: ‹$ps \vdash q \Longrightarrow p \# ps \vdash q$›
⟨*proof*⟩

lemma *imply-head* [*intro*]: ‹$p \# ps \vdash p$›
⟨*proof*⟩

lemma *imply-lift-Imp* [*simp*]:
assumes ‹$\vdash p \longrightarrow q$›
shows ‹$\vdash p \longrightarrow ps \rightsquigarrow q$›
⟨*proof*⟩

lemma *add-imply* [*simp*]: ‹$\vdash q \Longrightarrow ps \vdash q$›
⟨*proof*⟩

lemma *imply-mem* [*simp*]: ‹$p \in set\ ps \Longrightarrow ps \vdash p$›
⟨*proof*⟩

lemma *deduct1*: ‹$ps \vdash p \longrightarrow q \Longrightarrow p \# ps \vdash q$›
⟨*proof*⟩

lemma *imply-append* [*iff*]: ‹$(ps\ @\ qs \rightsquigarrow r) = (ps \rightsquigarrow qs \rightsquigarrow r)$›
⟨*proof*⟩

lemma *imply-swap-append*: ‹$ps\ @\ qs \vdash r \Longrightarrow qs\ @\ ps \vdash r$›
⟨*proof*⟩

lemma *deduct2*: ‹$p \# ps \vdash q \Longrightarrow ps \vdash p \longrightarrow q$›
⟨*proof*⟩

lemmas *deduct* [*iff*] = *deduct1 deduct2*

lemma *cut* [*trans*, *dest*]: ‹$p \# ps \vdash r \Longrightarrow q \# ps \vdash p \Longrightarrow q \# ps \vdash r$›
⟨*proof*⟩

lemma *Boole*: ‹$(\neg p) \# ps \vdash \bot \Longrightarrow ps \vdash p$›
⟨*proof*⟩

lemma *imply-weaken*: ‹$ps \vdash q \Longrightarrow set\ ps \subseteq set\ ps' \Longrightarrow ps' \vdash q$›
⟨*proof*⟩

23 Consistent

definition ‹*consistent* $S \equiv \nexists S'.\ set\ S' \subseteq S \wedge S' \vdash \bot$›

lemma *UN-finite-bound*:
assumes ‹*finite A*› **and** ‹$A \subseteq (\bigcup n.\ f\ n)$›
shows ‹$\exists\, m :: nat.\ A \subseteq (\bigcup n \leq m.\ f\ n)$›
⟨*proof*⟩

lemma *split-list*:
assumes ‹$x \in set\ A$›
shows ‹$set\ (x\ \#\ removeAll\ x\ A) = set\ A \wedge x \notin set\ (removeAll\ x\ A)$›
⟨*proof*⟩

lemma *imply-vars-fm*: ‹$vars\text{-}fm\ (ps \rightsquigarrow q) = concat\ (map\ vars\text{-}fm\ ps)\ @\ vars\text{-}fm\ q$›
⟨*proof*⟩

lemma *inconsistent-fm*:
assumes ‹*consistent S*› **and** ‹$\neg$ *consistent* $(\{p\} \cup S)$›
obtains S' **where** ‹$set\ S' \subseteq S$› **and** ‹$p\ \#\ S' \vdash \bot$›
⟨*proof*⟩

definition *max-set* :: ‹*nat set* ⇒ *nat*› **where**
‹*max-set X* ≡ *if X* = {} *then 0 else Max X*›

lemma *max-list-in-Cons*: ‹$xs \neq [] \Longrightarrow max\text{-}list\ xs \in set\ xs$›
⟨*proof*⟩

lemma *max-list-max*: ‹$\forall\, x \in set\ xs.\ x \leq max\text{-}list\ xs$›
⟨*proof*⟩

lemma *max-list-in-set*: ‹$finite\ S \Longrightarrow set\ xs \subseteq S \Longrightarrow max\text{-}list\ xs \leq max\text{-}set\ S$›
⟨*proof*⟩

lemma *consistent-add-witness*:
assumes ‹*consistent S*› **and** ‹$(\neg\ \forall\, p) \in S$›
and ‹$finite\ (vars\ S)$› **and** ‹$max\text{-}set\ (vars\ S) < n$›
shows ‹$consistent\ (\{\neg\ p\langle \#n/0\rangle\} \cup S)$›
⟨*proof*⟩

lemma *consistent-add-instance*:
assumes ‹*consistent S*› **and** ‹$\forall\, p \in S$›
shows ‹$consistent\ (\{p\langle t/0\rangle\} \cup S)$›
⟨*proof*⟩

24 Extension

fun *witness* **where**
‹$witness\ used\ (\neg\ \forall\, p) = \{\neg\ p\langle \#(SOME\ n.\ max\text{-}set\ used < n)/0\rangle\}$›
| ‹*witness - - =* {}›

primrec *extend* **where**

‹*extend S f 0 = S*›
| ‹*extend S f* (*Suc n*) =
(*let Sn = extend S f n in*
if consistent ({*f n*} ∪ *Sn*)
then witness (*vars* ({*f n*} ∪ *Sn*)) (*f n*) ∪ {*f n*} ∪ *Sn*
else Sn)›

definition ‹*Extend S f* ≡ ⋃ *n. extend S f n*›

lemma *Extend-subset*: ‹*S* ⊆ *Extend S f*›
⟨*proof*⟩

lemma *extend-bound*: ‹(⋃ *n* ≤ *m. extend S f n*) = *extend S f m*›
⟨*proof*⟩

lemma *finite-vars-witness* [*simp*]: ‹*finite* (*vars* (*witness used p*))›
⟨*proof*⟩

lemma *finite-vars-extend* [*simp*]: ‹*finite* (*vars S*) ⟹ *finite* (*vars* (*extend S f n*))›
⟨*proof*⟩

lemma *max-list-mono*: ‹*set xs* ⊆ *set ys* ⟹ *max-list xs* ≤ *max-list ys*›
⟨*proof*⟩

lemma *consistent-witness*:
fixes *p* :: ‹('*f*, '*p*) *fm*›
assumes ‹*consistent S*› **and** ‹*p* ∈ *S*› **and** ‹*vars S* ⊆ *used*› **and** ‹*finite used*›
shows ‹*consistent* (*witness used p* ∪ *S*)›
⟨*proof*⟩

lemma *consistent-extend*:
fixes *f* :: ‹*nat* ⇒ ('*f*, '*p*) *fm*›
assumes ‹*consistent S*› ‹*finite* (*vars S*)›
shows ‹*consistent* (*extend S f n*)›
⟨*proof*⟩

lemma *consistent-Extend*:
fixes *f* :: ‹*nat* ⇒ ('*f*, '*p*) *fm*›
assumes ‹*consistent S*› ‹*finite* (*vars S*)›
shows ‹*consistent* (*Extend S f*)›
⟨*proof*⟩

25 Maximal

definition ‹*maximal S* ≡ ∀ *p. p* ∉ *S* ⟶ ¬ *consistent* ({*p*} ∪ *S*)›

lemma *maximal-exactly-one*:
assumes ‹*consistent S*› **and** ‹*maximal S*›
shows ‹*p* ∈ *S* ⟷ (¬ *p*) ∉ *S*›

⟨*proof*⟩

lemma *maximal-Extend*:
assumes ‹*surj f*›
shows ‹*maximal* (*Extend S f*)›
⟨*proof*⟩

26 Saturation

definition ‹*saturated S* ≡ ∀ *p*. (¬ ∀ *p*) ∈ *S* ⟶ (∃ *n*. (¬ *p*⟨#*n*/*0*⟩) ∈ *S*)›

lemma *saturated-Extend*:
assumes ‹*consistent* (*Extend S f*)› **and** ‹*surj f*›
shows ‹*saturated* (*Extend S f*)›
⟨*proof*⟩

27 Hintikka

locale *Hintikka* =
fixes *H* :: ‹('*f*, '*p*) *fm set*›
assumes
NoFalsity: ‹⊥ ∉ *H*› **and**
ImpP: ‹(*p* ⟶ *q*) ∈ *H* ⟹ *p* ∉ *H* ∨ *q* ∈ *H*› **and**
ImpN: ‹(*p* ⟶ *q*) ∉ *H* ⟹ *p* ∈ *H* ∧ *q* ∉ *H*› **and**
UniP: ‹∀ *p* ∈ *H* ⟹ ∀ *t*. *p*⟨*t*/*0*⟩ ∈ *H*› **and**
UniN: ‹∀ *p* ∉ *H* ⟹ ∃ *n*. *p*⟨#*n*/*0*⟩ ∉ *H*›

27.1 Model Existence

abbreviation *hmodel* (‹⟦-⟧›) **where** ‹⟦*H*⟧ ≡ ⟦#, †, λ*P ts*. *Pre P ts* ∈ *H*⟧›

lemma *semantics-tm-id* [*simp*]:
‹(|#, †|) *t* = *t*›
⟨*proof*⟩

lemma *semantics-tm-id-map* [*simp*]: ‹*map* (|#, †|) *ts* = *ts*›
⟨*proof*⟩

theorem *Hintikka-model*:
assumes ‹*Hintikka H*›
shows ‹*p* ∈ *H* ⟷ ⟦*H*⟧ *p*›
⟨*proof*⟩

27.2 Maximal Consistent Sets are Hintikka Sets

lemma *inconsistent-head*:
assumes ‹*consistent S*› **and** ‹*maximal S*› **and** ‹*p* ∉ *S*›
obtains *S'* **where** ‹*set S'* ⊆ *S*› **and** ‹*p* # *S'* ⊢ ⊥›

⟨*proof*⟩

lemma *inconsistent-parts* [*simp*]:
assumes ‹$ps \vdash \bot$› **and** ‹*set ps* $\subseteq S$›
shows ‹$\neg$ *consistent S*›
⟨*proof*⟩

lemma *Hintikka-Extend*:
fixes H :: ‹(*'f*, *'p*) *fm set*›
assumes ‹*consistent H*› **and** ‹*maximal H*› **and** ‹*saturated H*›
shows ‹*Hintikka H*›
⟨*proof*⟩

28 Countable Formulas

instance *tm* :: (*countable*) *countable*
⟨*proof*⟩

instance *fm* :: (*countable*, *countable*) *countable*
⟨*proof*⟩

29 Completeness

theorem *strong-completeness*:
fixes p :: ‹(*'f* :: *countable*, *'p* :: *countable*) *fm*›
assumes ‹$\forall (E :: - \Rightarrow$ *'f tm*) *F G*. *Ball X* $[\![E, F, G]\!] \longrightarrow [\![E, F, G]\!]$ *p*›
and ‹*finite* (*vars X*)›
shows ‹$\exists$ *ps*. *set ps* $\subseteq X \wedge$ *ps* $\vdash$ *p*›
⟨*proof*⟩

theorem *completeness*:
fixes p :: ‹(*'f* :: *countable*, *'p* :: *countable*) *fm*›
assumes ‹$\forall (E :: - \Rightarrow$ *'f tm*) *F G*. $[\![E, F, G]\!]$ *p*›
shows ‹$\vdash$ *p*›
⟨*proof*⟩

corollary
fixes p :: ‹(*unit*, *unit*) *fm*›
assumes ‹$\forall (E ::$ *nat* $\Rightarrow$ *unit tm*) *F G*. $[\![E, F, G]\!]$ *p*›
shows ‹$\vdash$ *p*›
⟨*proof*⟩

30 Main Result

abbreviation *valid* :: ‹(*nat*, *nat*) *fm* $\Rightarrow$ *bool*› **where**
‹*valid p* $\equiv \forall (E ::$ *nat* $\Rightarrow$ *nat tm*) *F G*. $[\![E, F, G]\!]$ *p*›

theorem *main*: ‹*valid p* $\longleftrightarrow$ ($\vdash$ *p*)›

⟨*proof*⟩

end

References

[1] L. Henkin. The discovery of my completeness proofs. *Bulletin of Symbolic Logic*, 2(2):127–158, 1996.

[2] R. M. Smullyan. *First-Order Logic*. Springer-Verlag, 1968.