Completeness of the Q_0 higher-order logic

Asta Halkjær Boserup Jonathan Julián Huerta y Munive Anders Schlichtkrul

August 6, 2026

Abstract

We formalize the completeness of Peter B. Andrews' system Q_0, an implementation of Higher-Order Logic (also called simple type theory), following his textbook "An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof". Our work builds on Díaz's formalization of Q_0's syntax, semantics, soundness and consistency. The completeness is with respect to general models. We prove completeness by introducing an abstract consistency property for Q_0 using the framework for abstract consistency properties by From and Schlichtkrull to get a model existence theorem for Q_0. We adapt proofs from Andrews' book to this context.

Contents

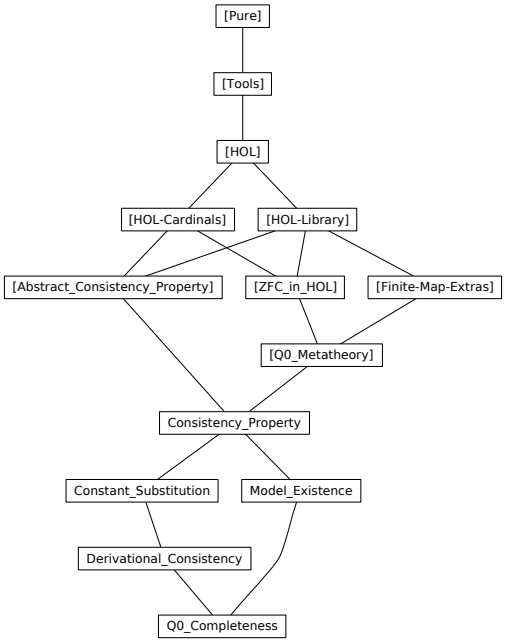

Figure 1: Theory dependency graph

1 Introduction

We give a completeness proof of Peter B. Andrews' system Q_0 for Higher-Order Logic (also called simple type theory) following his textbook "An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof' [And13]. The study of Higher-Order Logic and type theories includes works by Russell [Rus08] with Whitehead [WR13], Church [Chu40], Henkin [Hen50, Hen63, Hen96] and Andrews [And63, And13, And71, And72]. Andrews states in his book that the Q_0 system is specifically based on works by Church [Chu40] and Henkin [Hen50, Hen63] and also his own earlier works [And63, And72]. We build the completeness proof on Díaz's [Día23] formalization of Q_0, and thus also Paulson's Zermelo Fraenkel Set Theory in Higher-Order Logic [Pau19]. Díaz's formalization is combined with the framework by From and Schlichtkrull for abstract consistency properties [FS26, FS25]. This framework fits Andrews' completeness proof better than other such frameworks for logics. The framework for synthetic completeness [Fro23, Fro25] applies only to synthetic proof systems, a narrower range than for the chosen framework, and would limit the extensibility of the results. The frameworks for Beth/Hintikka style completeness proofs [BPT14a, BPT14b, BPT17] and for saturation-based provers [Tou20, BKT25, WTRB22, EBT21, TB21, BT20, WTRB20] were not designed with axiomatic systems like Q_0 in mind.

There are two formalizations of Q_0 in Isabelle, but they do not cover completeness. Specifically there is Díaz's [Día23] formalization of Q_0 on which this theory is built and Schlichtkrull's formalization of Q_0 [Sch23]. Formalizations of other systems for higher-order logic also do not cover completeness. There is specifically Arthan's specification of HOL in ProofPower [Art14a, Art14b, Art02], John Harrison's formalization of the proof system of HOL Light in HOL Light [Har06] and the formalization by Kumar et al. of HOL Light with definitions and a verified implementation in HOL4 [KAMO14, KAMO16, AMKS22, AM23]. Another formalization with the focus on implementation is by Roßkopf and Nipkow [NR21a, NR21b, RN23] who in Isabelle/HOL give both a formalized proof system for terms in Isabelle's metalogic and a formalized implementation of a checker and prove that the checker implements the system. Additionally there are works formalizing the metatheory of the calculus of constructions and the calculus of inductive constructions [Bar96b, Bar96a, BW96, Bar97, SBF+20].

The only other formalizations of completeness with general models we know of are for second-order logic by Kock and Kirst [KK22] in Rocq and by From and Schlichtkrull in Isabelle/HOL [FS26, FS25].

theory *Consistency_Property* **imports**
"Abstract_Consistency_Property.Abstract_Consistency_Property"
"Q0_Metatheory.Syntax"
begin

2 Consistency Property

inductive *confl_class* :: ‹*form list* ⇒ *form list* ⇒ *bool*› (**infix** ‹$\rightsquigarrow\!\!\!\!\times$› *50*) **where**
CFalse: ‹[] $\rightsquigarrow\!\!\!\!\times$ [F_o]›
| *CNot*: ‹[$\sim^{\mathcal{Q}}$ A] $\rightsquigarrow\!\!\!\!\times$ [A]› **if** ‹$A \in wffs_o$›

inductive *alpha_class* :: ‹*form list* ⇒ *form list* ⇒ *bool*› (**infix** ‹$\rightsquigarrow_\alpha$› *50*) **where**
CBool: ‹[A] $\rightsquigarrow_\alpha$ [$A =_o T_o$]› **if** ‹$A \in wffs_o$›
| *CIota*: ‹[] $\rightsquigarrow_\alpha$ [$\iota \cdot (Q_i \cdot A) =_i A$]› **if** ‹$A \in wffs_i$›
| *CRefl*: ‹[] $\rightsquigarrow_\alpha$ [$A =_\alpha A$]› **if** ‹$A \in wffs_\alpha$›
| *CTrans*: ‹[$A =_\alpha B$, $B =_\alpha C$] $\rightsquigarrow_\alpha$ [$A =_\alpha C$]› **if** ‹$A \in wffs_\alpha$› **and** ‹$B \in wffs_\alpha$› **and** ‹$C \in wffs_\alpha$›
| *CCong*: ‹[$A =_\alpha B$] $\rightsquigarrow_\alpha$ [$C \cdot A =_\beta C \cdot B$]› **if** ‹$A \in wffs_\alpha$› **and** ‹$B \in wffs_\alpha$› **and** ‹$C \in wffs_{\alpha \to \beta}$›
| *CSubst*: ‹[] $\rightsquigarrow_\alpha$ [$(\lambda x_\alpha.\ B) \cdot A =_\beta \mathbf{S}\ \{(x, \alpha) \rightarrowtail A\}\ B$]› **if**
‹$A \in wffs_\alpha$› **and** ‹$B \in wffs_\beta$› **and** ‹*free_vars* $A = \{\}$›

inductive *beta_class* :: ‹*form list* ⇒ *form list* ⇒ *bool*› (**infix** ‹$\rightsquigarrow_\beta$› *50*) **where**
CLEM: ‹[] $\rightsquigarrow_\beta$ [A, $\sim^{\mathcal{Q}} A$]› **if** ‹$A \in wffs_o$›

inductive *gamma_class* :: ‹*form list* ⇒ (*form set* ⇒ _) × (*form* ⇒ _) ⇒ *bool*› (**infix** ‹$\rightsquigarrow_\gamma$› *50*) **where**
CExt: ‹[$A =_{\alpha \to \beta} B$] $\rightsquigarrow_\gamma$ (λ_. $wffs_\alpha$, λC. [$A \cdot C =_\beta B \cdot C$])› **if**
‹$A \in wffs_{\alpha \to \beta}$› **and** ‹$B \in wffs_{\alpha \to \beta}$›

2.1 Negated Equality

inductive *ineq_match* :: ‹*form* ⇒ *type* × *type* × *form* × *form* ⇒ *bool*› **where**
‹*ineq_match* ($\sim^{\mathcal{Q}}$ ($A =_{\alpha \to \beta} B$)) (α, β, A, B)›

inductive-cases *ineq_matchE* [*elim*]: ‹*ineq_match* ($\sim^{\mathcal{Q}}$ ($A =_{\alpha \to \beta} B$)) (α', β', A', B')›

lemma *ineq_match_uniq* [*dest*]:
assumes ‹*ineq_match* C (α, β, A, B)›
and ‹*ineq_match* C (α', β', A', B')›
shows ‹$\alpha = \alpha' \wedge \beta = \beta' \wedge A = A' \wedge B = B'$›
⟨*proof*⟩

lemma *THE_ineq_match*:
assumes ‹*ineq_match* C (α, β, A, B)›
shows ‹(*THE* (α, β, A, B). *ineq_match* C (α, β, A, B)) = (α, β, A, B)›
⟨*proof*⟩

lemma *ineq_matchD* [*dest*]:
assumes ‹*ineq_match* C (α, β, A, B)›
shows ‹$C = \sim^{\mathcal{Q}}$ ($A =_{\alpha \to \beta} B$)›
⟨*proof*⟩

lemma *ineq_matchI* [*intro*]:
assumes ‹$C = \sim^{\mathcal{Q}}$ ($A =_{\alpha \to \beta} B$)›
shows ‹*ineq_match* C (α, β, A, B)›
⟨*proof*⟩

2.2 Delta

fun *delta* :: ‹*form* ⇒ *nat* ⇒ *form list*› **where**
CDelta: ‹*delta* C c =
(*if* $C \in wffs_o \wedge (\exists \alpha\ \beta\ A\ B.$ *ineq_match* C (α, β, A, B)) *then*
case THE (α, β, A, B). *ineq_match* C (α, β, A, B) *of*
(α, β, A, B) ⇒ [$\sim^{\mathcal{Q}}$ ($A \cdot \{\!|c|\!\}_\alpha =_\beta B \cdot \{\!|c|\!\}_\alpha$)]
else [])›

lemma *ineq_match_delta* [*simp*]:
assumes ‹$C \in wffs_o$› ‹*ineq_match* C (α, β, A, B)›
shows ‹*delta* C c = [$\sim^{\Omega}$ ($A \cdot \{\!|c|\!\}_\alpha =_\beta B \cdot \{\!|c|\!\}_\alpha$)]›
⟨*proof*⟩

lemma *delta*:
assumes ‹$A \in wffs_{\alpha \to \beta}$› ‹$B \in wffs_{\alpha \to \beta}$›
shows ‹*delta* ($\sim^{\Omega}$ ($A =_{\alpha \to \beta} B$)) c = [$\sim^{\Omega}$ ($A \cdot \{\!|c|\!\}_\alpha =_\beta B \cdot \{\!|c|\!\}_\alpha$)]›
⟨*proof*⟩

2.3 Operations

definition ‹*logical_names* ≡ {$\mathfrak{c}_Q$, $\mathfrak{c}_\iota$}›
abbreviation ‹*is_logical_name* $c \equiv c \in$ *logical_names*›

lemma *logical_name_simps*[*simp*]:
shows ‹*is_logical_name* $\mathfrak{c}_Q$›
and ‹*is_logical_name* $\mathfrak{c}_\iota$›
⟨*proof*⟩

definition ‹*is_param* $c \equiv \neg$ *is_logical_name* c›

fun *map_con* :: ‹(*nat* ⇒ *nat*) ⇒ *form* ⇒ *form*› **where**
‹*map_con* _ (x_α) = (x_α)›
| ‹*map_con* f ($\{\!|c|\!\}_\alpha$) = (*if is_logical_name* $c \vee$ *is_logical_name* (f c) *then* $\{\!|c|\!\}_\alpha$ *else* $\{\!|f\ c|\!\}_\alpha$)›
| ‹*map_con* f ($A \cdot B$) = *map_con* f $A \cdot$ *map_con* f B›
| ‹*map_con* f ($\lambda x_\alpha.\ A$) = $\lambda x_\alpha.$ *map_con* f A›

fun *cons_form* :: ‹*form* ⇒ *nat set*› **where**
‹*cons_form* (x_α) = {}›
| ‹*cons_form* ($\{\!|c|\!\}_\alpha$) = (*if is_logical_name* c *then* {} *else* {c})›
| ‹*cons_form* ($A \cdot B$) = *cons_form* $A \cup$ *cons_form* B›
| ‹*cons_form* ($\lambda x_\alpha.\ A$) = *cons_form* A›

2.4 Lemmas

This property is really what dodging the logical constants is all about.

proposition ‹*map_con* f ($\sim^{\Omega}$ A) = $\sim^{\Omega}$ (*map_con* f A)›
⟨*proof*⟩

lemma *map_con_id* [*simp*]: ‹*map_con id* = *id*›
⟨*proof*⟩

lemma *map_con_cong* [*simp*]:
assumes ‹$\forall x \in$ *cons_form* $A.\ f\ x = g\ x$›
shows ‹*map_con* f A = *map_con* g A›
⟨*proof*⟩

lemma *wff_map_con* [*iff*]: ‹*map_con* f $A \in wffs_\alpha \longleftrightarrow A \in wffs_\alpha$›
⟨*proof*⟩

lemma *finite_cons_form* [*simp*]: ‹*finite* (*cons_form* A)›
⟨*proof*⟩

lemma *map_con_ineq_match* [*intro*]:
assumes ‹*ineq_match* C (α, β, A, B)›
shows ‹*ineq_match* (*map_con* f C) (α, β, *map_con* f A, *map_con* f B)›
⟨*proof*⟩

lemma *free_vars_map_con* [*simp*]: ‹*free_vars* (*map_con* f A) = *free_vars* A›
⟨*proof*⟩

lemma *map_con_substitute* [*simp*]: ‹*map_con* f (*substitute* {(x, α) ⇀ A} B)
= *substitute* {(x, α) ⇀ *map_con* f A} (*map_con* f B)›

⟨*proof*⟩

2.5 Parameter Substitution Inversion

lemma *map_con_FVar* [*dest*]: ‹*map_con f A* = x_α ⟹ *A* = x_α›
⟨*proof*⟩

lemma *map_con_FCon_not_param* [*dest*]: ‹*map_con f A* = ⦇*c*⦈$_\alpha$ ⟹ ¬ *is_param c* ⟹ *A* = ⦇*c*⦈$_\alpha$›
⟨*proof*⟩

lemma *map_con_FApp* [*dest*!]:
assumes ‹*map_con f A* = *B* ∙ *C*›
shows ‹∃ *B' C'*. *map_con f B'* = *B* ∧ *map_con f C'* = *C* ∧ *A* = *B'* ∙ *C'*›
⟨*proof*⟩

lemma *map_con_FAbs* [*dest*!]:
assumes ‹*map_con f A* = λx_α. *B*›
shows ‹∃ *B'*. *map_con f B'* = *B* ∧ *A* = λx_α. *B'*›
⟨*proof*⟩

lemma *map_con_cQ* [*dest*]: ‹*map_con f A* = ⦇$\mathfrak{c}_Q$⦈$_\alpha$ ⟹ *A* = ⦇$\mathfrak{c}_Q$⦈$_\alpha$›
⟨*proof*⟩

lemma *map_con_equality_of_type* [*dest*]:
assumes ‹*map_con f A* = *B* =$_\alpha$ *C*›
shows ‹∃ *B' C'*. *map_con f B'* = *B* ∧ *map_con f C'* = *C* ∧ *A* = *B'* =$_\alpha$ *C'*›
⟨*proof*⟩

lemma *map_con_neg* [*dest*]: ‹*map_con f A* = ∼$^{\mathfrak{Q}}$ *B* ⟹ ∃ *B'*. *map_con f B'* = *B* ∧ *A* = ∼$^{\mathfrak{Q}}$ *B'*›
⟨*proof*⟩

lemma *ineq_match_map_con* [*dest*]:
assumes ‹*ineq_match* (*map_con f C*) (α, β, *A*, *B*)›
shows ‹∃ *A' B'*. *map_con f A'* = *A* ∧ *map_con f B'* = *B* ∧ *ineq_match C* (α, β, *A'*, *B'*)›
⟨*proof*⟩

2.6 Interpretations

interpretation *P*: *Params map_con cons_form is_param*
⟨*proof*⟩

interpretation *C*: *Confl map_con cons_form is_param confl_class*
⟨*proof*⟩

interpretation *A*: *Alpha map_con cons_form is_param alpha_class*
⟨*proof*⟩

interpretation *B*: *Beta map_con cons_form is_param beta_class*
⟨*proof*⟩

interpretation *G*: *Gamma map_con map_con cons_form is_param gamma_class*
⟨*proof*⟩

interpretation *D*: *Delta map_con cons_form is_param delta*
⟨*proof*⟩

abbreviation *Kinds* :: ‹(*nat*, *form*) *kind list*› **where**
‹*Kinds* ≡ [*C.kind*, *A.kind*, *B.kind*, *G.kind*, *D.kind*]›

lemma *prop$_E$_Kinds*:
assumes ‹*P.sat$_E$ C.kind C*› ‹*P.sat$_E$ A.kind C*› ‹*P.sat$_E$ B.kind C*› ‹*P.sat$_E$ G.kind C*› ‹*P.sat$_E$ D.kind C*›
shows ‹*P.prop$_E$ Kinds C*›
⟨*proof*⟩

interpretation *Consistency_Kinds map_con cons_form is_param Kinds*

⟨*proof*⟩

interpretation *Maximal_Consistency map_con cons_form is_param Kinds*
⟨*proof*⟩

end
theory *Constant_Substitution* **imports**
Consistency_Property
"Q0_Metatheory.Elementary_Logic"
begin

fun *const_subst* :: ‹*con* ⇒ *nat* ⇒ *form* ⇒ *form*› (‹$\mathbf{S}_c$ _ _ _› [*51*, *51*, *51*])
where ‹$\mathbf{S}_c$ (*c*, *β*) *x* (y_α) = y_α›
| ‹$\mathbf{S}_c$ (*c*, *β*) *x* ($\{\!|d|\!\}_\alpha$) = (*if c* = *d* ∧ *β* = *α then* (x_α) *else* ($\{\!|d|\!\}_\alpha$))›
| ‹$\mathbf{S}_c$ (*c*, *β*) *x* (*A* ▪ *B*) = ($\mathbf{S}_c$ (*c*, *β*) *x A*) ▪ ($\mathbf{S}_c$ (*c*, *β*) *x B*)›
| ‹$\mathbf{S}_c$ (*c*, *β*) *x* (λy_α. *A*) = (λy_α. $\mathbf{S}_c$ (*c*, *β*) *x A*)›

lemma *idemp_const_subst*:
assumes ‹*c* ∉ *cons_form F*›
and ‹¬ *is_logical_name c*›
shows ‹$\mathbf{S}_c$ (*c*, *α*) *x F* = *F*›
⟨*proof*⟩

lemma *const_subst_laws*:
assumes ‹¬ *is_logical_name c*›
shows ‹$\mathbf{S}_c$ (*c*, *τ*) *x* (*A* ∧$^{\mathcal{Q}}$ *B*) = ($\mathbf{S}_c$ (*c*, *τ*) *x A*) ∧$^{\mathcal{Q}}$ ($\mathbf{S}_c$ (*c*, *τ*) *x B*)›
and ‹$\mathbf{S}_c$ (*c*, *τ*) *x* (*A* ⊃$^{\mathcal{Q}}$ *B*) = ($\mathbf{S}_c$ (*c*, *τ*) *x A*) ⊃$^{\mathcal{Q}}$ ($\mathbf{S}_c$ (*c*, *τ*) *x B*)›
and ‹$\mathbf{S}_c$ (*c*, *τ*) *x* (*A* ≡$^{\mathcal{Q}}$ *B*) = ($\mathbf{S}_c$ (*c*, *τ*) *x A*) ≡$^{\mathcal{Q}}$ ($\mathbf{S}_c$ (*c*, *τ*) *x B*)›
and ‹$\mathbf{S}_c$ (*c*, *τ*) *x* (T_o) = T_o›
and ‹$\mathbf{S}_c$ (*c*, *τ*) *x* (F_o) = F_o›
and ‹$\mathbf{S}_c$ (*c*, *τ*) *x* (∀ z_α. *A*) = (∀ z_α. $\mathbf{S}_c$ (*c*, *τ*) *x A*)›
and ‹$\mathbf{S}_c$ (*c*, *τ*) *x* (*A* =$_\alpha$ *B*) = (($\mathbf{S}_c$ (*c*, *τ*) *x A*) =$_\alpha$ ($\mathbf{S}_c$ (*c*, *τ*) *x B*))›
⟨*proof*⟩

lemma *const_subst_axiom_if_no_c*:
assumes ‹*c* ∉ *cons_form A*›
and ‹¬ *is_logical_name c*›
and ‹*A* ∈ *axioms*›
shows ‹($\mathbf{S}_c$ (*c*, *α*) *x A*) ∈ *axioms*›
⟨*proof*⟩

lemma *axiom_1_const_subst*:
assumes ‹¬ *is_logical_name c*›
shows ‹$\mathbf{S}_c$ (*c*, *τ*) *x* ($\mathfrak{g}_{o\to o}$ ▪ T_o ∧$^{\mathcal{Q}}$ $\mathfrak{g}_{o\to o}$ ▪ F_o ≡$^{\mathcal{Q}}$ ∀ $\mathfrak{x}_o$. $\mathfrak{g}_{o\to o}$ ▪ $\mathfrak{x}_o$) ∈ *axioms*›
⟨*proof*⟩

lemma *axiom_2_const_subst*:
assumes ‹¬ *is_logical_name c*›
shows ‹$\mathbf{S}_c$ (*c*, *τ*) *x* (($\mathfrak{x}_\alpha$ =$_\alpha$ $\mathfrak{y}_\alpha$) ⊃$^{\mathcal{Q}}$ ($\mathfrak{h}_{\alpha\to o}$ ▪ $\mathfrak{x}_\alpha$ ≡$^{\mathcal{Q}}$ $\mathfrak{h}_{\alpha\to o}$ ▪ $\mathfrak{y}_\alpha$)) ∈ *axioms*›
⟨*proof*⟩

lemma *axiom_3_const_subst*:
assumes ‹¬ *is_logical_name c*›
shows ‹$\mathbf{S}_c$ (*c*, *τ*) *x* (($\mathfrak{f}_{\alpha\to\beta}$ =$_{\alpha\to\beta}$ $\mathfrak{g}_{\alpha\to\beta}$) ≡$^{\mathcal{Q}}$ ∀ $\mathfrak{x}_\alpha$. ($\mathfrak{f}_{\alpha\to\beta}$ ▪ $\mathfrak{x}_\alpha$ =$_\beta$ $\mathfrak{g}_{\alpha\to\beta}$ ▪ $\mathfrak{x}_\alpha$)) ∈ *axioms*›
⟨*proof*⟩

lemma *const_subst_wffs*:
assumes ‹*A* ∈ *wffs*$_\alpha$›
shows ‹$\mathbf{S}_c$ (*c*, *τ*) *x A* ∈ *wffs*$_\alpha$›
⟨*proof*⟩

lemma *axiom_4_1_con_const_subst*:
assumes ‹¬ *is_logical_name c*›
and ‹*A* ∈ *wffs*$_\alpha$›

and ‹$(x, \tau) \neq (y, \alpha)$›
shows ‹$\mathbf{S}_c\ (c, \tau)\ x\ ((\lambda y_\alpha.\ \{\!|d|\!\}_\beta) \cdot A =_\beta \{\!|d|\!\}_\beta) \in axioms$›
⟨*proof*⟩

lemma *axiom_4_1_var_const_subst*:
assumes ‹$\neg$ *is_logical_name* c›
and ‹$A \in wffs_\alpha$›
and ‹$y_\beta \neq z_\alpha$›
shows ‹$\mathbf{S}_c\ (c, \tau)\ x\ ((\lambda z_\alpha.\ y_\beta) \cdot A =_\beta y_\beta) \in axioms$›
⟨*proof*⟩

lemma *axiom_4_2_const_subst*:
assumes ‹$\neg$ *is_logical_name* c›
and ‹$A \in wffs_\alpha$›
shows ‹$\mathbf{S}_c\ (c, \tau)\ x\ ((\lambda z_\alpha.\ z_\alpha) \cdot A =_\alpha A) \in axioms$›
⟨*proof*⟩

lemma *axiom_4_3_const_subst*:
assumes ‹$\neg$ *is_logical_name* c›
and ‹$A \in wffs_\alpha$›
and ‹$B \in wffs_{\gamma\to\beta}$›
and ‹$C \in wffs_\gamma$›
shows ‹$\mathbf{S}_c\ (c, \tau)\ x\ ((\lambda y_\alpha.\ B \cdot C) \cdot A =_\beta ((\lambda y_\alpha.\ B) \cdot A) \cdot ((\lambda y_\alpha.\ C) \cdot A)) \in axioms$›
⟨*proof*⟩

lemma *in_var_const_subst*:
assumes ‹$(y, \gamma) \notin vars\ A$›
and ‹$(y, \gamma) \in vars\ (\mathbf{S}_c\ (c, \tau)\ x\ A)$›
shows ‹$y = x \wedge \gamma = \tau$›
⟨*proof*⟩

lemma *axiom_4_4_const_subst*:
assumes ‹$\neg$ *is_logical_name* c›
and ‹$A \in wffs_\alpha$› **and** ‹$B \in wffs_\delta$› **and** ‹$(y, \gamma) \notin \{(z, \alpha)\} \cup vars\ A$›
and ‹$(x,\tau) \notin vars\ ((\lambda z_\alpha.\ \lambda y_\gamma.\ B) \cdot A =_{\gamma\to\delta} (\lambda y_\gamma.\ (\lambda z_\alpha.\ B) \cdot A))$›
shows ‹$\mathbf{S}_c\ (c, \tau)\ x\ ((\lambda z_\alpha.\ \lambda y_\gamma.\ B) \cdot A =_{\gamma\to\delta} (\lambda y_\gamma.\ (\lambda z_\alpha.\ B) \cdot A)) \in axioms$›
⟨*proof*⟩

lemma *axiom_4_5_const_subst*:
assumes ‹$\neg$ *is_logical_name* c›
and ‹$A \in wffs_\alpha$› **and** ‹$B \in wffs_\delta$›
shows ‹$\mathbf{S}_c\ (c, \tau)\ x\ ((\lambda z_\alpha.\ \lambda z_\alpha.\ B) \cdot A =_{\alpha\ \to\ \delta}\ \lambda z_\alpha.\ B) \in axioms$›
⟨*proof*⟩

lemma *axiom_5_const_subst*:
assumes ‹$\neg$ *is_logical_name* c›
shows ‹$\mathbf{S}_c\ (c, \tau)\ x\ (\iota \cdot (Q_i \cdot \mathfrak{y}_i) =_i \mathfrak{y}_i) \in axioms$›
⟨*proof*⟩

lemma *const_subst_axiom*:
assumes ‹$\neg$ *is_logical_name* c›
and ‹$(x,\tau) \notin vars\ A$›
and ‹$A \in axioms$›
shows ‹$(\mathbf{S}_c\ (c, \tau)\ x\ A) \in axioms$›
⟨*proof*⟩

lemma *is_subform_at_const_subst*:
assumes ‹$A \preceq_p C$›
shows ‹$\mathbf{S}_c\ (c, \tau)\ x\ A \preceq_p \mathbf{S}_c\ (c, \tau)\ x\ C$›
⟨*proof*⟩

lemma *is_replacement_at_const_subst*:
assumes ‹$C\langle\!|p \leftarrow B|\!\rangle \rhd D$›
shows ‹$(\mathbf{S}_c\ (c, \tau)\ x\ C)\langle\!|p \leftarrow \mathbf{S}_c\ (c, \tau)\ x\ B|\!\rangle \rhd \mathbf{S}_c\ (c, \tau)\ x\ D$›

⟨*proof*⟩

lemma *is_rule_R_app_const_subst*:
assumes ‹$c \notin$ *logical_names*›
and ‹$(x, \tau) \notin$ *vars* $D \cup$ *vars* $C \cup$ *vars* E›
and ‹*is_rule_R_app* p D C E›
shows ‹*is_rule_R_app* p ($\mathbf{S}_c$ (c, τ) x D) ($\mathbf{S}_c$ (c, τ) x C) ($\mathbf{S}_c$ (c, τ) x E)›
⟨*proof*⟩

fun *const_subst_proof* :: ‹*con* ⇒ *nat* ⇒ *form list* ⇒ *form list*› (‹$\mathbf{S}_{cp}$ _ _ _› [*51*, *51*, *51*]) **where**
‹$\mathbf{S}_{cp}$ (c, β) x $\mathcal{P}$ = *map* (λA. $\mathbf{S}_c$ (c, β) x A) $\mathcal{P}$›

lemma *nil_is_proof*:
‹*is_proof* []›
⟨*proof*⟩

thm *theorem_is_derivable_form*
lemma *is_proof_induct* [*consumes 1*, *case_names p_nil p_axiom p_rule_R*]:
assumes ‹*is_proof* $\mathcal{P}$›
and *p_nil*: ‹P []›
and *p_axiom*: ‹($\bigwedge A$ $\mathcal{P}$. $A \in$ *axioms* ⟹ *is_proof* $\mathcal{P}$ ⟹ P $\mathcal{P}$ ⟹ P ($\mathcal{P}$ @ [A]))›
and *p_rule_R*: ‹($\bigwedge \mathcal{P}$ $\mathcal{P}'$ E $\mathcal{P}''$ C p D. *is_proof* $\mathcal{P}$ ⟹ P $\mathcal{P}$ ⟹ *prefix* ($\mathcal{P}'$ @ [E]) $\mathcal{P}$ ⟹ *prefix* ($\mathcal{P}''$ @ [C]) $\mathcal{P}$ ⟹ *is_rule_R_app* p D C E ⟹ P ($\mathcal{P}$ @ [D]))›
shows ‹P $\mathcal{P}$›
⟨*proof*⟩

lemma *is_proof_R_intro*:
assumes ‹*is_rule_R_app* p D C E›
and ‹*is_proof* S›
and ‹*prefix* (S' @ [E]) S›
and ‹*prefix* (S'' @ [C]) S›
shows ‹*is_proof* (S @ [D])›
⟨*proof*⟩

definition ‹$vars_p$ ($\mathcal{S}$::*form list*) = *vars* (*List.set* $\mathcal{S}$)›

lemma *is_proof_const_subst*:
assumes ‹*is_proof* $\mathcal{P}$›
and ‹$c \notin$ *logical_names*›
and ‹$(x, \beta) \notin vars_p$ $\mathcal{P}$›
shows ‹*is_proof* ($\mathbf{S}_{cp}$ (c, β) x $\mathcal{P}$)›
⟨*proof*⟩

lemma *finite_vars*$_p$: ‹*finite* ($vars_p$ $\mathcal{S}$)›
⟨*proof*⟩

lemma *fresh_free_vars_const_subst*:
assumes ‹$(x, \tau) \notin$ *vars* A›
shows ‹*free_vars* ($\mathbf{S}_c$ (c, τ) x A) = *free_vars* A ∨ *free_vars* ($\mathbf{S}_c$ (c, τ) x A) = *free_vars* $A \cup \{(x, \tau)\}$›
⟨*proof*⟩

lemma *const_subst_binders_at*:
shows ‹*binders_at* ($\mathbf{S}_c$ (c, τ) x C) p = *binders_at* C p›
⟨*proof*⟩

lemma *in_binders_at_in_vars*:
assumes ‹$(x, \tau) \in$ *binders_at* C p›
shows ‹$(x, \tau) \in$ *vars* C›
⟨*proof*⟩

lemma *const_subst_preserves_binders_at*:
assumes ‹$C' = \mathbf{S}_c$ (c, τ) x C›
shows ‹*binders_at* C p = *binders_at* C' p›
⟨*proof*⟩

lemma *capture_exposed_vars_at_const_subst1*:
assumes ‹$p \in$ *positions* C›
and ‹$C' = \mathbf{S}_c\ (c,\ \beta)\ x\ C$›
shows ‹*capture_exposed_vars_at* $p\ C\ \mathcal{H}$ = *capture_exposed_vars_at* $p\ C'\ \mathcal{H}$›
⟨*proof*⟩

lemma *capture_exposed_vars_at_const_subst2*:
assumes ‹$p \in$ *positions* C›
and ‹$C' = \mathbf{S}_c\ (c,\ \beta)\ x\ C$›
and ‹$E' = \mathbf{S}_c\ (c,\ \beta)\ x\ E$›
and ‹$(x,\ \beta) \notin$ *vars* $C \cup$ *vars* E›
shows ‹*capture_exposed_vars_at* $p\ C\ E$ = *capture_exposed_vars_at* $p\ C'\ E'$›
⟨*proof*⟩

lemma *capture_exposed_vars_at_intersection_const_subst*:
assumes ‹$p \in$ *positions* C›
and ‹*capture_exposed_vars_at* $p\ C\ E \cap$ *capture_exposed_vars_at* $p\ C\ As = \{\}$›
and ‹$C' = \mathbf{S}_c\ (c,\ \tau)\ x\ C$›
and ‹$E' = \mathbf{S}_c\ (c,\ \tau)\ x\ E$›
and ‹$(x,\ \tau) \notin$ *vars* $C \cup$ *vars* E›
shows ‹*capture_exposed_vars_at* $p\ C'\ E' \cap$ *capture_exposed_vars_at* $p\ C'\ As = \{\}$›
⟨*proof*⟩

lemma *is_rule_R'_app_const_subst*:
assumes ‹$C' = (\mathbf{S}_c\ (c,\ \tau)\ x\ C)$›
and ‹$D' = (\mathbf{S}_c\ (c,\ \tau)\ x\ D)$›
and ‹$E' = (\mathbf{S}_c\ (c,\ \tau)\ x\ E)$›
and ‹*is_rule_R'_app As* $p\ D\ C\ E$›
and ‹*is_hyps As*›
and ‹$c \notin$ *logical_names*›
and ‹$(x,\ \tau) \notin$ *vars* $D \cup$ *vars* $C \cup$ *vars* E›
and ‹$c \notin$ *P.params As*›
shows ‹*is_rule_R'_app As* $p\ D'\ C'\ E'$›
⟨*proof*⟩

lemma *is_hyp_proof_induct* [*consumes 1*, *case_names hp_nil hp_hyp hp_seq hp_rule_R'*]:
assumes ‹*is_hyp_proof* $\mathcal{H}\ \mathcal{P}_1\ \mathcal{P}_2$›
and ‹$P\ []$›
and ‹$\bigwedge A\ \mathcal{P}_2.\ A \in \mathcal{H} \Longrightarrow$ *is_hyp_proof* $\mathcal{H}\ \mathcal{P}_1\ \mathcal{P}_2 \Longrightarrow P\ \mathcal{P}_2 \Longrightarrow P\ (\mathcal{P}_2\ @\ [A])$›
and ‹$\bigwedge A\ \mathcal{P}_2.\ A \in$ *lset* $\mathcal{P}_1 \Longrightarrow$ *is_hyp_proof* $\mathcal{H}\ \mathcal{P}_1\ \mathcal{P}_2 \Longrightarrow P\ \mathcal{P}_2 \Longrightarrow P\ (\mathcal{P}_2\ @\ [A])$›
and ‹$\bigwedge S'\ E\ \mathcal{P}_2\ S''\ C\ p\ D.$ *prefix* $(S'\ @\ [E])\ \mathcal{P}_2 \Longrightarrow$ *prefix* $(S''\ @\ [C])\ \mathcal{P}_2$
$\Longrightarrow$ *is_rule_R'_app* $\mathcal{H}\ p\ D\ C\ E \Longrightarrow$ *is_hyp_proof* $\mathcal{H}\ \mathcal{P}_1\ \mathcal{P}_2 \Longrightarrow P\ \mathcal{P}_2 \Longrightarrow P\ (\mathcal{P}_2\ @\ [D])$›
shows ‹$P\ \mathcal{P}_2$›
⟨*proof*⟩

lemma *is_hyp_proof_R'_intro*:
assumes ‹*is_rule_R'_app H p D C E*›
and ‹*is_hyp_proof H S1 S*›
and ‹*prefix* $(S'\ @\ [E])\ S$›
and ‹*prefix* $(S''\ @\ [C])\ S$›
shows ‹*is_hyp_proof H S1* $(S\ @\ [D])$›
⟨*proof*⟩

lemma *is_hyp_proof_const_subst*:
assumes ‹*is_hyp_proof* $\mathcal{H}\ \mathcal{P}_1\ \mathcal{P}_2$›
and ‹*is_hyps* $\mathcal{H}$›
and ‹$c \notin$ *logical_names*›
and ‹$(x,\ \beta) \notin$ *vars*$_p$ $\mathcal{P}_2$›
and ‹$c \notin$ *P.params* $\mathcal{H}$›
shows ‹*is_hyp_proof* $\mathcal{H}\ (\mathbf{S}_{cp}\ (c,\ \beta)\ x\ \mathcal{P}_1)\ (\mathbf{S}_{cp}\ (c,\ \beta)\ x\ \mathcal{P}_2)$›
⟨*proof*⟩

lemma *is_hyp_proof_of_const_subst*:

assumes ‹$P' = \mathbf{S}_{cp}\ (c,\ \alpha)\ x\ P$›
and ‹$Ts' = \mathbf{S}_{cp}\ (c,\ \alpha)\ x\ Ts$›
and ‹$form' = \mathbf{S}_{c}\ (c,\ \alpha)\ x\ A$›
and ‹$is\_hyp\_proof\_of\ As\ Ts\ P\ A$›
and ‹$(x,\ \alpha) \notin vars\ As$›
and ‹$(x,\ \alpha) \notin vars\ B$›
and ‹$c \notin logical\_names$›
and ‹$(x,\ \alpha) \notin vars_p\ Ts$›
and ‹$(x,\ \alpha) \notin vars_p\ P$›
and ‹$c \notin P.params\ As$›
shows ‹$is\_hyp\_proof\_of\ As\ Ts'\ P'\ form'$›
⟨*proof*⟩

end
theory *Derivational_Consistency* **imports**
Constant_Substitution
"Q0_Metatheory.Consistency"
begin

3 Derivational Consistency

lemma *inconsistent_imp_hyps*:
assumes ‹$is\_inconsistent\_set\ \mathcal{H}$›
shows ‹$is\_hyps\ \mathcal{H}$›
⟨*proof*⟩

Instead of introducing derivations from infinite sets of hypotheses, we consider all subsets of possibly infinite consistent sets.

definition *is_consistent_set* :: ‹*form set* ⇒ *bool*› **where**
‹$is\_consistent\_set\ \mathcal{G} \equiv \forall \mathcal{H} \subseteq \mathcal{G}.\ \neg\ is\_inconsistent\_set\ \mathcal{H}$›

lemma *is_consistent_dest* [*dest*]:
assumes ‹$is\_consistent\_set\ \mathcal{G}$›
and ‹$\mathcal{H} \subseteq \mathcal{G}$›
shows ‹$\neg\ is\_inconsistent\_set\ \mathcal{H}$›
⟨*proof*⟩

lemma *is_consistent_intro* [*intro*]:
assumes ‹$\bigwedge \mathcal{H}.\ \mathcal{H} \subseteq \mathcal{G} \Longrightarrow is\_hyps\ \mathcal{H} \Longrightarrow \neg\ is\_inconsistent\_set\ \mathcal{H}$›
shows ‹$is\_consistent\_set\ \mathcal{G}$›
⟨*proof*⟩

lemma *is_inconsistent_set_insert*:
assumes ‹$is\_inconsistent\_set\ (\{A\} \cup \mathcal{H})$›
and ‹$\mathcal{H} \vdash A$›
shows ‹$is\_inconsistent\_set\ \mathcal{H}$›
⟨*proof*⟩

lemma *is_consistent_set_insert*:
assumes $\mathcal{G}$: ‹$is\_consistent\_set\ \mathcal{G}$›
and $\mathcal{H}$: ‹$\mathcal{H} \subseteq \mathcal{G}$› ‹$\mathcal{H} \vdash A$›
shows ‹$is\_consistent\_set\ (\{A\} \cup \mathcal{G})$›
⟨*proof*⟩

lemma *is_consistent_set_union*:
assumes X: ‹$finite\ X$›
and $\mathcal{G}$: ‹$is\_consistent\_set\ \mathcal{G}$›
and $\mathcal{H}$: ‹$\mathcal{H} \subseteq \mathcal{G}$› ‹$\forall A \in X.\ \mathcal{H} \vdash A$›
shows ‹$is\_consistent\_set\ (X \cup \mathcal{G})$›
⟨*proof*⟩

lemma *is_inconsistent_set_mono*:
assumes ‹$is\_inconsistent\_set\ \mathcal{H}$›

and ‹$\mathcal{H} \subseteq \mathcal{G}$›
and ‹*is_hyps* $\mathcal{G}$›
shows ‹*is_inconsistent_set* $\mathcal{G}$›
⟨*proof*⟩

3.1 Conflicts

interpretation *DC*: *Derivational_Confl map_con cons_form is_param confl_class is_consistent_set*
⟨*proof*⟩

3.2 Conjunctive Consistency

lemma *pre_is_taut*:
assumes ‹$A \in$ *pwffs*›
and ‹$B \in$ *pwffs*›
shows ‹*is_tautology* $((\sim^{\mathfrak{Q}} (A \supset^{\mathfrak{Q}} B)) \supset^{\mathfrak{Q}} A)$›
and ‹*is_tautology* $((\sim^{\mathfrak{Q}} (A \supset^{\mathfrak{Q}} B)) \supset^{\mathfrak{Q}} \sim^{\mathfrak{Q}} B)$›
and ‹*is_tautology* $((A \supset^{\mathfrak{Q}} F_o) \supset^{\mathfrak{Q}} \sim^{\mathfrak{Q}} A)$›
and ‹*is_tautology* $(\sim^{\mathfrak{Q}} A \supset^{\mathfrak{Q}} (A \supset^{\mathfrak{Q}} F_o))$›
and ‹*is_tautology* $(A \vee^{\mathfrak{Q}} \sim^{\mathfrak{Q}} A)$›
and ‹*is_tautology* $(\sim^{\mathfrak{Q}}\sim^{\mathfrak{Q}} A \supset^{\mathfrak{Q}} A)$›
⟨*proof*⟩

lemma *is_taut*:
assumes ‹$A \in$ *wffs*$_o$›
and ‹$B \in$ *wffs*$_o$›
shows ‹*is_tautologous* $((\sim^{\mathfrak{Q}} (A \supset^{\mathfrak{Q}} B)) \supset^{\mathfrak{Q}} A)$›
and ‹*is_tautologous* $((\sim^{\mathfrak{Q}} (A \supset^{\mathfrak{Q}} B)) \supset^{\mathfrak{Q}} \sim^{\mathfrak{Q}} B)$›
and ‹*is_tautologous* $((A \supset^{\mathfrak{Q}} F_o) \supset^{\mathfrak{Q}} \sim^{\mathfrak{Q}} A)$›
and ‹*is_tautologous* $(\sim^{\mathfrak{Q}} A \supset^{\mathfrak{Q}} (A \supset^{\mathfrak{Q}} F_o))$›
and ‹*is_tautologous* $(A \vee^{\mathfrak{Q}} \sim^{\mathfrak{Q}} A)$›
and ‹*is_tautologous* $(\sim^{\mathfrak{Q}}\sim^{\mathfrak{Q}} A \supset^{\mathfrak{Q}} A)$›
⟨*proof*⟩

lemma *axiom_5_wff*:
assumes *A*: ‹$A \in$ *wffs*$_i$›
shows ‹$\vdash \iota \cdot (Q_i \cdot A) =_i A$›
⟨*proof*⟩

interpretation *DA*: *Derivational_Alpha map_con cons_form is_param alpha_class is_consistent_set*
⟨*proof*⟩

3.3 Disjunctive Consistency

lemma *prop_LEM*:
assumes ‹*is_hyps* H›
and ‹$A \in$ *wffs*$_o$›
shows ‹$H \vdash A \vee^{\mathfrak{Q}} \sim^{\mathfrak{Q}} A$›
⟨*proof*⟩

lemma *Qdouble_negE*:
assumes ‹*is_hyps* H›
and ‹$A \in$ *wffs*$_o$›
and ‹$H \vdash \sim^{\mathfrak{Q}} \sim^{\mathfrak{Q}} A$›
shows ‹$H \vdash A$›
⟨*proof*⟩

lemma *QnegD*:
assumes ‹*is_hyps* H›
and ‹$A \in$ *wffs*$_o$›
and ‹$H \vdash \sim^{\mathfrak{Q}} A$›
shows ‹$H \vdash A \supset^{\mathfrak{Q}} F_o$›
⟨*proof*⟩

lemma *QnegI*:

assumes ‹*is_hyps H*›
and ‹$A \in wffs_o$›
and ‹$H \cup \{A\} \vdash F_o$›
shows ‹$H \vdash \sim^{\mathcal{Q}} A$›
⟨*proof*⟩

interpretation *DB*: *Derivational_Beta map_con cons_form is_param beta_class is_consistent_set*
⟨*proof*⟩

3.4 Universal Consistency

interpretation *DG*: *Derivational_Gamma map_con map_con*
cons_form is_param gamma_class is_consistent_set
⟨*proof*⟩

3.5 Existential Consistency

lemma *axiom_3_x*: ‹$\vdash (\mathfrak{f}_{\alpha\to\beta} =_{\alpha\to\beta} \mathfrak{g}_{\alpha\to\beta}) \equiv^{\mathcal{Q}} \forall x_\alpha.\ (\mathfrak{f}_{\alpha\to\beta} \cdot x_\alpha =_\beta \mathfrak{g}_{\alpha\to\beta} \cdot x_\alpha)$›
⟨*proof*⟩

lemma *axiom_3_wff*:
assumes *A*: ‹$A \in wffs_{\alpha\to\beta}$› **and** *B*: ‹$B \in wffs_{\alpha\to\beta}$›
and *x*: ‹$(x, \alpha) \notin free\_vars\ A$› ‹$(x, \alpha) \notin free\_vars\ B$›
shows ‹$\vdash (A =_{\alpha\to\beta} B) \equiv^{\mathcal{Q}} \forall x_\alpha.\ (A \cdot x_\alpha =_\beta B \cdot x_\alpha)$›
⟨*proof*⟩

lemma *axiom_3_right_to_left*:
assumes ‹$A \in wffs_{\alpha\to\beta}$›
and ‹$B \in wffs_{\alpha\to\beta}$›
and ‹$S \vdash \forall x_\alpha.\ (A \cdot x_\alpha =_\beta B \cdot x_\alpha)$›
and ‹$(x, \alpha) \notin free\_vars\ A$›
and ‹$(x, \alpha) \notin free\_vars\ B$›
shows ‹$S \vdash (A =_{\alpha\to\beta} B)$›
⟨*proof*⟩

lemma *is_subform_at_vars*:
assumes ‹$A \preceq_p B$›
shows ‹$vars\ A \subseteq vars\ B$›
⟨*proof*⟩

lemma *is_subform_vars*:
assumes ‹$A \preceq B$›
shows ‹$vars\ A \subseteq vars\ B$›
⟨*proof*⟩

lemma *is_hyps_from_derivable*:
assumes ‹$H \vdash A$›
shows ‹*is_hyps H*›
⟨*proof*⟩

lemma *fresh_var_derivable_from_derivable_const_eq*:
assumes ‹$H \vdash (A \cdot \{\!|c|\!\}_\alpha =_\beta B \cdot \{\!|c|\!\}_\alpha)$› (**is** ‹$H \vdash$ *?form*›)
and ‹$c \notin P.params\ H$› **and** ‹$c \notin logical\_names$›
shows ‹$\exists x.\ (x,\alpha) \notin vars\ A \wedge (x,\alpha) \notin vars\ B \wedge (x,\alpha) \notin vars\ H \wedge H \vdash \mathbf{S}_c\ (c, \alpha)\ x\ (A \cdot \{\!|c|\!\}_\alpha =_\beta B \cdot \{\!|c|\!\}_\alpha)$›
⟨*proof*⟩

lemma *fresh_const_on_subset_remains_inconsistent*:
assumes ‹$p \in As$› ‹*is_param c*› ‹$c \notin P.params\ As$›
and *consistent*: ‹*is_consistent_set As*›
and *wff_p*: ‹$p \in wffs_o$›
and *p_eq*: ‹$p = \sim^{\mathcal{Q}} (A =_{\alpha\ \to\ \beta} B)$›
and *H*: ‹$H \subseteq As$› ‹*is_inconsistent_set* ((*lset* (*delta p c*)) ∪ *H*)›
shows ‹$\neg$ *lset* (*delta p c*) ∪ *H* ⊢ F_o›
⟨*proof*⟩

lemma *ineq_remains_consistent*:
  assumes ‹$p \in As$› ‹*is_param c*› ‹$c \notin P.params\ As$›
    and *consistent*: ‹*is_consistent_set As*›
    and *wff_p*: ‹$p \in wffs_o$›
    and *ex_ineq*: ‹$\exists\, \alpha\ \beta\ A\ B.$ *ineq_match p* $(\alpha,\ \beta,\ A,\ B)$›
  shows ‹*is_consistent_set* (*lset* (*delta p c*) ∪ *As*)›
⟨*proof*⟩

interpretation *DD*: *Derivational_Delta map_con cons_form is_param delta is_consistent_set*
⟨*proof*⟩

interpretation *Derivational_Consistency map_con cons_form is_param Kinds is_consistent_set*
⟨*proof*⟩

end
theory *Model_Existence* **imports**
  Consistency_Property
  "Q0_Metatheory.Soundness"
begin

4 Prelude

instance *type* :: *countable*
  ⟨*proof*⟩

instance *form* :: *countable*
  ⟨*proof*⟩

instance *type* :: *small* ⟨*proof*⟩
instance *type* :: *embeddable* ⟨*proof*⟩
instance *form* :: *small* ⟨*proof*⟩
instance *form* :: *embeddable* ⟨*proof*⟩

definition *is_frugal* :: ‹*model_structure* ⇒ *bool*› **where**
  ‹*is_frugal* $\mathcal{M} \equiv$ *case* $\mathcal{M}$ *of* $(\mathcal{D},\ \mathcal{J},\ \mathcal{V}) \Rightarrow \forall\, \alpha.\ |elts\ (\mathcal{D}\ \alpha)| \leq o\ |UNIV :: form\ set|$›

lemma *is_frugal_countable*: ‹*is_frugal* $(\mathcal{D},\ \mathcal{J},\ \mathcal{V}) \longleftrightarrow (\ \forall\, \alpha.\ |elts\ (\mathcal{D}\ \alpha)| \leq o\ |UNIV :: nat\ set|\ )$›
  ⟨*proof*⟩

definition *extensionally_complete_membership* :: ‹*form set* ⇒ *bool*› **where**
  ‹*extensionally_complete_membership H* ⟷
    $(\forall\, A\ B\ \alpha\ \beta.$ *is_closed_wff_of_type A* $(\beta \to \alpha) \longrightarrow$
            is_closed_wff_of_type B $(\beta \to \alpha) \longrightarrow$
            $(\exists\, C.$ *is_closed_wff_of_type C* $\beta \wedge$
                $((A \cdot C) =_\alpha (B \cdot C) \in H \longrightarrow (A =_{\beta \to \alpha} B) \in H)))$›

lemma *substitute_cong*:
  assumes ‹$A \in wffs_\alpha$›
    and ‹$\forall\, x \in$ *free_vars A*. *F* $ x = G $ x›
  shows ‹*substitute F A* = *substitute G A*›
  ⟨*proof*⟩

lemma *fmran'_fmdrop_subset*: ‹*fmran'* (*fmdrop* $(x,\ \alpha)\ \vartheta$) ⊆ *fmran'* ϑ›
  ⟨*proof*⟩

lemma *free_vars_substitute*: ‹*free_vars* (*substitute* φ *A*)
  ⊆ (*free_vars A* − *fmdom'* φ) ∪ ⋃(*free_vars* ‘ *fmran'* φ)›
⟨*proof*⟩

5 Hintikka

locale *MyHintikka* = *Hintikka map_con cons_form is_param Kinds H*
  for *H* :: ‹*form set*›

begin

lemmas *confl* = sat_H[*of C.kind*]
  and *alpha* = sat_H[*of A.kind*]
  and *beta* = sat_H[*of B.kind*]
  and *gamma* = sat_H[*of G.kind*]
  and *delta* = sat_H[*of D.kind*]

theorem *consistent*:
  assumes ‹$A \in wffs_o$›
  shows ‹$A \notin H \vee \sim^{\mathcal{Q}} A \notin H$›
  ⟨*proof*⟩

lemma *cFalse*: ‹$F_o \notin H$›
  ⟨*proof*⟩

lemma *cBool*:
  assumes ‹$A \in wffs_o$›
    and ‹$A \in H$›
  shows ‹$A =_o T_o \in H$›
  ⟨*proof*⟩

lemma *cTrans*:
  assumes ‹$A \in wffs_\alpha$› ‹$B \in wffs_\alpha$› ‹$C \in wffs_\alpha$›
    and ‹$A =_\alpha B \in H$› ‹$B =_\alpha C \in H$›
  shows ‹$A =_\alpha C \in H$›
  ⟨*proof*⟩

lemma *cCong*:
  assumes ‹$A \in wffs_\alpha$› ‹$B \in wffs_\alpha$› ‹$C \in wffs_{\alpha \to \beta}$›
    and ‹$A =_\alpha B \in H$›
  shows ‹$C \cdot A =_\beta C \cdot B \in H$›
  ⟨*proof*⟩

lemma *cIota*:
  assumes ‹$A \in wffs_\iota$›
  shows ‹$(\iota \cdot (Q_\iota \cdot A) =_\iota A) \in H$›
  ⟨*proof*⟩

lemma *cSubst*:
  assumes ‹$A \in wffs_\alpha$› ‹$B \in wffs_\beta$›
    and ‹*free_vars* $A = \{\}$›
  shows ‹$(\lambda x_\alpha.\ B) \cdot A =_\beta$ *substitute* $\{(x,\ \alpha) \rightarrowtail A\}\ B \in H$›
  ⟨*proof*⟩

lemma *cExt*:
  assumes ‹$A \in wffs_{\alpha \to \beta}$› ‹$B \in wffs_{\alpha \to \beta}$› ‹$C \in wffs_\alpha$›
    and ‹$(A =_{\alpha \to \beta} B) \in H$›
  shows ‹$(A \cdot C =_\beta B \cdot C) \in H$›
  ⟨*proof*⟩

lemma *cIneq*:
  assumes ‹$A \in wffs_{\alpha \to \beta}$› ‹$B \in wffs_{\alpha \to \beta}$›
    and ‹$\sim^{\mathcal{Q}} (A =_{\alpha \to \beta} B) \in H$›
  shows ‹$\exists c.$ *is_param* $c \wedge \sim^{\mathcal{Q}} (A \cdot \{\!|c|\!\}_\alpha =_\beta B \cdot \{\!|c|\!\}_\alpha) \in H$›
⟨*proof*⟩

lemma *complete*:
  assumes ‹$A \in wffs_o$›
  shows ‹$A \in H \vee \sim^{\mathcal{Q}} A \in H$›
  ⟨*proof*⟩

lemma *cRefl*:

assumes ‹$A \in wffs_\alpha$›
shows ‹$A =_\alpha A \in H$›
⟨*proof*⟩

lemma *cIrr*:
assumes ‹$A \in wffs_\alpha$›
shows ‹$\sim^{\mathcal{Q}} (A =_\alpha A) \notin H$›
⟨*proof*⟩

lemma *cTop*: ‹$T_o \in H$›
⟨*proof*⟩

lemma *cSym*:
assumes ‹$A \in wffs_\alpha$› ‹$B \in wffs_\alpha$›
and ‹$A =_\alpha B \in H$›
shows ‹$B =_\alpha A \in H$›
⟨*proof*⟩

lemma *cEqv*:
assumes ‹$A \in wffs_o$› ‹$B \in wffs_o$›
and ‹$A \in H$› ‹$B \in H$›
shows ‹$A \equiv^{\mathcal{Q}} B \in H$›
⟨*proof*⟩

lemma *extensionally_complete_membership*: ‹*extensionally_complete_membership H*›
⟨*proof*⟩

6 The universe of Sets

definition *V_of_form* :: ‹*form* ⇒ *V*› **where**
‹*V_of_form* ≡ *SOME V_of*. *inj V_of*›

definition *V_of_form_set* :: ‹*form set* ⇒ *V*› **where**
‹*V_of_form_set As* ≡ *set* (*V_of_form* ‘ *As*)›

fun
$\mathcal{D}$ :: ‹*type* ⇒ *V*› **and**
$\mathcal{V}$ :: ‹*form* ⇒ *type* ⇒ *V*› **and**
get_rep :: ‹*V* ⇒ *type* ⇒ *form*› **where**
‹$\mathcal{D}$ *o* = $\mathbb{B}$›
| ‹$\mathcal{D}$ *i* = *set* {$\mathcal{V}$ *A i* | *A*. *is_closed_wff_of_type A i*}›
| ‹$\mathcal{D}$ ($\beta \to \alpha$) = *set* {$\mathcal{V}$ *A* ($\beta \to \alpha$) | *A*. *is_closed_wff_of_type A* ($\beta \to \alpha$)}›
| ‹$\mathcal{V}$ *A o* = (*if A* ∈ *H then* **T** *else* **F**)›
| ‹$\mathcal{V}$ *A i* = *V_of_form_set* {*B*. *is_closed_wff_of_type B i* ∧ *A* =$_i$ *B* ∈ *H*}›
| ‹$\mathcal{V}$ *A* ($\beta \to \alpha$) = (**λ***VC*β **:** $\mathcal{D}$ β**.** (*let C* = *get_rep VC*β β *in* $\mathcal{V}$ (*A* ▪ *C*) α))›
| ‹*get_rep VC*β β = (*SOME C*. $\mathcal{V}$ *C* β = *VC*β ∧ *is_closed_wff_of_type C* β)›

lemma *one_o*: ‹$\mathcal{D}$ *o* = *set* {$\mathcal{V}$ *A o*| *A*. *is_closed_wff_of_type A o*}›
⟨*proof*⟩

lemma *bool_to_V_distinct*: ‹*bool_to_V False* ≠ *bool_to_V True*›
⟨*proof*⟩

lemma *two_o*:
assumes ‹$A \in wffs_o$› ‹$B \in wffs_o$›
shows ‹$\mathcal{V}$ *A o* = $\mathcal{V}$ *B o* ⟷ *A* ≡$^{\mathcal{Q}}$ *B* ∈ *H*›
⟨*proof*⟩

lemma *one_i*: ‹$\mathcal{D}$ *i* = *set* {$\mathcal{V}$ *A i*| *A*. *is_closed_wff_of_type A i*}›
⟨*proof*⟩

lemma *inj_V_of_form*: ‹*inj V_of_form*›
⟨*proof*⟩

lemma *V_of_form_set_inj*:
assumes ‹*V_of_form_set As* = *V_of_form_set Bs*›
shows ‹*As* = *Bs*›
⟨*proof*⟩

lemma *two_i*:
assumes ‹*is_closed_wff_of_type A i*›
and ‹*is_closed_wff_of_type B i*›
shows ‹$\mathcal{V}$ *A i* = $\mathcal{V}$ *B i* ⟷ *A* =$_i$ *B* ∈ *H*›
⟨*proof*⟩

lemma *one_fun*:
‹$\mathcal{D}$ (*β* → *α*) = *set* {$\mathcal{V}$ *A* (*β* → *α*)| *A*. *is_closed_wff_of_type A* (*β* → *α*)}›
⟨*proof*⟩

lemma *fun_ext_vfuncset*:
assumes ‹*f* ∈ *elts* (*A* ⟼ *B*)› ‹*g* ∈ *elts* (*A* ⟼ *B*)›
and ‹⋀*x*. *x* ∈ *elts A* ⟹ *app f x* = *app g x*›
shows ‹*f* = *g*›
⟨*proof*⟩

lemma *well_typed*:
assumes ‹*is_closed_wff_of_type A γ*›
shows ‹$\mathcal{V}$ *A γ* ∈ *elts* ($\mathcal{D}$ *γ*)›
⟨*proof*⟩

6.1 1γ

lemma *one_gamma*: ‹$\mathcal{D}$ *γ* = *set* {$\mathcal{V}$ *A γ*| *A*. *is_closed_wff_of_type A γ*}›
⟨*proof*⟩

lemma *wff_for_elts*:
assumes ‹*x* ∈ *elts* ($\mathcal{D}$ *α*)›
shows ‹∃ *A*. *is_closed_wff_of_type A α* ∧ $\mathcal{V}$ *A α* = *x*›
⟨*proof*⟩

lemma *fun_typed*:
shows ‹*elts* ($\mathcal{D}$ (*β* → *α*)) ⊆ *elts* ($\mathcal{D}$ *β* ⟼ $\mathcal{D}$ *α*)›
⟨*proof*⟩

6.2 2γ

lemma *two_gamma*:
assumes ‹*is_closed_wff_of_type A γ*›
and ‹*is_closed_wff_of_type B γ*›
shows ‹$\mathcal{V}$ *A γ* = $\mathcal{V}$ *B γ* ⟷ *A* =*γ* *B* ∈ *H*›
⟨*proof*⟩

lemma *unambiguity*:
assumes ‹*is_closed_wff_of_type A* (*β* → *α*)›
and ‹*is_closed_wff_of_type B β*›
and ‹*is_closed_wff_of_type C β*›
and ‹$\mathcal{V}$ *B β* = $\mathcal{V}$ *C β*›
shows ‹$\mathcal{V}$ (*A* ∙ *B*) *α* = $\mathcal{V}$ (*A* ∙ *C*) *α*›
⟨*proof*⟩

6.3 M is interpretation

fun $\mathcal{J}$ :: ‹*nat* × *Syntax.type* ⇒ *V*› **where**
‹$\mathcal{J}$ (*c*,*τ*) = $\mathcal{V}$ (*FCon* (*c*,*τ*)) *τ*›

lemma *non_logical_constant_denotation_*$\mathcal{V}$:
assumes ‹¬ *is_logical_constant* (*c*, *α*)›
shows ‹$\mathcal{V}$ (*FCon* (*c*, *α*)) *α* ∈ *elts* ($\mathcal{D}$ *α*)›

⟨*proof*⟩

lemma *non_logical_constant_denotation_*$\mathcal{J}$:
assumes ‹¬ *is_logical_constant* (*c*, α)›
shows ‹$\mathcal{J}$ (*c*, α) ∈ *elts* ($\mathcal{D}$ α)›
⟨*proof*⟩

lemma *function_domain*: ‹$\mathcal{D}$ (α → β) ≤ $\mathcal{D}$ α ⟼ $\mathcal{D}$ β›
⟨*proof*⟩

lemma *domain_nonemptiness*: ‹$\mathcal{D}$ α ≠ *0*›
⟨*proof*⟩

lemma *domain_frame*: ‹*frame* $\mathcal{D}$›
⟨*proof*⟩

lemma *distrib_*$\mathcal{V}$*_app*:
assumes ‹*is_closed_wff_of_type A* (α → β)› ‹*is_closed_wff_of_type B* α›
shows ‹$\mathcal{V}$ (*A* ⋅ *B*) β = $\mathcal{V}$ *A* (α → β) ⋅ $\mathcal{V}$ *B* α›
⟨*proof*⟩

lemma *Q_denotation_*$\mathcal{V}$*_two*:
assumes ‹*x* ∈ *elts* ($\mathcal{D}$ α)› ‹*y* ∈ *elts* ($\mathcal{D}$ α)›
shows ‹$\mathcal{V}$ (Q_α) (α→α→*o*) ⋅ *x* ⋅ *y* = ($q_\alpha{}^{\mathcal{D}}$) ⋅ *x* ⋅ *y*›
⟨*proof*⟩

lemma *Q_denotation_*$\mathcal{V}$*_one*:
assumes ‹*x* ∈ *elts* ($\mathcal{D}$ α)›
shows ‹$\mathcal{V}$ (Q_α) (α→α→*o*) ⋅ *x* = ($q_\alpha{}^{\mathcal{D}}$) ⋅ *x*›
⟨*proof*⟩

lemma *Q_denotation_*$\mathcal{V}$: ‹$\mathcal{V}$ (Q_α) (α→α→*o*) = $q_\alpha{}^{\mathcal{D}}$›
⟨*proof*⟩

lemma *Q_denotation_*$\mathcal{J}$: ‹$\mathcal{J}$ (*Q_constant_of_type* α) = $q_\alpha{}^{\mathcal{D}}$›
⟨*proof*⟩

lemma ι*_denotation_*$\mathcal{V}$: ‹*frame.is_unique_member_selector* $\mathcal{D}$ ($\mathcal{V}$ ι ((*i*→*o*)→*i*))›
⟨*proof*⟩

lemma ι*_denotation_*$\mathcal{J}$: ‹*frame.is_unique_member_selector* $\mathcal{D}$ ($\mathcal{J}$ *iota_constant*)›
⟨*proof*⟩

sublocale *premodel* $\mathcal{D}$ $\mathcal{J}$
⟨*proof*⟩

6.4 M is general model

definition *fun_E* :: ‹(*var* ⇒ *V*) ⇒ (*var* ⇒ *form*)›
where ‹*fun_E* φ ≡ λ(*x*,δ). (*SOME A*. φ (*x*,δ) = $\mathcal{V}$ *A* δ ∧ *is_closed_wff_of_type A* δ)›

definition *map_E* :: ‹*var set* ⇒ (*var* ⇒ *V*) ⇒ (*var* ⇀ *form*)›
where ‹*map_E xs* φ ≡ *map_restrict_set xs* (*Some* ∘ *fun_E* φ)›

definition *subst_E* :: ‹*var set* ⇒ (*var* ⇒ *V*) ⇒ *substitution*›
where ‹*subst_E xs* φ ≡ *Abs_fmap* (*map_E xs* φ)›

definition ϑ_E :: ‹(*var* ⇒ *V*) ⇒ *form* ⇒ *substitution*›
where ‹ϑ_E φ *C* ≡ *subst_E* (*free_vars C*) φ›

definition *close_E* :: ‹*form* ⇒ (*var* ⇒ *V*) ⇒ *form*›

where ‹*close_E C φ* ≡ **S** (*ϑ_E φ C*) *C*›

definition *type_of* :: ‹*form* ⇒ *type*›
where ‹*type_of A* ≡ (*SOME γ*. *A* ∈ *wffs_γ*)›

definition *V_φ* :: ‹(*var* ⇒ *V*) ⇒ *form* ⇒ *V*› (‹*V_*›)
where ‹*V_φ C* ≡ *V* (*close_E C φ*) (*type_of C*)›

lemma *fmdom'_map_restrict_set*:
assumes ‹*finite xs*›
and ‹*x* ∈ *fmdom'* (*Abs_fmap* (*map_restrict_set xs* (*Some* ∘ *f*)))›
shows ‹*x* ∈ *xs*›
⟨*proof*⟩

lemma *ϑ_E_is_substitution*:
assumes ‹*φ* ⇝ *D*›
shows ‹*is_substitution* (*ϑ_E φ C*)›
⟨*proof*⟩

lemma *assignment_some_wff*:
assumes *φ*: ‹*φ* ⇝ *D*›
obtains *E* **where**
‹(*SOME A*. *φ* (*x*, *α*) = *V A α* ∧ *is_closed_wff_of_type A α*) = *E*›
‹*is_closed_wff_of_type E α*› ‹*φ* (*x*,*α*) = *V E α*›
⟨*proof*⟩

no-notation *substitute* (‹**S** _ _› [*51*, *51*])

lemma *finite_dom_map_E*:
assumes ‹*finite xs*›
shows ‹*finite* (*dom* (*map_E xs φ*))›
⟨*proof*⟩

lemma *finite_dom_map_E_free_vars*:
fixes *C* :: *form*
shows ‹*finite* (*dom* (*map_E* (*free_vars C*) *φ*))›
⟨*proof*⟩

lemma *ϑ_E_lookup*: ‹*ϑ_E φ C* $$ *x* = *map_E* (*free_vars C*) *φ x*›
⟨*proof*⟩

lemma *subst_E_Some*:
assumes ‹*finite xs*›
and ‹*subst_E xs φ* $$ (*x*, *α*) = *Some A*›
shows ‹*A* = *fun_E φ* (*x*, *α*)›
⟨*proof*⟩

lemma *closed_fmran'_subst_E*:
assumes ‹*A* ∈ *fmran'* (*subst_E xs φ*)›
and ‹*finite xs*›
and ‹*φ* ⇝ *D*›
shows ‹*free_vars A* = {}›
⟨*proof*⟩

lemma *dom_map_restrict_set*: ‹*dom* (*map_restrict_set xs* (*Some* ∘ *f*)) = *xs*›
⟨*proof*⟩

lemma *fmdom'_ϑ_E*: ‹*fmdom'* (*ϑ_E φ A*) = *free_vars A*›
⟨*proof*⟩

lemma *close_E_closes*:
assumes *φ*: ‹*φ* ⇝ *D*›
shows ‹*free_vars* (*close_E A φ*) = {}›

⟨*proof*⟩

lemma *close_E_wff*:
assumes φ: ‹$\varphi \rightsquigarrow \mathcal{D}$›
and A: ‹$A \in wffs_\alpha$›
shows ‹*close_E A* $\varphi \in wffs_\alpha$›
⟨*proof*⟩

lemma *close_E_closes_wff*:
assumes φ: ‹$\varphi \rightsquigarrow \mathcal{D}$›
and A: ‹$A \in wffs_\alpha$›
shows ‹*is_closed_wff_of_type* (*close_E A* φ) α›
⟨*proof*⟩

lemma *g*:
assumes φ: ‹$\varphi \rightsquigarrow \mathcal{D}$›
and A: ‹$A \in wffs_\alpha$›
shows ‹$\mathcal{V}_\varphi$ $A \in$ *elts* ($\mathcal{D}$ α)›
⟨*proof*⟩

lemma *denotation_function_a*:
assumes φ: ‹$\varphi \rightsquigarrow \mathcal{D}$›
shows ‹$\mathcal{V}_\varphi$ (x_α) = φ (x, α)›
⟨*proof*⟩

lemma *denotation_function_b*: ‹$\mathcal{V}_\varphi$ ($\{\!|c|\!\}_\alpha$) = $\mathcal{J}$ (c, α)›
⟨*proof*⟩

lemma *denotation_function_c*:
assumes φ: ‹$\varphi \rightsquigarrow \mathcal{D}$›
and A: ‹$A \in wffs_{\beta \to \alpha}$›
and B: ‹$B \in wffs_\beta$›
shows ‹$\mathcal{V}_\varphi$ (A ▪ B) = $\mathcal{V}_\varphi$ A ⋅ $\mathcal{V}_\varphi$ B›
⟨*proof*⟩

lemma *fmdom'_ϑ_E_lam*: ‹(x, α) ∉ *fmdom'* (ϑ_E φ (λx_α. B))›
⟨*proof*⟩

lemma *empty_subst_E*:
assumes ‹*free_vars* C = {}›
shows ‹*subst_E* (*free_vars* C) φ = {$$}›
⟨*proof*⟩

lemma *empty_close_E*:
assumes ‹*free_vars* A = {}›
shows ‹*close_E A* φ = A›
⟨*proof*⟩

lemma *close_E_lam*: ‹*close_E* (λx_α. B) φ = λx_α. *substitute* (*subst_E* (*free_vars* B − {(x, α)}) φ) B›
⟨*proof*⟩

lemma *substitute_id_disjoint*:
assumes ‹*free_vars* A ∩ *fmdom'* φ = {}›
shows ‹*substitute* φ A = A›
⟨*proof*⟩

corollary *substitute_id_closed*:
assumes ‹*free_vars* A = {}›
shows ‹*substitute* φ A = A›
⟨*proof*⟩

lemma *map_E_fun_upd*:
assumes ‹$(x, \alpha) \in xs$›
and ‹*fun_E* $(\varphi((x, \alpha) := A))\ (x, \alpha) = E$›
shows ‹*map_E* $xs\ (\varphi((x, \alpha) := A)) = ((map\_E\ (xs - \{(x, \alpha)\})\ \varphi)((x, \alpha) \mapsto E))$›
⟨*proof*⟩

lemma *substitute_fm_upd*:
assumes B: ‹$B \in wffs_\beta$›
and E: ‹$E \in wffs_\alpha$› ‹*free_vars* $E = \{\}$› ‹*fun_E* $(\varphi((x, \alpha) := \mathcal{V}\ E\ \alpha))\ (x, \alpha) = E$›
and φ: ‹$\varphi \rightsquigarrow \mathcal{D}$›
shows ‹*substitute* $((subst\_E\ (free\_vars\ B - \{(x, \alpha)\})\ \varphi)((x, \alpha) \rightarrowtail E))\ B =$
substitute $(subst\_E\ (free\_vars\ B)\ (\varphi((x, \alpha) := \mathcal{V}\ E\ \alpha)))\ B$›
⟨*proof*⟩

lemma *cSubst_close_E*:
assumes B: ‹$B \in wffs_\beta$›
and E: ‹$E \in wffs_\alpha$› ‹*free_vars* $E = \{\}$› ‹*fun_E* $(\varphi((x, \alpha) := \mathcal{V}\ E\ \alpha))\ (x, \alpha) = E$›
and φ: ‹$\varphi \rightsquigarrow \mathcal{D}$›
shows ‹*close_E* $(\lambda x_\alpha.\ B)\ \varphi \cdot E =_\beta close\_E\ B\ (\varphi((x, \alpha) := \mathcal{V}\ E\ \alpha)) \in H$›
⟨*proof*⟩

lemma *denotation_function_d*:
assumes φ: ‹$\varphi \rightsquigarrow \mathcal{D}$›
and B: ‹$B \in wffs_\beta$›
shows ‹$\mathcal{V}_\varphi\ (\lambda x_\alpha.\ B) = (\boldsymbol{\lambda} z{::}\mathcal{D}\ \alpha\ \boldsymbol{.}\ \mathcal{V}_{\varphi((x, \alpha) := z)}\ B)$›
⟨*proof*⟩

lemma *denotation_function*: ‹*is_wff_denotation_function* $\mathcal{V}\varphi$›
⟨*proof*⟩

sublocale M: *general_model* $\mathcal{D}\ \mathcal{J}\ \mathcal{V}\varphi$
⟨*proof*⟩

lemma *sat_closed_formulas*:
assumes A: ‹$A \in wffs_o$› ‹*free_vars* $A = \{\}$›
and H: ‹$A \in H$›
shows ‹$\mathcal{V}_\varphi\ A = \mathbf{T}$›
⟨*proof*⟩

lemma *canon_model_for*: ‹*is_model_for* $(\mathcal{D},\mathcal{J},\mathcal{V}\varphi)\ \{A \in H.\ A \in wffs_o \wedge free\_vars\ A = \{\}\}$›
⟨*proof*⟩

lemmas *is_general_model* $= M.general\_model\_axioms$

lemma $\mathcal{V}\varphi$*_consistent*:
assumes A: ‹$A \in wffs_o$› ‹*free_vars* $A = \{\}$›
shows ‹$\neg\ (\mathcal{V}_\varphi\ A = \mathbf{T} \wedge \mathcal{V}_\varphi\ (\sim^{\mathcal{Q}}\ A) = \mathbf{T})$›
⟨*proof*⟩

lemma *model_consistent*:
assumes A: ‹$A \in wffs_o$› ‹*free_vars* $A = \{\}$›
shows ‹$\neg\ ((\mathcal{D},\mathcal{J},\mathcal{V}\varphi) \models A \wedge (\mathcal{D},\mathcal{J},\mathcal{V}\varphi) \models \sim^{\mathcal{Q}}\ A)$›
⟨*proof*⟩

lemma *elts_in_wffs*: ‹*elts* $(\mathcal{D}\ \alpha) \subseteq (\lambda A.\ \mathcal{V}\ A\ \alpha)\ `\ wffs_\alpha$›
⟨*proof*⟩

lemma *frugal_wffs*: ‹$|elts\ (\mathcal{D}\ \alpha)| \leq o\ |wffs_\alpha|$›
⟨*proof*⟩

theorem *is_frugal*: ‹*is_frugal* $(\mathcal{D},\mathcal{J},\mathcal{V}\varphi)$›
⟨*proof*⟩

end

7 Model Existence

theorem *model_existence*:
fixes S :: ‹*form set*›
assumes *cprop*: ‹$P.prop_E$ *Kinds* C›
and S: ‹$S \in C$› ‹*P.enough_new* S›
shows ‹$\exists M.$ *is_general_model* $M \wedge$ *is_frugal* $M \wedge$
$(\forall A \in S.$ *is_sentence* $A \longrightarrow M \models A) \wedge$
$(\forall A.$ *is_sentence* $A \longrightarrow \neg\ (M \models A \wedge M \models \sim^{\mathcal{Q}} A))$›
⟨*proof*⟩

end
theory *Q0_Completeness* **imports**
Derivational_Consistency
Model_Existence
begin

8 Completeness

theorem *strong_completeness*:
assumes *mod*: ‹$\bigwedge M.$ *is_general_model* $M \Longrightarrow$ *is_frugal* $M \Longrightarrow \forall B \in \mathcal{G}.\ M \models B \Longrightarrow M \models A$›
and A: ‹*is_sentence* A›
and $\mathcal{G}$: ‹$\forall B \in \mathcal{G}.$ *is_sentence* B› ‹*P.enough_new* $\mathcal{G}$›
shows ‹$\exists \mathcal{H} \subseteq \mathcal{G}.\ \mathcal{H} \vdash A$›
⟨*proof*⟩

lemma *infinite_params*: ‹*infinite* (*Collect is_param*)›
⟨*proof*⟩

lemma *is_hyps_enough_new*:
assumes ‹*is_hyps* $\mathcal{H}$›
shows ‹*P.enough_new* $\mathcal{H}$›
⟨*proof*⟩

corollary *completeness*:
assumes ‹$\bigwedge M.$ *is_general_model* $M \Longrightarrow$ *is_frugal* $M \Longrightarrow M \models A$› ‹*is_sentence* A›
shows ‹$\vdash A$›
⟨*proof*⟩

9 Addendum

hyp_derivability_implies_validity in *Q0_Metatheory.Soundness* mechanizes Andrews' 5402 Soundness Theorem (b). However, unlike Andrews', it assumes the set $\mathcal{G}$ to be finite by assuming *is_hyps* $\mathcal{G}$. On page 229, Andrews lifts derivability to infinite sets by simply requiring derivability from a finite subset. We state this version of the theorem (all of the work having been done already).

theorem *hyp_derivability_implies_validity_general*:
assumes ‹*is_model_for* $\mathcal{M}$ $\mathcal{G}$›
and ‹$\exists \mathcal{H} \subseteq \mathcal{G}.\ \mathcal{H} \vdash A$›
and ‹*is_general_model* $\mathcal{M}$›
shows ‹$\mathcal{M} \models A$›
⟨*proof*⟩

model_existence_implies_set_consistency in *Q0_Metatheory.Consistency* assumes *is_hyps* $\mathcal{G}$ for the set of formulas $\mathcal{G}$. This limits the result to finite sets. Andrews does not make this assumption in his Consistency Theorem (5403). We give a version without this finiteness assumption by once again taking derivability from an infinite set to mean derivability from a finite subset. Consistency of a set then means that *no subset* proves falsity. Similarly, we remove this finiteness assumption from the principle of explosion.

lemma *is_consistent_set*: ‹*is_consistent_set* $\mathcal{G}$

$\longleftrightarrow$ ($\nexists \mathcal{H}.\ \mathcal{H} \subseteq \mathcal{G} \wedge$ *finite* $\mathcal{H} \wedge \mathcal{H} \subseteq$ *wffs*$_o \wedge \mathcal{H} \vdash F_o$)›
⟨*proof*⟩

corollary *model_existence_implies_set_consistency_general*:
assumes ‹*is_general_model* $\mathcal{M}$› ‹*is_model_for* $\mathcal{M}$ $\mathcal{G}$›
shows ‹*is_consistent_set* $\mathcal{G}$›
⟨*proof*⟩

corollary *principle_of_explosion_general*:
‹*is_inconsistent_set* $\mathcal{G} \longleftrightarrow (\forall A \in (wffs_o).\ \mathcal{G} \vdash A)$›
⟨*proof*⟩

We note that infinite sets are always consistent under Díaz's formulation, since nothing can be derived from them. This is again, why we take subsets above.

lemma *infinite_sets_underivable*: ‹*infinite* $\mathcal{G} \Longrightarrow \neg\ \mathcal{G} \vdash A$›
⟨*proof*⟩

lemma *infinite_sets_consistent*: ‹*infinite* $\mathcal{G} \Longrightarrow \neg$ *is_inconsistent_set* $\mathcal{G}$›
⟨*proof*⟩

We might finally remark, that even if we stick to finite sets, then *hyp_derivability_implies_validity* in *Q0_Metatheory.Soundness* carries a redundant assumption *is_hyps* $\mathcal{G}$ since this follows from the derivation $\mathcal{G} \vdash A$. Likewise, *model_existence_implies_set_consistency* in *Q0_Metatheory.Consistency* needlessly assumes *is_hyps* $\mathcal{G}$ since when proving $\neg$ *is_inconsistent_set* $\mathcal{G}$ we get to assume $\mathcal{G} \vdash F_o$ and the same argument as above applies. To showcase these redundancies, we remove the extra assumptions for strong soundness and consistency.

lemma *hyp_derivability_implies_validity2*:
assumes *"is_model_for* $\mathcal{M}$ $\mathcal{G}$*"*
and *"*$\mathcal{G} \vdash A$*"*
and *"is_general_model* $\mathcal{M}$*"*
shows *"*$\mathcal{M} \models A$*"*
⟨*proof*⟩

lemma *model_existence_implies_set_consistency2*:
assumes *"is_general_model* $\mathcal{M}$*"* *"is_model_for* $\mathcal{M}$ $\mathcal{G}$*"*
shows *"*$\neg$ *is_inconsistent_set* $\mathcal{G}$*"*
⟨*proof*⟩

end

References

[AM23] Oskar Abrahamsson and Magnus O. Myreen. Fast, verified computation for Candle. In Adam Naumowicz and René Thiemann, editors, *14th International Conference on Interactive Theorem Proving, ITP 2023, July 31 to August 4, 2023, Białystok, Poland*, volume 268 of *LIPIcs*, pages 4:1–4:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[AMKS22] Oskar Abrahamsson, Magnus O. Myreen, Ramana Kumar, and Thomas Sewell. Candle: A verified implementation of HOL Light. In June Andronick and Leonardo de Moura, editors, *13th International Conference on Interactive Theorem Proving, ITP 2022, August 7-10, 2022, Haifa, Israel*, volume 237 of *LIPIcs*, pages 3:1–3:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[And63] Peter Andrews. A reduction of the axioms for the theory of propositional types. *Fundamenta Mathematicae*, 52:345–350, 1963.

[And71] Peter B. Andrews. Resolution in type theory. *Journal of Symbolic Logic*, 36(3):414432, 1971.

[And72] Peter B Andrews. General models and extensionality. *The Journal of Symbolic Logic*, 37(2):395–397, 1972.

[And13] Peter B. Andrews. *An introduction to mathematical logic and type theory: to truth through proof*, volume 27 of *Applied Logic Series*. Springer Science & Business Media, 2nd edition, 2013.

[Art02] Rob Arthan. HOL formalised: Deductive system, 1993, revised 2002. http://www.lemma-one.com/ProofPower/specs/specs.html.

[Art14a] Rob Arthan. HOL formalised: Language and overview, 1993, revised 2014. http://www.lemma-one.com/ProofPower/specs/specs.html.

[Art14b] Rob Arthan. HOL formalised: Semantics, 1993, revised 2014. http://www.lemma-one.com/ProofPower/specs/specs.html.

[Bar96a] Bruno Barras. Coq en Coq. Research Report RR-3026, INRIA, 1996. Projet COQ, https://inria.hal.science/inria-00073667.

[Bar96b] Bruno Barras. Verification of the interface of a small proof system in Coq. In Eduardo Giménez and Christine Paulin-Mohring, editors, *Types for Proofs and Programs, International Workshop TYPES'96, Aussois, France, December 15-19, 1996, Selected Papers*, volume 1512 of *Lecture Notes in Computer Science*, pages 28–45. Springer, 1996.

[Bar97] Bruno Barras. coq-in-coq. In *coq-contribs*, 1997. https://github.com/coq-contribs/coq-in-coq.

[BKT25] Ghilain Bergeron, Florent Krasnopol, and Sophie Tourret. A modular splitting framework for saturation theorem proving. *Archive of Formal Proofs*, June 2025. https://isa-afp.org/entries/Splitting_Framework.html, Formal proof development.

[BPT14a] Jasmin Christian Blanchette, Andrei Popescu, and Dmitriy Traytel. Abstract completeness. *Archive of Formal Proofs*, April 2014. https://isa-afp.org/entries/Abstract_Completeness.html, Formal proof development.

[BPT14b] Jasmin Christian Blanchette, Andrei Popescu, and Dmitriy Traytel. Unified classical logic completeness - A coinductive pearl. In Stéphane Demri, Deepak Kapur, and Christoph Weidenbach, editors, *Automated Reasoning - 7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 19-22, 2014. Proceedings*, volume 8562 of *Lecture Notes in Computer Science*, pages 46–60. Springer, 2014.

[BPT17] Jasmin Christian Blanchette, Andrei Popescu, and Dmitriy Traytel. Soundness and completeness proofs by coinductive methods. *J. Autom. Reason.*, 58(1):149–179, 2017.

[BT20] Jasmin Christian Blanchette and Sophie Tourret. Extensions to the comprehensive framework for saturation theorem proving. *Archive of Formal Proofs*, August 2020. https://isa-afp.org/entries/Saturation_Framework_Extensions.html, Formal proof development.

[BW96] Bruno Barras and Benjamin Werner. Coq in Coq (manuscript). Technical report, 1996. http://www.lix.polytechnique.fr/Labo/Bruno.Barras/publi/coqincoq.pdf.

[Chu40] Alonzo Church. A formulation of the simple theory of types. *The journal of symbolic logic*, 5(2):56–68, 1940.

[Día23] Javier Díaz. Metatheory of Q0. *Archive of Formal Proofs*, November 2023. https://isa-afp.org/entries/Q0_Metatheory.html, Formal proof development.

[EBT21] Gabriel Ebner, Jasmin Blanchette, and Sophie Tourret. A unifying splitting framework. In André Platzer and Geoff Sutcliffe, editors, *Automated Deduction - CADE 28 - 28th International Conference on Automated Deduction, Virtual Event, July 12-15, 2021, Proceedings*, volume 12699 of *Lecture Notes in Computer Science*, pages 344–360. Springer, 2021.

[Fro23] Asta Halkjær From. Synthetic completeness. *Archive of Formal Proofs*, January 2023. https://isa-afp.org/entries/Synthetic_Completeness.html, Formal proof development.

[Fro25] Asta Halkjær From. An Isabelle/HOL framework for synthetic completeness proofs. In Kathrin Stark, Amin Timany, Sandrine Blazy, and Nicolas Tabareau, editors, *Proceedings of the 14th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2025, Denver, CO, USA, January 20-21, 2025*, pages 171–186. ACM, 2025.

[FS25] Asta Halkjær From and Anders Schlichtkrull. Abstract, compositional consistency: Isabelle/HOL locales for completeness à la fitting. In Yannick Forster and Chantal Keller, editors, *16th International Conference on Interactive Theorem Proving, ITP 2025, Reykjavik, Iceland, September 28 - October 1, 2025*, volume 352 of *LIPIcs*, pages 8:1–8:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025.

[FS26] Asta Halkjær From and Anders Schlichtkrull. Abstract consistency properties. *Archive of Formal Proofs*, March 2026. https://isa-afp.org/entries/Abstract_Consistency_Property.html, Formal proof development.

[Har06] John Harrison. Towards self-verification of HOL Light. In Ulrich Furbach and Natarajan Shankar, editors, *Automated Reasoning, Third International Joint Conference, IJCAR 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings*, volume 4130 of *Lecture Notes in Computer Science*, pages 177–191. Springer, 2006.

[Hen50] Leon Henkin. Completeness in the theory of types. *The Journal of Symbolic Logic*, 15(2):81–91, 1950.

[Hen63] Leon Henkin. A theory of propositional types. *Fundamenta Mathematicae*, 52:323–344, 1963.

[Hen96] Leon Henkin. The discovery of my completeness proofs. *Bull. Symb. Log.*, 2(2):127–158, 1996.

[KAMO14] Ramana Kumar, Rob Arthan, Magnus O. Myreen, and Scott Owens. HOL with definitions: Semantics, soundness, and a verified implementation. In Gerwin Klein and Ruben Gamboa, editors, *Interactive Theorem Proving - 5th International Conference, ITP 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, volume 8558 of *Lecture Notes in Computer Science*, pages 308–324. Springer, 2014.

[KAMO16] Ramana Kumar, Rob Arthan, Magnus O. Myreen, and Scott Owens. Self-formalisation of higher-order logic - semantics, soundness, and a verified implementation. *J. Autom. Reason.*, 56(3):221–259, 2016.

[KK22] Mark Koch and Dominik Kirst. Undecidability, incompleteness, and completeness of second-order logic in Coq. In Andrei Popescu and Steve Zdancewic, editors, *CPP '22: 11th ACM SIGPLAN International Conference on Certified Programs and Proofs, Philadelphia, PA, USA, January 17 - 18, 2022*, pages 274–290. ACM, 2022.

[NR21a] Tobias Nipkow and Simon Roßkopf. Isabelle's metalogic: Formalization and proof checker. In André Platzer and Geoff Sutcliffe, editors, *Automated Deduction - CADE 28 - 28th International Conference on Automated Deduction, Virtual Event, July 12-15, 2021, Proceedings*, volume 12699 of *Lecture Notes in Computer Science*, pages 93–110. Springer, 2021.

[NR21b] Tobias Nipkow and Simon Roßkopf. Isabelle's metalogic: Formalization and proof checker. *Archive of Formal Proofs*, April 2021. https://isa-afp.org/entries/Metalogic_ProofChecker.html, Formal proof development.

[Pau19] Lawrence C. Paulson. Zermelo Fraenkel set theory in higher-order logic. *Archive of Formal Proofs*, October 2019. https://isa-afp.org/entries/ZFC_in_HOL.html, Formal proof development.

[RN23] Simon Roßkopf and Tobias Nipkow. A formalization and proof checker for Isabelle's metalogic. *J. Autom. Reason.*, 67(1):1, 2023.

[Rus08] Bertrand Russell. Mathematical logic as based on the theory of types. *American journal of mathematics*, 30(3):222–262, 1908.

[SBF+20] Matthieu Sozeau, Simon Boulier, Yannick Forster, Nicolas Tabareau, and Théo Winterhalter. Coq Coq correct! verification of type checking and erasure for Coq, in Coq. *Proc. ACM Program. Lang.*, 4(POPL):8:1–8:28, 2020.

[Sch23] Anders Schlichtkrull. Soundness of the Q0 proof system for higher-order logic. *Archive of Formal Proofs*, November 2023. https://isa-afp.org/entries/Q0_Soundness.html, Formal proof development.

[TB21] Sophie Tourret and Jasmin Blanchette. A modular Isabelle framework for verifying saturation provers. In Catalin Hritcu and Andrei Popescu, editors, *CPP '21: 10th ACM SIGPLAN International Conference on Certified Programs and Proofs, Virtual Event, Denmark, January 17-19, 2021*, pages 224–237. ACM, 2021.

[Tou20] Sophie Tourret. A comprehensive framework for saturation theorem proving. *Archive of Formal Proofs*, April 2020. https://isa-afp.org/entries/Saturation_Framework.html, Formal proof development.

[WR13] Alfred North Whitehead and Bertrand Russell. *Principia Mathematica*. Cambridge University Press, Cambridge England, 1913. 3 volumes; first edition 1913, second edition 1927.

[WTRB20] Uwe Waldmann, Sophie Tourret, Simon Robillard, and Jasmin Blanchette. A comprehensive framework for saturation theorem proving. In Nicolas Peltier and Viorica Sofronie-Stokkermans, editors, *Automated Reasoning - 10th International Joint Conference, IJCAR 2020, Paris, France, July 1-4, 2020, Proceedings, Part I*, volume 12166 of *Lecture Notes in Computer Science*, pages 316–334. Springer, 2020.

[WTRB22] Uwe Waldmann, Sophie Tourret, Simon Robillard, and Jasmin Blanchette. A comprehensive framework for saturation theorem proving. *J. Autom. Reason.*, 66(4):499–539, 2022.