Formalized Completeness of Peter B. Andrews' Higher-Order Logic in Isabelle/HOL

Asta Halkjær Boserup
asfr@di.ku.dk
Department of Computer Science,
University of Copenhagen
Copenhagen, Denmark

Jonathan Julián Huerta y Munive
jjhymuni@cs.aau.dk
Department of Computer Science,
Aalborg University Copenhagen
Copenhagen, Denmark

Anders Schlichtkrull
andsch@cs.aau.dk
Department of Computer Science,
Aalborg University Copenhagen
Copenhagen, Denmark

Abstract

Higher-order logic (also called simple type theory) is an expressive family of formal systems whose members serve as the foundation for notable proof assistants. In turn, people use these tools extensively to formalize mathematics, computer science and logic. For instance, the Isabelle/HOL proof assistant has been used to implement a framework for proving completeness of formal systems via abstract consistency properties (ACPs). Yet, completeness has not been formalized before mid-2026 for any higher-order logic. Notably, higher-order logics are incomplete under their standard semantics, but complete with respect to so-called general semantics. Our contribution is twofold: we provide the first formalization of the completeness proof of a higher-order logic, and we do so by stress-testing the ACP framework. Specifically, we demonstrate the framework's applicability by proving the completeness of the higher-order logic Q_0 from Peter B. Andrews' book "An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof".

Keywords: Logic, Higher-order logic, completeness, abstract consistency property, Isabelle/HOL, locales

1 Introduction

Higher-order logic (also called simple type theory) [12, 45] is a highly expressive logic used to formalize mathematics and computer science. Pioneers within their study include Bertrand Russell [37] with Alfred N. Whitehead [46], Alonzo Church [15], Leon A. Henkin [23–25] and Peter B. Andrews [4–7]. Formulations of higher-order logic typically include the syntax of a simply typed lambda calculus with fixed types for booleans and individuals. There are also proof systems for higher-order logic, e.g. axiomatic systems consisting of a set of axioms and inference rules. Proof assistants, such as the HOL systems [3, 20, 22, 39] and Isabelle/HOL [30, 34], are based on a higher-order logic. Formalizations of results in proof assistants serve as arguments for increased trust in those results. Thus, such arguments ultimately rely on our trust in proof systems for higher-order logic. In particular, proof assistants are used extensively in research on programming languages [16, 44]. For this reason, higher-order logic becomes a critical component of mathematics and computer science. An argument for trusting higher-order logic is that its meta-theory is well studied. Here, in particular, the correspondence between proof systems and semantics is studied. The ideal correspondence is to have both a soundness theorem (all that can be proven is semantically valid) and a completeness theorem (all that is semantically valid can be proven), as is the case for weaker logics such as propositional logic and first-order logic. When we have both, then we have an equivalence — a formula is provable if and only if it is semantically valid. This means that someone convinced by the definition of semantic validity must also be convinced by the proof system — because they are equivalent. But the converse is also the case! Someone convinced by the definition of the proof system must also be convinced by the semantics — again because they are equivalent. The justification of a logic thus has two pieces, soundness and completeness.

With respect to its standard semantics, higher-order logic is famously sound but incomplete [12, 45]. However, there is another semantics which allows for so-called general models — a broader class than standard models — for which higher-order logic is both sound and complete. One influential work on the meta-theory of higher-order logic is Peter B. Andrews' chapter on type theory in his textbook on logic [7]. There, Andrews introduces a higher-order logic called Q_0, defines a proof system and semantics for it and gives a soundness and completeness proof for it with respect to general models. The system is influenced by works of Church [15] and Henkin [23, 24] as well as earlier works by Andrews [4, 6].

The soundness and completeness proofs in Andrews' book are convincing because of their rigor, but the ultimate test of a proof is its formalization in a proof assistant. A formalization of Q_0's soundness and completeness would cement our trust in higher-order logic. Steps have been taken in this direction: Díaz [17] and Schlichtkrull [38] have each formalized Q_0 proving, among other things, its soundness. Both are in Isabelle/HOL and formalize Q_0 (the object logic) by extending Isabelle's HOL (the meta logic used) with a type of set-theoretical sets. Yet these works do not formalize completeness, even though completeness proofs for other

CPP 2027, Mexico City, Mexico
2018. https://doi.org/XXXXXXX.XXXXXXX

logics have appeared before. In fact, a recent paper by From and Schlichtkrull [19] introduces a general framework for conducting such proofs, but it has not been applied to any higher-order logic. Thus, the formal justification of Q_0 is missing one of its two pieces: completeness. *We provide here this missing piece, cementing our trust in higher-order logic.*

We do this exactly by applying the aforementioned framework to Díaz's formalization of Q_0. We henceforth refer to the framework as "the abstract consistency framework", or simply, "the framework". It generalizes ideas from Smullyan [40] and Fitting [18], and provides an abstract version of a theorem used to prove completeness: the model existence theorem. Concrete model existence theorems are interesting beyond their utility to prove completeness because they state that consistent sets can be satisfied by some model. Intuitively, their significance is that they express that any consistent set actually describes something. The framework builds on consistency properties [18, 40]. Intuitively, a consistency property is any set similar enough to $\{X \mid X \text{ is consistent}\}$ to admit a model existence theorem showing that its members are satisfiable. An *abstract* consistency property (ACP) is the specification of such consistency properties.

However, applying the framework is not automatic — the framework builds on several choices, and on assumptions that we need to prove of Q_0 and our choices.

Our paper describes related work (section 2) and explains necessary background (section 3 and section 4). *In proving completeness, we provide several contributions:*

- For the framework's applicability to Díaz's formalization of Q_0, we (slightly) generalize the framework's requirements of parameter substitutions (section 4).
- We define an ACP for Q_0 (section 5).
- We prove a new model existence theorem for Q_0, that is, from such consistency property (section 6).
- We prove that consistency with respect to Q_0's proof system constitutes a consistency property as we define it (section 7). This yields the concrete model existence theorem for Q_0's proof system.
- Applying the framework to Q_0 shows its flexibility in handling formalizations of logics with types, lambda expressions and named variable representation.
- The end result is the completeness theorem (section 8).

We conclude (section 9) by noting that it was possible to adapt Andrews' proofs from his chapter to fit the needs of the framework. Andrews' work is very thorough, which helped in the formalization effort. Yet, we discovered several places where further thoroughness was required for the formalization process, and we describe these in the paper.

2 Related Work

Andrews' soundness proof has been formalized and appears in the Archive of Formal Proofs twice: by Díaz [17] and by Schlichtkrull [38]. Here, we build on Díaz's formalization because it covers a larger part of Andrews' chapter. Additionally, it relies on Paulson's ZFC set theory for Isabelle/HOL [33], which includes convenient tooling for converting between Isabelle/HOL types and the sets in his formalization — Schlichtkrull's formalization relies on a port of a set theory by Kumar et al. [27] which does not have such tooling.

Benzmüller and Kirchner [14] formalize soundness, completeness and consistency of a natural deduction system for their BKK higher-order logic [13]. We use a Hilbert-style approach and focus on completeness via the framework. Other related formalizations have a different focus, e.g. the soundness of HOL Light's proof system by John Harrison in HOL Light [21] and Kumar et al.'s work in HOL4 [1, 2, 27, 28] including a verified proof assistant. Roßkopf and Nipkow [31, 32, 36] formalize a proof system and a proof checker for terms of Isabelle's metalogic, and show that the proof checker indeed implements this proof system (not a semantic focus). Recently, Popescu has used Henkin-style general models, and gives an unformalized proof for the completeness of HOL [35] (which has more features than Andrews' Q_0).

Metatheory of the calculus of constructions and the calculus of inductive constructions has been formalized in Rocq, but it is significantly different from the one considered in this paper. Those works focus on consistency, strong normalization, and checkers implementing the calculus [8–11, 41].

There are soundness and completeness formalizations of second-order logics with general models in Rocq, by Koch and Kirst [26], and Isabelle [19], by From and Schlichtkrull [19]. The latter uses the same framework as us.

3 The System Q_0

We recount here Q_0's syntax, proof system, and semantics [7], and our conventions following Díaz's formalization [17]. We summarize his soundness result and present our generalization of his formalization of Q_0's consistency theorem.

Syntax of Q_0**.** The types of Q_0 are recursively defined from the types of individuals (ι), booleans (o), and function types $\alpha \to \beta$ for α and β. Their formalization uses a datatype:

```
datatype type = TInd (‹ι›) | TBool (‹o›) | TFun type type (infixr ‹→› 101)
```

For each type α, Andrews assumes that a denumerable amount of typed variables and constants is available. Henceforth, $x_\alpha, y_\alpha, f_\alpha, g_\alpha$ range over variables and c_α, d_α, over constants (written $\lfloor\!\lfloor d \rfloor\!\rfloor_\alpha$ in Isabelle), with concrete variable instances formalized as natural numbers in Isabelle and denoted with Fraktur letter such as $\mathfrak{x}_\alpha, \mathfrak{y}_\alpha, \mathfrak{f}_\alpha, \mathfrak{g}_\alpha$ or $\mathfrak{h}_\alpha$. Then, Andrews recursively defines the *well-formed formulas of type* α. Nowadays, these are more commonly called the *terms* of type α, but here we follow Andrews' nomenclature and abbreviation ($wffs_\alpha$) for faithfulness. Using A_α, B_α, and C_α as variables ranging over $wffs_\alpha$, the definition is as follows:

- variables and constants of type α are $wffs_\alpha$,

- the application $A_{\alpha\to\beta} \cdot B_\alpha$ belongs to $wffs_\beta$,
- the abstraction $\lambda x_\alpha.\ B_\beta$ is in $wffs_{\alpha\to\beta}$.

Their formalization is a two-step definition with a datatype capturing the type of formulas *form* and thereafter an inductive predicate capturing the well-formedness. Finally, the notation $wffs_\alpha$ is used for the corresponding set of formulas.

The *logical constants* of Q_0 are the *identity relation* $Q_{\alpha\to\alpha\to o}$ for each type α, and the *selection operator for individuals* $\iota_{(i\to o)\to i}$. We follow Díaz [17] in denoting the former as Q_α and the latter as ι while deeming other constants as *nonlogical*. We also refer to nonlogical constants as *parameters*. Q_0 defines typical logical connectives and quantifiers as abbreviations in terms of Q_α and ι, such as equality ($=_\alpha$), equivalence ($\equiv^Q$), truth (T_o), falsity (F_o), universal quantification ($\forall\, x_\alpha$), conjunction ($\wedge^Q$), implication ($\supset^Q$), negation ($\sim^Q$), disjunction ($\vee^Q$), existential quantification ($\exists\, x_\alpha$) and inequality ($\neq_\alpha$). For instance, equality $A_\alpha =_\alpha B_\alpha$ is the abbreviation for $Q_\alpha \cdot A_\alpha \cdot B_\alpha$ while logical equivalence $\equiv^Q$ abbreviates $=_o$.

Díaz recursively defines the free and bound variables of a formula over the type *form*. In contrast, Andrews opts for asserting that x_α is *free in* A_γ if it is not in a well-formed part of A_γ with a shape $\lambda x_\alpha.\ B_\beta$. A closed wff (denoted cwff) is one without free variables. A sentence is a formula in $cwffs_o$.

In the present paper, we will sometimes omit the type subscripts on wffs and operators if they are clear from context. In particular, the formalization uses assumptions of the form $A \in wffs_\alpha$ to indicate the types of wffs instead of subscripts.

3.1 Proof system of Q_0

Q_0's axioms appear in figure 1. Intuitively, the first one restricts the sets of truth values to two, while the second one is the equational principle of indiscernibility of identicals, i.e. identical entities share the same properties. The third axiom is the principle of functional extensionality asserting equality of functions as point-wise equality. Axioms ($axiom\text{-}4_1$) to ($axiom\text{-}4_5$) establish the behavior of lambda-abstractions. They specify constant functions, identity functions, the distribution of abstractions over applications, how abstractions can commute, and variable shadowing. The last axiom characterizes the selection operator. Díaz's formalization of the axioms is faithfully implemented as an inductive set.

Andrews imposes one rule of inference on Q_0: **Rule R**. Namely, that given an equality $A_\alpha =_\alpha B_\alpha$, one can replace a single instance of the left-hand side with the right-hand side in any formula C, as long as "the occurrence of A_α in C is not (an occurrence of a variable) immediately preceded by λ" [7]. Díaz formalizes this as a single predicate *is-rule-R-app* that requires several definitions starting from whether a formula A is a subformula of another C at a given position p (a list of left or right directions), denoted $A \preceq_p C$, to a predicate establishing that C' is the result of replacing the subformula of C at position p with B, denoted $C\langle\!|p \leftarrow B|\!\rangle \rhd C'$ [17]. We represent Díaz's **Rule R** in figure 1 and note that the "immediately preceded by λ" proviso holds by his construction of the recursive definitions of position and formulas.

As customary, a *proof of* $A \in wffs_o$ *in* Q_0 is a sequence of $wffs_0$ such that each is either an axiom or is inferred from earlier $wffs_0$ in the sequence by **Rule R**. If there is a proof of $A \in wffs_o$ in Q_0, then A is a *theorem* of Q_0. From here, derivability $\mathcal{H} \vdash A$ from a finite set $\mathcal{H} \subseteq wffs_o$ of hypotheses is an inductive predicate in Isabelle, faithful to Andrews' prose, which we summarize below:

- if $A \in \mathcal{H}$, then $\mathcal{H} \vdash A$,
- if A is a theorem of Q_0, then $\mathcal{H} \vdash A$, and
- **Rule R'**: if $\mathcal{H} \vdash A_\alpha =_\alpha B_\alpha$ and $\mathcal{H} \vdash C$ and $A \preceq_p C$ and $C\langle\!|p \leftarrow B|\!\rangle \rhd C'$ and there is no variable x_β, free in $A_\alpha =_\alpha B_\alpha$ and $\mathcal{H}$, such that $A \preceq_q (\lambda x_\beta.\ D) \preceq_r C$ for some D (where p is the result of appending the non-empty q to r), then $\mathcal{H} \vdash C'$.

Andrews alternative definition involves two sequences of $wffs_0$ such that the first is a proof in Q_0 and the second has A as its last element, where each formula of the second comes from $\mathcal{H}$, the proof, or an application of **Rule R'**. Díaz formalizes this notion as the predicate *is-hyp-proof-of* and shows its equivalence with the inductive predicate $\mathcal{H} \vdash A$.

3.2 Set-theoretic background

Here, we summarize Paulson's formalization of Zermelo-Fraenkel (ZF) Set Theory [33], that extends Isabelle/HOL, as this is the backbone for Díaz's proof of the soundness of Q_0 [17]. Paulson's formalization declares a type V representing ZF-sets and axiomatizes ZF on it. The type V is different from the type $'a\ set$ of HOL sets. Intuitively, the axiomatization uses a function $elts :: V \Rightarrow V\ set$ that maps each ZF-set to its elements. That is, $elts\ x$ is axiomatized to be the HOL-term of type $V\ set$ whose members are the members of $x :: V$. For instance, part of the axiomatization asserts the extensionality principle for sets as $elts\, x = elts\, y \Rightarrow x = y$, i.e. two sets are equal if they share the same elements. An HOL-set, $X :: 'a\ set$, is *small* if it can be injected into the extension of a V-set. In particular, a V-set is small if it is equal to $elts\ Y$ for some Y. Additionally, Paulson defines the function *set* of type $V\ set \Rightarrow V$, where $set\ x$ has the same members as $x :: V\ set$ if x is small. Moreover, he builds common ZF constructs such as pairs and disjoint unions. In particular, he encodes the set of functions $A \longmapsto B$ with domain $A :: V$ and codomain $B :: V$, as well as the corresponding abstraction and application functions: the operator $\bullet$ such that $f \bullet x$ is the application of $f \in elts(A \longmapsto B)$ to $x \in elts\, A$, and the lambda expression $\boldsymbol{\lambda} x : y\,.\, f\, x$ creates such encoded functions, i.e. x is an Isabelle/HOL variable being bound, the function created has domain $y :: V$ and is the ZF-function mapping members of x:y to $f\ x$ (where $f :: V\ \Rightarrow V$).

(*axiom-1*)	$(\mathfrak{g}_{o\to o} \cdot T_o \wedge^Q \mathfrak{g}_{o\to o} \cdot F_o) \equiv^Q \forall \mathfrak{x}_o.\ \mathfrak{g}_{o\to o} \cdot \mathfrak{x}_0$	
(*axiom-2*)	$(\mathfrak{x}_\alpha =_\alpha \mathfrak{y}_\alpha) \supset^Q \mathfrak{h}_{\alpha\to o} \cdot \mathfrak{x}_\alpha \equiv^Q \mathfrak{h}_{\alpha\to o} \cdot \mathfrak{y}_\alpha$	
(*axiom-3*)	$(\mathfrak{f}_{\alpha\to\beta} =_\alpha \mathfrak{g}_{\alpha\to\beta}) \equiv^Q \forall \mathfrak{x}_\alpha.\ \mathfrak{f}_{\alpha\to\beta} \cdot \mathfrak{x}_\alpha =_\beta \mathfrak{g}_{\alpha\to\beta} \cdot \mathfrak{x}_\alpha$	
(*axiom-4*$_1$)	$(\lambda x_\alpha.\ B_\beta) \cdot A_\alpha =_\beta B_\beta$	where B_β is a constant or a variable different from x_α
(*axiom-4*$_2$)	$(\lambda x_\alpha.\ x_\alpha) \cdot A_\alpha =_\alpha A_\alpha$	
(*axiom-4*$_3$)	$(\lambda x_\alpha.\ B \cdot C) \cdot A_\alpha =_\beta ((\lambda x_\alpha.\ B) \cdot A_\alpha) \cdot ((\lambda x_\alpha.\ C) \cdot A_\alpha)$	with $B \in wffs_{\gamma\to\beta}$ and $C \in wffs_{C_\gamma}$
(*axiom-4*$_4$)	$(\lambda x_\alpha.\ \lambda y_\gamma.\ B_\delta) \cdot A_\alpha =_{\gamma\to\delta} (\lambda y_\gamma.\ (\lambda x_\alpha.\ B_\delta) \cdot A_\alpha)$	where y_γ is not x_α nor any variable in A_α
(*axiom-4*$_5$)	$(\lambda x_\alpha.\ \lambda x_\alpha.\ B_\delta) \cdot A_\alpha =_{\alpha\to\delta} (\lambda x_\alpha.\ B_\delta)$	
(*axiom-5*)	$\iota \cdot (Q_{i\to i\to o} \cdot \mathfrak{y}_i) =_i \mathfrak{y}_i$	
(Rule R)	$\dfrac{A_\alpha =_\alpha B_\alpha \qquad C}{C'}$	where $A \preceq_p C$ and $C\langle\!\langle p \leftarrow B\rangle\!\rangle \rhd C'$

Figure 1. Axioms and rules of inference of Q_0

3.3 Semantics of Q_0

Andrews defines a *frame* as a family $\mathcal{D}_\alpha$ of nonempty domains parametrized by the types α of Q_0 such that $\mathcal{D}_o$ is a two-element set and $\mathcal{D}_{\alpha\to\beta}$ is a set of functions from $\mathcal{D}_\alpha$ to $\mathcal{D}_\beta$. Díaz collects these assumptions in a locale using Paulson's type V as the codomain of $\mathcal{D}$. In this locale, he can closely follow Andrews' definitions of the one-element set $\{x\}_\alpha \in elts\,(\mathcal{D}_\alpha \longmapsto \mathcal{D}_o)$ and the identity relation q_α such that $\mathrm{q}_\alpha \bullet x = \{x\}_\alpha$, for $x \in elts\,(\mathcal{D}\,\alpha)$. From here, an *interpretation* is a pairing of a frame $\mathcal{D}$ with a function $\mathcal{J}$ such that $\mathcal{J}\,Q_\alpha = \mathrm{q}_\alpha$, $(\mathcal{J}\,\iota) \bullet \{x\}_i = x$ for each $x \in \mathcal{D}_i$, and $\mathcal{J}\,c_\alpha \in elts(\mathcal{D}_\alpha)$ for each parameter c_α. Additionally, an *assignment into* $\mathcal{D}$ is a function φ from typed variables to V such that $\varphi(x_\alpha) \in elts\,(\mathcal{D}_\alpha)$. As customary, with an interpretation, if assignments can be structurally extended to specify values to all $wffs_\alpha$ as a function $\mathcal{V}$ such that

- $\mathcal{V}\,\varphi\,x_\alpha = \varphi\,x_\alpha$,
- $\mathcal{V}\,\varphi\,c_\alpha = \mathcal{J}\,c_\alpha$,
- $\mathcal{V}\,\varphi\,(A_{\alpha\to\beta} \cdot B_\beta) = (\mathcal{V}\,\varphi\,A_{\alpha\to\beta}) \bullet (\mathcal{V}\,\varphi\,B_\beta)$, and
- $\mathcal{V}\,\varphi\,(\lambda x_\alpha.\ B_\beta) = \boldsymbol{\lambda} z : \mathcal{D}_\alpha.\ \mathcal{V}\,\varphi[x_\alpha \mapsto z]\,B_\beta$,

then the pairing $\mathcal{M} = \langle \mathcal{D}, \mathcal{J} \rangle$ is a *general model* for Q_0. From these, the *satisfiability*, *validity* in a model, and *general validity* follow—see figure 2.

Soundness of Q_0**.** From the proof system and semantics of Q_0, these results hold: (1) Every theorem of Q_0 is valid in the general sense, i.e. if $\vdash A_o$, then $\models A_o$, and (2) if $\mathcal{M}$ is a model for the set $\mathcal{G} \subseteq wffs_o$, and if $\mathcal{G} \vdash A$, then $\mathcal{G} \models A$.

These statements' proof essentially consists of showing the axioms' validity and preservation of validity by **Rule R**.

3.4 Formalized soundness and consistency

A set $\mathcal{G} \subseteq wffs_o$ is *inconsistent* if falsity can be derived from some subset of hypotheses $\mathcal{H} \subseteq \mathcal{G}$. That is, if there is some $\mathcal{H} \subseteq \mathcal{G}$ with $\mathcal{H} \vdash F_o$. The system Q_0 is inconsistent if $\vdash F_o$. From here, being *consistent* means not being inconsistent. The consistency theorem then states that every set of $wffs_o$ which has a model is consistent. Díaz's formalization [17] does not cover a paragraph in Andrews's textbook [7] where he overloads the turnstile: for finite sets $\mathcal{G}$, $\mathcal{G} \vdash A$ has the meaning given above, but for infinite sets $\mathcal{G}$, Andrews takes $\mathcal{G} \vdash A$ to mean that there exists a subset of hypotheses $\mathcal{H} \subseteq \mathcal{G}$ such that $\mathcal{H} \vdash A$. Therefore the formalized soundness and consistency theorems are for finite sets only in contrast to Andrews' stronger counterparts. Since we wish to prove *strong* completeness in the end, we extend Diaz's formalization by formalizing the more general notion of consistency:

```
definition is-consistent-set :: ‹form set ⇒ bool› where
  ‹is-consistent-set 𝒢 ≡ ∀ℋ ⊆ 𝒢. ¬ is-inconsistent-set ℋ›
```

Unfolding definitions, this means that the set contains no finite subset of sentences from which we can derive falsity:

```
lemma is-consistent-set: ‹is-consistent-set 𝒢
  ⟷ (∄ℋ. ℋ ⊆ 𝒢 ∧ finite ℋ ∧ ℋ ⊆ wffs₀ ∧ ℋ ⊢ F₀)›
```

We have used this more general notion of consistency in our proofs of the following strengthenings of Díaz's work. These are small generalizations, almost all of the work having been done by Díaz, but they bring the formalized statements in line with Andrews' book. First, the second half of Andrews' soundness theorem, 5402(b): if we have a model for a *possible infinite* set and a derivation from *a subset of* that set, then the model satisfies the derived formula:

```
theorem hyp-derivability-implies-validity-general:
  assumes ‹is-model-for ℳ 𝒢›
    and ‹∃ℋ ⊆ 𝒢. ℋ ⊢ A› and ‹is-general-model ℳ›
  shows ‹ℳ ⊨ A›
```

Second, the second part of Andrews' consistency theorem, 5403: every set which has a model is consistent:

```
corollary model-existence-implies-set-consistency-general:
  assumes ‹is-general-model ℳ› ‹is-model-for ℳ 𝒢›
  shows ‹is-consistent-set 𝒢›
```

4 Abstract Consistency Framework

From and Schlichtkrull [19] recently developed a framework for completeness and other meta-logical properties of proof systems based on Smullyan's [40] and Fitting's [18] work on

$$\varphi \text{ satisfies } A_0 \text{ in } \mathcal{M} \text{ (denoted } \mathcal{M} \models_\varphi A_o) \longleftrightarrow \mathcal{V}\ \varphi\ A_o = \mathcal{V}\ \varphi\ T_o,$$
$$A_0 \text{ is valid in } \mathcal{M} \text{ (denoted } \mathcal{M} \models A_o) \longleftrightarrow (\forall \varphi.\ \varphi \text{ assigns into } \mathcal{M} \longrightarrow \mathcal{M} \models_\varphi A_o),$$
$$\mathcal{M} \text{ is a model for the set } \mathcal{G} \subseteq \mathit{wffs}_o \longleftrightarrow (\forall A \in \mathcal{G}.\ \mathcal{M} \models A), \text{ and}$$
$$A_0 \text{ is valid in the general sense (denoted } \models A_o) \longleftrightarrow (\forall \mathcal{M}.\ \mathcal{M} \text{ is a general model} \longrightarrow \mathcal{M} \models A_o).$$

Figure 2. *Satisfiability*, *validity* in a model, and *general validity*.

abstract consistency properties. These are intuitively "set-membership" descriptions of what is needed to prove model existence for formal systems. ACPs are closure conditions imposed on collections C of sets of formulas. Examples of possible conditions are the requirement that the conjuncts A_1 and A_2 of a conjunction (or conjunction-equivalent formula) $A_1 \wedge A_2$ such that $(A_1 \wedge A_2) \in \mathcal{G} \in C$ also satisfy $\mathcal{G} \cup \{A_1, A_2\} \in C$, or dually, that a disjunct A_ℓ of a disjunction (or disjunction-equivalent formula) $A_1 \vee A_2$ (with $\ell \in \{1, 2\}$) such that $(A_1 \vee A_2) \in \mathcal{G} \in C$ also satisfies $\mathcal{G} \cup \{A_\ell\} \in C$. Our ACP for Q_0 appears in Figure 3. This basis allowed From and Schlichtkrull to mechanize the Lindenbaum-Tarski construction only once and build maximal consistent sets for any logic that such an ACP can characterize. They applied the framework to first-order logic, second-order logic and a hybrid logic with propositional quantification, and wondered whether the framework would scale to lambda terms. The examples were all developed by the authors themselves and have common features: de Bruijn indices for the binders and type variables for the non-logical constants (*parameters*). We ask whether the framework is flexible enough to apply to (Díaz's mechanization of) the Q_0 system, which, as seen above, has a different flavor. With a slight generalization of their work, we can answer both questions in the affirmative.

Model existence. Applying the framework has three steps:

Step 1 define an ACP based on Q_0 and prove that it satisfies the framework's requirements (see Section 5),

Step 2 use the automatically derived Hintikka properties of maximal consistent sets to prove abstract model existence (see Section 6), and

Step 3 show that sets which are consistent with respect to Q_0 constitute a consistency property (see Section 7).

Where Andrews can use the full power of Q_0's proof system for his model existence result, we crystallize our assumptions in **Step 1** and can only work with the framework's output (based on our **Step 1** input). We then discharge those assumptions in **Step 3**. The ACP's role as *interface* between the proof's two parts allows us to reuse the model existence for any calculus meeting the same requirements.

Parameter substitution. Díaz's Q_0 system uses natural numbers for constants. Two of these, 7 and 8, are the logical constants for Q_α and ι, respectively. The rest are the uninterpreted parameters. From and Schlichtkrull's original version of the framework [19] expects a distinct type parameter for logical parameters with free reign over all values. To meet step (2) above, we had to introduce a predicate *is-param* that distinguishes the logical and non-logical constants, and update the definitions and proofs to work with only a subset of the full type. From and Schlichtkrull have applied our updates in in the newest version of the framework. The extra flexibility impairs no extra burden in the case where all values of the type are parameters, and only causes minor busywork in the case of Q_0. Alternatively, we could have updated the Q_0 mechanization to distinguish these two types of constants structurally in the data type, but if we want to apply the framework to other logics in the future, the work was better spent on allowing arbitrary subsets of the constant type as parameters. The tension between types and sets in Isabelle/HOL [29] seems to be an ever-present one.

5 Abstract Consistency Property

We start at **Step 1** of applying the framework to build a maximal consistent set instead of mechanizing the Lindenbaum-Tarski construction ourselves. Curiously, while Andrews [7] uses ACPs in the first-order logic part of his book, he hand-rolls the extension lemma for higher-order logic. Figure 3 lists the conditions of our ACP. The works of Andrews [5] (on resolution) and Steen [42, 43] (on paramodulation) have inspired us but our definition ultimately comes from pragmatism. We state the ACP to avoid manually mechanizing the Lindenbaum-Tarski's construction of maximal consistent sets, but the rest of the model existence proof follows his textbook [7] closely. Figure 3's properties link model/semantics with calculus/syntactics; the stronger we make the ACP, the more information we have for building the model, but the more obligations the proof system must fulfill.

5.1 Consistency conditions

Figure 3 has four types of conditions, and a specification of witnessing. The first type, conflicts, says that falsity must not appear in a consistent set and if the formula $\sim^Q A$ occurs, A must not. The second type, α, says that when all formulas on the left occur in a consistent set, there must be a consistent set where all formulas on the right also occur. The first case embeds Andrews' rule T (5219), every true formula is equal to truth, and the second one embeds the axiom of descriptions (*axiom*-5, but for arbitrary wffs_i). Notice that the letter α on the arrows corresponds to the framework's notation, but that in the equalities corresponds to Díaz's formalization.

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

(alpha) **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*: ‹[] $\rightsquigarrow_\alpha$ [$(\lambda x_\alpha.\ B) \cdot A =_\beta \mathbf{S}\ \{(x, \alpha) \mapsto A\}\ B$]› **if** ‹$A \in wffs_\alpha$› **and** ‹$B \in wffs_\beta$› **and** ‹*free-vars* $A = \{\}$›

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

(gamma) **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** ‹$B \in wffs_{\alpha \to \beta}$›

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

Figure 3. Abstract consistency property: Conflicts α-, β-, γ- and δ-formulas and their respective parts.

The third case asks for reflexivity of equality (5200) and the fourth for transitivity (5201(3)). The fifth says functions are equal on equal arguments (5201(6)) and the sixth represents β-reduction (5207). The third type, β, says that if all the formulas on the left are in a consistent set, then at least one of the formulas on the right must be consistent with it. We abuse it here to include the law of excluded middle (tautology). The fourth, singular γ case works like α but with a parameterized set of formulas all consistent with the left-hand side of the arrow $\rightsquigarrow$. Here, we embed function extensionality (5201(5)): equal functions are equal on all arguments (of the right type). Finally, the δ function (*delta*) *characterized* at very end of figure 3 witnesses inequalities: if two functions are unequal, then they are unequal at some argument. The framework provides parameter c—guaranteed to be fresh. The actual specification of the function involves unpleasant pattern matching via a separate inductive definition, since the negated equality is a "non-constructor pattern", so we chose to show the much clearer lemma here.

We have tried to be economical in our selection above. In Andrews' [5] work on resolution, he includes properties for all the logical connectives and their negations. We only talk about equality (Q_α), definite description (ι), and negation as best we can, recalling that negation just means equal to falsity. Since Q_α and ι are our only logical constants, this seems natural. Then come the types: propositions o, individuals i and functions $\alpha \to \beta$. We need to know something about equality on all of these, as well as about equality itself.

Steen [42, 43] does not include the law of excluded middle as we do but derives it from other properties. One states that if two propositions are equal, they are either both true (both included) or both false (both negations included). This statement is a disjunction of conjunctions and does not fit the framework, as it is based on Smullyan's uniform notation; we have α for conjunctions, β for disjunctions, but not their combination. We could use β and put syntactical conjunctions on the right-hand side, but then we would need to include rules for conjunction. Including LEM seemed cleaner.

5.2 Framework obligations

We must convince the framework of the fidelity of our specification in figure 3. Crucially, we must explain what parameter substitution means and show our properties are invariant under it. For previous framework applications the authors used Isabelle's built-in mapping function generated for data-types with type variables. This comes with the expected properties such that substituting by the identity function is the identity operation (*map-fm id = id* in the framework), and a congruence property ($\forall x \in$ *params-fm p*. *f x = g x* ⟹ *map-fm f p = map-fm g p*). Building on Díaz mechanization [17], which hard-codes the type of constants, we have no such luxury and code the substitution ourselves:

fun *map-con* :: ‹(*nat* ⇒ *nat*) ⇒ *form* ⇒ *form*›
where ‹*map-con* - (x_α) = (x_α)›
| ‹*map-con f* ($\{\!|c|\!\}_\alpha$) = (*if is-logical-name c* ∨ *is-logical-name* (*f c*) *then* $\{\!|c|\!\}_\alpha$ *else* $\{\!|f\,c|\!\}_\alpha$)›
| ‹*map-con f* ($A \cdot B$) = *map-con f A* · *map-con f B*›
| ‹*map-con f* ($\lambda x_\alpha.\ A$) = λx_α. *map-con f A*›

The case for constants avoids replacing or introducing logical constants. Thus we only touch parameters, and do not change any wff's logical structure. We could also have restricted the allowed arguments f to only functions not violating the property. We chose the above to make more properties hold unconditionally, e.g. preservation of typing:

lemma *wff-map-con* [*iff*]:
‹*map-con f A* ∈ $wffs_\alpha$ ⟷ $A \in wffs_\alpha$›

We supply this parameter substitution to the framework along with a function that extracts the parameters of a given

wff and predicate on constants that holds only for parameters. We must then prove the above identity and congruence properties, and that wffs have finitely many parameters.

For every relation in figure 3, we prove that when two lists are related by $\rightsquigarrow$, their parameter substitutions are also related. The framework needs this property to build maximal consistent sets [19]. Showing it for the *delta* function turned out somewhat involved, but having explained to the framework which constants are logical and which are parameters, we are guaranteed it never gives us a logical constant to be used as a parameter. For the universal condition (γ), we use the framework's ability to restrict the set of terms used for instantiation; in our case to only wffs of the proper type. This set is constant —it does not rely on the set of formulas that contained the hypotheses— so the additional proof obligations from the framework are trivial. All but the *delta* cases are automatically discharged by Isabelle's proof automation.

These results compose to show that we have defined an ACP supported by the framework.

5.3 Extension lemma

With the framework accepting our ACP, we immediately get a version of the Lindenbaum-Tarski lemma. Let us first look at Andrews' 5500 extension lemma. He calls a sentence set $\mathcal{H}$ *complete in* Q_0 if for each sentence A of Q_0, either $\mathcal{H} \vdash A$ or $\mathcal{H} \vdash {\sim}^Q A$. From here, Lemma 5500 asserts that if $\mathcal{G}$ is a consistent set of sentences of Q_0, then Q_0's language can be expanded with parameters and one can obtain a superset $\mathcal{H}$ of $\mathcal{G}$ in the expansion that is consistent, complete and extensionally complete (see the definition below). Andrews' proof of this requires extending Q_0's language with language-of-Q_0-sized well-ordered sets C_α, one for each type α, and it uses transfinite induction to build the superset $\mathcal{H}$ of $\mathcal{G}$.

In contrast to Andrews, we do not extend the language, because we inherit the framework's assumption that the starting set $\mathcal{G}$ has "enough" new constants unused, a hypothesis (*P.enough-new S*) reflected in Section 6's model existence result. Also, the framework does the transfinite induction for us and provides the maximal consistent set, with some notable differences to the above. While Andrews shows that each of his extensions to $\mathcal{G}$ are consistent, our consistency requirements are transferred instead to **Step 3**. Plus, the maximal consistent set is consistent and maximal with respect to the ACP we wrote down, not to the proof system. That is, consistency hinges on our stated conflicts of figure 3, and maximality means there is no superset of the resulting set that remains consistent with respect to the ACP. We see now how the Hintikka properties arise—e.g. the extension only contains a conjunction if it also contains its conjuncts. As we have added the law of excluded middle to our ACP, the result is *complete* in the same sense as Andrews'.

Finally, we approach "extensional completeness" in a set-theoretical approach, where we opt for Hintikka-set membership ($\in$) instead of derivability via turnstiles ($\vdash$). Andrews states that the set $\mathcal{H}$ is *extensionally complete in* Q_0 if each sentence $A_{\alpha\to\beta} = B_{\alpha\to\beta}$ admits a cwff C_α of Q_0 such that:

$$\mathcal{H} \vdash (A_{\alpha\to\beta} \cdot C_\alpha =_\beta B_{\alpha\to\beta} \cdot C_\alpha) \supset^Q A_{\alpha\to\beta} =_{\alpha\to\beta} B_{\alpha\to\beta}.$$

We work instead with the following definition, notably with no object-logic implication:

```
definition extensionally-complete-membership ::
  ‹form set ⇒ bool› where
  ‹extensionally-complete-membership H ⟷
    (∀ A B α β. is-closed-wff-of-type A (β → α) ⟶
      is-closed-wff-of-type B (β → α) ⟶
        (∃ C. is-closed-wff-of-type C β ∧
          ((A ⋅ C) =α (B ⋅ C) ∈ H ⟶ (A =β → α B) ∈ H)))›
```

That our maximal consistent sets are extensionally complete follows from the law of excluded middle and the *delta*-witnessing of unequal functions given in figure 3.

6 Model Existence Theorem

Now to **Step 2** of applying the framework. Andrews proves a model existence theorem (5501 Henkin's Theorem). We also prove an (abstract) model existence theorem—our proof and model construction adapts Andrews' proof to Paulson's Isabelle/HOL set theory and Hintikka sets. We discuss the proof and our adaptation here. We discuss how to use formulas in our model by converting to and from type *V*. Then we define the canonical model, and prove its defining properties, before showing it to correctly interpret constants. We prove that we have a *general* model, that it is frugal and satisfies the set of sentences it was built from. This last part is the *abstract* model existence result, which applies to any proof system whose consistency forms a consistency property.

6.1 Conversion from *form* to *V*

A prerequisite for our definition of the canonical model is a conversion from *form* to *V*. Andrews' proof of 5501, Henkin's theorem, defines the canonical model and his valuation evaluates wffs to elements built from wffs—e.g. a wff of type *i* is evaluated to a set of wffs. We can express this set as a *form set* but valuations must return terms of *V* in the formalization. Therefore we need to convert between, on the one hand, *form* (an HOL datatype) and *form set* (an HOL type), and on the other hand, *V* (set-theoretical sets). We introduce these conversions as follows: we prove that *type* and *form* are members of the *countable* type class using the *countable-datatype* proof method. Paulson defines a type class *embeddable* for types that have an injective function into *V*, and he shows that *countable* is a subclass of *embeddable*. We get access to an injection from *form* into *V* using Hilbert's choice operator and lift it to sets by taking the image on the set and converting the resulting *V set* to *V* using the *set* function.

```
fun 𝒟 :: ‹type ⇒ V› and 𝒱 :: ‹form ⇒ type ⇒ V›
 and get-rep :: ‹V ⇒ type ⇒ form›
 where ‹𝒟 o = 𝔹›
| ‹𝒟 i = set {𝒱 A i | A. is-closed-wff-of-type A i}›
| ‹𝒟 (β → α) = set {𝒱 A (β → α) | A. is-closed-wff-of-type
A (β → α)}›
| ‹𝒱 A o = (if A ∈ H then T else F)›
| ‹𝒱 A i = V-of-form-set {B. is-closed-wff-of-type B i ∧ A =ᵢ
B ∈ H}›
| ‹𝒱 A (β → α) = (λVCβ : 𝒟 β. (let C = get-rep VCβ β in
𝒱 (A · C) α))›
| ‹get-rep VCβ β = (SOME C. 𝒱 C β = VCβ ∧
is-closed-wff-of-type C β)›
```

Figure 4. Mutually recursive definitions for the frame and valuation in our model-existence result.

```
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)›
```

6.2 Defining the canonical model

Andrews defines a canonical model by defining by induction on the type γ the domains $\mathcal{D}_\gamma$ and a function $\mathcal{V}_\gamma$ so that

1^γ: $\mathcal{D}_\gamma = \{\mathcal{V}\ A_\gamma |\ A_\gamma$ is a $cwff_\gamma\}$

2^γ: for all $cwff$s A_γ and B_γ, $\mathcal{V}A_\gamma = \mathcal{V}A_\gamma$ iff $H \vdash A_\gamma = B_\gamma$.

The overview of Andrews' proof is: (First base case — booleans) Andrews defines $\mathcal{D}_o$ and $\mathcal{V}_o$ and then proves 1° and 2°. (Second base case—individuals) Andrews defines $\mathcal{D}_i$ and $\mathcal{V}_i$ and then proves 1^i and 2^i. (Induction case—functions) Andrews defines $\mathcal{D}_{\beta\to\alpha}$ and $\mathcal{V}_{\beta\to\alpha}$ using recursion on β and α and proves $1^{\beta\to\alpha}$ and $2^{\beta\to\alpha}$, using the induction hypothesis.

Isabelle does not have a command for mixing definition and proof as Andrews does. We do not introduce such a command and rather show that the present tooling suffices. A strategy for "simulating" such a definition in Isabelle would be to prove the theorem $\exists D\ V.\ 1^\gamma \wedge 2^\gamma$ by induction on γ, thus hiding the definition behind the existential quantifier. In the recursive case ($\beta \to \alpha$), one could then `obtain` the $\mathcal{D}_\alpha$, $\mathcal{D}_\beta$, $\mathcal{V}_\alpha$ and $\mathcal{V}_\beta$ that the induction hypothesis would claim exists. After proving that, one could define $\mathcal{D}_\gamma$ and $\mathcal{V}_\gamma$ outside of the scope of the theorem using Hilbert choice. Yet, we follow a different strategy that does not excessively introduce existential quantifiers: we separate definitions and proofs. The mutually recursive (in the *type*) functions in Figure 4 capture the construction from Andrews' proof with changes: Compared to Andrews, we have replaced provability ($\vdash A$) with Hintikka set membership ($A \in H$) and adapted the definitions to Isabelle/HOL and Paulson's set theory.

We directly use Andrews' definitions for o and i, except for the use of the *set* function discussed above in section 3.2. We explain here for all types the intuition and formalization:

(Type o) The domain of booleans, $\mathbb{B}$: the two-element set from Díaz's formalization containing two arbitrary different elements **T** and **F**. The valuation of $A :: form$ of type o maps it to **T** if it is a member of H and **F** otherwise.

(Type i) The valuation of an individual A is the V of closed *wffs* of type i that the Hintikka set considers equal to A. The domain of i is the image of the valuation (on $cwffs_i$).

(Type $\beta \to \alpha$) In Andrews' words but our notation:

> To define $\mathcal{D}_{\beta\to\alpha}$, suppose $\mathcal{D}_\alpha$ and $\mathcal{D}_\beta$ are defined, and $A_{\beta\to\alpha}$ is a $\text{cwff}_{\beta\to\alpha}$. Let $\mathcal{V}A_{\beta\to\alpha}$ be the function from $\mathcal{D}_\beta$ to $\mathcal{D}_\alpha$ whose value, for any argument $\mathcal{V}C_\beta \in \mathcal{D}_\beta$, is $\mathcal{V}[A_{\beta\to\alpha}C_\beta]$.
>
> ...
>
> $\mathcal{D}_{\beta\to\alpha} = \{\mathcal{V}A_{\beta\to\alpha} \mid A_{\beta\to\alpha}$ is a $\text{cwff}_{\beta\to\alpha}\}$.

We represent $\mathcal{V}A_{\beta\to\alpha}$ as a lambda ($\boldsymbol{\lambda}$) expression denoting a ZF-function. Andrews can write "for any argument $\mathcal{V}C_\beta \in \mathcal{D}_\beta$" since any D_β member, *by induction* (specifically 1^γ), is the valuation of some C_β. Thus, he implicitly chooses C_β. In our case, we explicitly choose C_β using the function *get-rep*, which chooses the appropriate *form* using Hilbert choice. We explain the intuition for function types:

The valuation of $A_{\beta\to\alpha}$ is $\boldsymbol{\lambda} b : \mathcal{D}\ \beta.$ ($let\ C = get\text{-}rep\ b\ \beta\ in\ \mathcal{V}\ (A \cdot C)\ \alpha$)). This function (encoded as a V) takes any member of β's domain, then finds C, a *form* that represents that member, syntactically applies A to C, and recursively evaluates $\mathcal{V}$ on $A \cdot C$ and α. This is possible because we recurse on the type, and $A \cdot C$'s type α is a part of $\beta \to \alpha$.

The domain of $\beta \to \alpha$ is the image of the valuation (on the set of closed wffs of type i). Thus the domain will only contain those functions expressible in Q_0's syntax. It is thus not the full function space $\mathcal{V}\ \beta \longmapsto \mathcal{V}\ \alpha$ but only a subset.

6.3 The defining properties of the canonical model

With $\mathcal{V}$ and $\mathcal{D}$, we prove our adapted versions of 1^γ and 2^γ:

```
lemma one-gamma: ‹𝒟 γ =
 set {𝒱 A γ| A. is-closed-wff-of-type A γ}›

lemma two-gamma:
 assumes ‹is-closed-wff-of-type A γ›
  and ‹is-closed-wff-of-type B γ›
 shows ‹𝒱 A γ = 𝒱 B γ ⟷ A =γ B ∈ H›
```

Lemma *one-gamma* is faithful to Andrews' book except the application of *set*. Lemma *two-gamma* is as in Andrews' book but with Hintikka set membership rather than provability. Andrews proved 1^γ and 2^γ in the induction defining $\mathcal{V}$ and $\mathcal{D}$—but we prove them in sequence after the definition of $\mathcal{V}$ and $\mathcal{D}$. As *one-gamma* in all cases follows essentially "by definition" no induction is needed. For *two-gamma* the proof is by induction and an adaptation of Andrews' proof using

properties in figure 3. The most interesting case is the inductive case, where $\gamma = \beta \to \alpha$. Here, Andrews proves using the induction hypothesis that $\mathcal{V}A_{\beta\to\alpha}$ is unambiguous; i.e. the definition does not depend on the aforementioned implicit choice of a representative wff for the argument it receives. If it were not unambiguous then one could not meaningfully call $\mathcal{V}A_{\beta\to\alpha}$ a function as it would have different candidates for its output on the same input. Arguably our definition does not have this "problem" as we let Hilbert choice pick the representative formula and thus it is clear what is returned because there is no implicit choice — *get-rep* explicitly lets Hilbert choice choose. We prove unambiguity none-the-less:

```
assume ‹is-closed-wff-of-type A (β → α)›
 ‹is-closed-wff-of-type B β›  ‹is-closed-wff-of-type C β›
 ‹𝒱 B β = 𝒱 C β›
then have ‹𝒱 (A ⋅ B) α = 𝒱 (A ⋅ C) α›
```

At first it was not clear to us if this was useful—Andrews does not explicitly state the use of unambiguity beyond for defining the function, however we used unambiguity to prove that $\mathcal{V}$ distributes over function application, and this is used several times in Andrews' proof and in our adapted formalization. To have access to unambiguity and distributivity of $\mathcal{V}$ over function application outside *two-gamma*'s proof's scope we reprove it outside this scope—this time using directly *two-gamma* rather than its induction hypothesis.

6.4 Interpretation of constants

Following the definitions in section 3.3, we must build an *interpretation* [7] that maps each constant to an element of the corresponding type, interprets Q_α as the identity relation and ι as the selection operator for individuals. We use $\mathcal{V}$:

```
fun 𝒥 :: ‹nat × Syntax.type ⇒ V›
 where ‹𝒥 (c,τ) = 𝒱 (FCon (c,τ)) τ›
```

The well typedness of $\mathcal{J}$ follows from the well typedness of $\mathcal{V}$ (which holds by a straight-forward induction):

```
lemma well-typed:
 assumes ‹is-closed-wff-of-type A γ›
 shows ‹𝒱 A γ ∈ elts (𝒟 γ)›

lemma non-logical-constant-denotation-𝒥:
 assumes ‹¬ is-logical-constant (c, α)›
 shows ‹𝒥 (c, α) ∈ elts (𝒟 α)›
```

That Q_α denotes the identity relation follows from the 2^γ property above, among other things. We want to prove the following, where ${q_\alpha}^{\mathcal{D}}$ is the set-theoretic identity relation:

```
lemma Q-denotation-𝒥: ‹𝒥 (Q-constant-of-type α) = q_α^𝒟›
```

We know from 2^γ that $\mathcal{V}$ sends arguments A and B to the same element in V iff we have a witness to their equality in H: $A =_\alpha B \in H$. By the definition of $\mathcal{V}$, this holds exactly when $\mathcal{V}\ A\ B\ o = T$, which, by the distribution of $\mathcal{V}$ across application, holds when $\mathcal{V}\ Q_\alpha$ applied to the valuations of A and B is true. Taking it together, the denotation of Q_α behaves exactly as the identity relation. The proof for ι follows similar lines, but using the condition *CIota* of ι in figure 3 instead:

```
lemma ι-denotation-𝒥:
 ‹frame.is-unique-member-selector 𝒟 (𝒥 iota-constant)›
```

6.5 The canonical model is a model

Closing time. Our valuation $\mathcal{V}$ is only defined on closed wffs, but for the general model, we need a valuation on all wffs. To get around this, Andrews defines $\mathcal{V}_\varphi$, which closes its argument wff C using φ before passing it to $\mathcal{V}$. To do this, Andrews assumes a fixed enumeration of closed wffs and explicitly substitutes every free variable x_α in the argument C with *the first* cwff E_α such that $\varphi\, x_\alpha = \mathcal{V}E_\alpha$ for an assignment φ into the model. That is, while multiple *cwffs* might hit the same element of the domain under the valuation, Andrews asks for the first one. We use Hilbert's choice once more, and simply ask for *some* E_α such that $\varphi\, x_\alpha = \mathcal{V}E_\alpha$. What matters is not that we pick the first, but that we always pick the same and Hilbert's choice suffices.

Andrews writes that the result of the above process, replacing every free variable with a closed wff (of the same type), is "clearly" a closed wff itself (of the same type). While intuitive, proving this in Isabelle/HOL required a bit of work, which includes an inductive proof stating that substituting a free variable with a term replaces that free variable with those of the term, and a busywork proof saying that all this replacement constitutes a *substitution* in the technical sense.

General model properties. We must prove five properties of $\mathcal{D}$, $\mathcal{J}$ and $\mathcal{V}_\varphi$ to show that they constitute a general model. Firstly, we need to show a simple closure property saying that our $\mathcal{V}_\varphi$ maps wffs to elements of the domain with the correct type. This follows from the above work asserting that closing a formula is a type-preserving substitution.

```
lemma g:
 assumes φ: ‹φ ⇝ 𝒟› and A: ‹A ∈ wffs_α›
 shows ‹𝒱_φ A ∈ elts (𝒟 α)›
```

Next, the four proper conditions (a-d) on general models: that the valuation acts *appropriately* on variables, constants, applications and lambda abstractions. Let us take a look:

```
lemma denotation-function-a:
 assumes φ: ‹φ ⇝ 𝒟›
 shows ‹𝒱_φ (x_α) = φ (x, α)›
```

Variables x_α are replaced by a cwff E_α s.t. $\varphi x_\alpha = \mathcal{V}E_\alpha$.

```
lemma denotation-function-b: ‹𝒱_φ (⦇c⦈_α) = 𝒥 (c, α)›
```

As constants have no free variables this is by definition.

```
lemma denotation-function-c:
 assumes φ: ‹φ ⇝ 𝒟› and A: ‹A ∈ wffs_β → α›
  and B: ‹B ∈ wffs_β›
 shows ‹𝒱_φ (A ⋅ B) = 𝒱_φ A • 𝒱_φ B›
```

Andrews proves this property in one step by simply redistributing the closing operation over application. In Isabelle, this is not as simple: the closing substitution for $A \cdot B$ replaces the free variables of both A and B with *cwffs*, but it only applies to A and B, separately. A congruence principle via a distinct induction is required to show that extra free variables do not affect the result and the redistribution holds.

```
lemma denotation-function-d:
 assumes φ: ‹φ ⇝ 𝒟› and B: ‹B ∈ wffs_β›
 shows ‹𝒱_φ (λx_α. B) = (λz:𝒟 α . 𝒱_φ((x, α) := z) B)›
```

Finally, evaluating a lambda actually produces a function. Here, Andrews uses that *first* cwff E to represent an element of the domain, which for us is whichever Hilbert's choice picks. Andrews uses β-reduction to prove the two sides equal. We do not speak of the proof system and instead use the substitution condition *CSubst* from figure 3. However, combining β-reduction with the closing substitution is more complicated than Andrews' prose suggests and involves consolidating the two substitutions using an induction (via a lemma from Díaz's mechanization [17]).

6.6 Frugality

We have built a *frugal* model, that is, we have added fewer elements to the domain of each type symbol than formulas in our language for talking about those elements:

```
definition is-frugal :: ‹model-structure ⇒ bool›
where ‹is-frugal ℳ ≡ case ℳ of
  (𝒟, 𝒥, 𝒱) ⇒ ∀ α. |elts (𝒟 α)| ≤o |UNIV :: form set|›

theorem is-frugal: ‹is-frugal (𝒟,𝒥,𝒱φ)›
```

This follows from the stronger property that we have no more elements at each type symbol than *wffs* of that type:

```
lemma frugal-wffs: ‹|elts (𝒟 α)| ≤o |wffs_α|›
```

Our model is syntactic and our domain is of *wffs*, so this should come as no surprise, but it allows for a stronger completeness result later on. Since Díaz's mechanization [17] fixes the type of parameters (nonlogical constants) to be natural numbers, his language is, unfortunately, inherently countable—so here frugality is just countability:

```
lemma is-frugal-countable: ‹is-frugal (𝒟, 𝒥, 𝒱)
 ⟷ ( ∀ α. |elts (𝒟 α)| ≤o |UNIV :: nat set| )›
```

We want to lift this language restriction and allow any type of parameters, more in line with Andrews. The framework [19] supports uncountable languages, but changing Díaz's mechanization is future work. We build on his formulation as-is, showing a strength of the framework.

6.7 The canonical model satisfies the Hintikka set

The fruit of this section is an *abstract* model existence result:

```
theorem model-existence:
 fixes S :: ‹form set›
 assumes cprop: ‹P.prop_E Kinds C›
  and S: ‹S ∈ C› ‹P.enough-new S›
 shows ‹∃ M. is-general-model M ∧ is-frugal M ∧
  (∀ A ∈ S. is-sentence A ⟶ M ⊨ A) ∧
  (∀ A. is-sentence A ⟶ ¬ (M ⊨ A ∧ M ⊨ ~^Q A))›
```

The *cprop* assumption certifies C as a consistency property in our sense. For a consistent set S (in the C sense), leaving enough new parameters witnessing all existentials, there is a general, frugal model satisfying all sentences in S acting as expected under negation. The last part, needed for completeness, is not derivable from Díaz's [17] characterization of general models, but holds for our concrete canonical model.

We restrict ourselves to sentences here since $\mathcal{V}_\varphi$'s closing operation does not affect those, so $\mathcal{V}_\varphi A = \mathcal{V} A$, which entails that $A \in H$ ensures A's satisfaction. For wffs with free variables, $A \in H$ does not imply $\mathcal{V}_\varphi\, A \in H$ as the latter is a different wff and H is not necessarily closed under such substitutions of free variables for "equivalent" closed terms. Exploring this possible extension of the proof to open wffs is future work. Andrews also restricts himself to sentences.

7 Derivational Consistency

The last framework application step is **Step 3**. In section 6, we obtained an *abstract* model existence result contingent on a consistency property that fulfils the specification in figure 3. Here, we prove that any set of formulas from which no subset proves falsity is such a consistency property. The result above then gives us a model for such sets.

The framework asks us to instantiate a locale for each kind of condition —conflicts (confl), conjunctive (alpha), disjunctive (beta), existential (delta) and universal (gamma)— proving the connection between the chosen notions of concrete and abstract consistency. For conflicts, we must prove that if a set contains both every formula on the left-hand side of $\rightsquigarrow\!\!\!\!\times$ and any formula on the right-hand side, the set is inconsistent. For conjunctive (alpha) properties, we prove that if a consistent set contains every formula on the left of our alpha specification, then adding all formulas on the right preserves consistency. For disjunctive (beta) properties, we prove that some formula on the right of $\rightsquigarrow_\beta$ preserves consistency. Universal properties (gamma) rely on a notion of terms (here the *wffs* themselves), and we must show that extending a consistent set with any instantiation preserves consistency. The next section covers existentials (gamma).

The proofs are mostly busywork, but two things stand out. (1) We use various tautologies to get formulas into correct shapes or to prove outright theorems e.g. the law of excluded middle—Andrews gives it only as a tautology. (2) He defines *axiom*-5 (for ι) at constants, but we use it at any wff, so we generalize it explicitly with the Rule of Substitution (5221). Andrews rather states after Rule 5221 that he will, when

confusion is unlikely, designate by theorem numbers also their alpha-renamings and substitution instances without explicit reference to these—we do not have this luxury.

7.1 Delta formulas and constant substitution

As promised, we now cover our instantiation of the locale for existentials to prove the connection between the chosen notions of concrete and abstract consistency. We must prove that if a consistent set C contains the inequality $A_{\beta\to\gamma} \neq B_{\beta\to\gamma}$ of two functions (c.f. the first argument of the right-hand side of the equality characterizing delta) then it is consistent to add an inequality $A_{\beta\to\gamma}{\cdot}c_\beta \neq B_{\beta\to\gamma}{\cdot}c_\beta$ on a "witness" c_β (c.f. the left-hand side of the equality characterizing delta). The witness is a fresh constant c — intuitively we can think of this as naming the point on which the two functions differ.

The reason for this also occurs inside Andrews' 5500 Extension Lemma proof. He proves it by contradiction and so does our formalization, but we find the exposition clearer with a contrapositive proof: Assume C contains $A_{\beta\to\alpha} \neq B_{\beta\to\alpha}$, that c is fresh, and contrapositively that $C \cup \{A_{\beta\to\alpha}{\cdot}c_\beta \neq B_{\beta\to\alpha}{\cdot}c_\beta\}$ is inconsistent. We then want to prove that also C is inconsistent. Our figure 5 sketches the contrapositive proof.

Extensionality. The extensionality step in figure 5 requires more rigor. Andrews uses the derived Rule RR (not shown) with *axiom-3* (extensionality axiom). The formalization demands further rigor due to the generalization of an axiom applying to *wffs*, not only variables. Andrews uses *axiom-3* generalized to any wff and we explicitly prove this generalization using the Rule of Substitution (5221), alpha conversion (5206) and **Rule R**. The generalization of *axiom-3* requires $A_{\beta\to\alpha}$ and $B_{\beta\to\alpha}$ to not have x_β as a free variable.

Constant substitutions. In substituting by a fresh variable, Andrews is more rigorous than the sketch above: (1) He lets $\mathcal{P}$ denote the proof justifying $C \vdash A_{\beta\to\alpha} \cdot c_\beta = B_{\beta\to\alpha} \cdot c_\beta$. (2) He lets x denote a variable occuring neither in C nor $\mathcal{P}$. (3) He applies the notation $\mathbf{S}^{c_\beta}_{x_\beta}\mathcal{P}$ as "the result of substituting x_β for c_β throughout $\mathcal{P}$" for a proof $\mathcal{P}$. (4) He states that as c_β does not occur in C, $A_{\beta\to\alpha}$ or $B_{\beta\to\alpha}$, it is easily seen that $\mathbf{S}^{c_\beta}_{x_\beta}\mathcal{P}$ proves $A_{\beta\to\alpha} \cdot x_\beta = B_{\beta\to\alpha} \cdot x_\beta$ from C.

The formalization requires even further rigor as this last "easily seen" claim became the main part of a ~1100 LOC theory file *Constant_Substitution.thy* consisting of the two definitions below and their accompanying lemmas and proofs.

First we need to formalize $\mathbf{S}^{c_\beta}_{x_\beta}\mathcal{P}$ as "the result of substituting x_β for c_β throughout $\mathcal{P}$" where $\mathcal{P}$ is a proof". We refer to such substitutions as *constant substitutions*. Constant substitutions are an instance of the substitutions in Andrews' book (also in his chapters on propositional and first-order logic). They simultaneously replace all occurrences of a given formula with another given formula, avoiding variable capture. This is in addition to a different notion of substitution of terms for variables, that also takes variable capture into account (used everywhere else where that is a concern). Díaz covers the latter but not the former as the first time the former appears in Andrews' chapter on Q_0 is in his Extension lemma proof. Since the former is a substitution for constants and the latter is not for constants but for variables, the latter cannot be reused from Díaz's formalization in a definition of the former—and Díaz did not need to define them on whole proofs as we do. Another candidate for reuse would be the notion of replacement from Q_0's derivation rules, but that would not easily work out because it replaces a single occurrence (in one position) rather than all occurrences—and Díaz did not need to apply replacements on whole proofs as we do with constant substitutions.

We therefore formalize constant substitutions ourselves. Rather than general term-for-term substitutions, we formalize constant substitutions as substituting a single variable for all occurrences of a single constant—this suffices. In Díaz's formalization proofs are *form lists*. We can thus formalize the desired constant substitution of a constant c_β for a variable x_β (x's type is implicitly β) followed by the wanted property:

```
fun const-subst :: ‹con ⇒ nat ⇒ form ⇒ form› (‹S_c - - -›
[51, 51, 51]) where ‹S_c (c, β) x (y_α) = y_α›
 | ‹S_c (c, β) x (⦇d⦈_α) = (if c = d ∧ β = α then (x_α) else
(⦇d⦈_α))›
 | ‹S_c (c, β) x (A · B) = (S_c (c, β) x A) · (S_c (c, β) x B)›
 | ‹S_c (c, β) x (λy_α. A) = (λy_α. S_c (c, β) x A)›

fun const-subst-proof :: ‹con ⇒ nat ⇒ form list ⇒ form list›
(‹S_cp - - -› [51, 51, 51])
 where ‹S_cp (c, β) x P = map (λA. S_c (c, β) x A) P›

lemma is-proof-const-subst:
 assumes ‹is-proof P› and ‹c ∉ logical-names›
  and ‹(x, β) ∉ vars_p P›
 shows ‹is-proof (S_cp (c, β) x P)›
```

Our proof uses the induction principle below:

```
lemma is-proof-induct :
 assumes ‹is-proof P› and p-nil: ‹P []›
  and p-axiom: ‹(⋀A P. A ∈ axioms ⟹ is-proof P
⟹ P P ⟹ P (P @ [A]))›
  and p-rule-R: ‹(⋀P P' E P'' C p D. is-proof P
⟹ P P ⟹ prefix (P' @ [E]) P ⟹ prefix (P'' @ [C]) P
⟹ is-rule-R-app p D C E ⟹ P (P @ [D]))›
 shows ‹P P›
```

We had to prove this induction principle because Díaz defines *is-proof* $\mathcal{P}$ by quantifying over indices rather than inductively. His formalization contains an inductive derivability definition that allows us to show properties of the proved formula—not its proof. We prove *is-proof-induct* by induction on $\mathcal{P}$'s length—similarly to Díaz's proof relating a formula having a proof and being derivable. Proving

$C \cup \{A_{\beta\to\alpha} \cdot c_\beta \neq B_{\beta\to\alpha} \cdot c_\beta\} \vdash F_o$ — by definition of being inconsistent
$C \vdash A_{\beta\to\alpha} \cdot c_\beta = B_{\beta\to\alpha} \cdot c_\beta$ — proof by contradiction inside the proof system
$C \vdash A_{\beta\to\alpha} \cdot x_\beta = B_{\beta\to\alpha} \cdot x_\beta$ — fresh constant c_β might as well have been a fresh variable x_β
$C \vdash \forall x_\beta.\ A_{\beta\to\alpha} \cdot x_\beta = B_{\beta\to\alpha} \cdot x_\beta$ — by generalization
$C \vdash A_{\beta\to\alpha} = B_{\beta\to\alpha}$ — by extensionality
$C \vdash F_o$ — because C contains $A_{\beta\to\alpha} \neq B_{\beta\to\alpha}$

Figure 5. Derivations proving C inconsistent.

is-proof-const-subst relies on auxiliary lemmas saying (1) constant substitution instances of axioms are axioms (under present freshness assumptions), (2) if replacement can be done at a position in a wff, the corresponding "constant substituted replacement" can be done in the "constant substituted wff" giving the "constant substituted result", and (3) if rule R can be applied, so can the constant substituted version (under present freshness assumptions).

Lemma *is-proof-const-subst* allows applying constant substitution to proofs from axioms, but not to proofs from sets of hypothesis (see section 3.1), because Díaz's *is-proof* predicate captures only the former and not the latter. Notice from section 3.1 that **Rule R** differs from **Rule R'** as **Rule R'** requires inferences to avoid capturing variables free in the hypotheses (**Rule R** does not involve inferences from hypotheses). Thus, for substitutions on hypotheses, we must prove that constant substitutions avoid such capture. Díaz's defines the predicate *capture-exposed-vars-at p C* $\mathcal{H}$ denoting the set of free variables in $\mathcal{H}$ that can be captured at position p in A. We show in two variants that constant substitutions do not change this set under fair available assumptions:

```
lemma capture-exposed-vars-at-const-subst1:
 assumes ‹p ∈ positions C› and ‹C' = S_c (c, β) x C›
 shows ‹capture-exposed-vars-at p C H =
capture-exposed-vars-at p C' H›
```

```
lemma capture-exposed-vars-at-const-subst2:
 assumes ‹p ∈ positions C› and ‹C' = S_c (c, β) x C›
  and ‹E' = S_c (c, β) x E› and ‹(x, β) ∉ vars C ∪ vars E›
 shows ‹capture-exposed-vars-at p C E =
capture-exposed-vars-at p C' E'›
```

We also make an induction principle for such proofs with a similar argument as before. Thus, applying constant substitution to proofs from hypotheses is valid:

```
lemma is-hyp-proof-const-subst:
 assumes ‹is-hyp-proof H P_1 P_2› and ‹is-hyps H›
  and ‹c ∉ logical-names› and ‹(x, β) ∉ vars_p P_2›
  and ‹c ∉ P.params H›
 shows ‹is-hyp-proof H (S_cp (c, β) x P_1) (S_cp (c, β) x P_2)›
```

8 Completeness

From the above results we show strong completeness:

```
theorem strong-completeness:
 assumes mod: ‹⋀M. is-general-model M ⟹ is-frugal M
⟹ ∀ B ∈ G. M ⊨ B ⟹ M ⊨ A› and A: ‹is-sentence A›
  and G: ‹∀ B ∈ G. is-sentence B› ‹P.enough-new G›
 shows ‹∃ H ⊆ G. H ⊢ A›
```

If sentence A holds in all general, frugal models that satisfy all members of the set of sentences $\mathcal{G}$ (that leave enough parameters new), then we can derive A from a subset $\mathcal{H}$ of $\mathcal{G}$. Since the proof system only allows derivations from sets of sentences, then $\mathcal{H}$ must be such a set.

The result follows from model existence in the standard way: assume towards a contradiction that we cannot derive A from any subset $\mathcal{G}$. Then $\sim^Q A$ is consistent with every subset, hence with $\mathcal{G}$ itself. Thus $\{\sim^Q A\} \cup \mathcal{G}$ is in our consistency property and there is a general, frugal model for it. This contradicts A's validity, concluding the proof. Note that this last part of the proof is where we need the model requirement of not satisfying both a wff and its negation.

Weak completeness with no hypothesis set is a corollary:

```
corollary completeness:
 assumes ‹⋀M. is-general-model M ⟹ is-frugal M
  ⟹ M ⊨ A› ‹is-sentence A›
 shows ‹⊢ A›
```

9 Conclusion

We have studied higher-order logic, which is implemented in several proof assistants and is thus a critical component of modern formal methods. We defined consistency properties for Andrews' Q_0 higher-order logic and used the abstract consistency framework to obtain a model existence theorem for such consistent properties. We used this to obtain a model existence theorem for Q_0's proof system. From this theorem, we proved completeness of Q_0. Immediate extensions of this work include modifying the formalized language of Q_0 to allow for uncountably many parameters, proving completeness for another proof system, and proving the weak compactness theorem (Andrews' 5503). The first of these would make our frugality results carry more weight, the second would showcase the flexibility of the abstract model existence result, and the third would simply formalize more of Andrews' textbook. Another direction is to add more features, approaching the logics of the HOL provers.

From the outset we knew it was possible to go far in formalizing proof assistants' meta-theory as there were already

several soundness proofs for higher-order logic. We proved completeness in this paper. Our success builds on top of and reiterates the thoroughness of Andrews' book and the flexibility of Smullyan's and Fitting's consistency properties. It also demonstrates the robustness of Díaz's formalization. We hope that our work adds to the strengthening of higher-order logics' and type theories' status as trusted companions in the exploration of computer science and mathematics.

AI disclosure. No generative AI was used to produce the proofs or text for this work.

Acknowledgments

This work was partially supported by a Novo Nordisk Fonden Start Package grant (NNF20OC0063462).

References

[1] Oskar Abrahamsson and Magnus O. Myreen. 2023. Fast, Verified Computation for Candle. In *14th International Conference on Interactive Theorem Proving, ITP 2023, July 31 to August 4, 2023, Białystok, Poland (LIPIcs, Vol. 268)*, Adam Naumowicz and René Thiemann (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 4:1–4:17. doi:10.4230/LIPICS.ITP.2023.4

[2] Oskar Abrahamsson, Magnus O. Myreen, Ramana Kumar, and Thomas Sewell. 2022. Candle: A Verified Implementation of HOL Light. In *13th International Conference on Interactive Theorem Proving, ITP 2022, August 7-10, 2022, Haifa, Israel (LIPIcs, Vol. 237)*, June Andronick and Leonardo de Moura (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 3:1–3:17. doi:10.4230/LIPIcs.ITP.2022.3

[3] Oskar Abrahamsson, Magnus O. Myreen, Ramana Kumar, and Thomas Sewell. 2025. Candle: A Verified Implementation of HOL Light (Extended Version). *J. Autom. Reason.* 69, 4 (2025), 32. doi:10.1007/S10817-025-09743-8

[4] Peter B. Andrews. 1963. A reduction of the axioms for the theory of propositional types. *Fundamenta Mathematicae* 52 (1963), 345–350.

[5] Peter B. Andrews. 1971. Resolution in type theory. *Journal of Symbolic Logic* 36, 3 (1971), 414–432. doi:10.2307/2269949

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

[7] Peter B. Andrews. 2013. *An introduction to mathematical logic and type theory: to truth through proof* (2nd ed.). Applied Logic Series, Vol. 27. Springer Science & Business Media.

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

[9] Bruno Barras. 1996. Verification of the Interface of a Small Proof System in Coq. In *Types for Proofs and Programs, International Workshop TYPES'96, Aussois, France, December 15-19, 1996, Selected Papers (Lecture Notes in Computer Science, Vol. 1512)*, Eduardo Giménez and Christine Paulin-Mohring (Eds.). Springer, 28–45. doi:10.1007/BFB0097785

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

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

[12] Christoph Benzmüller and Peter B. Andrews. 2025. Church's Type Theory. In *The Stanford Encyclopedia of Philosophy* (Summer 2025 ed.), Edward N. Zalta and Uri Nodelman (Eds.). Metaphysics Research Lab, Stanford University.

[13] Christoph Benzmüller, Chad E. Brown, and Michael Kohlhase. 2004. Higher-order semantics and extensionality. *Journal of Symbolic Logic* 69, 4 (2004), 1027–1088. doi:10.2178/jsl/1102022211

[14] Christoph Benzmüller and Daniel Kirchner. 2026. A deep embedding of HOL in HOL: soundness, completeness, consistency. *Archive of Formal Proofs* (August 2026). https://isa-afp.org/entries/HOL_in_HOL_Deep.html, Formal proof development.

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

[16] Thibault Dardinier, Michael Sammler, Gaurav Parthasarathy, Alexander J. Summers, and Peter Müller. 2025. Formal Foundations for Translational Separation Logic Verifiers. *Proc. ACM Program. Lang.* 9, POPL, Article 20 (Jan. 2025), 31 pages. doi:10.1145/3704856

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

[18] Melvin Fitting. 1996. *First-Order Logic and Automated Theorem Proving, Second Edition.* Springer. doi:10.1007/978-1-4612-2360-3

[19] Asta Halkjær From and Anders Schlichtkrull. 2025. Abstract, Compositional Consistency: Isabelle/HOL Locales for Completeness à la Fitting. In *16th International Conference on Interactive Theorem Proving, ITP 2025, Reykjavik, Iceland, September 28 - October 1, 2025 (LIPIcs)*, Yannick Forster and Chantal Keller (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 8:1–8:20. doi:10.4230/LIPICS.ITP.2025.8

[20] Mike Gordon. 2000. From LCF to HOL: a short history. In *Proof, Language, and Interaction, Essays in Honour of Robin Milner*, Gordon D. Plotkin, Colin Stirling, and Mads Tofte (Eds.). The MIT Press, 169–186.

[21] John Harrison. 2006. Towards Self-verification of HOL Light. In *Automated Reasoning, Third International Joint Conference, IJCAR 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings (Lecture Notes in Computer Science, Vol. 4130)*, Ulrich Furbach and Natarajan Shankar (Eds.). Springer, 177–191. doi:10.1007/11814771_17

[22] John Harrison. 2009. HOL Light: An Overview. In *Theorem Proving in Higher Order Logics, 22nd International Conference, TPHOLs 2009, Munich, Germany, August 17-20, 2009. Proceedings (Lecture Notes in Computer Science, Vol. 5674)*, Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel (Eds.). Springer, 60–66. doi:10.1007/978-3-642-03359-9_4

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

[24] Leon Henkin. 1963. A Theory of Propositional Types. *Fundamenta Mathematicae* 52 (1963), 323–344.

[25] Leon Henkin. 1996. The discovery of my completeness proofs. *Bull. Symb. Log.* 2, 2 (1996), 127–158. doi:10.2307/421107

[26] Mark Koch and Dominik Kirst. 2022. Undecidability, incompleteness, and completeness of second-order logic in Coq. In *CPP '22: 11th ACM SIGPLAN International Conference on Certified Programs and Proofs, Philadelphia, PA, USA, January 17 - 18, 2022*, Andrei Popescu and Steve Zdancewic (Eds.). ACM, 274–290. doi:10.1145/3497775.3503684

[27] Ramana Kumar, Rob Arthan, Magnus O. Myreen, and Scott Owens. 2014. HOL with Definitions: Semantics, Soundness, and a Verified Implementation. In *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 (Lecture Notes in Computer Science, Vol. 8558)*, Gerwin Klein and Ruben Gamboa (Eds.). Springer, 308–324. doi:10.1007/978-3-319-08970-6_20

[28] Ramana Kumar, Rob Arthan, Magnus O. Myreen, and Scott Owens. 2016. Self-Formalisation of Higher-Order Logic - Semantics, Soundness, and a Verified Implementation. *J. Autom. Reason.* 56, 3 (2016), 221–259. doi:10.1007/s10817-015-9357-x

[29] Ondrej Kuncar and Andrei Popescu. 2019. From Types to Sets by Local Type Definition in Higher-Order Logic. *J. Autom. Reason.* 62, 2 (2019), 237–260. doi:10.1007/S10817-018-9464-6

[30] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. 2002. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic.* Lecture Notes in Computer Science, Vol. 2283. Springer. doi:10.1007/3-540-45949-9

[31] Tobias Nipkow and Simon Roßkopf. 2021. Isabelle's Metalogic: Formalization and Proof Checker. In *Automated Deduction - CADE 28 - 28th International Conference on Automated Deduction, Virtual Event, July 12-15, 2021, Proceedings (Lecture Notes in Computer Science, Vol. 12699)*, André Platzer and Geoff Sutcliffe (Eds.). Springer, 93–110. doi:10.1007/978-3-030-79876-5_6

[32] Tobias Nipkow and Simon Roßkopf. 2021. Isabelle's Metalogic: Formalization and Proof Checker. *Archive of Formal Proofs* (April 2021). https://isa-afp.org/entries/Metalogic_ProofChecker.html, Formal proof development.

[33] Lawrence C. Paulson. 2019. 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.

[34] Lawrence C. Paulson, Tobias Nipkow, and Makarius Wenzel. 2019. From LCF to Isabelle/HOL. *Formal Aspects Comput.* 31, 6 (2019), 675–698. doi:10.1007/S00165-019-00492-1

[35] Andrei Popescu. 2025. Completing Gordon's Higher-Order Logic. In *40th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2025, Singapore, June 23-26, 2025*. IEEE, 1–15. doi:10.1109/LICS65433.2025.00008

[36] Simon Roßkopf and Tobias Nipkow. 2023. A Formalization and Proof Checker for Isabelle's Metalogic. *J. Autom. Reason.* 67, 1 (2023), 1. doi:10.1007/s10817-022-09648-w

[37] Bertrand Russell. 1908. Mathematical Logic as Based on the Theory of Types. *American Journal of Mathematics* 30, 3 (1908), 222–262. http://www.jstor.org/stable/2369948

[38] Anders Schlichtkrull. 2023. 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.

[39] Konrad Slind and Michael Norrish. 2008. A Brief Overview of HOL4. In *Theorem Proving in Higher Order Logics, 21st International Conference, TPHOLs 2008, Montreal, Canada, August 18-21, 2008. Proceedings (Lecture Notes in Computer Science, Vol. 5170)*, Otmane Aït Mohamed, César A. Muñoz, and Sofiène Tahar (Eds.). Springer, 28–32. doi:10.1007/978-3-540-71067-7_6

[40] Raymond M. Smullyan. 1963. A Unifying Principal in Quantification Theory. *Proceedings of the National Academy of Sciences* 49, 6 (1963), 828–832. doi:10.1073/pnas.49.6.828

[41] Matthieu Sozeau, Simon Boulier, Yannick Forster, Nicolas Tabareau, and Théo Winterhalter. 2020. Coq Coq correct! verification of type checking and erasure for Coq, in Coq. *Proc. ACM Program. Lang.* 4, POPL (2020), 8:1–8:28. doi:10.1145/3371076

[42] Alexander Steen. 14 September 2018. *Extensional Paramodulation for Higher-Order Logic and its Effective Implementation Leo-III*. AKA-Verlag. hdl:https://orbilu.uni.lu/handle/10993/40251 http://www.aka-verlag.de/index.php?option=com_virtuemart&view=productdetails&virtuemart_product_id=701&virtuemart_category_id=4&Itemid=482&lang=en Dissertation, Freie Universität Berlin.

[43] Alexander Steen and Christoph Benzmüller. 2021. On Reductions of Hintikka Sets for Higher-Order Logic. arXiv:2004.07506 [cs.LO] https://arxiv.org/abs/2004.07506

[44] Jan van Brügge, James McKinna, Andrei Popescu, and Dmitriy Traytel. 2025. Barendregt Convenes with Knaster and Tarski: Strong Rule Induction for Syntax with Bindings. *Proc. ACM Program. Lang.* 9, POPL, Article 57 (Jan. 2025), 32 pages. doi:10.1145/3704893

[45] Jouko Väänänen. 2024. Second-order and Higher-order Logic. In *The Stanford Encyclopedia of Philosophy* (Winter 2024 ed.), Edward N. Zalta and Uri Nodelman (Eds.). Metaphysics Research Lab, Stanford University.

[46] Alfred North Whitehead and Bertrand Russell. 1913. *Principia mathematica*. Cambridge University Press, Cambridge.