Completing Gordon's Higher-Order Logic

Andrei Popescu
School of Computer Science, University of Sheffield, UK. Email: a.popescu@sheffield.ac.uk

***Abstract*—Mike Gordon's Higher-Order Logic (HOL) is one of the most important logical foundations for interactive theorem proving. The standard semantics of HOL, due to Andrew Pitts, employs a downward closed universe of sets, and interprets HOL's Hilbert choice operator via a global choice function on the universe. In this paper we fill a gap in the meta-theory of HOL: We provide a natural Henkin-style notion of general model corresponding to the standard models, and discover an enrichment of HOL deduction that we prove to be sound and complete w.r.t. these general models.**

I. Introduction

Interactive theorem provers, also known as proof assistants, are being increasingly deployed to verify meta-theoretic properties of operating systems, programming languages, compilers and hardware architectures, and to formalize mathematical theories. The list of major verification successes includes the verified compilers CompCert [61] and CakeML [55] and the verified operating system kernel seL4 [52] in computer science, and formal proofs of the the Four Color [33], Kepler [42] and Odd Order [34] theorems in mathematics. Solid and well-understood logical foundations for such provers are of paramount importance for the reliability of these results.

We can distinguish two kinds of logical foundations that dominate today's interactive theorem proving landscape. On the one hand, provers based on dependent type theories such as Agda [21], Coq [12], Lean [68] and Matita [6] rely on sophisticated type systems to manage both the data of interest and the proofs themselves. Developments in these provers often emphasize constructiveness and the computational content of the proofs. On the other hand, provers such as HOL4 [36], [83], HOL Light [43], Isabelle/HOL [69], ProofPower-HOL [5] and HOL Zero [2], all rely on Higher-Order Logic (HOL), a foundation with a different flavor: It is classical rather than intuitionistic, employs simple types and custom-defined types with rank-1 polymorphism (hence a less sophisticated type system), relies essentially on the Hilbert choice operator and the corresponding choice axiom, and (consequently) relies much more heavily than dependent type theory on *semantic intuitions coming from set-based models*—indeed, HOL is often colloquially referred to as "typed set theory".

In this paper, we are concerned with the latter foundation. More precisely, we focus on what the theorem proving community refers to as Higher-Order Logic (HOL), a logic proposed by Mike Gordon at the end of the eighties inspired by practical verification needs [35], [36]. At its core, HOL is a rank-1 polymorphic extension of Church's Simple Type Theory [23], featuring built-in axioms of equality, Hilbert choice and infinity and some minimalistic deduction rules that can bootstrap the entire apparatus of classical logic connectives and quantifiers (Section II). In addition to this initial layer, users can further populate the HOL theories with constant and type declarations and definitions, as well as non-definitional axioms that underspecify constants or types [4], postulate the existence of large types [44], [54], [70], [71], model system behavior [20], etc. Even leaving non-definitional axioms aside, dialects of HOL differ in the exact forms of the definitional axioms. For example, while traditional HOL practice prescribes upfront proofs of non-emptiness for newly introduced types, Isabelle/HOL introduces type definitions in conditional form, conditioned by the nonemptyness of the defining predicate; moreover, overloaded constant and type definitions are permitted, and Haskell-style type classes are introduced by extending the theory with uninterpreted constants and axioms about them [90], [40].

This variety of types of axioms that are or have been featured in HOL-based systems all have in common the concern for logical consistency, and many of them are instances or mild extensions of the definitional approach to theorem proving [28], [65], [37], which insists on the use of definitions or otherwise safe-by-construction axioms as opposed to unrestricted axioms. For example, HOL's traditional type definitions are not equational definitions but axioms asserting that the newly introduced type is isomorphic to a subset of the underlying type (akin to the Axiom of Separation from set theory); they nevertheless can be regarded as "definitionally safe". Indeed, the concern for consistency and stronger properties such as conservativity have been a major theme in HOL from the very beginning (see [38], [4], [58] for some history). Also from early on, HOL has benefited from a powerful mechanism of ensuring consistency and conservativity: a natural set-theoretic semantics worked out by Andrew Pitts [73], which employs a *downward closed* universe of sets, and interprets type constructors as operators on this universe. The Pitts semantics captures intuitively the models that HOL users are likely to have in mind when introducing concepts and performing proofs, while also hosting interpretations for various safe axioms, including constant and type definitions and underspecifications—and facilitate simple semantic arguments for consistency using the soundness of HOL deduction in these models.

Here, we study not consistency, but a related problem that represents a gap in the current meta-theory of HOL: the completeness of deduction with respect to semantic models. Our main result will be the discovery of an enrichment of HOL deduction that achieves completeness w.r.t. to the Henkin-style generalization of the standard models of HOL, the Pitts models. We call this generalization *Pitts-Henkin models*

(Section III), and we provide a proof of completeness of HOL deduction w.r.t. these models (Section IV) along the route pioneered by Henkin: showing how a (syntactically) consistent theory extends to a maximally consistent one satisfying additional well-behavedness conditions (that later came to be) called *Hintikka conditions* [84], [9], which in turn admit a *syntactic model* consisting of sets of deductive equivalence classes of terms. Our saga in this paper will be fitting HOL with its Pitts-Henkin models into this scheme. It will turn out that this requires (1) the reduction of completeness to ground (non-polymorphic) conclusion completeness, (2) a downward closure of the syntactic model, and an infrastructure on the closure that allows for interpretation "by proxy" to the original syntactic model, (4) additional Hintikka conditions for *witness types* and representation terms that make the downward closure possible, and ultimately (5) an enrichment of the HOL deduction with a rule for *uniform local type definitions* and an axiom for trivial choice in order to support extensions satisfying the Hintikka witness conditions. Our result joins a rich body of completeness results for logics and calculi that adapt the methods of Henkin and Birkhoff (Section V).

More details about our constructions, as well as detailed proofs, can be found in a technical report available online [76].

II. HOL Preliminaries

A. Syntax

We fix an infinite set TVar, of *type-variables*, ranged by α, β, and an infinite set Var, of *(term) variables*, ranged by x, y, z or by other symbols, as suitable for the context. A *type structure* is a pair $(\mathsf{K}, \mathsf{arOf})$ where K, ranged by κ, is the set of *type constructors*, among which bool, ind and $\Rightarrow$ (representing the type of booleans, an infinite type of individuals, and the function type constructor) and $\mathsf{arOf} : \mathsf{K} \to \mathbb{N}$ associates arities to the type constructors, such that $\mathsf{arOf}(\mathsf{bool}) = \mathsf{arOf}(\mathsf{ind}) = 0$ and $\mathsf{arOf}(\Rightarrow) = 2$. We will call bool, ind and $\Rightarrow$ the *built-in type constructors*, and let $\mathsf{K}_0 = \{\mathsf{bool}, \mathsf{ind}, \Rightarrow\}$.

The *types* of $(\mathsf{K}, \mathsf{arOf})$, ranged by σ, τ, are defined by the following grammar: $\sigma ::= \alpha \mid (\sigma_1, \ldots, \sigma_{\mathsf{arOf}(\kappa)})\,\kappa$. Thus, a type is either a type-variable or an n-ary type constructor κ postfix-applied to a number of types corresponding to its arity. $\mathsf{Type}_{(\mathsf{K},\mathsf{arOf})}$ denotes the set of types of $(\mathsf{K}, \mathsf{arOf})$.

A *signature* is a tuple $\Sigma = (\mathsf{K}, \mathsf{arOf}, \mathsf{C}, \mathsf{tpOf})$, where: $(\mathsf{K}, \mathsf{arOf})$ is a type structure; C, ranged over by c, is a set of symbols called *constants*, containing two special symbols, $=$ and ε (aimed at representing equality and Hilbert choice of some element from a type, respectively); and $\mathsf{tpOf} : \mathsf{C} \to \mathsf{Type}_{(\mathsf{K},\mathsf{arOf})}$ is a function associating a type to every constant, such that $\mathsf{tpOf}(=) = \alpha \Rightarrow \alpha \Rightarrow \mathsf{bool}$ and $\mathsf{tpOf}(\varepsilon) = (\alpha \Rightarrow \mathsf{bool}) \Rightarrow \alpha$ (where α is a type-variable). We will call $=$ and ε the *built-in constants*, and let $\mathsf{C}_0 = \{=, \varepsilon\}$. We usually write Type_Σ instead of $\mathsf{Type}_{(\mathsf{K},\mathsf{arOf})}$ for the set of types associated to the type structure part $(\mathsf{K}, \mathsf{arOf})$ of a signature Σ.

Given a signature $\Sigma = (\mathsf{K}, \mathsf{arOf}, \mathsf{C}, \mathsf{tpOf})$ and an item u, we write $u \in \Sigma$ to mean that $u \in \mathsf{K}$ or $u \in \mathsf{C}$. Given signatures $\Sigma = (\mathsf{K}, \mathsf{arOf}, \mathsf{C}, \mathsf{tpOf})$ and $\Sigma' = (\mathsf{K}', \mathsf{arOf}', \mathsf{C}', \mathsf{tpOf}')$, we say Σ' *extends* Σ when $\mathsf{K} \subseteq \mathsf{K}'$, $\mathsf{C} \subseteq \mathsf{C}'$ and the functions arOf' and tpOf' are extensions of arOf and tpOf. We write $\Sigma_{\mathsf{init}} = (\mathsf{K}_0, \mathsf{arOf}, \mathsf{C}_0, \mathsf{tpOf})$ for the *initial signature*, containing only built-in type constructors and constants.

In this section, we fix a signature $\Sigma = (\mathsf{K}, \mathsf{arOf}, \mathsf{C}, \mathsf{tpOf})$, and all the concepts we introduce will be relative to Σ; so we write, e.g., Type and Term instead of Type_Σ and Term_Σ.

Given a type σ, $\mathsf{TV}(\sigma)$ denotes the set of its type-variables. σ is called *ground*, or *monomorphic* if $\mathsf{TV}(\sigma) = \emptyset$, and *polymorphic* otherwise. Let GType be the set of ground types. A *type-substitution* is a function $\rho : \mathsf{TVar} \to \mathsf{Type}$ with finite support, i.e., such that it changes only finitely many type-variables. The application of ρ to a type σ, written $\sigma[\rho]$, replaces in σ each type-variable α with $\rho(\alpha)$. We say that σ is an *instance of* τ *via* ρ, written $\sigma \leq_\rho \tau$, when $\tau[\rho] = \sigma$; and that it is an *instance of* τ when there exists ρ such that $\sigma \leq_\rho \tau$. A *constant instance* is a pair of a constant and a type, written c_σ, such that $\sigma \leq \mathsf{tpOf}(c)$. We let CInst denote the set of constant instances. A *typed variable* is a pair of a variable x and a type σ, written x_σ. We let VarT denote the set of typed variables.

The signature's *terms*, ranged over by s, t, are defined by the grammar $t ::= x_\sigma \mid c_\sigma \mid t_1\, t_2 \mid \lambda x_\sigma.\, t$. Thus, a term is either a variable, or a constant instance, or an application, or an abstraction. As usual, we identify terms modulo alpha-equivalence. Typing relates terms and types, is written $t : \sigma$ and is defined inductively in the expected way:

$$\frac{x_\sigma \in \mathsf{VarT}}{x_\sigma : \sigma} \qquad \frac{c_\sigma \in \mathsf{CInst}}{c_\sigma : \sigma} \qquad \frac{t_1 : \sigma \Rightarrow \tau \quad t_2 : \sigma}{t_1\, t_2 : \tau} \qquad \frac{t : \tau}{\lambda x_\sigma.\, t : \sigma \Rightarrow \tau}$$

A term t is *typable* if there exists a (necessarily unique) type σ such that $t : \sigma$. We let Term denote the set of all typable terms, and Term_σ the set of terms of type σ. In what follows, we implicitly assume that the terms we consider are typable.

$\mathsf{FTV}(t)$ denotes the set of t's free typed variables. For example, assuming $x \neq y$, we have $\mathsf{FTV}(\lambda x_\sigma.\, x_\sigma = y_\sigma) = \{y_\sigma\}$. A term t is *closed* if it has no free typed variables: $\mathsf{FTV}(t) = \emptyset$. $\mathsf{TV}(t)$ is the set of type-variables occurring in (any type that occurs in) t. A term is *ground*, or *monomorphic*, if $\mathsf{TV}(t) = \emptyset$, and *polymorphic* otherwise. GTerm denotes the set of (typable) ground terms, and GCTerm that of (typable) ground closed terms; moreover, GTerm_σ and GCTerm_σ denote the sets of ground and ground closed terms of type σ. We apply a type-substitution ρ to a term t, written $t[\rho]$, by applying it to the types of all typed variables and constant instances in t.

A *term-substitution* is a function $\delta : \mathsf{VarT} \to \mathsf{Term}$ of finite support (i.e., changing only finitely many typed variables) such that $\delta(x_\sigma) : \sigma$ for all $x_\sigma \in \mathsf{VarT}$. The application of δ to a term t, written $t[\delta]$, proceeds by replacing all free variables x_σ of t with $\delta(x_\sigma)$ in a capture-avoiding way. t/x is term-substitution that sends x_σ to t and any other typed variable to itself.

A *formula* is a term of type bool. Fmla, ranged by φ, ψ, is the set of formulas; and GFmla the set of ground formulas. The formula connectives (e.g., $\wedge$ and $\longrightarrow$) and quantifiers ($\forall$ and $\exists$) are defined from HOL primitives. (App. A1 from [76] gives details.)

We often omit the types of variables and constants if they can be inferred—e.g, writing $\lambda x_\sigma.\, x$ instead of $\lambda x_\sigma.\, x_\sigma$. We use

Equality	$x_\alpha = x$
	$x_\alpha = y_\alpha \longrightarrow p_{\alpha\Rightarrow\mathsf{bool}}\ x \longrightarrow p\ y$
Infinity	$\exists s_{\mathsf{ind}\Rightarrow\mathsf{ind}}, z_{\mathsf{ind}}.\ (\forall x_{\mathsf{ind}}, y_{\mathsf{ind}}.\ s\ x = s\ y \longrightarrow x = y)$ $\wedge\ (\forall x_{\mathsf{ind}}.\ s\ x \neq z)$
Choice	$p_{\alpha\Rightarrow\mathsf{bool}}\ x_\alpha \longrightarrow p\ (\varepsilon\ p)$

Fig. 1. The axioms of HOL, forming the set Ax

infix for equality, and standard functional notations: 1_σ abbreviates $\lambda x_\sigma.\ x$, and $t \circ s$ abbreviates $\lambda x_\sigma.\ t(s\ x)$ when $s : \sigma \Rightarrow \tau$.

Since HOL was designed to formalize mathematical practice, we have a correspondence between standard concepts used in the meta-language of this paper and HOL syntax—reflected in some notation overlap, e.g., $=$ for both meta-level and HOL equality, $\circ$ for both meta-level and HOL composition, and : for both meta-level function (co)domain indication and HOL typing. To help with disambiguation, we use single arrow for meta-level function spaces, as in $A \to B$ where A and B are sets, but double arrow for HOL function types, as in $\sigma \Rightarrow \tau$; and use parenthesized function application notation at the meta-level, as in $f(a)$ when $f : A \to B$ and $a \in A$, but non-parenthesized one in HOL, as in $t\ s$ when $t : \sigma \Rightarrow \tau$ and $s : \sigma$.

A *predicate* is a function from a set A to the two-element set $\{\top, \bot\}$ (where $\top$ represents "true" and $\bot$ represents "false"). The *extent* (or *extension*) of a predicate $p : A \to \{\top, \bot\}$ is the subset B of A on which p is true, i.e., $B = \{a \in A \mid p(a) = \top\}$. We also say that B was defined by *comprehension* from p, and that p is the *indicator* of B in A. The HOL counterparts of predicates are terms of type $\sigma \Rightarrow \mathsf{bool}$ for given types σ. HOL models interpret HOL predicates as meta-level predicates.

B. Axioms and deduction

At its core, HOL contains axioms for equality, infinity and choice, forming the set Ax shown in Fig. 1. Thus, we have the usual introduction (reflexivity) and elimination rules for equality. The infinity axiom ensures that ind is an infinite type, by stating that there exists an injective but non-surjective function between the type ind and itself. Finally, the choice axiom regulates the behavior of the Hilbert choice operator ε, indicating that if the argument predicate holds true for some item (i.e., is not vacuously false), then it will hold for the element returned by ε—in other words, this operator chooses an element satisfying its argument predicate if at all possible.

A *theory* over Σ is a set of Σ-formulas. A *context* Γ is a finite set of formulas. A type-variable α is *fresh* for Γ when it does not appear in any formula from Γ. A typed variable x_σ is fresh for Γ when x_σ does not appear free in any formula from Γ.

The HOL *deduction* is a ternary relation $\vdash$ between theories T, contexts Γ and formulas φ, written $T; \Gamma \vdash \varphi$, defined inductively by the rules shown in Fig. 2—referring to the use of formulas from the underlying theory or context, and to the application of type- or term- substitution, β-reduction, extensionality, and implication introduction and elimination. The familiar rules for the other connectives and quantifiers (such as $\neg$ and $\forall$) are derived. We write $T \vdash \varphi$ instead of $T; \emptyset \vdash \varphi$.

C. Monomorphic HOL (MHOL)

The MHOL syntax and deduction are obtained by modifying those of HOL: types no longer contain type-variables; the equality and choice axioms are modified by replacing the type-variable α with an arbitrary type σ (so they become axiom schemas); the type instantiation rule (T-INST) is removed.

Thus, what we call MHOL is a small extension and variation of Church's Simple Type Theory [23]. The extension consists of allowing non-built-in type constructors and constants, and including the axiom of infinity as primitive. The variation consists of taking equality as primitive instead of $\neg$, $\wedge$ and $\forall$, and of using a Gentzen rather than a Hilbert system for deduction; such alternative presentations are equivalent [10], [86].

D. Type definitions

Def 1: Given types τ and σ and a closed term $t : \sigma \Rightarrow \mathsf{bool}$, we let $\tau \cong (t, \sigma)$ be the formula $\exists rep_{\tau\Rightarrow\sigma}.\ \mathsf{tdef}_{\tau,\sigma,t,rep}$ where $\mathsf{tdef}_{\tau,\sigma,t,rep}$ is $(\forall y_\tau, y'_\tau.\ rep\ y = rep\ y' \longrightarrow y = y') \wedge (\forall x_\sigma.\ t\ x \leftrightarrow (\exists y_\tau.\ x = rep\ y))$. We call $\tau \cong (t, \sigma)$ a *type definition*, provided τ has the form $(\alpha_1, \ldots, \alpha_m)\kappa$ such that κ is a non-built-in type constructor, the α_i's are all distinct type-variables and $\mathsf{TV}(t) \subseteq \{\alpha_1, \ldots, \alpha_m\}$.

In a HOL development, a type definition is performed while extending the current signature with a new type constructor κ. So the type definition expresses that the (new) type $\tau = (\alpha_1, \ldots, \alpha_m)\kappa$ is in bijection with the extent of t in σ. Since types in HOL must be nonempty, the definition is only accepted if the user provides a proof that $\exists x_\sigma.\ t\ x$ holds.

E. Standard semantics: Pitts models

Using our notations, we follow Pitts's account [73], which is part of the standard documentation of the HOL system [74].

Def 2: A *Pitts type-model* for a type structure $(\mathsf{K}, \mathsf{arOf})$ is a tuple $\mathcal{M} = (\mathcal{U}, M_{\mathsf{bool}}, M_{\mathsf{ind}}, (M_\kappa)_{\kappa\in\mathsf{K}\setminus\mathsf{K}_0})$ where

- $\mathcal{U}$, the *universe* of $\mathcal{M}$, is a set of nonempty sets which is:
  - closed under function space, i.e., $(A \to B) \in \mathcal{U}$ for all $A, B \in \mathcal{U}$,
  - downward closed (closed under inclusion), i.e., $B \in \mathcal{U}$ for all A, B such that $B \subseteq A \in \mathcal{U}$;
- $M_{\mathsf{bool}} \in \mathcal{U}$ is a two-element set and $M_{\mathsf{ind}} \in \mathcal{U}$ is an infinite set;
- $M_\kappa : \mathcal{U}^{\mathsf{arOf}(\kappa)} \to \mathcal{U}$ (i.e., M_κ is an $\mathsf{arOf}(\kappa)$-ary operator on the universe) for all $\kappa \in \mathsf{K} \setminus \mathsf{K}_0$.

Recall that K_0 is the set of built-in type constructors $\{\mathsf{bool},\mathsf{ind},\Rightarrow\}$. So a Pitts type-model provides custom semantic interpretations for the non-built-in type constructors, while for the built-in ones it has the expected requirements: bool must be interpreted as a two-element set, ind as an infinite set, and implicitly $\Rightarrow$ as the (full) function space operator—under which the universe $\mathcal{U}$ is required to be closed. A main distinguishing feature of Pitts type-models is that their universe is also required to be downward closed—which, as we will discuss shortly, offers semantics (and semantic intuition) for type definitions.

$$\frac{}{T;\Gamma\vdash\varphi}\ [\varphi\in\mathsf{Ax}\cup T]\ (\textsc{Fact}) \qquad \frac{}{T;\Gamma\vdash\varphi}\ [\varphi\in\Gamma]\ (\textsc{Assum}) \qquad \frac{T;\Gamma\vdash\varphi}{T;\Gamma\vdash\varphi[\sigma/\alpha]}\ [\alpha\text{ fresh for }\Gamma]\ (\textsc{T-Inst}) \qquad \frac{T;\Gamma\vdash\varphi}{T;\Gamma\vdash\varphi[t/x_\sigma]}\ [x_\sigma\text{ fresh for }\Gamma]\ (\textsc{Inst})$$

$$\frac{}{T;\Gamma\vdash(\lambda x_\sigma.\,t)\ s=t[s/x_\sigma]}\ (\textsc{Beta}) \qquad \frac{T;\Gamma\vdash y_{\sigma\Rightarrow\tau}\ x_\sigma=z_{\sigma\Rightarrow\tau}\ x_\sigma}{T;\Gamma\vdash y_{\sigma\Rightarrow\tau}=z_{\sigma\Rightarrow\tau}}\ [x_\sigma\text{ fresh for }\Gamma]\ (\textsc{Ext}) \qquad \frac{T;\Gamma\cup\{\varphi\}\vdash\psi}{T;\Gamma\vdash\varphi\longrightarrow\psi}\ (\textsc{ImpI}) \qquad \frac{T;\Gamma\vdash\varphi\longrightarrow\psi\quad T;\Gamma\vdash\varphi}{T;\Gamma\vdash\psi}\ (\textsc{MP})$$

Fig. 2. Deduction rules for HOL

Given a Pitts type-model $\mathcal{M}=(\mathcal{U},M_{\mathsf{bool}},M_{\mathsf{ind}},(M_\kappa)_{\kappa\in\mathsf{K}\setminus\mathsf{K}_0})$ for $(\mathsf{K},\mathsf{arOf})$, a *type-valuation* in $\mathcal{M}$ is a function $\theta:\mathsf{TVar}\to\mathcal{U}$. We define the *interpretation of types* $\langle\_\rangle^{\mathcal{M}}:\mathsf{Type}_{(\mathsf{K},\mathsf{arOf})}\to(\mathsf{TVar}\to\mathcal{U})\to\mathcal{U}$ recursively, where θ is a type-valuation (omitting the superscript $\mathcal{M}$):

- $\langle\alpha\rangle(\theta)=\theta(\alpha)$; • $\langle\mathsf{bool}\rangle(\theta)=M_{\mathsf{bool}}$; • $\langle\mathsf{ind}\rangle(\theta)=M_{\mathsf{ind}}$;
- $\langle\sigma_1\Rightarrow\sigma_2\rangle(\theta)=(\langle\sigma_1\rangle(\theta)\to\langle\sigma_2\rangle(\theta))$;
- $\langle(\sigma_1,\ldots,\sigma_{\mathsf{arOf}(\kappa)})\kappa\rangle(\theta)=M_\kappa\,(\langle\sigma_1\rangle(\theta),\ldots,\langle\sigma_{\mathsf{arOf}(\kappa)}\rangle(\theta))$ if $\kappa\in\mathsf{K}\setminus\mathsf{K}_0$.

The interpretation actually only depends on the type-variables occurring in the type, in that $\langle\sigma\rangle^{\mathcal{M}}(\theta)=\langle\sigma\rangle^{\mathcal{M}}(\theta')$ if $\theta(\alpha)=\theta'(\alpha)$ for all $\alpha\in\mathsf{TV}(\sigma)$. Therefore, given $\theta:\mathsf{TV}(\sigma)\to\mathcal{U}$, we write $\langle\sigma\rangle^{\mathcal{M}}(\theta)$ for $\langle\sigma\rangle^{\mathcal{M}}(\theta')$, where $\theta':\mathsf{TVar}\to\mathcal{U}$ is some/any extension of θ.

Def 3: Given a signature $\Sigma=(\mathsf{K},\mathsf{arOf},\mathsf{C},\mathsf{tpOf})$, a *Pitts model* for Σ is a tuple $\mathcal{M}=(\mathcal{U},M_{\mathsf{bool}},M_{\mathsf{ind}},(M_\kappa)_{\kappa\in\mathsf{K}\setminus\mathsf{K}_0},M_\top,M_\bot,M_\varepsilon,(M_c)_{c\in\mathsf{C}\setminus\mathsf{C}_0})$ where:

- $(\mathcal{U},M_{\mathsf{bool}},M_{\mathsf{ind}},(M_\kappa)_{\kappa\in\mathsf{K}\setminus\mathsf{K}_0})$ is a Pitts type-model for $(\mathsf{K},\mathsf{arOf})$ whose type interpretation we denote by $\langle\_\rangle$;
- $M_\top$ and $M_\bot$ are the two (distinct) elements of M_{bool};
- M_ε is a family $(M_{\varepsilon,A})_{A\in\mathcal{U}}$ such that $M_{\varepsilon,A}\in A$;
- if $c\in\mathsf{C}\setminus\mathsf{C}_0$ with $\mathsf{tpOf}(c)=\sigma$, then M_c is a family $(M_{c,\theta})_{\theta:\mathsf{TV}(\sigma)\to\mathcal{U}}$ such that $M_{c,\theta}\in\langle\sigma\rangle(\theta)$.

Recall that C_0 is the set of built-in constants $\{=,\varepsilon\}$. So a Pitts model provides custom semantic interpretations for the the non-built-in constants and a global choice function for ε (while implicitly assuming that $=$ will be interpreted as equality). Moreover, although true and false are not primitive but derived symbols in HOL, a Pitts model has semantic counterparts for them, $M_\top$ and $M_\bot$ (and indeed, it can be shown from the definition of interpretation given below that true and false will necessarily be interpreted as $M_\top$ and $M_\bot$).

Given a Pitts model $\mathcal{M}=(\mathcal{U},M_{\mathsf{bool}},M_{\mathsf{ind}},(M_\kappa)_{\kappa\in\mathsf{K}\setminus\mathsf{K}_0},M_\top,M_\bot,M_\varepsilon,(M_c)_{c\in\mathsf{C}\setminus\mathsf{C}_0})$ and letting $\mathcal{N}=(\mathcal{U},M_{\mathsf{bool}},M_{\mathsf{ind}},(M_\kappa)_{\kappa\in\mathsf{K}\setminus\mathsf{K}_0})$ be its underlying Pitts type-model, we will refer to the universe $\mathcal{U}$ of $\mathcal{N}$ as also the universe of $\mathcal{M}$, and to any type-valuation $\theta:\mathsf{TVar}\to\mathcal{U}$ in $\mathcal{N}$ as a type-valuation in $\mathcal{M}$; and also write $\langle\sigma\rangle^{\mathcal{M}}$ instead of $\langle\sigma\rangle^{\mathcal{N}}$.

A *term-valuation* in $\mathcal{M}$ is a function $\xi:\mathsf{VarT}\to\bigcup_{A\in\mathcal{U}}A$. For any type-valuation θ and term-valuation ξ, both in $\mathcal{M}$, we say ξ is *θ-compatible* when $\xi(x_\sigma)\in\langle\sigma\rangle^{\mathcal{M}}(\theta)$ for all $x_\sigma\in\mathsf{VarT}$. We let $\mathsf{Compat}(\mathcal{M},\theta)$ be the set of θ-compatible term-valuations in $\mathcal{M}$. We define the *interpretation of terms* as a function $\langle\_\rangle^{\mathcal{M}}:\mathsf{Term}_\Sigma\to(\sum_{\theta:\mathsf{TVar}\to\mathcal{U}}\mathsf{Compat}(\mathcal{M},\theta))\to\mathcal{U}$ recursively as follows, where θ is a type-valuation and ξ a θ-compatible term-valuation (again omitting the superscript $\mathcal{M}$):

- $\langle x_\sigma\rangle(\theta,\xi)=\xi(x_\sigma)$.
- $\langle=_{\sigma\Rightarrow\sigma\Rightarrow\mathsf{bool}}\rangle(\theta,\xi)$ is the function $f:\langle\sigma\rangle(\theta)\to(\langle\sigma\rangle(\theta)\to M_{\mathsf{bool}})$ where $f(a)(b)=\begin{cases}M_\top, & \text{if } a=b\\ M_\bot, & \text{otherwise}\end{cases}$ (the $(M_\top,M_\bot)$-encoded equality predicate on $\langle\sigma\rangle(\theta)$).
- $\langle\varepsilon_{(\sigma\Rightarrow\mathsf{bool})\Rightarrow\sigma}\rangle(\theta,\xi)$ is the function $f:(\langle\sigma\rangle(\theta)\to M_{\mathsf{bool}})\to\langle\sigma\rangle(\theta)$ where $f(p)=\begin{cases}M_{\varepsilon,\{a\in\langle\sigma\rangle(\theta)\mid p(a)=M_\top\}},\\ \quad\text{if there exists } a\in\langle\sigma\rangle(\theta) \text{ such that } p(a)=M_\top\\ M_{\varepsilon,\langle\sigma\rangle(\theta)}, \text{ otherwise.}\end{cases}$
- If $c\in\mathsf{C}\setminus\mathsf{C}_0$ and $\sigma\leq_\rho\mathsf{tpOf}(c)$, then $\langle c_\sigma\rangle(\theta,\xi)=M_{c,\theta'}$, where $\theta':\mathsf{TV}(\mathsf{tpOf}(c))\to\mathcal{U}$ is the restriction of $\rho\cdot\theta$ to $\mathsf{TV}(\mathsf{tpOf}(c))$ and $\rho\cdot\theta$ is the type-valuation taking any α to $\langle\rho(\alpha)\rangle(\theta)$.
- $\langle t_1\ t_2\rangle(\theta,\xi)=\langle t_1\rangle(\theta,\xi)\,(\langle t_2\rangle(\theta,\xi))$.
- $\langle\lambda x_\sigma.\,t\rangle(\theta,\xi)$ the function $f:\langle\sigma\rangle(\theta)\to\langle\tau\rangle(\theta)$, where τ is the type of t, given by $f(a)=\langle t\rangle(\theta,\xi[x_\sigma\leftarrow a])$.

In the clause for λ-abstraction, $\xi[x_\sigma\leftarrow a]$ is the term-valuation obtained by updating ξ to send x_σ to a, which is θ-compatible because ξ is θ-compatible and $a\in\langle\sigma\rangle(\theta)$. In the clause for non-built-in constant instances c_σ, basic properties of type-interpretation and type-substitution ensure that $\langle c_\sigma\rangle(\theta,\xi)\in\langle\sigma\rangle(\theta)$ (because $\langle\mathsf{tpOf}(c)\rangle(\theta')=\langle\mathsf{tpOf}(c)\rangle(\rho\cdot\theta)=\langle\mathsf{tpOf}(c)[\rho]\rangle(\theta)=\langle\sigma\rangle(\theta)$) and that $\langle c_\sigma\rangle(\theta,\xi)$ does not depend on the choice of ρ (because $\sigma\leq_\rho\mathsf{tpOf}(c)$ determines uniquely the action of ρ on $\mathsf{TV}(\mathsf{tpOf}(c))$).

In the clause for ε, the important case is when the argument predicate is not vacuously false, allowing one to chose an element from its extent. On the other hand, when the predicate is vacuously false, one still needs to provide an element in the interpretation of the type—and in Pitts models this case is handled by applying the choice function M_ε to the entire set.

Note that $M_\varepsilon\in\prod_{A\in\mathcal{U}}A$ is a global choice function on the universe $\mathcal{U}$ of a Pitts model, i.e., it selects an element from each set $A\in\mathcal{U}$. This global choice follows set-theoretical tradition, and it matches the type $(\alpha\to\mathsf{bool})\to\alpha$ of the ε constant if we think of predicates via their extent. On the other hand, a more ordinary interpretation M_ε of a constant such as ε would belong, by virtue of its type $(\alpha\to\mathsf{bool})\to\alpha$, not to $\prod_{A\in\mathcal{U}}A$ but to $\prod_{A\in\mathcal{U}}(A\to M_{\mathsf{bool}})\to A$; that is, M_ε would choose not from any (nonempty) set, but from any *combination* of a set and a nonempty-extent predicate on that set. Such an "ordinary" interpretation would not guarantee, like the one in the Pitts models does, that the choice is the same for any two predicates $p:A\to M_{\mathsf{bool}}$ and $q:B\to M_{\mathsf{bool}}$ that are defined on two distinct (yet overlapping) underlying sets A and B, and have the same extent, i.e., $\{a\in A\mid p(a)=M_\top\}=\{b\in B\mid q(b)=M_\top\}$. We call this feature of the Pitts models *Uniform Choice*.

The term-valuation ξ does not matter when interpreting

closed terms, so when t is closed we write $\langle t \rangle^{\mathcal{M}}(\theta)$ to mean that $\langle t \rangle^{\mathcal{M}}(\theta, \xi)$ for some/any θ-compatible term-valuation ξ.

Given a Pitts model $\mathcal{M}$ and a Σ-formula φ, we say that $\mathcal{M}$ *satisfies* φ, written $\mathcal{M} \models_\Sigma \varphi$, when we have $\langle \varphi \rangle^{\mathcal{M}}(\theta, \xi) = M_\top$ for all type-valuations θ and θ-compatible term-valuations ξ; which, if φ is closed, becomes $\langle \varphi \rangle^{\mathcal{M}}(\theta) = M_\top$ for all type-valuations θ. We extend satisfaction to theories, defining $\mathcal{M} \models_\Sigma T$ to mean that $\mathcal{M} \models_\Sigma \varphi$ for all $\varphi \in T$.

The HOL rules of deduction are sound for Pitts models. Moreover, these models provide semantic justification for several safe specification mechanisms, including type definitions:

Prop 4: [73] If T is a Σ-theory that has a Pitts model $\mathcal{M}$, in that $\mathcal{M} \models_\Sigma T$, then the extension of Σ and T with a provably nonempty type definition also has a Pitts model $\mathcal{M}'$ which is an extension of $\mathcal{M}$.

Thus allows the HOL users to work as if *in* a Pitts model $\mathcal{M}$ with persistent identity, growing along with the theory.

III. Henkin-Style Generalization of Pitts Models

It follows from Gödel's first incompleteness theorem [32] that any recursively enumerable notion of deduction is incomplete w.r.t. standard models (for sufficiently expressive logics). Henkin-style generalizations, pioneered by Leon Henkin for Simple Type Theory [46] and then applied to many logics, are a technique for turning a logic's standard models into more abstract models for achieving completeness.

There are two features of the Pitts models that require generalization: (1) closedness under *full function spaces*, and (2) unrestricted downward closedness, which means that for any set in the universe, its *full powerset* is included in the universe.

Generalizing the first feature is well understood: We weaken closedness under full function spaces into closedness under restricted function spaces, containing not all functions but only enough of them to make term interpretation possible, i.e., to comprise all *definable functions*. We take a similar approach with the second feature: We do not consider all subsets, i.e., all elements of the full powerset, but only enough of them to comprise the *definable subsets*, i.e., those that are extents of definable predicates. This is achieved by first introducing structures called *frames* that feature restricted function space and restricted downward closedness, and then defining Henkin-style models to be frames that allow the interpretation of all terms. The next definitions use gray to highlight points of generalization from standard (Pitts) models.

Def 5: Given a type structure $(\mathsf{K}, \mathsf{arOf})$, a *type-frame* for $(\mathsf{K}, \mathsf{arOf})$ is a tuple $\mathcal{M} = (\mathcal{U}, M_{\mathsf{bool}}, M_{\mathsf{ind}}, M_\Rightarrow, (M_\kappa)_{\kappa \in \mathsf{K} \setminus \mathsf{K}_0})$ defined like a Pitts type-model (Def. 2), except that:

- there is an additional component $M_\Rightarrow : \mathcal{U}^2 \to \mathcal{U}$ such that $M_\Rightarrow(A, B) \subseteq (A \to B)$ for all $A, B \in \mathcal{U}$;
- both the closedness of $\mathcal{U}$ under function space and its downward closedness are removed.

Thus, type-frames are more abstract than Pitts type-models in that by abstracting closedness under function space to closedness under a type-frame-specific operator $M_\Rightarrow$. As for downward closedness, while this is removed from type-frames, a Henkin-style weakening of it will be introduced below for frames, when we can talk about sets definable from predicates.

The interpretation of types in type-frames $\mathcal{M}$ $\langle \_ \rangle = \langle \_ \rangle^{\mathcal{M}} : \mathsf{Type}_{(\mathsf{K},\mathsf{arOf})} \to (\mathsf{TVar} \to \mathcal{U}) \to \mathcal{U}$, is defined like the one for Pitts type-models, except that $\Rightarrow$ is interpreted not as function space but as $M_\Rightarrow$, i.e., $\langle \sigma_1 \Rightarrow \sigma_2 \rangle(\theta) = M_\Rightarrow(\langle \sigma_1 \rangle(\theta), \langle \sigma_2 \rangle(\theta))$.

Def 6: A *frame* for the signature $\Sigma = (\mathsf{K}, \mathsf{arOf}, \mathsf{C}, \mathsf{tpOf})$, is a tuple $\mathcal{M} = (\mathcal{U}, M_{\mathsf{bool}}, M_{\mathsf{ind}}, M_\Rightarrow, (M_\kappa)_{\kappa \in \mathsf{K} \setminus \mathsf{K}_0}, M_\top, M_\bot, M_\varepsilon, (M_c)_{c \in \mathsf{C} \setminus \mathsf{C}_0})$ defined like a Pitts model (Def. 3), except that:

- $(\mathcal{U}, M_{\mathsf{bool}}, M_{\mathsf{ind}}, M_\Rightarrow, (M_\kappa)_{\kappa \in \mathsf{K} \setminus \mathsf{K}_0})$ is (not a Pitts type-model but) a type-frame for $(\mathsf{K}, \mathsf{arOf})$;
- $\mathcal{U}$ is $(M_\Rightarrow, M_\top)$-downward closed, i.e., for all $A \in \mathcal{U}$ and $p \in M_\Rightarrow(A, M_{\mathsf{bool}})$, if $B = \{a \in A \mid p(a) = M_\top\}$ and $\iota_{B,A} : B \to A$ is the inclusion function, then $B \in \mathcal{U}$ and $\iota_{B,A} \in M_\Rightarrow(B, A)$.

Note that $(M_\Rightarrow, M_\top)$-downward closedness is essentially a dual of the subobject classifier property from toposes [51], as it says there are enough subobjects of A in the universe to account for all the classifying morphisms in $M_\Rightarrow(A, M_{\mathsf{bool}})$.

Given a frame $\mathcal{M} = (\mathcal{U}, M_\Rightarrow, M_{\mathsf{bool}}, M_{\mathsf{ind}}, (M_\kappa)_{\kappa \in \mathsf{K} \setminus \mathsf{K}_0}, M_\top, M_\bot, M_\varepsilon, (M_c)_{c \in \mathsf{C} \setminus \mathsf{C}_0})$ for $\Sigma = (\mathsf{K}, \mathsf{arOf}, \mathsf{C}, \mathsf{tpOf})$, we define the *interpretation of terms* as a partial function $\langle \_ \rangle^{\mathcal{M}} : \mathsf{Term}_\Sigma \to (\sum_{\theta : \mathsf{TVar} \to \mathcal{U}} \mathsf{Compat}(\mathcal{M}, \theta)) \rightharpoonup \bigcup_{A \in \mathcal{U}} A$ recursively by the following clauses, where θ is a type-valuation and ξ a θ-compatible term-valuations ξ (as usual, omitting the superscript $\mathcal{M}$):

- The clauses for variables x_σ, non-built-in constants c_σ and applications $t_1\ t_2$ are like for Pitts models.
- Consider the function $f : \langle \sigma \rangle(\theta) \to \langle \sigma \rangle(\theta) \to M_{\mathsf{bool}}$ given by $f(a)(b) = \begin{cases} M_\top, & \text{if } a = b \\ M_\bot, & \text{otherwise.} \end{cases}$
  Define $\langle =_{\sigma \Rightarrow \sigma \Rightarrow \mathsf{bool}} \rangle(\theta, \xi) =$
  $\begin{cases} f, \text{if } f \in M_\Rightarrow(\langle \sigma \rangle(\theta), M_\Rightarrow(\langle \sigma \rangle(\theta), M_{\mathsf{bool}})) \\ \text{undefined, otherwise.} \end{cases}$
- Consider the function $f : M_\Rightarrow(\langle \sigma \rangle(\theta), M_{\mathsf{bool}}) \to \langle \sigma \rangle(\theta)$,
  $f(p) = \begin{cases} M_{\varepsilon,\, \{a \in \langle \sigma \rangle(\theta) \mid p(a) = M_\top\}}, \\ \quad \text{if there exists } a \in \langle \sigma \rangle(\theta) \text{ such that } p(a) = M_\top \\ M_{\varepsilon, \langle \sigma \rangle(\theta)}, \text{ otherwise.} \end{cases}$
  Define $\langle \varepsilon_{(\sigma \Rightarrow \mathsf{bool}) \Rightarrow \sigma} \rangle(\theta, \xi) =$
  $\begin{cases} f, \text{ if } f \in M_\Rightarrow(M_\Rightarrow(\langle \sigma \rangle(\theta), M_{\mathsf{bool}}), \langle \sigma \rangle(\theta)) \\ \text{undefined, otherwise.} \end{cases}$
- Consider the function $f : \langle \sigma \rangle(\theta) \to \langle \tau \rangle(\theta)$, where τ is the type of t, given by $f(a) = \langle t \rangle(\theta, \xi[x_\sigma \leftarrow a])$.
  Define $\langle \lambda x_\sigma.\, t \rangle(\theta, \xi) = \begin{cases} f, \text{ if } f \in M_\Rightarrow(\langle \sigma \rangle(\theta), \langle \tau \rangle(\theta)) \\ \text{undefined, otherwise.} \end{cases}$

Implicit above is that the interpretation is undefined on a term if undefined on one of its immediate subterms.

Def 7: A *Pitts-Henkin model* is a frame for which the term interpretation partial function is actually a total function $\langle \_ \rangle^{\mathcal{M}} : \mathsf{Term}_\Sigma \to (\sum_{\theta : \mathsf{TVar} \to \mathcal{U}} \mathsf{Compat}(\mathcal{M}, \theta)) \to \bigcup_{A \in \mathcal{U}} A$.

Roughly speaking, a Pitts-Henkin model is a frame whose sets of the universe contain all the definable functions (and therefore, thanks to the frame properties, the universe contains all the definable subsets). The notions of satisfaction by a Pitts-Henkin model of a formula or a theory are defined similarly to those for Pitts models. For a theory T over Σ and

a Σ-formula φ, we define the *semantic Pitts-Henkin model deduction of φ from T*, written $T \models_\Sigma \varphi$, to mean that $\mathcal{M} \models_\Sigma T$ implies $\mathcal{M} \models_\Sigma \varphi$ for all Pitts-Henkin models $\mathcal{M}$.

With the Henkin-style generalization in place, we ask about soundness and completeness. While deduction is sound for Pitts-Henkin models, completeness fails immediately:

Prop 8: HOL deduction is not complete w.r.t. the Pitts-Henkin models, (i.e., it is *not* true that, for all signatures Σ, and all theories T and formulas φ over Σ, $T \models_\Sigma \varphi$ implies $T \vdash_\Sigma \varphi$).

Proof: Let φ be the Σ_{init}-formula $\forall x_\alpha.\ \exists y_\alpha.\ x \neq y$, which is polymorphic in α and says that α is not a singleton. The satisfaction of φ by a Pitts-Henkin model means that there is no singleton set in its universe—but this is false: $\{M_\top\}$ is a definable subset of M_{bool} hence it belongs to the universe. So we have $\{\varphi\} \models_{\Sigma_{\mathsf{init}}} \mathsf{false}$.

However, it is not true that $\{\varphi\} \vdash_{\Sigma_{\mathsf{init}}} \mathsf{false}$. Indeed, this would imply $\{\varphi[\sigma_1/\alpha], \ldots, \varphi[\sigma_n/\alpha]\} \vdash_{\Sigma_{\mathsf{init}}} \mathsf{false}$ for some ground types $\sigma_1, \ldots, \sigma_n$. But since the only ground types in Σ_{init} are built from bool, ind and $\Rightarrow$ which all provably have more than one element, we would obtain that $\emptyset \vdash_{\Sigma_{\mathsf{init}}} \varphi[\sigma_i/\alpha]$ for each i, and therefore $\emptyset \vdash_{\Sigma_{\mathsf{init}}} \mathsf{false}$, which contradicts the consistency of (initial) HOL. ∎

IV. Developing the Completeness Result

We aim to prove a completeness theorem of the form: A *suitable enrichment* of HOL deduction is complete w.r.t. the Pitts-Henkin models, i.e., $T \models_\Sigma \varphi$ implies $T \vdash_\Sigma \varphi$ (where $\vdash_\Sigma$ now refers to the enriched HOL deduction). As usual, we will often omit the signature Σ, writing $\vdash$ and $\models$ for $\vdash_\Sigma$ and $\models_\Sigma$.

(We will discover such a suitable enrichment from attempting a proof of completeness in the style of Henkin. As shown by Thm 9 from the end of this section, it will consist of:

- a major addition: the Strong Local Typedef rule (shown in Fig. 6), which allows performing type definitions in local proof contexts while preserving the behavior of the ε function of the host type;
- a minor addition: the Trivial Choice Axiom, $\varepsilon\,(\lambda x_\alpha.\ \mathsf{false}) = \varepsilon\,(\lambda x_\alpha.\ \mathsf{true})$, equating the behavior of ε on the vacuously false and vacuously true predicates.)

We are after a Henkin-style proof of completeness, so let us recall the steps required by Henkin's original proofs [46], [45]:

Step 1. Reducing the completeness problem to the model existence problem, i.e., the problem of showing that any (syntactically) consistent theory has a model.

Step 2. Showing that any consistent theory can be extended to a (maximally consistent) Hintikka theory.

Step 3. Showing that any Hintikka theory has a model.

And since Steps 2 and 3 solve the model existence problem, completeness is proved thanks to Step 1.

A. Dealing with polymorphism in the deduction system

For many logics, FOL and Simple Type Theory included, Step 1 (a generalization of the refutability-based reformulation from Gödel's original proof for FOL [31]) is straightforward: Assuming $T \models \varphi$ and wanting to prove $T \vdash \varphi$, for a contradiction assume that $T \vdash \varphi$ does not hold. From this, we obtain $T \cup \{\neg\,\varphi\}$ is consistent, often expressed as $T \cup \{\neg\,\varphi\} \nvdash \mathsf{false}$ (or an equivalent formulation). By the model existence property, there is a model $\mathcal{M}$ with $\mathcal{M} \models T \cup \{\neg\,\varphi\}$, i.e., $\mathcal{M} \models T$ and $\mathcal{M} \models \neg\,\varphi$, i.e., $\mathcal{M} \models T$ and $\mathcal{M} \not\models \varphi$, contradicting $T \models \varphi$.

However, the above argument does not work for HOL, specifically because it is not true that the failure of $T \vdash \varphi$ implies the consistency of $T \cup \{\neg\,\varphi\}$. E.g., consider again the α-polymorphic formula φ that says α is not a singleton type. Then $\emptyset \nvdash_{\Sigma_{\mathsf{init}}} \varphi$, yet $\{\neg\,\varphi\}$ is inconsistent, as it requires all types to be singletons. The culprit is the implicit polymorphism at the top of HOL judgments: a formula φ in $T \vdash \varphi$ implicitly refers to what could be written as $\forall \mathsf{TV}(\varphi).\ \varphi$, the universal quantification of φ over all its type-variables. In this notation, $T \vdash \neg\,\varphi$ means $T \vdash \forall \mathsf{TV}(\varphi).\neg\,\varphi$, and not $T \vdash \neg(\forall \mathsf{TV}(\varphi).\ \varphi)$.

So the above argument for reducing completeness to model existence can only be applied to ground formulas. To address this, we reduce completeness to *ground-conclusion completeness*, namely the property that $T \models \varphi$ implies $T \vdash \varphi$ assuming φ is ground, along the following lines. Given a polymorphic formula φ, we extend the signature with fresh ground types u_α corresponding to the type-variables α of φ and prove a fact in the style of Shoenfield's "theorem on constants" from FOL [82], [67] showing that deduction of φ (within the original signature) is implied by deduction of $\varphi[\rho]$, i.e., of φ type-substituted via ρ, in the extended signature—where ρ is a "grounding substitution", replacing every type-variable α from φ with u_α. This implies our desired reduction. In conclusion, we complete Step 1 with the help of a preliminary step:

Step 0. Reducing completeness to ground-conclusion completeness.

B. Traditional Hintikka theories and syntactic Henkin models

For engaging in Steps 2 and 3, we need a suitable notion of Hintikka theory. Traditionally, Hintikka conditions on a theory T refer to the model-like behavior of deduction from T—e.g., ensuring that $T \vdash \varphi$ iff not $T \vdash \neg\,\varphi$ (which mimics the semantics of negation), and that $T \vdash \exists x_\sigma.\ \varphi$ iff there exists a (closed) term t such that $T \vdash \varphi[t/x]$ (mimicking the semantics of existential quantification).

This behavior makes it possible to construct an actual model. Namely, for any Hintikka theory T one defines the *T-deductive equivalence* on closed (typable) terms, according to which two terms t, t' are equivalent, written $t \simeq t'$, when they have the same type and their equality is inferable from T, i.e., $T \vdash t = t'$. Finally, based on these equivalence classes, one should (ideally) be able to define a so-called *syntactic model* for T. In Henkin's completeness proof for Simple Type Theory, each type σ is interpreted as a set isomorphic to the set $\mathsf{CTerm}_{\sigma/\simeq}$ of $\simeq$-equivalence classes of closed terms—not equal but only isomorphic because function types of the form $\sigma_1 \Rightarrow \sigma_2$ are interpreted not as $\mathsf{CTerm}_{\sigma_1 \Rightarrow \sigma_2/\simeq}$, but as a restricted function space between $\mathsf{CTerm}_{\sigma_1/\simeq}$ and $\mathsf{CTerm}_{\sigma_2/\simeq}$, namely the subset of $\mathsf{CTerm}_{\sigma_1/\simeq} \to \mathsf{CTerm}_{\sigma_2/\simeq}$ containing all functions determined by the syntactic application of

a term in $\mathsf{CTerm}_{\sigma_1 \Rightarrow \sigma_2}$. This induces a bijection between $\mathsf{CTerm}_{\sigma_1 \Rightarrow \sigma_2 /\simeq}$ and this restricted function space. Overall, the syntactic model interprets each type σ as a set U_σ for which we have the mutually inverse to-and-fro bijections with the sets of equivalence classes of terms, $\mathsf{to}_\sigma : U_\sigma \to \mathsf{CTerm}_{\sigma/\simeq}$ and $\mathsf{fro}_\sigma : \mathsf{CTerm}_{\sigma/\simeq} \to U_\sigma$. For the non-function types σ, including the built-ins bool and ind, we have actual equality, in that $U_\sigma = \mathsf{CTerm}_{\sigma/\simeq}$, and to_σ and fro_σ are the identity functions.

For the Henkin models of Simple Type Theory the above is enough, in that it can be proved that the sets U_σ form the basis of a model of T. Moreover, it is not difficult to factor in the rank-1 polymorphism in this scheme, by (1) building the sets U_σ only for ground types σ and ground closed terms (so replacing CTerm with GCTerm) and then, taking advantage of the fact that each set U_σ determines σ uniquely, (2) proving that the interpretation of polymorphic types and formulas can be reduced to substitution.

C. Constructing a downward closed model

The syntactic model sketched above, whose universe, let's denote it by $\mathcal{U}$, is the set of all sets U_σ, is not downward closed—far from it, since the sets U_σ are mutually disjoint. We address this in two stages:

Substep 3.1. At the syntactic level, we enrich the notion of Hintikka theory to witness, via certain ground types $\tau_{\sigma,t}$, all extents of (ground closed) predicates $t : \sigma \Rightarrow \mathsf{bool}$ (i.e., $\tau_{\sigma,t}$ will be in bijection with the extent of t in σ, like with type definitions).

Substep 3.2. At the semantic level, we produce a downward closure $\mathcal{V}$ of the syntactic universe $\mathcal{U}$ using the interpretation of the witness types $\tau_{\sigma,t}$ as "proxies" whose structure we copy.

So $\mathcal{V}$ is (definably) downward closed by construction, while at the same time it does not satisfy any new syntactically-definable properties compared to $\mathcal{U}$, since the proxy infrastructure delegates the interpretation in $\mathcal{V}$ to the $\mathcal{U}$ subuniverse (which is well-behaved by construction). This allows us to conclude Step 3 with:

Substep 3.3. We define a Pitts-Henkin model with universe $\mathcal{V}$, and prove that it is a model of our Hintikka theory T.

The above three substeps are discussed below. Substep 3.1 is about bringing witness types into the Hintikka conditions. We start by adding the requirement that, for every ground closed predicate $t : \sigma \Rightarrow \mathsf{bool}$ whose extent's non-emptiness is provable from the given theory (in that $T \vdash \exists x_\sigma.\, t\ x$) there exists a ground type $\tau_{\sigma,t}$ that is T-provably isomorphic to the extent of t in σ. Recall from Def. 1 that this is exactly what type definitions express, so we can write it as $T \vdash \tau_{\sigma,t} \cong (\sigma, t)$, where $\tau_{\sigma,t} \cong (\sigma, t)$ denotes $\exists rep_{\tau_\sigma \Rightarrow \sigma}.\, \mathsf{tdef}_{\tau_{\sigma,t},\sigma,t,rep}$, and $\mathsf{tdef}_{\tau_{\sigma,t},\sigma,t,rep}$ says that *rep* is a bijection between $\tau_{\sigma,t}$ and the extent of t in σ. We will actually consider some concrete (ground closed) terms $\mathsf{Rep}_{\sigma,t} : \tau_{\sigma,t} \Rightarrow \sigma$ such that $T \vdash \mathsf{tdef}_{\tau_{\sigma,t},\sigma,t,\mathsf{Rep}_{\sigma,t}}$, and the terms $\mathsf{Abs}_{\sigma,t} : \sigma \Rightarrow \tau_{\sigma,t}$ expressing the inverses of $\mathsf{Rep}_{\sigma,t}$. We call $\mathsf{Rep}_{\sigma,t}$ the *representation function* of the witness type $\tau_{\sigma,t}$.

Since the identity of a term in the syntactic universe is up to T-deductive equivalence, we do not want our witness types to break this abstraction layer, so we require the following.

Witness Abstractness: $\tau_{\sigma,t} = \tau_{\sigma,t'}$ and $\mathsf{Rep}_{\sigma,t} = \mathsf{Rep}_{\sigma,t'}$ whenever σ is a ground type, $t, t' : \sigma \Rightarrow \mathsf{bool}$ are ground closed terms, and $T \vdash t = t'$.

Now we move to discussing Substep 3.2 (and will get back to Substep 3.1 later, after Substep 3.2 suggests more Hintikka conditions). First, let us gather some facts and notations about the syntactic universe $\mathcal{U}$: Recall that $\mathcal{U}$ consists of the sets U_σ where σ is a ground type. Each U_σ determines σ uniquely and is isomorphic to $\mathsf{GCTerm}_{\sigma/\simeq}$ via the to-and-fro bijections to_σ and fro_σ. Moreover, U_{bool} is actually equal to $\mathsf{GCTerm}_{\mathsf{bool}/\simeq}$, and consists of exactly two elements: the T-deductive equivalence classes of true and false—we denote these by $\top$ and $\bot$. Finally, $\mathcal{U}$ is closed under definable function space; we let $\mathsf{Fun} : \mathcal{U} \times \mathcal{U} \to \mathcal{U}$ denote this restricted function space operator, defined as $\mathsf{Fun}(U_{\sigma_1}, U_{\sigma_2}) = U_{\sigma_1 \Rightarrow \sigma_2} \subseteq (U_{\sigma_1} \to U_{\sigma_2})$.

We want to close $\mathcal{U}$ under definable subsets. To this end, by induction-recursion we define **(1)** the extended universe $\mathcal{V} \supseteq \mathcal{U}$ together with **(2)** its restricted function space operator $\mathsf{Fun}' : \mathcal{V} \times \mathcal{V} \to \mathcal{V}$ (so that $\mathsf{Fun}'(A, B) \subseteq (A \to B)$ for all $A, B \in \mathcal{V}$), **(3)** the function $\mathsf{Proxy} : \mathcal{V} \to \mathcal{U}$, and **(4)** for each $A \in \mathcal{V}$, the mutually inverse bijections $\mathsf{toU}_A : \mathsf{Proxy}(A) \to A$ and $\mathsf{froU}_A : A \to \mathsf{Proxy}(A)$. The definition has three inductive-recursive rules, summarized below.

The base rule: This rule deals with the sets in $\mathcal{U}$, taking the proxy infrastructure to be trivial for $\mathcal{U}$. Namely for all $A \in \mathcal{U}$, $\mathsf{Proxy}(A) = A$ and $\mathsf{toU}_A = \mathsf{froU}_A = 1_A$; in particular, $\mathsf{Proxy}(U_{\mathsf{bool}}) = U_{\mathsf{bool}}$. And Fun' extends Fun, i.e., coincides with Fun on sets from $\mathcal{U}$.

Rule for downward closedness: This ensures that $\mathcal{V}$ contains every nonempty subset $B = \{a \in A \mid p(a) = \top\}$ of an existing set A definable as the extent of an existing predicate p, i.e., $B \in \mathcal{V}$ whenever $A \in \mathcal{V}$ and $p \in \mathsf{Fun}'(A, U_{\mathsf{bool}})$. As the proxy for B, $\mathsf{Proxy}(B)$, we take the set $U_{\tau_{\sigma,t}}$, where σ is the unique ground type such that $\mathsf{Proxy}(A) = U_\sigma$ and $t : \sigma \Rightarrow \mathsf{bool}$ is the syntactic counterpart of the predicate that corresponds to p via the proxy infrastructure, i.e., $t_{/\simeq} = \mathsf{to}_{\sigma \Rightarrow \mathsf{bool}}(p \circ \mathsf{froU}_A)$. By Witness Abstractness, $\tau_{\sigma,t}$ is uniquely identified by A and p—because t is uniquely identified by p up to T-deductive equivalence.

Rule for restricted-function-space closedness: This rule keeps $\mathcal{V}$ closed under the restricted function space operator Fun' which copies the one from $\mathcal{U}$ via the proxy infrastructure. In particular, we have that, for all $A, B \in \mathcal{V}$, $\mathsf{Proxy}(\mathsf{Fun}'(A, B)) = \mathsf{Fun}(\mathsf{Proxy}(A), \mathsf{Proxy}(B))$.

Correctness of the definition. To guarantee that our inductive-recursive definition of $\mathcal{V}$ and its proxy infrastructure is correct, we must prove that the rule for downward closedness does not introduce any ambiguity. Indeed, the rule offers two potentials for ambiguity.

One arises in the degenerate case when the predicate $p : A \to U_{\mathsf{bool}}$ is vacuously true, so that the newly introduced set $B = \{a' \in A \mid p(a)\}$ is equal to A. In

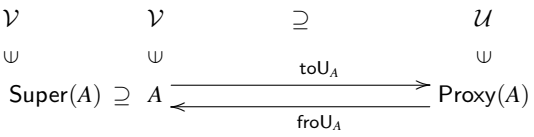

Fig. 3. The extension $\mathcal{V}$ of the universe $\mathcal{U}$.

this case, since $\mathsf{Proxy}(A) = U_\sigma$ and $\mathsf{Proxy}(B) = U_{\tau_{\sigma,t}}$ where $t_{/\simeq} = \mathsf{to}_{\sigma\Rightarrow\mathsf{bool}}(p \circ \mathsf{froU}_A)$, we have $U_{\tau_{\sigma,t}} = \mathsf{Proxy}(B) = \mathsf{Proxy}(A) = U_\sigma$, hence an overlap between $U_{\tau_{\sigma,t}}$ and U_σ. So the ambiguity would be avoided provided $\tau_{\sigma,t} = \sigma$. Because, thanks to p being vacuosuly true, we have $t_{/\simeq} = (\lambda x_\sigma.\,\mathsf{true})_{/\simeq}$ hence $T \vdash t = (\lambda x_\sigma.\,\mathsf{true})$, and also factoring in Witness Abstractness, it suffices to assume the following, which we add to our Hintikka conditions (as part of Substep 3.1).

Witness Identity: $\tau_{\sigma,\,\lambda x_\sigma.\mathsf{true}} = \sigma$ and $T \vdash \mathsf{Rep}_{\sigma,\,\lambda x_\sigma.\mathsf{true}} = 1_\sigma$ whenever σ is a ground type.

The other potential for ambiguity goes deeper into the structure of our definition. We must prove that, for the newly added set B, we obtain the same proxy infrastructure $\mathsf{Proxy}(B), \mathsf{toU}_B, \mathsf{froU}_B$ regardless of the superset A and predicate $p : A \to U_{\mathsf{bool}}$ used to produce B (as the extent of p in A).

To achieve this, we introduce another bit of infrastructure, namely for each set $A \in \mathcal{V}$, we also define (as part of the inductive-recursive definition), the set $\mathsf{Super}(A) \in \mathcal{V}$ as the maximal superset of A in $\mathcal{V}$. In fact, $\mathsf{Super}(A)$ will be the original superset ancestor of A via applications of the downward closedness rule, in that there will exist a chain $\mathsf{Super}(A) = A_0 \supseteq A_1 \supseteq \ldots \supseteq A_n = A$ leading from $\mathsf{Super}(A)$ to A such that each A_i was introduced (by comprehension) as a subset of A_{i-1}. Moreover, we maintain as an invariant the fact that the indicator $q : \mathsf{Super}(A) \to U_{\mathsf{bool}}$ of A in $\mathsf{Super}(A)$ is definable in $\mathcal{V}$, in that $q \in \mathsf{Fun}'(\mathsf{Super}(A), U_{\mathsf{bool}})$. (This is true because q is the conjunction of the all the indicator predicates along the aforementioned ancestry chain.) Note that $\mathsf{Super}(A) = A$ for all $A \in U$. Fig. 3 depicts the extended universe $\mathcal{V}$ and its proxy and superset infrastructures.

As another invariant of the definition (a consequence of the fact that new subsethood relations appear only via the downward closedness rule), we have the diamond-like property that non-disjoint sets always have the same superset, i.e., $A_1 \cap A_2 \neq \emptyset$ implies $\mathsf{Super}(A_1) = \mathsf{Super}(A_2)$ for all $A_1, A_2 \in \mathcal{V}$. Thanks to this, we can normalize the process of comprehension in $\mathcal{V}$ by reducing every comprehension to one from the superset. Indeed, now we have the opportunity to state the following: **(1)** introducing B by the downward closedness rule from some set $A_1 \in \mathcal{V}$ and predicate $p_1 : A_1 \to U_{\mathsf{bool}}$ *should give the same result (i.e., same* $\mathsf{Proxy}(B), \mathsf{toU}_B, \mathsf{froU}_B$*) as* **(2)** introducing B by the downward closedness rule from any other set $A_2 \in \mathcal{V}$ and predicate $p_2 : A_2 \to U_{\mathsf{bool}}$, *because they both give the same result as* introducing B from the common ancestor $A = \mathsf{Super}(A_1) = \mathsf{Super}(A_2)$, using, as predicate on A, **(1)** the conjunction between p_1 and the indicator predicate $q_1 : A \to U_{\mathsf{bool}}$ of A_1 in A, *which is the same as* **(2)** the conjunction between p_2 and the indicator predicate $q_2 : A \to U_{\mathsf{bool}}$ of A_2 in A.

To make the emphasized claims true, we again go back, via the proxy infrastructure, to the syntactic model, and further to the underlying theory T. Since the proxies for comprehension are $U_{\tau_{\sigma,t}}$ for suitable witness types $\tau_{\sigma,t}$ where t is determined by the defining predicates, we can to reduce these claims to the following property depicted in Fig. 4 (right), which becomes a new Hintikka condition.

Witness Compositionality: $\tau_{\tau_{\sigma,t},\,t'} = \tau_{\sigma,\,t\,\wedge_2\,(t'\,\circ\,\mathsf{Abs}_{\sigma,t})}$ and $\mathsf{Rep}_{\sigma,\,t\,\wedge_2\,(t'\,\circ\,\mathsf{Abs}_{\sigma,t})} = \mathsf{Rep}_{\sigma,t} \circ \mathsf{Rep}_{\tau_{\sigma,t},\,t'}$ whenever σ is a ground type, $t : \sigma \Rightarrow \mathsf{bool}$ and $t' : \tau_{\sigma,t} \Rightarrow \mathsf{bool}$ are ground closed terms, $T \vdash \exists x_\sigma.\, t\ x$ and $T \vdash \exists y_{\tau_{\sigma,t}}.\, t'\ y$.

Above, $\wedge_2$ denotes componentwise conjunction of predicates (i.e., $s_1 \wedge_2 s_2$ denotes $\lambda x_\sigma.\ s_1\ x \wedge s_2\ x$). To explain why this condition solves our non-ambiguity problem, we chase the diagrams of Fig. 4. The figure focuses on the route of introducing B from A_1 versus that of introducing B from $A = \mathsf{Super}(A_1)$, and we will show that the two routes yield the same result. (A similar argument works for A_2 versus $A = \mathsf{Super}(A_2)$, and the two therefore prove what we want.) The left edges of the three triangles (center, left and right) account for the first route, tracing the superset chain $A \supseteq A_1 \supseteq B$, where $q_1 : A \to U_{\mathsf{bool}}$ and $p_1 : A_1 \to U_{\mathsf{bool}}$ are the indicator predicates of A_1 in A, and of B in A_1, respectively:

- σ is the type corresponding to $A = \mathsf{Super}(A_1)$, i.e., $\mathsf{Proxy}(A) = U_\sigma$; and $t : \sigma \Rightarrow \mathsf{bool}$ is the (unique up to T-deducibility) syntactic predicate corresponding to q_1, i.e., $t_{/\simeq} = \mathsf{to}_{\sigma\Rightarrow\mathsf{bool}}(\mathsf{toU}_{\mathsf{Fun}'(A,U_{\mathsf{bool}})}(q_1)) = \mathsf{to}_{\sigma\Rightarrow\mathsf{bool}}(q_1 \circ \mathsf{froU}_A)$;
- $\tau_{\sigma,t}$ corresponds to A_1, i.e., $\mathsf{Proxy}(A_1) = U_{\tau_{\sigma,t}}$; and $t' : \tau_{\sigma,t} \Rightarrow \mathsf{bool}$ corresponds to p_1, i.e., $t'_{/\simeq} = \mathsf{to}_{\tau_{\sigma,t}\Rightarrow\mathsf{bool}}(\mathsf{toU}_{\mathsf{Fun}'(A_1,U_{\mathsf{bool}})}(p_1)) = \mathsf{to}_{\tau_{\sigma,t}\Rightarrow\mathsf{bool}}(p_1 \circ \mathsf{froU}_{A_1})$;
- $\tau_{\tau_{\sigma,t},t'}$ is the type corresponding to B along this route, i.e., $\mathsf{Proxy}(B) = U_{\tau_{\tau_{\sigma,t},t'}}$.

On the other hand, the right edges of the three triangles account for the second route, tracing the direct superset relationship $A \supseteq B$, where $r_1 : A \to U_{\mathsf{bool}}$, the indicator of B in A, is the conjunction of q_1 and p_1, and we let s be the syntactic conjunction $t \wedge_2 (t' \circ \mathsf{Abs}_{\sigma,t})$:

- $s : \sigma \Rightarrow \mathsf{bool}$ can be shown to be (T-provably equal to) the syntactic predicate corresponding to r_1, i.e., $s_{/\simeq} = \mathsf{to}_{\sigma\Rightarrow\mathsf{bool}}(\mathsf{toU}_{\mathsf{Fun}'(A,U_{\mathsf{bool}})}(r_1)) = \mathsf{to}_{\sigma\Rightarrow\mathsf{bool}}(r_1 \circ \mathsf{froU}_A)$;
- $\tau_{\sigma,s}$ is the type corresponding to B along this route, i.e., $\mathsf{Proxy}(B) = U_{\tau_{\sigma,s}}$.

Now, the two alternative definitions of $\mathsf{Proxy}(B)$ and froU_B (and consequently of toU_B too), emerging from the two routes of introducing B, are seen to be reconciled thanks to Witness Compositionality. Indeed, the two definitions of $\mathsf{Proxy}(B)$ are equivalent because $\tau_{\sigma,s} = \tau_{\tau_{\sigma,t},t'}$. Moreover, the ($T$-provable) commutativity of the right triangle implies the commutativity of the center triangle. In turn, this ensures that defining froU_B based on froU_{A_1} (so to make the rectangle of vertices B, A_1, $\mathsf{Proxy}(A_1)$ and $\mathsf{Proxy}(B)$ commutative) gives the same result as defining it based on froU_A (so to make the rectangle of vertices B, A, $\mathsf{Proxy}(A)$ and $\mathsf{Proxy}(B)$ commutative).

We are finally ready for Substep 3.3, namely organizing $\mathcal{V}$ into a Pitts-Henkin model for the Hintikka theory T. First, we

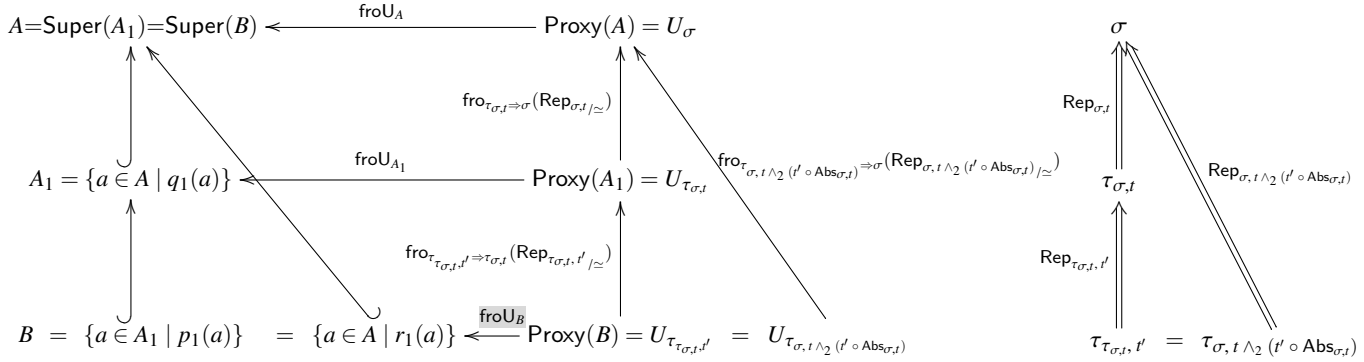

Fig. 4. Tracing back the desired comprehension unambiguity in $\mathcal{V}$ (left), via $\mathcal{U}$ (center) to the Witness Compositionality property (right).

define a type-frame $\mathcal{M}$ having universe $\mathcal{V}$ and interpreting the type constructors using the natural syntactic intepretation in $\mathcal{U}$, via proxy. Thus, we take $M_{\Rightarrow}$ to be Fun' (which had already been set up to relate to the Fun operator on $\mathcal{U}$ via proxy), $M_{\mathsf{bool}} = U_{\mathsf{bool}}$ and $M_{\mathsf{ind}} = U_{\mathsf{ind}}$. Moreover, if κ is a non-built-in n-ary type constructor, $M_\kappa : \mathcal{V}^n \to \mathcal{V}$ is defined by $M_\kappa(A_1, \ldots, A_n) = U_{(\sigma_1,\ldots,\sigma_n)\kappa}$ where each σ_i is the unique ground type such that $\mathsf{Proxy}(A_i) = U_{\sigma_i}$.

We write $\langle\sigma\rangle$ for the semantic interpretation of (not necessarily ground) types σ in our type-frame $\mathcal{M}$; thus, for any type-valuation $\theta : \mathsf{TVar} \to \mathcal{U}$, we have that $\langle\sigma\rangle(\theta) \in \mathcal{V}$. The above "by-proxy" interpretation of the type constructors extends to the interpretation of types, in that we can prove the following by structural induction on types:

The type interpretation lemma. $\mathsf{Proxy}(\langle\sigma\rangle(\theta)) = U_{\sigma[\rho_\theta]}$, where $\rho_\theta : \mathsf{TVar} \to \mathsf{GType}$ is the syntactic counterpart of θ (via proxy), i.e., for all α, $\rho_\theta(\alpha)$ is the unique ground type σ' with $\mathsf{Proxy}(\theta(\alpha)) = U_{\sigma'}$.

On top of this type-frame, we define a frame, also denoted by $\mathcal{M}$, using again the proxy infrastructure to borrow the natural syntactic interpretation from $\mathcal{U}$. We take $M_\top$ and $M_\bot$ to be $\top$ and $\bot$. If c is a non-built-in constant of type σ, we define $M_c \in \prod_{\theta:\mathsf{TV}(\sigma)\to\mathcal{V}}\langle\sigma\rangle(\theta)$ by $M_{c,\theta} = \mathsf{froU}_{\langle\sigma\rangle(\theta)}(\mathsf{fro}_{\sigma[\rho_\theta]}(c_{\sigma[\rho_\theta]/\simeq}))$. We postpone the definition of $M_\varepsilon \in \prod_{A\in\mathcal{V}} A$, as we will infer it from proof requirements.

We write $\langle t\rangle$ for the semantic interpretation of (not necessarily ground or closed) terms t in our frame $\mathcal{M}$; thus, for any type-valuation $\theta : \mathsf{TVar} \to \mathcal{V}$ and θ-compatible term-valuation $\xi : \mathsf{VarT} \to \bigcup_{A\in\mathcal{V}} A$, we have that $\langle t\rangle(\theta,\xi)$ is either undefined or is an element of $\langle\sigma\rangle(\theta)$ where σ is the type of t.

The term interpretation lemma. $\langle t\rangle(\theta,\xi)$ is defined and $\langle t\rangle(\theta,\xi) = \mathsf{froU}_{\langle\sigma\rangle(\theta)}(\mathsf{fro}_{\sigma[\rho_\theta]}(t[\rho_\theta,\delta_{\theta,\xi}]/_\simeq))$, where σ is the type of t, and $\delta_{\theta,\xi} : \mathsf{VarT} \to \mathsf{GCTerm}$ is a syntactic counterpart of ξ (again via the proxy infrastructure), i.e.: for all $x_{\sigma'} \in \mathsf{VarT}$, $\delta_{\theta,\xi}(x_{\sigma'}) = s$ where $s \in \mathsf{GCTerm}_{\sigma'[\rho_\theta]}$ is such that $s/_\simeq = \mathsf{to}_{\sigma'[\rho_\theta]}(\mathsf{toU}_{\langle\sigma'\rangle(\theta)}(\xi(x_{\sigma'})))$.

Above, $t[\rho_\theta,\delta_{\theta,\xi}]$ denotes the application of the type-substitution ρ_θ and the quasi-term-substitution $\delta_{\theta,\xi}$ *in tandem* to the term t, namely $\delta_{\theta,\xi}$ is applied to all the free typed variables x_σ (in a capture-avoiding fashion) while ρ_θ is applied to all occurrences of types in t that are not within free typed variables. We call $\delta_{\theta,\xi}$ a *quasi*-term-substitution because it does not preserve the types directly, but only modulo ρ_θ, i.e., each $\delta_{\theta,\xi}(x_\sigma)$ has type not σ but $\sigma[\rho_\theta]$; this mirrors syntactically the semantic compatibility relationship between ξ and δ, i.e., the fact that $\xi(x_\sigma) \in \langle\sigma\rangle(\theta)$.

While the definedness part of the term interpretation lemma means that $\mathcal{M}$ is a Pitts-Henkin model, the lemma also yields the final desired fact from Substep 3.3: that $\mathcal{M}$ is a model of T. Indeed, for any $\varphi \in T$, any θ and any θ-compatible ξ, using that $\mathsf{bool}[\rho_\theta] = \mathsf{bool}$ and that $\mathsf{fro}_{\mathsf{bool}}$ and $\mathsf{froU}_{U_{\mathsf{bool}}}$ are $1_{U_{\mathsf{bool}}}$, the lemma gives $\langle\varphi\rangle(\theta,\xi) = \mathsf{froU}_{\langle\mathsf{bool}\rangle(\theta)}(\mathsf{fro}_{\mathsf{bool}[\rho_\theta]}(\varphi[\rho_\theta,\delta_{\theta,\xi}]/_\simeq)) = \varphi[\rho_\theta,\delta_{\theta,\xi}]/_\simeq$. Moreover, using that $T \vdash \varphi$ and therefore (by a substitution lemma) $T \vdash \varphi[\rho_\theta,\delta_{\theta,\xi}]$, i.e., $T \vdash \varphi[\rho_\theta,\delta_{\theta,\xi}] = \mathsf{true}$, we obtain that $\varphi[\rho_\theta,\delta_{\theta,\xi}]/_\simeq = \mathsf{true}/_\simeq = \top = M_\top$. Hence $\langle\varphi\rangle(\theta,\xi) = M_\top$, as desired.

The proof of the term interpretation lemma goes by structural induction on terms. The cases different from ε are fairly routine, using the properties of $\mathcal{U}$ and $\mathcal{V}$. Now let us look into the ε case, which will inform our definition of M_ε.

Letting σ' denote $(\sigma \Rightarrow \mathsf{bool}) \Rightarrow \sigma$, we must prove $\langle\varepsilon_{\sigma'}\rangle(\theta,\xi)$ defined and equal to $\mathsf{froU}_{\langle\sigma'\rangle(\theta)}(\mathsf{fro}_{(\sigma')[\rho_\theta]}(\varepsilon_{\sigma'}[\rho_\theta,\delta_{\theta,\xi}]/_\simeq))$. Expanding the definitions, this amounts to proving $f = g$, where f and g are functions in $\mathsf{Fun}'(\langle\sigma\rangle(\theta), M_{\mathsf{bool}}) \to \langle\sigma\rangle(\theta)$ acting as follows, for any $p \in \mathsf{Fun}'(\langle\sigma\rangle(\theta), M_{\mathsf{bool}})$:

$$f(p) = \begin{cases} M_{\varepsilon,\,\{a\in\langle\sigma\rangle(\theta)\,\mid\,p(a)=\top\}}, \text{ if } p \text{ has non-empty extent} \\ M_{\varepsilon,\,\langle\sigma\rangle(\theta)}, \text{ otherwise.} \end{cases}$$

$g(p) = \mathsf{froU}_{\langle\sigma\rangle(\theta)}(\mathsf{fro}_{\sigma[\rho_\theta]}((\varepsilon\, t)/_\simeq))$, where $t : \sigma[\rho_\theta] \Rightarrow \mathsf{bool}$ corresponds to p, i.e., $t/_\simeq = \mathsf{to}_{\sigma[\rho_\theta]\Rightarrow\mathsf{bool}}(p \circ \mathsf{froU}_{\langle\sigma\rangle(\theta)})$.

Case 1: Assume p has nonempty extent. We need to prove

$$M_{\varepsilon,\,\{a\in\langle\sigma\rangle(\theta)\,\mid\,p(a)=\top\}} = \mathsf{froU}_{\langle\sigma\rangle(\theta)}(\mathsf{fro}_{\sigma[\rho_\theta]}((\varepsilon\, t)/_\simeq)) \quad \textbf{(*)}$$

So we are compelled to define M_ε such that, for all $B \in \mathcal{V}$, $M_{\varepsilon,B} = \mathsf{froU}_{\langle\sigma\rangle(\theta)}(\mathsf{fro}_{\sigma[\rho_\theta]}((\varepsilon\, t)/_\simeq))$ whenever B has the form $\{a \in \langle\sigma\rangle(\theta) \mid p(a) = \top\}$ for some type σ and predicate p (where t is the term corresponding to p). The problem is that such σ and p are not unique, i.e., we can have σ_1, p_1 and σ_2, p_2 with $B = \{a \in A_1 \mid p_1(a) = \top\} = \{a \in A_2 \mid p_2(a) = \top\}$, where $A_1 = \langle\sigma_1\rangle(\theta)$ and $A_2 = \langle\sigma_2\rangle(\theta)$. So in such cases we would like to have $\mathsf{froU}_{\langle\sigma_1\rangle(\theta)}(\mathsf{fro}_{\sigma_1[\rho_\theta]}((\varepsilon\, t_1)/_\simeq)) = \mathsf{froU}_{\langle\sigma_2\rangle(\theta)}(\mathsf{fro}_{\sigma_2[\rho_\theta]}((\varepsilon\, t_2)/_\simeq))$, for any t_1 and t_2 such that

$t_1/_\simeq = \mathsf{to}_{\sigma_1[\rho_\theta]\Rightarrow\mathsf{bool}}(p_1 \circ \mathsf{froU}_{\langle\sigma_1\rangle(\theta)})$ and $t_2/_\simeq = \mathsf{to}_{\sigma_2[\rho_\theta]\Rightarrow\mathsf{bool}}$ $(p_2 \circ \mathsf{froU}_{\langle\sigma_2\rangle(\theta)})$. We faced a similar situation when proving the definition of $\mathcal{V}$ unambiguous, and the solution was to "normalize" via the common superset $A = \mathsf{Super}(A_1) = \mathsf{Super}(A_2)$ and show that either comprehension (from A_1 and A_2) yields the same result as comprehension from A.

This suggests that, when defining $M_\varepsilon \in \prod_{A\in\mathcal{V}} A$, we again take the superset as reference point for comprehension. So for all $A \in \mathcal{V}$, we define $M_{\varepsilon,A} = \mathsf{froU}_{\mathsf{Super}(A)}(\mathsf{fro}_{\sigma'}((\varepsilon\, s)_{/\simeq}))$, where σ' is the unique ground type such that $\mathsf{Proxy}(\mathsf{Super}(A)) = U_{\sigma'}$, and $s : \sigma' \Rightarrow \mathsf{bool}$ and $r : \mathsf{Super}(A) \to U_{\mathsf{bool}}$ are such that $A = \{a \in \mathsf{Super}(A) \mid r(a) = \top\}$ and $s_{/\simeq} = \mathsf{to}_{\sigma'\Rightarrow\mathsf{bool}}(r \circ \mathsf{froU}_{\mathsf{Super}(A)})$. Thus, in our model, the act of choosing an element of a set $A \in \mathcal{V}$ is delegated to the proxy of the maximal superset of A, $\mathsf{Proxy}(\mathsf{Super}(A))$, where a syntactic choice is being made via the syntactic counterpart s of the indicator r of A within $\mathsf{Super}(A)$; and then copied back to $\mathsf{Super}(A)$ via the proxy infrastructure. The result is guaranteed to be not only in $\mathsf{Super}(A)$ but also in A, because it satisfies r (courtesy of using its syntactic counterpart s).

With this definition, the desired equality (*) becomes $\mathsf{froU}_C(\mathsf{fro}_{\sigma'}((\varepsilon\, s)_{/\simeq})) = \mathsf{froU}_{\langle\sigma\rangle(\theta)}(\mathsf{fro}_{\sigma[\rho_\theta]}((\varepsilon\, t)/_\simeq))$ where

- the left side uses the indicator of B in $C = \mathsf{Super}(\langle\sigma\rangle(\theta)) = \mathsf{Super}(B)$, i.e.: $B = \{c \in C \mid r(c) = \top\}$, and $s_{/\simeq} = \mathsf{to}_{\sigma'\Rightarrow\mathsf{bool}}(r \circ \mathsf{froU}_C)$, where $\mathsf{Proxy}(C) = U_{\sigma'}$;
- the right side uses the indicator of B in $\langle\sigma\rangle(\theta)$, i.e.: $B = \{a \in \langle\sigma\rangle(\theta) \mid p(a) = \top\}$ and $t/_\simeq = \mathsf{to}_{\sigma[\rho_\theta]\Rightarrow\mathsf{bool}}(p \circ \mathsf{froU}_{\langle\sigma\rangle(\theta)})$.

Let $q \in \mathsf{Fun}'(C, U_{\mathsf{bool}})$ be the indicator of $\langle\sigma\rangle(\theta)$ in C, and $u : \sigma' \Rightarrow \mathsf{bool}$ its syntactic counterpart, in that $u/_\simeq = \mathsf{to}_{\sigma'\Rightarrow\mathsf{bool}}(q \circ \mathsf{froU}_C)$. By the definition of $\mathcal{V}$ and the type interpretation lemma, we have that $\tau_{\sigma',u} = \sigma[\rho_\theta]$ and $\mathsf{Proxy}(\langle\sigma\rangle(\theta)) = U_{\tau_{\sigma',u}} = U_{\sigma[\rho_\theta]}$, yielding the situation depicted in Fig. 5. Chasing the figure's commutative diagram with vertices $\mathsf{Proxy}(\langle\sigma\rangle(\theta)), \mathsf{Proxy}(C), C, \langle\sigma\rangle(\theta)$ and using that froU_C is injective, the desired equality reduces to $\mathsf{fro}_{\sigma'}((\varepsilon\, s)/_\simeq) = \mathsf{fro}_{\sigma[\rho_\theta]\Rightarrow\sigma'}(\mathsf{Rep}_{\sigma,u/_\simeq})\,(\mathsf{fro}_{\sigma[\rho_\theta]}((\varepsilon\, t)/_\simeq))$; hence, by the definition of $\mathsf{fro}_{\sigma[\rho_\theta]\Rightarrow\sigma'}$, to $\mathsf{fro}_{\sigma'}(\varepsilon\, s)/_\simeq = \mathsf{fro}_{\sigma'}((\mathsf{Rep}_{\sigma,u}\,(\varepsilon\, t))/_\simeq))$; hence, by the injectiveness of $\mathsf{fro}_{\sigma'}$ and the definition of $\simeq$, to $T \vdash \mathsf{Rep}_{\sigma',u}\,(\varepsilon\, t) = \varepsilon\, s$. Moreover, because s is T-provably equal to $u \wedge_2 (t \circ \mathsf{Abs}_{\sigma,t})$ (since, on the semantic counterpart level, r is the conjunction of q and p), this further reduces to $T \vdash \mathsf{Rep}_{\sigma',u}\,(\varepsilon\, t) = \varepsilon\,(u \wedge_2 (t \circ \mathsf{Abs}_{\sigma',u}))$. Thus, going back to our Hintikka conditions, we add one more.

Uniform Choice: $T \vdash \mathsf{Rep}_{\sigma,t}\,(\varepsilon\, s) = \varepsilon\,(t \wedge_2 (s \circ \mathsf{Abs}_{\sigma,t}))$ if σ is a ground type, $t : \sigma \Rightarrow \mathsf{bool}$ and $s : \tau_{\sigma,t} \Rightarrow \mathsf{bool}$ are ground closed terms, $T \vdash \exists x_\sigma.\, t\, x$ and $T \vdash \exists y_{\tau_\sigma}.\, s\, y$.

This condition says that the behavior of ε on the type $\tau_{\sigma,t}$ is uniform along the translation $\mathsf{Rep}_{\sigma,t}$ in that, for any predicate $s : \tau_{\sigma,t} \Rightarrow \mathsf{bool}$ with nonempty extent: **(1)** making a choice for (an element satisfying) s in $\tau_{\sigma,t}$ and translating it via $\mathsf{Rep}_{\sigma,t}$, is T-provably the same as **(2)** making a choice for (an element satisfying) its counterpart $t \wedge_2 (s \circ \mathsf{Abs}_{\sigma,t})$ on σ. Essentially, Uniform Choice allows our syntactically constructed model to comply with the semantic feature of Pitts-Henkin models which we called choice uniformity: that choice is performed on each set, and not on each combination of set and predicate on it separately.

Case 2: Assume p has empty extent, i.e., is vacuously $\bot$. We must prove $M_{\varepsilon,\langle\sigma\rangle(\theta)} = \mathsf{froU}_{\langle\sigma\rangle(\theta)}(\mathsf{fro}_{\sigma[\rho_\theta]}((\varepsilon\, t)/_\simeq))$, i.e., $\mathsf{froU}_{\mathsf{Super}(\langle\sigma\rangle(\theta))}(\mathsf{fro}_{\sigma'}((\varepsilon\, s)_{/\simeq})) = \mathsf{froU}_{\langle\sigma\rangle(\theta)}(\mathsf{fro}_{\sigma[\rho_\theta]}((\varepsilon\, t)/_\simeq))$, where $t_{/\simeq} = \mathsf{to}_{\sigma[\rho_\theta]\Rightarrow\mathsf{bool}}(p \circ \mathsf{froU}_{\langle\sigma\rangle(\theta)}) = (\lambda x_{\sigma[\rho_\theta]}.\, \mathsf{false})_{/\simeq}$, $\mathsf{Proxy}(\mathsf{Super}(\langle\sigma\rangle(\theta))) = U_{\sigma'}$, q is the indicator of $\langle\sigma\rangle(\theta)$ in $\mathsf{Super}(\langle\sigma\rangle(\theta))$ and $s_{/\simeq} = \mathsf{to}_{\sigma'\Rightarrow\mathsf{bool}}(q \circ \mathsf{froU}_{\mathsf{Super}(\langle\sigma\rangle(\theta))})$.

By Uniform Choice, $\mathsf{froU}_{\mathsf{Super}(\langle\sigma\rangle(\theta))}(\mathsf{fro}_{\sigma'}((\varepsilon\, s)_{/\simeq})) = \mathsf{froU}_{\langle\sigma\rangle(\theta)}(\mathsf{fro}_{\sigma[\rho_\theta]}((\varepsilon\,(\lambda x_{\sigma[\rho_\theta]}.\, \mathsf{true}))/_\simeq))$, so the desired fact becomes $\mathsf{froU}_{\langle\sigma\rangle(\theta)}(\mathsf{fro}_{\sigma[\rho_\theta]}((\varepsilon\,(\lambda x_{\sigma[\rho_\theta]}.\, \mathsf{true}))_{/\simeq})) = \mathsf{froU}_{\langle\sigma\rangle(\theta)}$ $(\mathsf{fro}_{\sigma[\rho_\theta]}((\varepsilon\,(\lambda x_{\sigma[\rho_\theta]}.\, \mathsf{false}))/_\simeq))$, i.e., $T \vdash \varepsilon\,(\lambda x_{\sigma[\rho_\theta]}.\, \mathsf{false}) = \varepsilon\,(\lambda x_{\sigma[\rho_\theta]}.\, \mathsf{true})$. Thus we need the following.

Trivial Choice property: $T \vdash \varepsilon\,(\lambda x_\alpha.\, \mathsf{false}) = \varepsilon\,(\lambda x_\alpha.\, \mathsf{true})$.

We call this "Trivial Choice" because it equates the applications of the choice constant ε on the two trivial extremes. For the vacuously true predicate, it is expected that ε simply returns an element about which we only know that the predicate holds, which is to say we do not know anything. But how about for a vacuously false predicate? Since HOL is a total logic, in that case ε has no choice but to also return an unknown element of the given type. So in both cases ε returns a completely unknown item, though for different reasons. Trivial Choice requires that these two "unknowns" be equated. It reflects the design decision underlying Pitts (hence Pitts-Henkin) models of interpreting impossible choice as choice on the whole type. And indeed, all Pitts-Henkin models validate this equation.

Thus, assuming Uniform Choice and Trivial Choice allows us to prove the term interpretation lemma, finishing Step 3.

D. Constructing a Hintikka theory extension

The only remaining part of the completeness proof is Step 2: showing that any consistent theory T admits an (also consistent) Hintikka theory extension T' (possibly over an extended signature). We need T' to satisfy the usual Hintikka conditions (allowing "model-like" behavior w.r.t. connectives and quantifiers); and additionally, as we discovered in §IV-C, to guarantee the existence of certain ground types $\tau_{\sigma,t}$ that witness all the extents of ground closed predicates $t : \sigma \Rightarrow \mathsf{bool}$, via ground closed terms $\mathsf{Rep}_{\sigma,t} : \tau_{\sigma,t} \Rightarrow \sigma$ and $\mathsf{Abs}_{\sigma,t} : \sigma \Rightarrow \tau_{\sigma,t}$ such that $\mathsf{Rep}_{\sigma,t}$ is a T-provable bijection between σ and the extent of t in σ, i.e., $T \vdash \mathsf{tdef}_{\tau_{\sigma,t},\sigma,t,\mathsf{Rep}_{\sigma,t}}$, and $\mathsf{Abs}_{\sigma,t}$ is its inverse. These must satisfy the Abstractness, Identity, Compositionality and Uniform Choice properties identified during our model construction exploration.

But why should a consistent theory T admit a consistent extension that proves the above bijections? What if, to the contrary, for *some* σ and t, T proves $\neg\, \mathsf{tdef}_{\tau,\sigma,t,\mathsf{Rep}}$ for *all* τ and Rep? For example, let T be $\{\varphi\}$ where φ is (yet again) the α-polymorphic formula φ saying that α is not a singleton, and let σ be bool and t be $\lambda x_{\mathsf{bool}}.\, x = \mathsf{true}$. Then $T \vdash \neg\, \mathsf{tdef}_{\tau,\sigma,t,\mathsf{Rep}}$ for all τ and Rep because $\mathsf{tdef}_{\tau,\sigma,t,\mathsf{Rep}}$ implies that τ is a singleton.

A generic way to exclude counterexamples as above would be postulating the *existence* of such types and representation functions for any σ and $t : \sigma \Rightarrow \mathsf{bool}$, i.e., adding an axiom

$$(\exists x_\sigma.\, t\, x) \longrightarrow (\exists\alpha.\, \exists rep_{\alpha\Rightarrow\sigma}.\, \mathsf{tdef}_{\alpha,\sigma,t,rep})$$

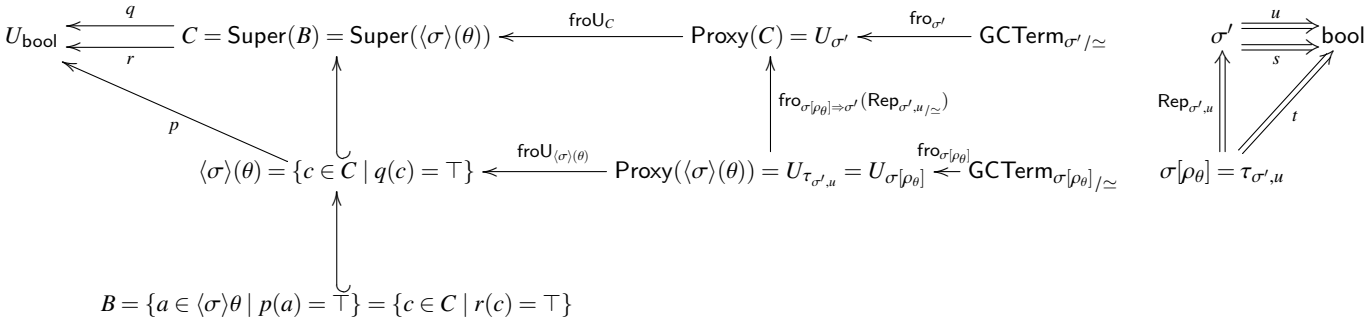

Fig. 5. The items from the proof of the term-interpretation lemma, the case of the ε constant and nonempty extent of the argument predicate.

However, existential quantification over types is not supported in HOL, and adding such support would take us too far afield, outside rank-1 polymorphism (a restriction of HOL celebrated for its simplicity). A less intrusive solution comes from the work of Kunçar and Popescu [57], who wanted the ability to assume the existence of types corresponding to nonempty predicate-definable subsets on a dynamic basis, inside proof contexts. They noticed that the effect of the above existential quantification over types can be achieved with the addition of a new rule, called Local Typedef, seen in Fig. 6 when ignoring the parts highlighted in gray. The rule has the following intuition, when applied backwards: While in a proof context with assumptions Γ and desired conclusion φ, given any nonempty-extent predicate t on some type σ, we are allowed to assume (that there exists) a type α which is in bijection with the extent of t.

The Local Typedef rule has the potential of helping us construct the Hintikka theory extension along the following lines. We would incrementally extend the signature with fresh types $\tau_{\sigma,t}$ for each σ and t such that $\exists x_\sigma.\ t\ x$ is inferrable, and postulate the existence of a witnessing bijection—while counting on Local Typedef to guarantee consistency of the extension. As for the other Hinitikka conditions about the witness types (besides bijectiveness), it will turn out that Witness Abstractness, Identity and Compositionality can be "mended" after the fact. On the other hand, Uniform Choice must be addressed early: Since this property requires that the behavior of ε on $\tau_{\sigma,t}$ be consistent (via the chosen bijection $\mathsf{Rep}_{\sigma,t}$) with its behavior on the larger type σ, we must ensure that in Local Typedef the existentially quantified *rep* satisfies this. So we strengthen Local Typedef as highlighted in Fig. 6, obtaining what we call Strong Local Typedef. It additionally requires choice uniformity for *rep*, expressed by the following formula which we denote by $\mathsf{chUni}_{\alpha,\sigma,t,rep}$:

$$\forall s_{\alpha\Rightarrow\mathsf{bool}}.\ rep\ (\varepsilon\ s) = \varepsilon\ (\lambda x_\sigma.\ \exists y_\alpha.\ rep\ y = x \wedge s\ y)$$

Note that $T \vdash (t \wedge_2 (s \circ \mathsf{Abs}_{\sigma,t})) = (\lambda x_\sigma.\ \exists y_{\tau_{\sigma,t}}.\ \mathsf{Rep}_{\sigma,t}\ y = x \wedge s\ y)$, so the conclusion of the Uniform Choice condition is equivalent to $T \vdash \mathsf{chUni}_{\tau_{\sigma,t},\sigma,t,\mathsf{Rep}_{\sigma,t}}$ (taking α and *rep* to be $\tau_{\sigma,t}$ and $\mathsf{Rep}_{\sigma,t}$). Overall:

We enrich HOL deduction with the Strong Local Typedef rule and the Trivial Choice axiom $\varepsilon\ (\lambda x_\alpha.\ \mathsf{false}) = \varepsilon\ (\lambda x_\alpha.\ \mathsf{true})$.

This allows us to complete Step 2 as sketched above. (The proof of Prop. 19 in App. G from [76] gives details.)

Summary. To obtain completeness w.r.t. the Pitts-Henkin models, we enriched HOL deduction with a rule for introducing new types in arbitrary proof contexts in a choice-uniform manner, and an axiom for handling vacuous choice. After reducing the problem to ground-conclusion completeness, we took the route prescribed by Henkin: extending any consistent theory to a Hintikka theory, and showing that any Hintikka theory has a model. Custom Hintikka conditions referring to witness types for nonempty extents have been extracted from the challenges of ensuring that the constructed model is downward closed and admits uniform choice. Our model's universe is the downward closure of a syntactic universe, connected to it via a proxy infratructure for copying its behavior. Consequently, our interpretation lemmas that connect substitution with semantic valuations, and ultimately show that the model satisfies the theory, proceed "via proxy".

Here is the completeness result in final form. Let us call the extension of HOL deduction to incorporate the Strong Local Typedef rule and the Trivial Choice axiom *enriched HOL deduction*, and denote it by $\Vdash$.

Thm 9: Enriched HOL deduction is complete w.r.t. the Pitts-Henkin models (i.e., for all signatures Σ, and theories T and formulas φ over Σ, we have that $T \models_\Sigma \varphi$ implies $T \Vdash_\Sigma \varphi$).

While completeness can be regarded as a largely theoretical goal, it is worth asking whether shifting from the original HOL deduction ($\vdash$) to the enriched one ($\Vdash$) would bring any practical improvements. As discussed, (S-L-TYPEDEF) is a stronger form of a previously proposed rule [57], (L-TYPEDEF)—which is a useful addition to HOL.

For example, consider the following formula expressing the functoriality of the map operator on lists:

$$\forall xs_{\alpha\ \mathsf{list}},\ f_{\alpha\Rightarrow\beta},\ \forall g_{\beta\Rightarrow\gamma}.\ \mathsf{map}\ (g \circ f)\ xs = \mathsf{map}\ g\ (\mathsf{map}\ f\ xs)$$

Often, a HOL proof developer needs a more flexible *set-relativized* version, which quantifies universally over sets (subsets of the types expressed by the type variables):

$$\forall A_{\alpha\ \mathsf{set}},\ B_{\beta\ \mathsf{set}},\ C_{\gamma\ \mathsf{set}}.\ \forall xs_{\alpha\ \mathsf{list}},\ f_{\alpha\Rightarrow\beta},\ g_{\beta\Rightarrow\gamma}.\ xs \in \mathsf{lists}\ A \wedge \mathsf{image}\ f\ A \subseteq B \wedge \mathsf{image}\ g\ B \subseteq C \longrightarrow \mathsf{map}\ (g \circ f)\ xs = \mathsf{map}\ g\ (\mathsf{map}\ f\ xs)$$

where $\mathsf{lists}\ A$ denotes the set of lists whose elements are all in A. The latter set-relativized version does not follow from the former "type-only" version using standard HOL deduc-

$$\frac{T;\Gamma\vdash\exists x_\sigma.\, t\; x_\sigma \qquad T;\Gamma\vdash(\exists rep_{\alpha\Rightarrow\sigma}.\,\mathsf{tdef}_{\alpha,\sigma,t,rep}\wedge\mathsf{chUni}_{\alpha,\sigma,t,rep})\longrightarrow\varphi}{T;\Gamma\vdash\varphi}\ \begin{array}{l}(\text{S-L-TYPEDEF})\\ [\alpha \text{ fresh for } t,\varphi,\Gamma]\end{array}$$

Fig. 6. The Strong Local Typedef rule

tion [59], but would follow with the help of (L-TYPEDEF). In general, (L-TYPEDEF) allows to relativize to sets statements for which there is sufficiently uniform infrastructure available—e.g., an operator such as lists for the list type constructor, and parametricity-like [79], [88] properties for the constants involved). Our stronger version of this rule, (S-L-TYPEDEF), guaranteeing consistency of choice across the typedef embedding, enables a local parametricity property for the otherwise notoriously non-parametric choice constant—with the potential of enlarging the scope of parametricity-based automatic transfer tools for HOL [50], [60], [64], [77], for which concepts defined using choice are a known bottleneck. (L-TYPEDEF) and (S-L-TYPEDEF) also have the potential of simplifying definitional packages for HOL provers [11], [62], [63], [39], [87], [16], [17], where with standard HOL deduction one often needs to maintain flexible, set-based axiomatizations in order to perform the necessary constructions. For example, Isabelle/HOL's current (co)datatype and corecursive function packages [16], [15] are based on bounded natural functors [85], [18], [19] (a HOL adaptation of accessible functors [1]), whose set-relativized axiomatization is necessary for constructing the required initial algebras and final coalgebras; simplifying this to a type-only axiomatization would lead to a sizable reduction of its code base.

V. RELATED WORK

Henkin's proofs of completeness for Simple Type Theory [46] and for FOL [45] (interestingly, the *former* having inspired the latter not vice versa [47]), led to a method that has been adapted to many logics (including modal [14], intuitionistic [53] and fuzzy [41]), with some of its high-level ideas also incorporated in abstract logic-independent frameworks [72], [56]. A variation of Simple Type Theory, featured e.g. in Andrews' Q_0 system [3], uses instead of the axiom of choice the weaker axiom of definite description, which only guarantees the correct behavior of the choice operator on predicates with singleton extents; this requires a slightly more complex proof of completeness, since extensionality of the syntactic model is no longer guaranteed unless a property called extensional completeness is included in the Hintikka conditions. What make our own adaptation of Henkin's method non-trivial are the polymorphism, downward closedness and choice uniformity features of the standard HOL semantics, none of which are present in Simple Type Theory.

As part of an effort to convert a model-theoretic into a proof-theoretic conservativity result for the Isabelle/HOl dialect of HOL, Gengelbach and Weber [27] proved completeness for standard HOL deduction $\vdash$ w.r.t. so-called *ground models* [57], which are essentially Henkin's original general models for Simple Type Theory adapted to MHOL (in particular, featuring no downward closedness or choice uniformity), with satisfaction defined as the MHOL satisfaction of all monomorphic instances—so that polymorphism is treated syntactically, as a form of infinitary universal quantification. By contrast, our completeness results refer to models that interpret polymorphism in line with the standard HOL semantics, in a native set-theoretical way, via dependent products over the considered universe.

In the area of programming language semantics, Henkin-style models and completeness theorems have been studied for systems with higher-rank or impredicative polymorphism, notably Girard and Reynold's System F [29], [78], [89], and Mitchell's Higher-Order Lambda Calculus generalization [66]. These systems feature a purely equational theory, without any (primitive or derived) deduction rules for first-order connectives or quantifiers. Consequently, their completeness theorems [22], [66], while also relying on syntactic models involving equivalence classes of terms, follow not Henkin's but Birkhoff's method for proving the completeness of equational logic [13]—initially adapted to handle higher-order features by Friedman [26] for the Simply-Typed Lambda Calculus (which incidentally, for its pure β-η equational theory, offers the pleasant exception that completeness holds not only for Henkin-style models, but also for the standard model when starting with an infinite base type). By contrast, standard set-theoretic models (with dependent products) do not even exist for System F [80], unless one abandons classical in favor of intuitionistic logic [75]. Outside the realm of purely set-theoretic semantics, domain-theoretic and categorical models have a rich tradition in the study of λ-calculi (System F included) [81], [7], [30] and some completeness theorems have been provided there, e.g., [8], [49], [75].

Unlike Henkin's, Birkhoff's method does not require maximally consistent (Hintikka-style) extensions, but it only applies to equational theories and variations such as Horn theories. (And indeed, Birkhoff-style completeness proofs have been performed not only for λ-calculi equational theories, but for their typing judgements too [48], [91], essentially because these follow a Horn-like format.) Even putting the connective/quantifier dimension aside, our completeness proof for Pitts-Henkin models requires a Hintikka extension anyway, in order to fill all downward closedness gaps with witness types.

To account for downward closedness, our Hintikka conditions on the witness representation functions enforce coherent inclusion-like behavior going beyond that of monomorphisms, resembling inclusion systems [25], [24].

Acknowledgments. We thank the reviewers for their thoughtful suggestions, including that of emphasizing the potential practical interest of our proposed deduction enrichment, independently of completeness. Work supported by EPSRC grants EP/V039156/1 (Security of Digital Twins in Manufacturing) and EP/X015114/1 (Safe and secure COncurrent programming for adVancEd aRchiTectures).

References

[1] Adamek, J., Rosicky, J.: Locally Presentable and Accessible Categories. London Mathematical Society Lecture Note Series, Cambridge University Press (1994)

[2] Adams, M.: Introducing HOL Zero (extended abstract). In: Fukuda, K., van der Hoeven, J., Joswig, M., Takayama, N. (eds.) ICMS 2010. LNCS, vol. 6327, pp. 142–143. Springer (2010)

[3] Andrews, P.B.: An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof. Applied Logic Series, Springer (2013)

[4] Arthan, R.: On definitions of constants and types in HOL. J. Autom. Reason. 56(3), 205–219 (2016)

[5] Arthan, R.D., Jones, R.B.: Z in HOL in ProofPower. In: The Newsletter of the Formal Aspects of Computing Science (FACS) Specialist Group (2005), https://web.archive.org/web/20221014122152/https://www.bcs.org/media/3096/facts200503.pdf

[6] Asperti, A., Ricciotti, W., Coen, C.S., Tassi, E.: The Matita interactive theorem prover. In: Bjørner, N.S., Sofronie-Stokkermans, V. (eds.) CADE-23. LNCS, vol. 6803, pp. 64–69. Springer (2011)

[7] Barendregt, H.P.: The lambda calculus - its syntax and semantics, Studies in logic and the foundations of mathematics, vol. 103. North-Holland (1985)

[8] Barendregt, H., Coppo, M., Dezani-Ciancaglini, M.: A filter lambda model and the completeness of type assignment. J. Symb. Log. 48(4), 931–940 (1983), https://doi.org/10.2307/2273659

[9] Bell, J.L., Machover, M.: A course in mathematical logic. North-Holland (1977)

[10] Benzmüller, C., Andrews, P.: Church's Type Theory. In: Zalta, E.N., Nodelman, U. (eds.) The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University, Spring 2024 edn. (2024)

[11] Berghofer, S., Wenzel, M.: Inductive datatypes in HOL - lessons learned in formal-logic engineering. In: Bertot, Y., Dowek, G., Hirschowitz, A., Paulin-Mohring, C., Théry, L. (eds.) Theorem Proving in Higher Order Logics, 12th International Conference, TPHOLs'99, Nice, France, September, 1999, Proceedings. LNCS, vol. 1690, pp. 19–36. Springer (1999)

[12] Bertot, Y., Castéran, P.: Interactive Theorem Proving and Program Development - Coq'Art: The Calculus of Inductive Constructions. Texts in Theoretical Computer Science. An EATCS Series, Springer (2004)

[13] Birkhoff, G.: On the structure of abstract algebras. Mathematical Proceedings of the Cambridge Philosophical Society 31(4), 433–454 (1935)

[14] Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic, Cambridge Tracts in Theoretical Computer Science, vol. 53. Cambridge University Press (2001), https://doi.org/10.1017/CBO9781107050884

[15] Blanchette, J.C., Bouzy, A., Lochbihler, A., Popescu, A., Traytel, D.: Friends with benefits - implementing corecursion in foundational proof assistants. In: Yang, H. (ed.) Programming Languages and Systems - 26th European Symposium on Programming, ESOP 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings. LNCS, vol. 10201, pp. 111–140. Springer (2017)

[16] Blanchette, J.C., Hölzl, J., Lochbihler, A., Panny, L., Popescu, A., Traytel, D.: Truly modular (co)datatypes for isabelle/hol. In: Klein, G., Gamboa, R. (eds.) 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. LNCS, vol. 8558, pp. 93–110. Springer (2014)

[17] Blanchette, J.C., Meier, F., Popescu, A., Traytel, D.: Foundational nonuniform (co)datatypes for higher-order logic. In: 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017. pp. 1–12. IEEE Computer Society (2017)

[18] Blanchette, J.C., Popescu, A., Traytel, D.: Foundational extensible corecursion: a proof assistant perspective. In: Fisher, K., Reppy, J.H. (eds.) Proceedings of the 20th ACM SIGPLAN International Conference on Functional Programming, ICFP 2015, Vancouver, BC, Canada, September 1-3, 2015. pp. 192–204. ACM (2015)

[19] Blanchette, J.C., Popescu, A., Traytel, D.: Witnessing (co)datatypes. In: Vitek, J. (ed.) Programming Languages and Systems - 24th European Symposium on Programming, ESOP 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings. LNCS, vol. 9032, pp. 359–382. Springer (2015)

[20] Bortin, M., Johnsen, E.B., Lüth, C.: Structured formal development in Isabelle. Nord. J. Comput. 13(1-2), 2–21 (2006)

[21] Bove, A., Dybjer, P., Norell, U.: A brief overview of Agda – a functional language with dependent types. In: Berghofer, S., Nipkow, T., Urban, C., Wenzel, M. (eds.) TPHOLs 2009. LNCS, vol. 5674, pp. 73–78. Springer (2009)

[22] Bruce, K.B., Meyer, A.R., Mitchell, J.C.: The semantics of second-order lambda calculus. Inf. Comput. 85(1), 76–134 (1990), https://doi.org/10.1016/0890-5401(90)90044-I

[23] Church, A.: A Formulation of the Simple Theory of Types. The Journal of Symbolic Logic 5(2), 56–68 (1940)

[24] Căzănescu, V.E., Roşu, G.: Weak inclusion systems. Math. Struct. Comput. Sci. 7(2), 195–206 (1997), https://doi.org/10.1017/S0960129596002253

[25] Diaconescu, R., Goguen, J., Stefaneas, P.: Logical support for modularization. In: Huet, G., Plotkin, G. (eds.) Logical Environments, pp. 83–130. Cambridge (1993)

[26] Friedman, H.: Equality between functionals. In: Parikh, R. (ed.) Logic Colloquium. pp. 22–37. Springer Berlin Heidelberg, Berlin, Heidelberg (1975)

[27] Gengelbach, A., Weber, T.: Proof-theoretic conservative extension of HOL with ad-hoc overloading. In: Pun, V.K.I., Stolz, V., Simão, A. (eds.) Theoretical Aspects of Computing - ICTAC 2020 - 17th International Colloquium, Macau, China, November 30 - December 4, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12545, pp. 23–42. Springer (2020), https://doi.org/10.1007/978-3-030-64276-1_2

[28] Geuvers, H.: Proof assistants: History, ideas and future. Sadhana 34(1), 3–25 (2009)

[29] Girard, J.Y.: Une extension de l'interpretation de Gödel a l'analyse, et son application a l'elimination des coupure dans l'analyse et la theorie des types. In: 2nd Scandinavian Logic Symposium. pp. 63–92 (1971)

[30] Girard, J.Y., Lafont, Y., Taylor, P.: Proofs and Types. No. 7 in Cambridge Tracts in Theoretical Computer Science, Cambridge University Press (1989)

[31] Gödel, K.: Über die Vollständigkeit des Logikkalküls. Ph.D. thesis, University Of Vienna (1929)

[32] Gödel, K.: Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. Monatshefte für Mathematik und Physik 38(1), 173–198 (1931)

[33] Gonthier, G.: The four colour theorem: Engineering of a formal proof. In: Kapur, D. (ed.) ASCM 2007. LNCS, vol. 5081, p. 333. Springer (2007)

[34] Gonthier, G., Asperti, A., Avigad, J., Bertot, Y., Cohen, C., Garillot, F., Roux, S.L., Mahboubi, A., O'Connor, R., Biha, S.O., Pasca, I., Rideau, L., Solovyev, A., Tassi, E., Théry, L.: A machine-checked proof of the odd order theorem. In: Blazy, S., Paulin-Mohring, C., Pichardie, D. (eds.) Interactive Theorem Proving - 4th International Conference, ITP 2013, Rennes, France, July 22-26, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7998, pp. 163–179. Springer (2013), https://doi.org/10.1007/978-3-642-39634-2_14

[35] Gordon, M.J.C.: Introduction to the HOL system. In: Archer, M., Joyce, J.J., Levitt, K.N., Windley, P.J. (eds.) Proceedings of the 1991 International Workshop on the HOL Theorem Proving System and its Applications, August 1991, Davis, California, USA. pp. 2–3. IEEE Computer Society (1991)

[36] Gordon, M.J.C., Melham, T.F. (eds.): Introduction to HOL: A theorem proving environment for higher order logic. Cambridge University Press (1993), http://www.cs.ox.ac.uk/tom.melham/pub/Gordon-1993-ITH.html

[37] Gordon, M.: From LCF to HOL: a short history. In: Plotkin, G.D., Stirling, C., Tofte, M. (eds.) Proof, Language, and Interaction, Essays in Honour of Robin Milner. pp. 169–186. The MIT Press (2000)

[38] Gordon, M.: Twenty years of theorem proving for hols past, present and future. In: Mohamed, O.A., Muñoz, C.A., Tahar, S. (eds.) 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, pp. 1–5. Springer (2008), https://doi.org/10.1007/978-3-540-71067-7_1

[39] Gunter, E.L.: A broader class of trees for recursive type definitions for HOL. In: Joyce, J.J., Seger, C.H. (eds.) Higher Order Logic Theorem Proving and its Applications, 6th International Workshop, HUG '93, Vancouver, BC, Canada, August 11-13, 1993, Proceedings. LNCS, vol. 780, pp. 141–154. Springer (1993)

[40] Haftmann, F., Wenzel, M.: Constructive type classes in Isabelle. In: Altenkirch, T., McBride, C. (eds.) Types for Proofs and Programs, International Workshop, TYPES 2006, Nottingham, UK, April 18-21, 2006, Revised Selected Papers. LNCS, vol. 4502, pp. 160–174. Springer (2006)

[41] Hájek, P.: Metamathematics of Fuzzy Logic, Trends in Logic, vol. 4. Kluwer (1998), https://doi.org/10.1007/978-94-011-5300-3

[42] Hales, T.C., Adams, M., Bauer, G., Dang, D.T., Harrison, J., Hoang, T.L., Kaliszyk, C., Magron, V., McLaughlin, S., Nguyen, T.T., Nguyen, T.Q., Nipkow, T., Obua, S., Pleso, J., Rute, J.M., Solovyev, A., Ta, A.H.T., Tran, T.N., Trieu, D.T., Urban, J., Vu, K.K., Zumkeller, R.: A formal proof of the Kepler conjecture. CoRR abs/1501.02155 (2015), http://arxiv.org/abs/1501.02155

[43] Harrison, J.: HOL Light: A tutorial introduction. In: Srivas, M.K., Camilleri, A.J. (eds.) FMCAD 1996. LNCS, vol. 1166, pp. 265–269. Springer (1996)

[44] Harrison, J.: Towards self-verification of HOL Light. In: Furbach, U., Shankar, N. (eds.) Automated Reasoning, Third International Joint Conference, IJCAR 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4130, pp. 177–191. Springer (2006), https://doi.org/10.1007/11814771_17

[45] Henkin, L.: The completeness of the first-order functional calculus. J. Symb. Log. 14(3), 159–166 (1949), https://doi.org/10.2307/2267044

[46] Henkin, L.: Completeness in the theory of types. J. Symb. Log. 15(2), 81–91 (1950), https://doi.org/10.2307/2266967

[47] Henkin, L.: The discovery of my completeness proofs. Bull. Symb. Log. 2(2), 127–158 (1996), https://doi.org/10.2307/421107

[48] Hindley, J.R.: The completeness theorem for typing lambda-terms. Theor. Comput. Sci. 22, 1–17 (1983), https://doi.org/10.1016/0304-3975(83)90136-6

[49] Honsell, F., Plotkin, G.D.: On the completeness of order-theoretic models of the lambda-calculus. Inf. Comput. 207(5), 583–594 (2009), https://doi.org/10.1016/j.ic.2008.03.027

[50] Huffman, B., Kuncar, O.: Lifting and transfer: A modular design for quotients in Isabelle/HOL. In: Gonthier, G., Norrish, M. (eds.) Certified Programs and Proofs - Third International Conference, CPP 2013, Melbourne, VIC, Australia, December 11-13, 2013, Proceedings. LNCS, vol. 8307, pp. 131–146. Springer (2013)

[51] Johnstone, P.T.: Topos theory. Journal of Symbolic Logic 47(2), 448–450 (1982)

[52] Klein, G., Andronick, J., Elphinstone, K., Heiser, G., Cock, D., Derrin, P., Elkaduwe, D., Engelhardt, K., Kolanski, R., Norrish, M., Sewell, T., Tuch, H., Winwood, S.: seL4: formal verification of an operating-system kernel. Commun. ACM 53(6), 107–115 (2010)

[53] Kripke, S.A.: Semantical Analysis of Intuitionistic Logic I. In: Dummett, M., Crossley, J.N. (eds.) Formal Systems and Recursive Functions, pp. 92–130. North Holland (1963)

[54] Kumar, R., Arthan, R., Myreen, M.O., Owens, S.: HOL with definitions: Semantics, soundness, and a verified implementation. In: Klein, G., Gamboa, R. (eds.) 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, pp. 308–324. Springer (2014), https://doi.org/10.1007/978-3-319-08970-6_20

[55] Kumar, R., Myreen, M.O., Norrish, M., Owens, S.: CakeML: a verified implementation of ML. In: Jagannathan, S., Sewell, P. (eds.) The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014. pp. 179–192. ACM (2014), https://doi.org/10.1145/2535838.2535841

[56] Kuncar, O.: An Isabelle/HOL framework for synthetic completeness proofs. In: Tabareau, N., Blazy, S. (eds.) Proceedings of the 2025 Conference on Certified Programs and Proofs, CPP 2025. To appear

[57] Kuncar, O., Popescu, A.: From types to sets by local type definitions in higher-order logic. In: Blanchette, J.C., Merz, S. (eds.) Interactive Theorem Proving - 7th International Conference, ITP 2016, Nancy, France, August 22-25, 2016, Proceedings. LNCS, vol. 9807, pp. 200–218. Springer (2016)

[58] Kuncar, O., Popescu, A.: A consistent foundation for Isabelle/HOL. J. Autom. Reason. 62(4), 531–555 (2019)

[59] Kuncar, O., Popescu, A.: From types to sets by local type definition in higher-order logic. J. Autom. Reason. 62(2), 237–260 (2019)

[60] Lammich, P.: Automatic data refinement. In: Blazy, S., Paulin-Mohring, C., Pichardie, D. (eds.) Interactive Theorem Proving - 4th International Conference, ITP 2013, Rennes, France, July 22-26, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7998, pp. 84–99. Springer (2013), https://doi.org/10.1007/978-3-642-39634-2_9

[61] Leroy, X.: Formal verification of a realistic compiler. Commun. ACM 52(7), 107–115 (2009)

[62] Melham, T.F.: Automating Recursive Type Definitions in Higher Order Logic, pp. 341–386. Springer (1989)

[63] Melham, T.F.: A package for inductive relation definitions in HOL. In: Archer, M., Joyce, J.J., Levitt, K.N., Windley, P.J. (eds.) Proceedings of the 1991 International Workshop on the HOL Theorem Proving System and its Applications, August 1991, Davis, California, USA. pp. 350–357. IEEE Computer Society (1991)

[64] Milehins, M.: An extension of the framework types-to-sets for Isabelle/HOL. In: Popescu, A., Zdancewic, S. (eds.) CPP '22: 11th ACM SIGPLAN International Conference on Certified Programs and Proofs, Philadelphia, PA, USA, January 17 - 18, 2022. pp. 180–196. ACM (2022)

[65] Milner, R.: LCF: A way of doing proofs with a machine. In: Becvár, J. (ed.) Mathematical Foundations of Computer Science 1979, Proceedings, 8th Symposium, Olomouc, Czechoslovakia, September 3-7, 1979. Lecture Notes in Computer Science, vol. 74, pp. 146–159. Springer (1979), https://doi.org/10.1007/3-540-09526-8_11

[66] Mitchell, J.C.: Lambda calculus models of typed programming languages. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, USA (1984), https://hdl.handle.net/1721.1/15394

[67] Monk, J.D.: Mathematical Logic. Springer (1976)

[68] de Moura, L.M., Kong, S., Avigad, J., van Doorn, F., von Raumer, J.: The Lean theorem prover (system description). In: Felty, A.P., Middeldorp, A. (eds.) CADE-25. LNCS, vol. 9195, pp. 378–388. Springer (2015)

[69] Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL - A Proof Assistant for Higher-Order Logic, LNCS, vol. 2283. Springer (2002)

[70] Obua, S.: Partizan games in Isabelle/HOLZF. In: Barkaoui, K., Cavalcanti, A., Cerone, A. (eds.) Theoretical Aspects of Computing - ICTAC 2006, Third International Colloquium, Tunis, Tunisia, November 20-24, 2006, Proceedings. LNCS, vol. 4281, pp. 272–286. Springer (2006)

[71] Paulson, L.C.: Zermelo Fraenkel Set Theory in Higher-Order Logic. Arch. Formal Proofs 2019 (2019), https://www.isa-afp.org/entries/ZFC_in_HOL.html

[72] Petria, M.: An institutional version of Gödel's completeness theorem. In: Mossakowski, T., Montanari, U., Haveraaen, M. (eds.) Algebra and Coalgebra in Computer Science, Second International Conference, CALCO 2007, Bergen, Norway, August 20-24, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4624, pp. 409–424. Springer (2007), https://doi.org/10.1007/978-3-540-73859-6_28

[73] Pitts, A.: Introduction to HOL: A theorem proving environment for higher order logic, chap. The HOL Logic, pp. 191–232. In: Gordon and Melham [36] (1993), http://www.cs.ox.ac.uk/tom.melham/pub/Gordon-1993-ITH.html

[74] Pitts, A.: The HOL System: Logic (1991), part of the HOL4 documentation. Available from http://sourceforge.net/projects/hol/files/hol/trindemossen-1/trindemossen-1-logic.pdf/download

[75] Pitts, A.M.: Polymorphism is set theoretic, constructively. In: Pitt, D.H., Poigné, A., Rydeheard, D.E. (eds.) Category Theory and Computer Science, Edinburgh, UK, September 7-9, 1987, Proceedings. Lecture Notes in Computer Science, vol. 283, pp. 12–39. Springer (1987), https://doi.org/10.1007/3-540-18508-9_18

[76] Popescu, A.: Completing Gordon's Higher-Order Logic – Extended Technical Report (2025), https://www.andreipopescu.uk/pdf/TRcompletingGordonHOL.pdf

[77] Popescu, A., Traytel, D.: Admissible types-to-PERs relativization in higher-order logic. Proc. ACM Program. Lang. 7(POPL), 1214–1245 (2023)

[78] Reynolds, J.C.: Towards a theory of type structure. In: Robinet, B.J. (ed.) Programming Symposium, Proceedings Colloque sur la Programmation, Paris, France, April 9-11, 1974. Lecture Notes in Computer Science, vol. 19, pp. 408–423. Springer (1974), https://doi.org/10.1007/3-540-06859-7_148

[79] Reynolds, J.C.: Types, abstraction and parametric polymorphism. In: Mason, R.E.A. (ed.) IFIP 1983, pp. 513–523. North-Holland/IFIP (1983)

[80] Reynolds, J.C.: Polymorphism is not set-theoretic. In: Kahn, G., MacQueen, D.B., Plotkin, G.D. (eds.) Semantics of Data Types, International Symposium, Sophia-Antipolis, France, June 27-29, 1984, Proceedings. Lecture Notes in Computer Science, vol. 173, pp. 145–156. Springer (1984), https://doi.org/10.1007/3-540-13346-1_7

[81] Scott, D.S.: Data types as lattices. SIAM J. Comput. 5(3), 522–587 (1976), https://doi.org/10.1137/0205037
[82] Shoenfield, J.R.: Mathematical Logic. Addison-Wesley, Reading, Mass., (1967)
[83] Slind, K., Norrish, M.: A brief overview of HOL4. In: Mohamed, O.A., Muñoz, C.A., Tahar, S. (eds.) TPHOLs 2008. LNCS, vol. 5170, pp. 28–32. Springer (2008)
[84] Smullyan, R.M.: First-Order Logic. Springer Verlag, New York [etc.] (1968)
[85] Traytel, D., Popescu, A., Blanchette, J.C.: Foundational, compositional (co)datatypes for higher-order logic: Category theory applied to theorem proving. In: Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012. pp. 596–605. IEEE Computer Society (2012)
[86] Troelstra, A.S.: Basic Proof Theory. Cambridge University Press, New York (2000)
[87] Urban, C., Tasson, C.: Nominal techniques in Isabelle/HOL. In: Nieuwenhuis, R. (ed.) Automated Deduction - CADE-20, 20th International Conference on Automated Deduction, Tallinn, Estonia, July 22-27, 2005, Proceedings. LNCS, vol. 3632, pp. 38–53. Springer (2005)
[88] Wadler, P.: Theorems for free! In: Stoy, J.E. (ed.) FPCA 1989. pp. 347–359. ACM (1989)
[89] Wadler, P.: The Girard-Reynolds isomorphism (second edition). Theor. Comput. Sci. 375(1-3), 201–226 (2007), https://doi.org/10.1016/j.tcs.2006.12.042
[90] Wenzel, M.: Type classes and overloading in higher-order logic. In: Gunter, E.L., Felty, A.P. (eds.) Theorem Proving in Higher Order Logics, 10th International Conference, TPHOLs'97, Murray Hill, NJ, USA, August 19-22, 1997, Proceedings. LNCS, vol. 1275, pp. 307–322. Springer (1997)
[91] Yokouchi, H.: Completeness of type assignment systems with intersection, union, and type quantifiers. Theor. Comput. Sci. 272(1-2), 341–398 (2002), https://doi.org/10.1016/S0304-3975(00)00356-X