Model Theory for Dependently-Typed Higher-Order Logic

Florian Rabe[https://orcid.org/0000-0003-3040-3655]

University Erlangen-Nuremberg, Germany

Abstract. Both higher-order logic and dependent types are popular features for expressive languages for automated reasoning. Recently DHOL was introduced as a language that provides the expressivity of dependent types while retaining the simplicity of higher-order logic. DHOL was introduced with a proof system and a translation to HOL, but it lacked a model-theoretical semantics.
The present paper fills in this gap. It introduces both standard and general (Henkin-like) models for DHOL and proves soundness and completeness. While the definitions and theorems appear straightforward, considerable care had to be taken to find a formulation that accommodates the subtleties introduced by dependent types while staying as close to the usual treatment for HOL as possible. Our results will allow for extending semantic HOL methods to DHOL and for giving model-theoretical correctness proofs for other DHOL calculi and for the existing DHOL-to-HOL translation.

1 Introduction and Related Work

Setting Recently dependently-typed higher-order logic (DHOL) was introduced in [RRB23]. It can be seen as an extension of HOL [Chu40,Gor88] that uses dependent function types $\Pi x : A.B$ instead of simple function types $A \to B$. It is designed to stay as simple and as close to HOL as possible while meeting the frequent user demand of supporting dependent types. Contrary to typical formulations of dependent type theory such as Martin-Löf type theory [ML74] and implementations in proof assistants [Nor05,Coq15,dKA+15], DHOL does not employ a sophisticated treatment of equality that keeps typing decidable. Instead, it uses a straightforward formulation of equality at the cost of making typing undecidable. Because theorem proving, the ultimate goal, is undecidable anyway, we do not consider that a deal-breaker.

Concretely, [RRB23] introduces the syntax and proof theory of DHOL and then gives a sound and complete translation of DHOL to HOL. The latter allows using existing HOL-theorem provers for DHOL. A native theorem prover for DHOL based on tableau calculus was recently developed in [NBK24]. However, a model-theoretical semantics of DHOL has so far been lacking.

Motivation Apart from the fact that it is well-established best practice to couple every logic with an appropriate model theory, there are serious practical reasons to do so for DHOL in particular.

The proof theory of DHOL, while conceptually straightforward, is finicky to get right due to the many subtleties entailed by dependent types and undecidable typing. Here a sound and complete model theory provides an alternative characterization of the language and acts as an independent check that no proof rules are missing. It can also serve as a reference semantics for proving future, more implementation-optimized calculi for DHOL correct, such as the one from [NBK24].

Moreover, a model theory for DHOL allows transferring model-theoretical techniques from HOL to DHOL. Individual models can be used as counterexamples for disproving conjectures or as example for establishing the consistency of theories. It also allows using DHOL as a dependently typed higher-order specification language.

Finally, it would allow using model-theoretical arguments to prove translations into or out of DHOL correct. For example, the correctness proof for the DHOL→HOL translation of [RRB23] is shown correct by painstakingly translating HOL-proofs back to DHOL. This is not only difficult and error-prone but also not robust under extensions of DHOL: even for minor additions (e.g., adding integers as a built-in base type), it is non-obvious if the translation is still sound and complete, let alone how the proof has to be adjusted. Experience has shown that model-theoretical correctness arguments can be much simpler, e.g., as done in [SR09].

Contribution and Overview We define the missing model theory and prove soundness and completeness. Sect. 2 gives a self-contained definition of DHOL, including all rules and extending [RRB23] with some details that are needed in our proofs. Then we define standard models in Sect. 3.

Standard models do not yield completeness for higher-order languages, and it is common to use a more general notion of model originally due to Henkin [Hen49]. We introduce an appropriate notion of general model for DHOL in Sect. 4. Our key idea is that a general model is an arbitrary, i.e., not necessarily compositional, interpretation of the syntax that preserves all judgments and commutes with substitution. That subsumes Henkin models for HOL. We establish a characterization of general models as certain functors out of the category of contexts, which creates a connection to categorical semantics in the style of [LS86,Hof94]. In Sect. 5, we prove completeness relative to general models via a term model construction.

Our treatment is deceptively simple: The presentation of our results feels natural and is smooth to read. But significant effort went into identifying exactly the one formulation in which the various subtleties can be treated elegantly.

2 Syntax and Proof Theory

DHOL was introduced in [RRB23], and we give a self-contained definition in Fig. 1 (grammar), 2 (defined connectives), 3 (judgments), 4 (structural rules), and 5 (rules for expressions). Intuitively, DHOL arises in a straightforward way

Theories			
T	$::=$	$\cdot$	empty theory
	$\mid$	$T,\ a : (\Gamma)\texttt{Type}$	dependent type constructor a with argument list Γ
	$\mid$	$T,\ c : A$	typed constant c
	$\mid$	$T,\ F$	axiom asserting formula F

Contexts			
Γ	$::=$	$\cdot$	empty context
	$\mid$	$\Gamma,\ x : A$	typed variable x
	$\mid$	$\Gamma,\ F$	local assumption of formula F

Substitutions			
γ	$::=$	$\cdot$	substitution for the empty context
	$\mid$	$\gamma,\ t$	substitution for a context $\Gamma, x : A$

Types			
A, B	$::=$	$\texttt{bool}$	booleans/formulas
	$\mid$	$a\ \gamma$	dependent type constructor applied to argument terms
	$\mid$	$\Pi x : A.B$	dependent function type

Terms (including formulas)			
s, t, F, G	$::=$	c	constant from theory
	$\mid$	x	variable from context
	$\mid$	$\lambda x : A.t$	function formation
	$\mid$	$t\ s$	function application
	$\mid$	$s =_A t$	equality of terms of type A
	$\mid$	$F \Rightarrow G$	dependent implication

Fig. 1. DHOL Grammar

from HOL by adding dependent function types $\Pi x : A.B$, and functions $\lambda x : A.t$ map each argument $x : A$ to a result in $B(x)$. In the sequel, we discuss only the subtleties that readers familiar with FOL or HOL must watch out for.

Most importantly, DHOL uses a single **equality** $s =_A t$ for typed terms that behaves like that of FOL or HOL. The most consequential rule of DHOL is number (2) in Fig. 5. It is part of the congruence rules, which we have grayed out because they are so straightforward. Indeed, (2) simply says that two applications $a\ \gamma$ and $a\ \gamma'$ of a dependent type constructor a are equal if their arguments are equal. But it makes equality of types depend on equality of terms, which in turn depends on axioms; thus all judgments become undecidable.

Theories, contexts, and substitutions are mostly routine. Users declare dependent base types $a : (\Gamma)\texttt{Type}$ where the context Γ lists the arguments of a, and then $a\ \gamma$ is the type obtained by supplying terms for the arguments declared in Γ. (Alternatively, we could use a curried application of base types. By not doing it, we can simplify the presentation because we do not need to type partially

Abbreviation	Definiens
$\texttt{true}$	$(\lambda x : \texttt{bool}.x) =_{\texttt{bool}} (\lambda x : \texttt{bool}.x)$
$\texttt{false}$	$(\lambda x : \texttt{bool}.x) =_{\texttt{bool}} (\lambda x : \texttt{bool}.\texttt{true})$
$\neg F$	$F =_{\texttt{bool}} \texttt{false}$
$F \vee G$	$\neg F \Rightarrow G$
$F \wedge G$	$\neg(F \Rightarrow \neg G)$
$\forall x : A.F$	$(\lambda x : A.F) =_{A \to \texttt{bool}} (\lambda x : A.\texttt{true})$
$\exists x : A.F$	$\neg\forall x : A.\neg F$

Fig. 2. Definable Connectives

Judgment	Intuition	Equality Judgment
$\vdash T : \texttt{Thy}$	T is a theory	
$\vdash_T \Gamma : \texttt{Cont}$	Γ is a context	$\vdash_T \Gamma \equiv \Gamma'$
$\vdash_T \gamma : \Gamma \to \Delta$	γ substitutes Γ-variables with Δ-terms	$\vdash_T \gamma \equiv \gamma' : \Gamma \to \Delta$
$\Gamma \vdash_T A : \texttt{Type}$	A is a type	$\Gamma \vdash_T A \equiv A'$
$\Gamma \vdash_T t : A$	term t has type A	$\Gamma \vdash t =_A t'$
$\Gamma \vdash F$	F is a theorem (including the special case where F is $t =_A t'$)	

Note that while $s =_A t$ is a formula, i.e., a term of type $\texttt{bool}$, the equality of contexts, substitutions, and types are not formulas and are instead handled by separate judgments all written using $\equiv$.

Fig. 3. DHOL Judgments (T will be dropped if clear from context)

applied base types.) Theories and contexts may also declare typed constants $c : A$ resp. typed variables $x : A$, and they may state axioms/assumptions.

A substitution $\vdash \gamma : (x_1 : A_1, \ldots, x_n : A_n) \to \Delta$ is a list $t_1, \ldots, t_n$ of Δ-terms that *simultaneously substitute* each x_i with t_i. For a type/term using the x_i, we write $A[\gamma]$ and $t[\gamma]$ for the Δ-expression resulting from applying γ. In the common case where $\gamma = x_1, \ldots, x_{n-1}, s$, we simply write this as $A[s]$ resp. $t[s]$.

The only subtlety is the *order of declarations*. Every declared identifier may occur in subsequent declarations, and well-typedness of declarations may depend on the preceding axioms/assumptions. Therefore, theories and contexts must be lists in which identifier declarations and axioms/assumptions may alternate. Consequently, substitutions must respect any assumptions declared in the domain context, i.e., well-formedness of substitutions depends on provability — this is handled by Rule (1) in Fig. 4. The order-sensitivity is harmless in practice: modern practical systems for HOL are processing theories as a list of declarations anyway. In fact, users already often choose a specific order of declarations for purposes of proving theorems or inserting extra-logical content like tactics or documentation.

Types and terms are also mostly straightforward. As usual for dependent functions, the typing rule for application needs to use a substitution to determine the return type, see Rule (3) in Fig. 5. This is the reason why typing and substitution must be intertwined in all algorithms and proofs.

Theories:

$$\frac{}{\vdash \cdot : \mathtt{Thy}} \qquad \frac{\vdash T : \mathtt{Thy} \qquad a \notin T \qquad \vdash_T \Gamma : \mathtt{Cont}}{\vdash T, a : (\Gamma)\mathtt{Type} : \mathtt{Thy}}$$

$$\frac{\vdash T : \mathtt{Thy} \qquad c \notin T \qquad \vdash_T A : \mathtt{Type}}{\vdash T, c : A} \qquad \frac{\vdash T : \mathtt{Thy} \qquad \vdash_T F : \mathtt{bool}}{\vdash T, F : \mathtt{Thy}}$$

Context formation and equality:

$$\frac{\vdash T : \mathtt{Thy}}{\vdash_T \cdot : \mathtt{Cont}} \qquad \frac{\vdash_T \Gamma : \mathtt{Cont} \qquad \Gamma \vdash_T A : \mathtt{Type}}{\vdash_T, x : A : \mathtt{Cont}} \qquad \frac{\vdash_T \Gamma : \mathtt{Cont} \qquad \Gamma \vdash_T F : \mathtt{bool}}{\vdash_T, F : \mathtt{Cont}}$$

$$\frac{\vdash T : \mathtt{Thy}}{\vdash_T \cdot \equiv \cdot} \qquad \frac{\vdash_T \Gamma \equiv \Gamma' \qquad \Gamma \vdash_T A \equiv A'}{\vdash_T \Gamma, x : A \equiv \Gamma', x : A'} \qquad \frac{\vdash_T \Gamma \equiv \Gamma' \qquad \Gamma \vdash_T F =_{\mathtt{bool}} F'}{\vdash_T \Gamma, F \equiv \Gamma', F'}$$

Substitution formation and equality:

$$\frac{\vdash_T \Delta : \mathtt{Cont}}{\vdash_T \cdot : \cdot : \to \Delta} \qquad \frac{\vdash_T \gamma : \Gamma \to \Delta \qquad \Delta \vdash_T t : A[\gamma]}{\vdash_T, \gamma, t : \Gamma, x : A \to \Delta} \qquad \frac{\vdash_T \gamma : \Gamma \to \Delta \qquad \Delta \vdash_T F[\gamma]}{\vdash_T, \gamma : \Gamma, F \to \Delta} \quad (1)$$

$$\frac{\vdash_T \Delta : \mathtt{Cont}}{\vdash_T \cdot \equiv \cdot : \cdot \to \Delta} \qquad \frac{\vdash_T \gamma \equiv \gamma' : \Gamma \to \Delta \qquad \Delta \vdash_T t =_{A[\gamma]} t'}{\vdash_T, (\gamma, t) \equiv (\gamma', t') : (\Gamma, x : A) \to \Delta} \qquad \frac{\vdash_T \gamma \equiv \gamma' : \Gamma \to \Delta \qquad \Delta \vdash_T F[\gamma]}{\vdash_T, \gamma \equiv \gamma' : \Gamma, F \to \Delta}$$

Fig. 4. DHOL Rules: Theories, contexts, substitutions

Following Andrews' definition of HOL [And86], we choose a simplified presentation that starts with equality $=_A$ and defines the connectives like `true`, `false`, $\neg$, $\forall$, $\exists$ from it as shown in Fig. 2. However, contrary to HOL, we have to make one binary connective primitive because DHOL requires *dependent* binary connectives: in the dependent implication $F \Rightarrow G$, the well-formedness of G may depend on the truth of F. This is important, e.g., to state a formula like $x =_A y \Rightarrow f(x) =_{B[x]} f(y)$ for a dependent function $f : \Pi x : A.B$: here, $f(x) =_{B[x]} f(y)$ is only well-formed if $f(x) : B(x)$ and $f(y) : B(y)$ have the same type; that follows from the assumption $x =_A y$ using Rule (2); but we need Rule (4) to actually appeal to that assumption while type-checking $f(x) =_{B[x]} f(y)$. The dependent connectives cannot be defined from equality. But dependent conjunction $F \wedge G$ and disjunction $F \vee G$, where the well-formedness of G may assume F resp. $\neg F$, can be defined from dependent implication.

The **proof rules** are the usual ones for HOL. It is straightforward to derive the usual proof rules for the defined connectives. We use classical logic with β and η-conversion, which implies functional extensionality. As usual for dependently-typed languages and contrary to HOL, we allow for *empty types*. Thus, e.g., $\forall x : A.F \Rightarrow \exists x : A.F$ is not a theorem. Maybe surprisingly, this requires no special treatment in the proof system — by systematically carrying the context Γ, we are already building a language in which types may be empty, and non-emptiness would require an additional axiom.

Example 1 (Vectors). The following theory can be used to specify a dependent type $V\ n$ holding vectors of dimension n over type U:

$$N : \mathtt{Type}, \quad 1 : N, \quad + : N \to N \to N, \quad U : \mathtt{Type}, \quad V : N \to \mathtt{Type},$$
$$\mathit{entry} : U \to V\ 1, \quad \mathit{concat} : \Pi m : N.\Pi n : N.(V\ m) \to (V\ n) \to (V\ (m + n))$$

Type formation and equality:

$$\frac{a : (\Delta)\texttt{Type} \in T \qquad \vdash_T \Gamma, \delta : \Gamma, \Delta \to \Gamma}{\Gamma \vdash_T a\ \delta : \texttt{Type}} \qquad \frac{\Gamma, x : A \vdash_T B : \texttt{Type}}{\Gamma \vdash \Pi x : A.B : \texttt{Type}} \qquad \frac{\vdash_T \Gamma : \texttt{Cont}}{\Gamma \vdash_T \texttt{bool} : \texttt{Type}}$$

$$\frac{\Gamma \vdash_T (id_\Gamma, \delta) \equiv (id_\Gamma, \delta') : (\Gamma, \Delta) \to \Gamma}{\Gamma \vdash_T a\ \delta \equiv a\ \delta'}(2) \qquad \frac{\Gamma \vdash_T A \equiv A' \qquad \Gamma, x : A \vdash_T B \equiv B'}{\Gamma \vdash \Pi x : A.B \equiv \Pi x : A'.B'} \qquad \frac{\vdash_T \Gamma : \texttt{Cont}}{\Gamma \vdash_T \texttt{bool} \equiv \texttt{bool}}$$

Term formation and equality, for identifiers:

$$\frac{c : A \in T \qquad \vdash_T \Gamma : \texttt{Cont}}{\Gamma \vdash_T c : A} \qquad \frac{x : A \in \Gamma \qquad \vdash_T \Gamma : \texttt{Cont}}{\Gamma \vdash_T x : A}$$

$$\frac{c : A \in T \qquad \vdash_T \Gamma : \texttt{Cont}}{\Gamma \vdash_T c =_A c} \qquad \frac{x : A \in \Gamma \qquad \vdash_T \Gamma : \texttt{Cont}}{\Gamma \vdash_T x =_A x}$$

Term formation and equality, for functions:

$$\frac{\Gamma \vdash_T A : \texttt{Type} \qquad \Gamma, x : A \vdash_T t : B}{\Gamma \vdash \lambda x : A.t : \Pi x : A.B} \qquad \frac{\Gamma \vdash_T t : \Pi x : A.B \qquad \Gamma \vdash_T s : A}{\Gamma \vdash_T t\ s : B[s]}(3)$$

$$\frac{\Gamma \vdash A \equiv A' \qquad \Gamma, x : A \vdash_T t =_B t'}{\Gamma \vdash \lambda x : A.t =_{\Pi x:A.B} \lambda x : A'.t'}(\xi) \qquad \frac{\Gamma \vdash_T t =_{\Pi x:A.B} t' \qquad \Gamma \vdash_T s =_A s'}{\Gamma \vdash_T t\ s =_{B[s]} t'\ s'}$$

$$\frac{\Gamma \vdash_T t : \Pi x : A.B}{\Gamma \vdash t =_{\Pi x:A.B} \lambda x : A.(t\ x)}(\eta) \qquad \frac{\Gamma, x : A \vdash_T t : B}{\Gamma \vdash_T (\lambda x : A.t)\ s =_{B[s]} t[s]}(\beta)$$

Term formation for Booleans:

$$\frac{\Gamma \vdash_T t : A \qquad \Gamma \vdash_T t' : A}{\Gamma \vdash t =_A t' : \texttt{bool}} \qquad \frac{\Gamma \vdash_T F : \texttt{bool} \qquad \Gamma, F \vdash_T G : \texttt{bool}}{\Gamma \vdash_T F \Rightarrow G : \texttt{bool}}(4)$$

Proof rules:

$$\frac{\Gamma, F \vdash_T G \qquad \Gamma, G \vdash_T F}{\Gamma \vdash_T F =_{\texttt{bool}} G}(\text{PrExt}) \qquad \frac{\Gamma \vdash_T F \qquad \Gamma \vdash_T F =_{\texttt{bool}} G}{\Gamma \vdash_T G}$$

$$\frac{\Gamma, F \vdash_T G}{\Gamma \vdash_T F \Rightarrow G} \qquad \frac{\Gamma \vdash_T F \Rightarrow G \qquad \Gamma \vdash_T F}{\Gamma \vdash_T G}$$

$$\frac{\Gamma \vdash_T t : A \qquad \Gamma \vdash_T t =_A t' \qquad \Gamma \vdash_T A \equiv A'}{\Gamma \vdash t' : A'} \qquad \frac{\Gamma \vdash_T F : \texttt{bool}}{\Gamma \vdash_T F \vee \neg F}(\text{TND})$$

Fig. 5. DHOL Rules: types, terms, proofs

Here we simplify and write $N : \texttt{Type}$ instead of $N : (\cdot)\texttt{Type}$ for a type without arguments. Now the associativity axiom for concatenation

$$concat\ l\ (m{+}n)\ x\ (concat\ m\ n\ y\ z) =_{V(l+(m+n))} concat\ (l{+}m)\ n\ (concat\ l\ m\ x\ y)\ z$$

is well-typed only after stating the associativity axiom $l+(m+n) =_N (l+m)+n$ for addition.

Remark 1 (HOL as a Fragment of DHOL). It is instructive to see how we can recover HOL. HOL-theories are the ones where all type declarations are of the form $a : (\cdot)A$, i.e., do not take any arguments. Then the syntax guarantees that terms never occur in types, and thus $\Pi x : A.B$ can always be written $A \to B$, and typing is decidable. Moreover, well-formedness of terms cannot depend on axioms/assumptions. Therefore, theories/contexts can be simplified by collecting the axioms/assumptions into a set.

Finally, we show that substitution has the usual properties, i.e., preserves all judgments on expressions:

Theorem 1 (Substitution). *Fix a theory T and assume $\vdash \gamma : \Gamma \to \Delta$. Then*

$$\begin{array}{rll} \Gamma \vdash A : \mathtt{Type} & \textit{implies} & \Delta \vdash A[\gamma] : \mathtt{Type} \\ \Gamma \vdash A \equiv A' & \textit{implies} & \Delta \vdash A[\gamma] \equiv A'[\gamma] \\ \Gamma \vdash t : A & \textit{implies} & \Delta \vdash t[\gamma] : A[\gamma] \\ \Gamma \vdash F & \textit{implies} & \Delta \vdash F[\gamma] \end{array}$$

Proof. All statements are proved together, by a straightforward joint induction on derivations.

3 Standard Models

Syntax	Semantics of typing
theory T	class $[\![T]\!]$ of models M
context Γ substitution $\gamma : \Gamma \to \Delta$	set $[\![\Gamma]\!]^M$ of assignments α for Γ into M function $[\![\gamma]\!]^M : [\![\Delta]\!]^M \to [\![\Gamma]\!]^M$
type A term t of type A	set $[\![A]\!]^M_\alpha$ element $[\![t]\!]^A_\alpha \in [\![A]\!]^M_\alpha$

Fig. 6. Class of Models and Structure of an Interpretation Function

Our definition of model follows the well-known structure of models for first- and higher-order logic. Essentially, a model interprets the identifiers and induces an inductive interpretation function for all expressions. The following definition makes the concept of the interpretation function precise:

Definition 1 (Interpretation Function). *An interpretation function for a theory T is a family of mappings defined for contexts, substitutions, types, and terms as described in Fig. 6.*

Because models and assignments are tuples, we frequently need to append an element to a tuple and introduce a shortcut for it:

Definition 2 (Extending a Tuple). *If t is a tuple $(t_1, \ldots, t_n)$, we write $t +^n a$ for the tuple $(t_1, \ldots, t_n, a)$. We omit n from the notation.*

Because terms can occur in types, because well-formedness depends on the assumptions in the context, and because contexts may contain alternations of variable declarations and assumptions, the definitions of models and interpretation functions are all mutually recursive and proceed by a joint induction on derivations. However, it is didactically preferable to ignore this technicality at first: we first state all definitions separately, and then prove well-definedness afterwards in Thm. 3.

First we define a standard model of T as a tuple that contains one interpretation for every T-identifier and satisfies all axioms:

Definition 3 (Standard Models). *We define* $[\![T]\!]$ *by induction on* T*:*
- $[\![\cdot]\!] = \{()\}$ *(empty tuple models the empty theory)*
- $[\![T,\, a : (\Gamma)A]\!] = \{M + m \,:\, M \in [\![T]\!]^M,\, m : [\![\Gamma]\!]^M \to \mathcal{SET}\}$ *(models interpret type constructors* a *as set-valued functions* m*)*
- $[\![T,\, c : A]\!] = \{M + m \,:\, M \in [\![T]\!],\, m \in [\![A]\!]^M_{()}\}$ *(models interpret constants* c *as elements* m *of their type)*
- $[\![T,\, F]\!] = \{M \in [\![T]\!] \mid [\![F]\!]^M_{()} = 1\}$ *(models must satisfy axioms)*

(where $()$ *is the assignment for the empty context used to interpret the closed expressions* A *and* F*).*

Given $M \in [\![T]\!]$ *and a* T*-identifier* a *or* c*, we write* a^M *resp.* c^M *for the corresponding component of* M*.*

The interpretation of a context Γ is the set $[\![\Gamma]\!]^M$ of assignments that map Γ-variables to values in M. An assignment is a tuple that contains one value in M for every variable in Γ in such a way that all assumptions in Γ are satisfied. A substitution $\gamma : \Gamma \to \Delta$ is interpreted contravariantly as the function that composes γ with a Δ-assignment to obtain a Γ-assignment:

Definition 4 (Assignments and Interpretation of Contexts). *We define* $[\![\Gamma]\!]^M$ *by induction on* Γ*:*
- $[\![\cdot]\!]^M = \{()\}$ *(empty assignment for the empty context)*
- $[\![\Gamma,\, x : A]\!]^M = \{\alpha + u \,:\, \alpha \in [\![\Gamma]\!]^M,\, u \in [\![A]\!]^M_\alpha\}$ *(assignments map variables* x *to values* u*)*
- $[\![\Gamma,\, F]\!]^M = \{\alpha \in [\![\Gamma]\!]^M \mid [\![F]\!]^M_\alpha = 1\}$ *(assignments must satisfy assumptions)*

Given $\alpha \in [\![\Gamma]\!]^M$ *and a* Γ*-variable* x*, we write* x^α *for the corresponding component of* α*.*

For a substitution $\vdash t_1, \ldots, t_n : \Gamma \to \Delta$*, the mapping* $[\![t_1 \ldots, t_n]\!]^M$ *is defined by* $[\![\Delta]\!]^M \ni \alpha \mapsto ([\![t_1]\!]^M_\alpha, \ldots, [\![t_n]\!]^M_\alpha) \in [\![\Gamma]\!]^M$*.*

Example 2 (Vectors). A model M for the theory from Ex. 1 is given by the tuple

$$\left(\mathbb{N} \setminus \{0\},\ 1\ ,+\ ,\mathbb{R}\ ,n \mapsto \mathbb{R}^n\ ,r \mapsto (r),\ m \mapsto n \mapsto (v \in \mathbb{R}^m) \mapsto (w \in \mathbb{R}^n) \mapsto \begin{pmatrix} v \\ w \end{pmatrix}\right)$$

Here we omit $() \mapsto$ when giving the interpretation of a type without arguments.

Types and terms in context Γ are interpreted relative to a model M and an assignment $\alpha \in [\![\Gamma]\!]^M$

Definition 5 (Interpretation of Types and Terms). *For types and terms, we define* $[\![-]\!]^M_\alpha$ *by induction:*
- $[\![\texttt{bool}]\!]^M_\alpha = \{0, 1\}$
- $[\![a\ t_1 \ldots t_n]\!]^M_\alpha = a^M([\![t_1]\!]^M_\alpha, \ldots, [\![t_n]\!]^M_\alpha)$
- $[\![\Pi x : A.B]\!]^M_\alpha = \{f : [\![A]\!]^M_\alpha \ni u \mapsto f(u) \in [\![B]\!]^M_{\alpha+u}\}$ *(i.e., the set of functions* f *mapping values* u *in the interpretation of* A *to values in the interpretation of* B *at* u*.*
- $[\![c]\!]^M_\alpha = c^M$
- $[\![x]\!]^M_\alpha = x^\alpha$

– $[\![\lambda x : A.t]\!]^M_\alpha = [\![A]\!]^M_\alpha \ni u \mapsto [\![t]\!]^M_{\alpha+u}$
– $[\![t\ s]\!]^M_\alpha = [\![t]\!]^M_\alpha([\![s]\!]^M_\alpha)$
– $[\![s =_A t]\!]^M_\alpha = 1$ *iff* $[\![s]\!]^M_\alpha = [\![t]\!]^M_\alpha$
– $[\![F \Rightarrow G]\!]^M_\alpha = 1$ *iff* $[\![F]\!]^M_\alpha = 0$, *or* $[\![F]\!]^M_\alpha = 1$ *and* $[\![G]\!]^M_\alpha = 1$

In the case for implication, note that $[\![F]\!]^M_\alpha = 1$ *implies* $\alpha \in [\![\Gamma, F]\!]$, *which is needed to use* α *to interpret* G *as required by Rule (4) in Fig. 5.*

We can now show that this yields the usual semantics for the definable connectives from Fig. 2:

Theorem 2 (Defined Operators). *We have the following interpretation rules:*
– $[\![\texttt{true}]\!]^M_\alpha = 1$
– $[\![\texttt{false}]\!]^M_\alpha = 0$
– $[\![\neg F]\!]^M_\alpha = 1$ *iff* $[\![F]\!]^M_\alpha = 0$
– $[\![F \wedge G]\!]^M_\alpha = 1$ *iff* $[\![F]\!]^M_\alpha = 1$ *and* $[\![G]\!]^M_\alpha = 1$
– $[\![F \vee G]\!]^M_\alpha = 1$ *iff* $[\![F]\!]^M_\alpha = 1$, *or* $[\![F]\!]^M_\alpha = 0$ *and* $[\![G]\!]^M_\alpha = 1$
– $[\![\forall x : A.F]\!]^M_\alpha = 1$ *iff* $[\![F]\!]^M_{\alpha+u} = 1$ *for every* $u \in [\![A]\!]^M_\alpha$
– $[\![\exists x : A.F]\!]^M_\alpha = 1$ *iff* $[\![F]\!]^M_{\alpha+u} = 1$ *for some* $u \in [\![A]\!]^M_\alpha$

Proof. All cases are straightforward. In the case for conjunction, note that $[\![F]\!]^M_\alpha = 1$ implies $\alpha \in [\![\Gamma, F]\!]$ so that α can indeed be used to interpret G. Accordingly, in the case for disjunction, $[\![F]\!]^M_\alpha = 0$ implies $\alpha \in [\![\Gamma, \neg F]\!]$ so that α can indeed be used to interpret G.

As mentioned above, the above definitions are mutually recursive. Moreover, when it comes to checking the well-definedness, the devil is in the details, and we must prove well-definedness, the substitution property, and even soundness in a single big induction:

Theorem 3 (Well-Definedness, Substitution Property, and Soundness). *Consider a model* $M \in [\![T]\!]$. *Then the interpretation function induced by* M
– *is well-defined (preserves typing), i.e.,*

$$\begin{aligned} [\![A]\!]^M_\alpha &\in \mathcal{SET} && \text{for } \Gamma \vdash_T A : \texttt{Type} \\ [\![t]\!]^M_\alpha &\in [\![A]\!]^M_\alpha && \text{for } \Gamma \vdash_T t : A \\ [\![\Gamma]\!]^M &\in \mathcal{SET} && \text{for } \vdash_T \Gamma : \texttt{Cont} \\ [\![\gamma]\!]^M : [\![\Delta]\!]^M &\to [\![\Gamma]\!]^M && \text{for } \vdash_T \gamma : \Gamma \to \Delta \end{aligned}$$

– *commutes with substitutions* $\vdash \gamma : \Gamma \to \Delta$, *i.e.,*

$$\begin{aligned} [\![t[\gamma]]\!]^M_\alpha &= [\![t]\!]^M_{[\![\gamma]\!]^M(\alpha)} && \text{for } \Gamma \vdash_T t : A \\ [\![A[\gamma]]\!]^M_\alpha &= [\![A]\!]^M_{[\![\gamma]\!]^M(\alpha)} && \text{for } \Gamma \vdash A : \texttt{Type} \end{aligned}$$

– *preserves equality, i.e.,*

$$\begin{aligned} [\![A]\!]^M_\alpha &= [\![A']\!]^M_\alpha && \text{for } \Gamma \vdash_T A \equiv A' \\ [\![t]\!]^M_\alpha &= [\![t']\!]^M_\alpha && \text{for } \Gamma \vdash_T t =_A t' \end{aligned}$$

$$[\![\Gamma]\!]^M = [\![\Gamma']\!]^M \qquad \text{for } \Gamma \vdash_T \Gamma \equiv \Gamma'$$
$$[\![\gamma]\!]^M = [\![\gamma']\!]^M \qquad \text{for } \Gamma \vdash_T \gamma \equiv \gamma'$$

(where the case for terms is a special case of soundness)

– *is sound, i.e.,*

$$[\![F]\!]^M_\alpha = 1 \qquad \text{for } \Gamma \vdash_T F$$

Proof. All proofs are mutually recursive and we proceed using a single induction on derivations.

Well-definedness: All cases are straightforward consequences of the respective induction hypotheses with the following exceptions.

For function application, we have to show $[\![t\ s]\!]^M_\alpha \in [\![B[s]]\!]^M_\alpha$ according to Rule (3) in Fig. 5. The induction hypotheses for $\Gamma \vdash t : \Pi x : A.B$ yields that $[\![t]\!]^M_\alpha$ maps every argument $u \in [\![A]\!]^M_\alpha$ to an element of $[\![B]\!]^M_{\alpha+u}$, and the one for $\Gamma \vdash s : A$ lets us put $u = [\![s]\!]^M_\alpha \in [\![A]\!]^M_\alpha$. To conclude the proof, we also need to use the induction hypothesis for the commutation with substitution for the type B and the substitution $id_\Gamma, s : \Gamma, x : A \to \Gamma$. That yields the missing link $[\![B]\!]^M_{\alpha+[\![s]\!]^M_\alpha} = [\![B[s]]\!]^M_\alpha$.

For substitutions $\vdash \gamma : \Gamma, F \to \Delta$, we have to show $[\![\gamma]\!]^M : [\![\Delta]\!]^M \to [\![\Gamma, F]\!]^M$. The induction hypothesis for $\vdash \gamma : \Gamma \to \Delta$ yields $[\![\gamma]\!]^M : [\![\Delta]\!]^M \to [\![\Gamma]\!]^M$ (1) and the one for $\Delta \vdash F[\gamma]$ yields $[\![F[\gamma]]\!]^M_\alpha = 1$ for all $\alpha \in [\![\Delta]\!]^M$ (2). We also have to use the substitution property for F and γ, which yields $[\![F[\gamma]]\!]^M_\alpha = [\![F]\!]^M_{[\![\gamma]\!]^M(\alpha)}$ (3). First of all, we use (1) and note that Def. 4 interprets $\vdash \gamma : \Gamma, F \to \Delta$ and $\vdash \gamma : \Gamma \to \Delta$ as the same function: the former is a restriction of the latter to a smaller codomain. Incidentally, this justifies using the same notation $[\![\gamma]\!]^M$ for both domain contexts. We have to show that this restriction is well-defined, i.e., that the image of $[\![\gamma]\!]^M$ is contained in $[\![\Gamma, F]\!]^M \subseteq [\![\Gamma]\!]^M$. Thus, we need $[\![\gamma]\!]^M(\alpha) \in [\![\Gamma, F]\!]^M$ for any $\alpha \in [\![\Delta]\!]^M$. That is equivalent to $[\![F]\!]^M_{[\![\gamma]\!]^M(\alpha)} = 1$, which follows from (3) and (2).

Note that these cases are the reason why well-definedness, substitution property, and soundness must be proved in a joint induction.

Commutation with substitution: All cases push the inductive definitions of substitution and interpretation function into the term/type. For the base case for variables, we assume Γ declares variables $x_1, \ldots, x_n$ and $\gamma = t_1, \ldots, t_n$ and have $[\![x_i[\gamma]]\!]^M_\alpha = [\![t_i]\!]^M_\alpha = x_i^{[\![\gamma]\!]^M(\alpha)} = [\![x_i]\!]^M_{[\![\gamma]\!]^M(\alpha)}$ as needed.

Equality preservation: All cases for $\equiv$ are straightforward, and the statement for terms is a special case of soundness.

Soundness: All cases for congruence and implication are straightforward, and we state the cases for the remaining rules.

For Rule β, we have to show that $[\![A]\!]^M_\alpha \ni u \mapsto [\![t]\!]^M_{\alpha+u}$ maps $u = [\![s]\!]^M_\alpha$ to $[\![t[s]]\!]^M_\alpha$. That follows from the substitution property.

For Rule η, the function $[\![A]\!]^M_\alpha \ni u \mapsto [\![t\ x]\!]^M_{\alpha+u}$ is equal to $u \mapsto [\![t]\!]^M_{\alpha+u}([\![x]\!]^M_{\alpha+u})$, which is equal to $u \mapsto [\![t]\!]^M_\alpha(u)$ and thus to $[\![t]\!]^M_\alpha$. Technically, this uses the substitution property for the term t and the identity substitution $\Gamma \to \Gamma, x : A$ to drop u from the assignment $\alpha + u$ when interpreting t.

For Rule (PrExt), we can use the implication introduction rule to derive $\Gamma \vdash F \Rightarrow G$ and $\Gamma \vdash G \Rightarrow F$. Then $[\![F]\!]^M_\alpha = [\![G]\!]^M_\alpha$ follows from the interpretation of implication.

The soundness of Rule (TND) follows from Thm. 2.

4 General Models

For HOL, it is relatively easy to define general models, usually called Henkin models after [Hen49]. The key idea is allow function types $A \to B$ to be interpreted as sufficiently large *subsets of* the function set $[\![A]\!]^M \to [\![B]\!]^M$. This allows retaining almost all definitions of the compositional interpretation function with little change.

Designing a good definition of general model for DHOL is significantly more difficult. The technical reason for this is, essentially, that DHOL-types cannot be interpreted in isolation (as exploited in Henkin models) and must be treated in joint inductions with terms and provability.

Our key observation that leads to an elegant notion of general model is that the interpretation function of a standard model (i) interprets contexts and variables via assignments, (ii) satisfies the substitution property, and (iii) is compositional, i.e., defined by induction on the syntax; and that, crucially, (i) and (ii) can be realized even if (iii) is given up. This motivates the following intuition for a general DHOL model: Like a standard model, a general model is given by an interpretation function like in Fig. 6. But contrary to a standard model, the interpretation function does not have to be compositional.

Concretely, we define:

Definition 6 (General Model). *Consider an interpretation function W for a theory T, written $[\![-]\!]^W$ and $[\![-]\!]^W_-$. We say that W*
- *preserves typing*
- *commutes with substitution*
- *preserves equality*
- *is sound*

if it satisfies the corresponding statements of Thm. 3.
We say W has standard variables if it satisfies the interpretation rules for contexts and substitutions from Def. 4 as well as the interpretation rule $[\![x]\!]^W_\alpha = x^\alpha$ for variables from Def. 5.
We say W has standard Booleans if it satisfies the interpretation rules for `bool`*, $=_A$, and $\Rightarrow$ from Def. 5.*
Finally, we call W a general model if satisfies all of the above.

Thus, a general model is the same as a standard model except for being free to deviate from the compositional interpretation rules for Π, λ, and application. Due to Thm. 3, every standard model is a general model.

Remark 2 (Henkin Models as a Special Case). Our general models are not a direct analogue of Henkin models for HOL: if we restrict attention to HOL theories, our general models are instead a substantial generalization of Henkin models. Henkin models require compositionality for λ and application and only relax the interpretation of function types, by allowing $[\![A \to B]\!]$ to be a subset of $[\![A]\!] \to [\![B]\!]$. Our definition is much more general, imposing only some high-level conditions on an arbitrary interpretation. That is a lot more flexible as the examples below and our definition of the term model in Sect. 5 show.

Nonetheless, Henkin models are quite similar to ours: Via higher-order abstract syntax, HOL's λ, application, and β-reduction correspond to substitution application. Thus, the compositional interpretation of λ and application in a Henkin model is closely related to the substitution property, and we conjecture there to be some kind of equivalence theorem. In any case, we suspect that our notion of general models will prove interesting for HOL as well.

Example 3 (Trivial Interpretation). The function that interprets every type as the singleton set $\{0\}$ and every term as 0 has all properties of a general model except for standard Booleans. With a little effort, it can be adapted to have standard Booleans as well.

Example 4 (Product Interpretation). Given two general models V and W, we can define an interpretation function $V \times W$ by putting $[\![A]\!]_\alpha^{V\times W} = [\![A]\!]_\alpha^V \times [\![A]\!]_\alpha^W$ and $[\![t]\!]_\alpha^{V\times W} = ([\![t]\!]_\alpha^V, [\![t]\!]_\alpha^W)$. In particular, $V \times W$ interprets a function as a pair of functions and not, as a standard model would, as a function between pairs. $V \times W$ has all properties of a general model except for standard Booleans, and it can be adapted to have standard Booleans as well.

The requirements of Def. 6 are not chosen arbitrarily: they are motivated by the following theorem that enables a high-level characterization of general models as functors out of the syntactic category, an approach commonly used both for higher-order logic [LS86] and for dependent type theory [Hof94]:

Definition 7 (Category of Contexts). *Given a theory T, the category $\mathcal{CTX}_T$ has as objects the contexts $\vdash_T \Gamma : \texttt{Cont}$ quotiented by $\vdash \Gamma \equiv \Gamma'$, and as morphisms from Γ to Δ the substitutions $\vdash \gamma : \Gamma \to \Delta$ quotiented by $\vdash \gamma \equiv \gamma : \Gamma \to \Delta$. Identity and composition are defined in the obvious way.*

Theorem 4 (General Models as Functors). *For a general model W of theory T, the function $[\![-]\!]^W$ induces a contravariant functor from $\mathcal{CTX}_T$ to $\mathcal{SET}$.*

Proof. The elements of $\mathcal{CTX}_T$ are equivalence classes, and $[\![-]\!]^W$ is defined on representatives. This is well-defined because W preserves equality. It is well-defined as a mapping of objects/morphisms to objects/morphisms because W preserves typing and has standard variables.

To show the contravariant functoriality, consider the identity substitution $x_1, \dots, x_n$ of Γ. $[\![x_1, \dots, x_n]\!]^W$ maps α to $([\![x_1]\!]_\alpha^W, \dots, [\![x_n]\!]_\alpha^W)$; because W has standard variables, this is equal to $(x_1^\alpha, \dots, x_n^\alpha) = \alpha$; thus $[\![x_1, \dots, x_n]\!]^W$ is indeed the identify function on $[\![\Gamma]\!]^W$.

Now consider substitutions $\gamma = t_1, \ldots, t_n : \Gamma \to \Delta$ and $\delta : \Delta \to E$. $[\![\gamma; \delta]\!]^W$ maps $\alpha \in [\![E]\!]^W$ to $(\ldots, [\![t_i[\delta]]\!]^W_\alpha, \ldots)$. Because W commutes with substitution, this equals $(\ldots, [\![t_i]\!]^W_{[\![\delta]\!]^W(\alpha)}, \ldots)$, which equals $[\![\gamma]\!]^W([\![\delta]\!]^W(\alpha))$ as needed.

Note that the proof does not actually use the fact W has standard Booleans.

Remark 3 (Functors as General Models). We can classify the general models as the contravariant functors from $\mathcal{CTX}_T$ to $\mathcal{SET}$ that satisfy a few additional properties. In fact, every contravariant functor Φ already induces an interpretation function: for $\Gamma \vdash A : \texttt{Type}$, $[\![A]\!]^\Phi_\alpha$ is the set of elements u such that $\alpha + u \in \Phi(\Gamma, x : A)$; and for $\Gamma \vdash t : A$, $[\![t]\!]^\Phi_\alpha$ is the element u such that $\Phi(id_\Gamma, t)(\alpha) = \alpha + u$. Moreover, this interpretation function already preserves typing and equality, has standard variables, and almost commutes with substitution. We say *almost* in the previous sentence because we have proved all necessary properties except for $[\![A]\!]^\Phi_{[\![\gamma]\!]^\Phi_\alpha} \subseteq [\![A[\gamma]]\!]^\Phi_\alpha$ $(*)$. We are currently investigating what natural conditions can be imposed on Φ to also prove $(*)$. For now, we can say that the general models are the functors that satisfy $(*)$ and have standard Booleans.

5 The Term Model and Completeness

The basic idea of the term model for theory T is to interpret every type as the set of its terms, quotiented by provable equality, and to make formulas true iff they are provable. The usual construction of term models is to extend T to a complete theory by enumerating all formulas F and adding an axiom asserting F whenever neither F nor $\neg F$ is provable. We need to make a few subtle adjustments to generalize this construction to DHOL. Most importantly, our enumeration of formulas must consider that adding an axiom may increase the set of well-formed formulas.

Definition 8. *We allow theories T that end with an infinite list of axioms.*
T is complete if $\vdash_T F$ or $\vdash_T \neg F$ for every formula F.
T is inconsistent if $\vdash_T F$ and $\vdash_T \neg F$ for some formula F.

Theorem 5 (Theory Completion). *Every consistent theory T can be extended to a complete consistent theory T^*.*

Proof. We call a term *possible* if it can be produced by the DHOL-grammar using the identifiers from T, i.e., possible terms need not be well-typed. We call a possible t *critical* if it has type `bool` and neither t not $\neg t$ are provable.

There is an enumeration $t_0, t_1, \ldots$ of possible terms in which every element occurs infinitely often. For example, we can enumerate the set $P \times \mathbb{N}$ where P is the set of all possible terms.

We put $T_0 = T$ and define T_{i+1} from T_i as follows. If t_i is critical over T_i, we put $T_{i+1} = T_i,\ t_i$. Otherwise, we put $T_{i+1} = T_i$. We define T^* as the union of all T_i.

T^* is consistent: If it were inconsistent, some T_i would be inconsistent because a DHOL-proof can use only finitely many axioms. But each T_i is consistent by construction.

T^* is complete: If there were a critical t over T^*, then there would be some T_i over which t would be well-formed, because the well-formedness of t can only depend on finitely many axioms. And then t would be critical over every subsequent T_j. But t occurs in the enumeration infinitely often and thus some time after i, and thus cannot be critical after that point.

Theorem 6 (Term Model). *Every consistent theory T has a general model.*

Proof. We construct a general model W for the infinite theory T^*, which can then be restricted to a general model for T.

We only have to define the interpretation function for types and terms. Then the interpretation of contexts and substitutions is forced by having standard variables.

For a type $\Gamma \vdash A : \mathtt{Type}$, $[\![A]\!]^W_\alpha$ is the set of terms $\vdash_{T^*} t : A[\alpha^!]$ quotiented by the relation $\Gamma \vdash_{T^*} s =_A t$ (which is an equivalence relation by the rules for equality). Here α, by construction, is a tuple of equivalence classes of terms, and $\alpha^!$ is any substitution formed from choosing a representative from each class. That is well-defined because the substitution rules for equality guarantee that any choice for $\alpha^!$ leads to $\equiv$-equal types $A[\alpha^!]$, and the rules for typing guarantee that $\equiv$-equal types have the same terms.

For a term $\Gamma \vdash t : A$, $[\![t]\!]^W_\alpha$ is the corresponding equivalence class of t. This is well-defined for the same reasons.

The interpretation function preserves typing and equality by construction. It commutes with substitution because of the properties of substitutions. To show that it has standard variables, it remains to show $[\![x]\!]^W_\alpha = x^\alpha$, which also holds.

We show that W has standard Booleans. Because T^* is complete, we have $\vdash_{T^*} F =_{\mathtt{bool}} \mathtt{true}$ or $\vdash_{T^*} F =_{\mathtt{bool}} \mathtt{false}$ for every Boolean F. And because T^* is consistent, $\mathtt{true}$ and $\mathtt{false}$ are unequal. Thus, $[\![\mathtt{bool}]\!]^W_\alpha$ has two elements. These elements are not 0 and 1 as technically required, and we omit the obvious but tedious adjustment to the definitions. The cases for $=_A$ and $\Rightarrow$ are straightforward (and the latter uses in particular the rules for $\Rightarrow$).

Finally, soundness holds because $\vdash_T F$ implies $\vdash_{T^*} F =_{\mathtt{bool}} \mathtt{true}$, and thus $[\![F]\!]^W_() = [\![\mathtt{true}]\!]^W_() = 1$.

Example 5 (Vectors). If we use the theory from Ex. 1 with associativity axioms for $+$ and *concat*, the term model W has $N^W \cong \mathbb{N} \setminus \{0\}$ and $V^M(n) \cong (U^W)^n$. U^W is the empty set, but we can model vectors over other sets by adding constants to the theory that construct terms of type U.

Theorem 7 (Completeness). *If $[\![F]\!]^W_\alpha = 1$ for every general model W of T, then $\Gamma \vdash_T F$.*

Proof. If T is inconsistent, the result is trivial. Otherwise, if $\Gamma \vdash_T F$ did not hold, the theory T, $\neg F$ would be consistent, and its term model would satisfy $\neg F$ in violation of the assumption.

Remark 4 (Quantifiers in the Term Model). Readers familiar with term models for FOL and HOL may be surprised that our construction of T^* does not add Skolem constants as witnesses for provable existentials. One might try to show a contradiction by using the theory

$$a : \texttt{Type},\ c : a,\ p : a \to \texttt{bool},\ p(c),\ \exists x : a.\neg p(x)$$

where p holds for all terms of type a and no term exists that witnesses the existential.

Indeed in the term model, we have that $[\![a]\!]^W$ is a singleton set (containing only the equivalence class of c), $[\![\exists x : a.\neg p(x)]\!]^W = 1$, and yet $[\![p\, x]\!]^W_{(u)} = 1$ for every $u \in [\![a]\!]^W$. This appears paradoxical because W has the standard interpretation of equality, and should thus have the standard interpretation of $\exists$ (in the sense of Thm. 2) as well.

The resolution is that the quantifiers, even though they are defined using only equality, do not necessarily have standard interpretations in general models. That is because their definitions use functions, and general models can have a non-standard interpretation of functions. For example, we have $[\![p]\!]^W \neq [\![\lambda x : a.\texttt{true}]\!]^W$ in line with $\vdash \neg\forall x : a.p\ x$. But even though W satisfies η, $[\![p]\!]^W$ is not equal to $u \mapsto [\![p\ x]\!]^W_{(u)}$. Indeed, a comparison of the two is not even well-typed: the former is an equivalence class of terms, and the latter is a function between equivalence classes of terms.

It is possible to require general models to have the standard interpretation of quantifiers. In that case, our definition of the term model would have to be adjusted to add Skolem constants.

Remark 5 (Relation to Term Models for HOL). Our term model construction is much simpler than existing constructions of term models for HOL such as in [BBK04]. The reason is that we can interpret all terms as themselves (up to equivalence), including function terms (see also Rem. 2). On the contrary, a term Henkin model for HOL must interpret $\lambda x : A.t$ as a function, usually the function $s \mapsto t[s]$. While this is intuitively straightforward, it makes the rigorous construction more technically demanding.

6 Conclusion and Future Work

The present paper can be seen as the reference definition of DHOL. It gives the syntax and proof theory in all detail, subsuming and extending the presentation in [RRB23], and adds the model theory with standard and general models as well as soundness and completeness proofs.

Future work will use the model theory to better understand and utilize DHOL. We are particularly interested in attempting to find model-theoretical correctness proofs of the existing DHOL→HOL translation and of other theorem provers for DHOL. We also want to extend DHOL and its model theory with additional features such as polymorphism, choice operator, and additional types like product, refinement, or quotient types.

References

And86. P. Andrews. *An introduction to mathematical logic and type theory: to truth through proof*. Academic Press, 1986.

BBK04. C. Benzmüller, C. Brown, and M. Kohlhase. Higher-order semantics and extensionality. *Journal of Symbolic Logic*, 69(4):1027, 2004.

Chu40. A. Church. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5(1):56–68, 1940.

Coq15. Coq Development Team. The Coq Proof Assistant: Reference Manual. Technical report, INRIA, 2015.

dKA+15. L. de Moura, S. Kong, J. Avigad, F. van Doorn, and J. von Raumer. The Lean Theorem Prover (System Description). In A. Felty and A. Middeldorp, editors, *Automated Deduction*, pages 378–388. Springer, 2015.

Gor88. M. Gordon. HOL: A Proof Generating System for Higher-Order Logic. In G. Birtwistle and P. Subrahmanyam, editors, *VLSI Specification, Verification and Synthesis*, pages 73–128. Kluwer-Academic Publishers, 1988.

Hen49. L. Henkin. The completeness of the first-order functional calculus. *Journal of Symbolic Logic*, 14:159–166, 1949.

Hof94. M. Hofmann. On the interpretation of type theory in locally cartesian closed categories. In *Computer Science Logic*, pages 427–441. Springer, 1994.

LS86. J. Lambek and P. Scott. *Introduction to Higher-Order Categorical Logic*, volume 7 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 1986.

ML74. P. Martin-Löf. An Intuitionistic Theory of Types: Predicative Part. In *Proceedings of the '73 Logic Colloquium*, pages 73–118. North-Holland, 1974.

NBK24. J. Niederhauser, C. Brown, and C. Kaliszyk. Tableaux for automated reasoning in dependently-typed higher-order logic, 2024. under review.

Nor05. U. Norell. The Agda WiKi, 2005. http://wiki.portal.chalmers.se/agda.

RRB23. C. Rothgang, F. Rabe, and C. Benzmüller. Theorem Proving in Dependently Typed Higher-Order Logic. In B. Pientka and C. Tinelli, editors, *Automated Dedution*, pages 438–455. Springer, 2023.

SR09. K. Sojakova and F. Rabe. Translating a Dependently-Typed Logic to First-Order Logic. In A. Corradini and U. Montanari, editors, *Recent Trends in Algebraic Development Techniques*, pages 326–341. Springer, 2009.