Semantics for Dependently-Typed HOL

Florian Rabe

University Erlangen-Nuremberg

July 2026

Motivation

Flashback to 2006

Home > Automated Reasoning > Conference paper

First-Order Logic with Dependent Types

Conference paper

pp 377–391 | Cite this conference paper

Automated Reasoning

(IJCAR 2006)

Save conference paper

Florian Rabe

Access this chapter

my first conference paper, presented at IJCAR 20 years ago

And today

20 Years Later

- Higher-order logic with dependent types — What took so long?

And today

20 Years Later

- Higher-order logic with dependent types
- idea in my head since $\approx$ 2010, but ...

What took so long?

And today

20 Years Later

- Higher-order logic with dependent types — What took so long?
- idea in my head since $\approx$ 2010, but ...

Theory (Surprisingly) Difficult

- sound/complete translation DHOL$\rightarrow$HOL (dependency erasure)
  false results in literature?
- additional issues due to classical Booleans

And today

20 Years Later

- Higher-order logic with dependent types — What took so long?
- idea in my head since $\approx$ 2010, but ...

Theory (Surprisingly) Difficult

- sound/complete translation DHOL$\rightarrow$HOL (dependency erasure)
  false results in literature?
- additional issues due to classical Booleans

Time now right for DHOL

- HOL ATPs applicable to DHOL
- DHOL in reach as a next frontier for ATPs
- Appetite for intermediate language between DTT-ITPs and ATPs

Connected Work

Theory

- Rothgang's MSc thesis 2022 — advised by Rabe, Benzmüller
  DHOL and DHOL→HOL — CADE 2023, ACM ToCL 2025
- Subtyping: Rothgang, Rabe — FroCoS 2025
- Polymorphism: Ranalter, Rabe, Kaliszyk — FSCD 2026
- Semantics: Rabe — this paper, IJCAR 2026

Practice

- Tableaux prover: Niederhauser, Brown, Kaliszyk — IJCAR 2024
- Choice, Leo embedding: Ranalter, Brown, Kaliszyk — LPAR 2024
- TPTP standard: Ranalter, Kaliszyk, Rabe, Sutcliffe — FroCoS 2025
- Lean⇀DHOL for hammering: Maio, Bentkamp, Blanchette, Tourret
  ongoing, `https://nekoka-project.github.io/pubs/dtt-to-dhol-report.pdf`

Example

Vectors over A of length n

polymorphic and dependently-typed

- $\texttt{Vec}(A : \texttt{Type}, n : \mathbb{N}) : \texttt{Type}$ notation: write $\texttt{Vec}\ A\ n$ as A^n
- $\texttt{empty}(A : \texttt{Type}) : \ A^0$
- $\texttt{prepend}(A : \texttt{Type}, n : \mathbb{N}) : \ A \to A^n \to A^{1+n}$
- $\texttt{conc}(A : \texttt{Type}, m : \mathbb{N}, n : \mathbb{N}) : \ A^m \to A^n \to A^{m+n}$

Undecidability of typing

- axioms to make $\texttt{empty}(A)$ neutral element for concatenation, e.g.,

$$\forall V : A^n. \quad \underbrace{\texttt{conc}\ V\ \texttt{empty}(A)}_{: A^{n+0}} \quad =_{A^n} \quad \underbrace{\texttt{conc}\ \texttt{empty}(A)\ V}_{: A^{0+n}}$$

- well-formedness of formula depends on truth of $n + 0 =_{\mathbb{N}} n =_{\mathbb{N}} 0 + n$

What's so Difficult About it?

HOL+dependent types: intuitively straightforward

What's so Difficult About it?

HOL+dependent types: intuitively straightforward

But we can have only 2 out of these 3:

- decidable type-checking — type checking needs type equality
- straightforward type equality rule, e.g.,

$$\frac{A \equiv B \qquad m =_{\mathbb{N}} n}{A^m \equiv B^n}$$

  if types depend on terms, type equality depends on term equality
- term equality in axioms/assumptions — undecidable in general

What's so Difficult About it?

HOL+dependent types: intuitively straightforward

But we can have only 2 out of these 3:

- decidable type-checking — type checking needs type equality
- straightforward type equality rule, e.g.,

$$\frac{A \equiv B \qquad m =_{\mathbb{N}} n}{A^m \equiv B^n}$$

if types depend on terms, type equality depends on term equality

- term equality in axioms/assumptions — undecidable in general

Rocq, Agda, Lean etc.: definitional vs. propositional equality
PVS, DHOL: undecidable typing
open question: is that a price worth paying?

Results

Strict and Lax Models

Either way

- interpret types as sets, terms as elements
- interpret formulas as Boolean truth values
- soundness: provable formulas are true in all models

Strict models

- inductive interpretation function
- standard interpretation of functions
- theorem: interpretation commutes with substitution
- not complete

Generalization: Lax models

akin to HOL's Henkin models

- arbitrary interpretation function
- arbitrary interpretation of functions
- required: interpretation commutes with substitution
- complete

DHOL Syntax in 1 Slide

3 kinds of toplevel declarations

- type symbol $a(\Phi) : \mathtt{Type}$ $\qquad \mathtt{Vec}(A : \mathtt{Type}, n : \mathbb{N}) : \mathtt{Type}$
- typed constant $c(\Phi) : A$ $\qquad \mathtt{empty}(A : \mathtt{Type}) : A^0$
- axiom $(\Phi) \triangleright F$

  $(A : \mathtt{Type}, n : \mathbb{N}, V : A^n) \triangleright \mathtt{conc}\,\mathtt{empty}(A)\,V =_{A^n} \mathtt{conc}\,V\,\mathtt{empty}(A)$

Φ: list of type variables $u : \mathtt{Type}$ and term variables $x : A$

Types and Terms (Formulas F are terms of type `bool`)

- Types $A, B ::= a(\varphi) \mid u \mid \mathtt{bool} \mid \Pi_{x:A}\, B$ $\qquad$ no computation on types
- Terms $s, t, F ::= c(\varphi) \mid x \mid \lambda_{x:A}\, t \mid t\ s \mid s =_A t$

polymorphism is shallow: no binding of type variables

all connectives definable (almost) as usual

φ: substitution for the parameters in Φ

Strict Models

A strict model M consists of

- universe $\mathcal{U}$ of sets
- one base case for each toplevel declaration, e.g.,

declaration	case in model M
$\texttt{Vec}(A : \texttt{Type}, n : \mathbb{N}) : \texttt{Type}$	$\texttt{Vec}^M$ maps $\mathcal{U} \times [\![\mathbb{N}]\!]^M \to \mathcal{U}$
$\texttt{empty}(A : \texttt{Type}) : A^0$	$\texttt{empty}^M(U) \in [\![A^0]\!]^M_{A:=U}$ for $U \in \mathcal{U}$
axiom F	$[\![F]\!]^M = 1$

M induces sound interpretation function $[\![-]\!]^M_-$

- types to sets, terms to elements

$$\text{given} \quad \Phi \vdash t : A : \texttt{Type} \quad \text{we have} \quad [\![t]\!]^M_\alpha \in [\![A]\!]^M_\alpha \in \mathcal{U}$$

- soundness: given proof of $\Phi \vdash F$, we have $[\![F]\!]^M_\alpha = 1$

for all assignments α to variables in Φ

Strict Models: Interpretation Function

Standard Booleans

- $[\![\texttt{bool}]\!]^M_\alpha = \{0, 1\}$
- $[\![s =_A t]\!]^M_\alpha = 1 \quad$ iff $\quad [\![s]\!]^M_\alpha = [\![t]\!]^M_\alpha \in [\![A]\!]^M_\alpha$

defined connectives interpreted as usual

Standard functions

- HOL: $[\![A \to B]\!]^M_\alpha$ is the set of functions f

$$[\![A]\!]^M_\alpha \ni v \mapsto f(v) \in [\![B]\!]^M_\alpha$$

- DHOL: $[\![\Pi_{x:A} B]\!]^M_\alpha$ is the set of dependent functions f

$$[\![A]\!]^M_\alpha \ni v \mapsto f(v) \in [\![B]\!]^M_{\alpha, x:=v}$$

- λ-abstraction interpreted as function
- application interpreted as application

Lax Models

For HOL: Henkin models

- $[\![A \to B]\!]^M_\alpha$ is subset of functions $[\![A]\!]^M_\alpha$ to $[\![B]\!]^M_\alpha$

  not inductive for types

- standard interpretation of functions and application

  inductive for terms

Lax Models

For HOL: Henkin models

- $[\![A \to B]\!]^M_\alpha$ is subset of functions $[\![A]\!]^M_\alpha$ to $[\![B]\!]^M_\alpha$

  not inductive for types

- standard interpretation of functions and application

  inductive for terms

For DHOL: non-standard semantics of functions

- model may interpret function types as arbitrary sets

  not inductive for types or terms

- interpretation still commutes with substitution

  now a requirement, not a theorem

Lax Models

For HOL: Henkin models

- $[\![A \to B]\!]^M_\alpha$ is subset of functions $[\![A]\!]^M_\alpha$ to $[\![B]\!]^M_\alpha$

  not inductive for types

- standard interpretation of functions and application

  inductive for terms

For DHOL: non-standard semantics of functions

- model may interpret function types as arbitrary sets

  not inductive for types or terms

- interpretation still commutes with substitution

  now a requirement, not a theorem

- not totally arbitrary: elements of $f \in [\![A \to B]\!]^M_\alpha$ still function-like

$$@ : [\![A \to B]\!]^M_\alpha \to [\![A]\!]^M_\alpha \to [\![B]\!]^M_\alpha \quad = \quad \text{(definable)}$$

$$[\![f(t)]\!]^M_\alpha = @([\![f]\!]^M_\alpha)([\![t]\!]^M_\alpha)$$

Completeness for Lax Models

Theorem: model existence

- if a theory cannot prove falsity, it has a model
- proof: construct term model

  overall easier than typical constructions for Henkin models
  because function terms can be interpreted as themselves

Corollary: completeness

- given $\Phi \vdash F : \texttt{bool}$
- if $[\![F]\!]^M_\alpha = 1$ in every lax model M for every assignment α
- then $\Phi \vdash F$

Side Note: Functions with Preconditions

Intuition

- functions $\lambda_{x:A|F}\, t \;:\; \Pi_{x:A|F}\, B$
- precondition $x : A \vdash F : \texttt{bool}$
  - assumed in λ and Π
  - must be discharged when applying functions

Motivation

- allows defining
  - partial functions
  - refinement types — like in PVS
- low-hanging fruit if typing is undecidable anyway

Challenge

- supported by this semantics
- not supported by translation DHOL→HOL

ATP support open problem

Conclusion

Shootout to Peer Review

Original version of this paper rejected at IJCAR 2024

- did not cover polymorphism
- lax models looked kind of weird

looked daunting then, but paper much better now

Nice surprise: only minor changes needed

- all definitions stated via contexts and substitutions

  directly reusable when context allowed type variables
- interpretation must commute with substitution
  - automatically applies to type variables too
  - sufficient to make lax models better behaved

Conclusion

This paper

- reference definition for DHOL = HOL+
  - dependent types
  - polymorphism
  - preconditions (key feature for subtyping)
- ND calculus
- sound and complete semantics

Current work

- improve tool support
- collect more examples
- generalize practical calculi to DHOL e.g., resolution, superposition

Future work

- frontier for ATPs: can your prover handle DHOL?
- hammering: can ITPs export proof obligations in DHOL?