DHOL: HOL $+$ dependent types $+$ subtyping

Florian Rabe

Computer Science, University Erlangen-Nürnberg, Germany

HIM 2024

HOL+Dependent Types

Is This Formula True?

$$x = y \;\Rightarrow\; f(x) = f(y)$$

Is This Formula True?

$$\vdash x : A \qquad \vdash y : A \qquad \vdash f : (x : A) \to B(x)$$

$$\vdash f(x) : B(x) \qquad \vdash f(y) : B(y)$$

$$x =_A y \;\Rightarrow\; f(x) =_{???} f(y)$$

Is This Formula True?

$$\vdash x : A \qquad \vdash y : A \qquad \vdash f : (x : A) \to B(x)$$
$$\vdash f(x) : B(x) \qquad \vdash f(y) : B(y)$$

$$x =_A y \;\Rightarrow\; f(x) =_{???} f(y)$$

not even well-typed

Should it be?

2 Paths

$x =_A y \not\vdash B(x) \equiv B(y)$

- $+$ typing stays decidable
- $-$ need to explicitly transfer $u : B(x)$ into $B(y)$
- $+/-$ equality behaves more like isomorphism
- $-$ need two equalities (one decidable, one undecidable)

$x =_A y \vdash B(x) \equiv B(y)$

- $+$ intuitive, simple
- $+$ corresponds to mathematical practice
- $-$ typing undecidable

How bad is undecidable typing?

Very bad

- needs major change to system design   very difficult to retrofit
- breaks static checking of programs
  a major argument for using types in the first place

How bad is undecidable typing?

Very bad

- needs major change to system design — very difficult to retrofit
- breaks static checking of programs — a major argument for using types in the first place

But it's being done successfully

- NuPRL, PVS, Mizar
- mathematics

How bad is undecidable typing?

Very bad

- needs major change to system design   very difficult to retrofit
- breaks static checking of programs
  a major argument for using types in the first place

But it's being done successfully

- NuPRL, PVS, Mizar
- mathematics

Motivation for DHOL: reevaluate once a decade or so

- ATPs have gotten a lot better
- ultimate task (theorem proving) undecidable anyway

Grammar

STT

$$
\begin{array}{lcll}
T & ::= & \cdot & \\
 & | & T,\ a : \mathtt{Type} & \text{type symbol} \\
 & | & T,\ c : A & \text{typed constant} \\
 & & & \\
\Gamma & ::= & \cdot & \\
 & | & \Gamma,\ x : A & \text{typed variable} \\
 & & & \\
A, B & ::= & a \mid A \to B & \\
s, t, F, G & ::= & c \mid x \mid \lambda x : A.t \mid t\ s &
\end{array}
$$

Grammar

STT
HOL = STT + Booleans and equality

$$
\begin{array}{llll}
T & ::= & \cdot & \\
 & | & T,\ a : \mathtt{Type} & \text{type symbol} \\
 & | & T,\ c : A & \text{typed constant} \\
 & | & T,\ F & \text{global assumption (axiom)} \\
\Gamma & ::= & \cdot & \\
 & | & \Gamma,\ x : A & \text{typed variable} \\
 & | & \Gamma,\ F & \text{local assumption} \\
A, B & ::= & a \mid A \to B \mid \mathtt{bool} & \\
s, t, F, G & ::= & c \mid x \mid \lambda x : A.t \mid t\ s \mid s =_A t &
\end{array}
$$

note: type equality $\equiv$ is a judgment, not a Boolean term

Grammar

STT
HOL = STT + Booleans and equality
DHOL = the same but start with dependent types

T	$::=$	$\cdot$	
	$\mid$	$T,\ a : (\Gamma) \to \mathtt{Type}$	type symbol
	$\mid$	$T,\ c : A$	typed constant
	$\mid$	$T,\ F$	global assumption (axiom)
Γ	$::=$	$\cdot$	
	$\mid$	$\Gamma,\ x : A$	typed variable
	$\mid$	$\Gamma,\ F$	local assumption
A, B	$::=$	$a\ t\ \ldots\ t \mid (x : A) \to B \mid \mathtt{bool}$	
s, t, F, G	$::=$	$c \mid x \mid \lambda x : A.t \mid t\ s \mid s =_A t$	

note: type equality $\equiv$ is a judgment, not a Boolean term

Typing

$$
\begin{array}{lll}
T & ::= & \cdot \mid T,\ a : (\Gamma) \to \mathtt{Type} \mid T,\ c : A \mid T,\ F \\
\Gamma & ::= & \cdot \mid \Gamma,\ x : A \mid \Gamma,\ F \\
A, B & ::= & a\ t\ \dots\ t \mid (x : A) \to B \mid \mathtt{bool} \\
s, t, F, G & ::= & c \mid x \mid \lambda x : A.t \mid t\ s \mid s =_A t
\end{array}
$$

equality is for terms of the same type:

$$
\frac{\Gamma \vdash_T s : A \qquad \Gamma \vdash_T t : A}{\Gamma \vdash_T s =_A t : \mathtt{bool}}
$$

congruence for type formation:

$$
\frac{\begin{array}{l}\Gamma \vdash_T a : (x_1 : A_1, \dots, x_n : A_n) \to \mathtt{Type} \text{ in } T \\ \Gamma \vdash_T s_1 =_{A_1} t_1 \qquad \cdots \qquad \Gamma \vdash_T s_n =_{A_n[x_1/s_1,\dots,x_{n-1}/s_{n-1}]} t_n\end{array}}{\Gamma \vdash_T a\, s_1\ \dots\ s_n \equiv a\, t_1\ \dots\ t_n}
$$

the key rule that solves/creates all the problems

Typing: Subtleties

$$
\begin{array}{lcl}
T & ::= & \cdot \mid T,\ a : (\Gamma) \to \mathtt{Type} \mid T,\ c : A \mid T,\ F \\
\Gamma & ::= & \cdot \mid \Gamma,\ x : A \mid \Gamma,\ F \\
A, B & ::= & a\ t\ \ldots\ t \mid (x : A) \to B \mid \mathtt{bool} \\
s, t, F, G & ::= & c \mid x \mid \lambda x : A.t \mid t\ s \mid s =_A t
\end{array}
$$

$$
\frac{\Gamma \vdash_T a : (x_1 : A_1, \ldots, x_n : A_n) \to \mathtt{Type}\ \text{in}\ T \qquad \Gamma \vdash_T s_1 =_{A_1} t_1 \ldots \Gamma \vdash_T s_n =_{A_n} t_n}{\Gamma \vdash_T a\, s_1\ \ldots\ s_n \equiv a\, t_1\ \ldots\ t_n}
$$

Typing depends on assumptions in theory/context

order of declarations matters now!

Typing: Subtleties

$$
\begin{array}{lll}
T & ::= & \cdot \mid T, a : (\Gamma) \to \mathtt{Type} \mid T, c : A \mid T, F \\
\Gamma & ::= & \cdot \mid \Gamma, x : A \mid \Gamma, F \\
A, B & ::= & a\ t\ \ldots\ t \mid (x : A) \to B \mid \mathtt{bool} \\
s, t, F, G & ::= & c \mid x \mid \lambda x : A.t \mid t\ s \mid s =_A t \mid F \Rightarrow G
\end{array}
$$

$$
\frac{\Gamma \vdash_T a : (x_1 : A_1, \ldots, x_n : A_n) \to \mathtt{Type}\ \text{in}\ T \qquad \Gamma \vdash_T s_1 =_{A_1} t_1 \ldots \Gamma \vdash_T s_n =_{A_n} t_n}{\Gamma \vdash_T a\, s_1\ \ldots\ s_n \equiv a\, t_1\ \ldots\ t_n}
$$

Typing depends on assumptions in theory/context

order of declarations matters now!

Implication must be dependent:

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

same for conjunction, maybe disjunction

not definable from equality, needs to primitie

DHOL

$$\frac{\Gamma \vdash_T a : (x_1 : A_1, \ldots, x_n : A_n) \to \texttt{Type in } T \qquad \Gamma \vdash_T s_1 =_{A_1} t_1 \ldots \Gamma \vdash_T s_n =_{A_n} t_n}{\Gamma \vdash_T a\, s_1 \ \ldots \ s_n \equiv a\, t_1 \ \ldots \ t_n}$$

$$\frac{\Gamma \vdash_T s : A \qquad \Gamma \vdash_T t : A}{\Gamma \vdash_T s =_A t : \texttt{bool}}$$

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

Type-checks now for $f : (x : A) \to B$:

$$x =_A y \Rightarrow f(x) =_{B(x)} f(y)$$

Theorem Proving

By translation DHOL $\rightsquigarrow$ HOL

- Colin Rothgang, Rabe, Benzmüller — CADE 2023
- translation to HOL via dependency erasure
  - $a\, t_1 \ldots t_n \rightsquigarrow a, \qquad (x : A) \to B \rightsquigarrow A \to B$
  - sound/complete (only) for well-typed DHOL formulas
- recover dependencies via PER semantics
- Type-checker
  - follow the typing rules as usual
  - collect proof obligations and discharge them with HOL ATP

Native DHOL theorem prover

- Niederhauser, Brown, Kaliszyk — IJCAR 2024
- Tableaux-based
- Challenge: synthesizing well-typed terms

HOL + Dependent Types + Subtyping

Undecidable Typing: Double Down

DHOL is expensive because typing is undecidable.

But once we've paid that prize, other features are relatively cheap, e.g.,

- refinement types $A|p$ for $p : A \to \texttt{bool}$
- quotient types A/r for $r : A \to A \to \texttt{bool}$

both inherently undecidable

Refinement Types

$$
\begin{array}{ll}
A, B & ::= a\ t\ \ldots\ t \mid (x : A) \to B \mid \texttt{bool} \mid A|p \\
s, t, F, G, p & ::= c \mid x \mid \lambda x : A.t \mid t\ s \mid s =_A t \mid F \Rightarrow G
\end{array}
$$

Typing:

$$\frac{\Gamma \vdash_T p : A \to \texttt{bool} \quad \Gamma \vdash_T s : A \quad \Gamma \vdash_T p\, s}{\Gamma \vdash_T s : A|p}$$

$$\frac{\Gamma \vdash_T u : A|p}{\Gamma \vdash_T u : A}$$

Subtyping: $A|p <: A$

Quotient Types

Dual to refinement types

- refinement: canonical injection $A|p \to A$ noop
- quotient: canonical projection $A \to A/r$ noop?

Quotient Types

Dual to refinement types

- refinement: canonical injection $A|p \to A$ noop
- quotient: canonical projection $A \to A/r$ noop?

Yes — if the A in $s =_A t$ defines the equivalence relation to use

Quotient Types

Dual to refinement types

- refinement: canonical injection $A|p \to A$ noop
- quotient: canonical projection $A \to A/r$ noop?

Yes — if the A in $s =_A t$ defines the equivalence relation to use

$$\begin{array}{lcl} A, B & ::= & a\ t\ \ldots\ t \mid (x : A) \to B \mid \texttt{bool} \mid A/s \\ s, t, F, G & ::= & c \mid x \mid \lambda x : A.t \mid t\ s \mid s =_A t \mid F \Rightarrow G \end{array}$$

Typing:

$$\frac{\Gamma \vdash_T t : A \quad \Gamma \vdash_T r : A \to A \to \texttt{bool}}{\Gamma \vdash_T t : A/r}$$

$=_{A/r}$ is equivalence closure of r

$$\frac{\Gamma \vdash_T u : A/r \quad \Gamma x : A, x' : A, x =_{A/r} x' \vdash_T f(x) : B}{\Gamma \vdash_T f(u) : B}$$

Subtyping: $A <: A/r$

Subtype Hierarchy

$$A/\lambda x, y.\top$$

$$\vdots$$

$$A/r$$

$$\vdots$$

$$A|\lambda x.\top \equiv A \equiv A/\lambda x, y.\bot$$

$$\vdots$$

$$A|p$$

$$\vdots$$

$$A|\lambda x.\bot$$

Subtype Characterization

$$\Gamma \vdash_T A <: B$$

iff

$$\Gamma, x : A \vdash_T x : B$$

iff

$$\Gamma, x : A, y : A, x =_A y \vdash_T x =_B y$$

iff

$$\Gamma \vdash_T \forall x, y : A.\ x =_A y \Rightarrow x =_B y \qquad *$$

* not ATP-ready because true iff well-formed
and DHOL$\rightsquigarrow$HOL only sound/complete for well-formed formulas

Theorem Prover

- Translation to HOL generalizes easily
- Soundness/completeness as before
- Type-checker: as before but
  - needs subtype-checking
  - open question: how to reduce $A <: B$ to $\vdash F$

Trying to Eliminate Refinement/Quotient Types

Provable:

$$
\begin{array}{lcl}
(x:A) \to B|p & \equiv & ((x:A) \to B)|_{\lambda f.\forall x{:}A.\ p\ (f\ x)} \\
(x:A/r) \to B & \equiv & ((x:A) \to B)|_{\lambda f.\forall x,y{:}A.\ r\ x\ y \Rightarrow (f\ x) =_B (f\ y)} \\
(x:A|p) \to B & :> & ((x:A) \to B)/_{\lambda f,g.\forall x{:}A.\ (p\ x) \Rightarrow (f\ x) =_B (g\ x)} \\
(x:A) \to B/r & :> & ((x:A) \to B)/_{\lambda f,g.\forall x{:}A.\ r\ (f\ x)\ (g\ x)}
\end{array}
$$

What about the other two directions?

- $<:$ for refined domain: not true — $B(x)$ might depend on $p\ x$
- $<:$ for quotiented codomain: can be axiom