universität innsbruck

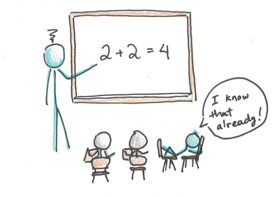
The Dependently Typed Higher-Order Form for the TPTP World

Daniel Ranalter, Cezary Kaliszyk, Florian Rabe, Geoff Sutcliffe

Overview

- Higher-Order Logic
- (Why) Going Beyond
- *Dependently-Typed* Higher-Order Logic
- TPTP Integration
- Conclusion

Syntax



HOL Syntax

- A HOL presentation suited for the extension
- Simple Type Theory a la Church with a base-type for booleans, implication and equality

$$
\begin{array}{llll}
T & ::= & \circ \mid T, a\ tp \mid T, c : A \mid T, F & \text{theory} \\
\Gamma & ::= & \bullet \mid \Gamma, x : A \mid \Gamma, F & \text{context} \\
A, B & ::= & a \mid o \mid A \to B & \text{types} \\
t, u, v & ::= & x \mid \lambda x : A.t \mid tu \mid t \Rightarrow u \mid t =_A u \mid \bot & \text{terms}
\end{array}
$$

- Con- and Disjunction, Quantification, etc. can be encoded
- $\forall f : nat \to nat \to nat.((\lambda n : nat.f\ 0\ n) =_{nat \to nat} f\ 0)$

Judgements

What can we do with it?

- $\forall f : nat \to nat \to nat.((\lambda n : nat.f\ 0\ n) =_{nat \to nat} f\ 0)$?
- How to reason about statements?
- Judgements:

$\Gamma \vdash t$	Well-formed boolean term t is provable
$\Gamma \vdash t : A$	Term t is of (well-formed) type A
$\Gamma \vdash A \equiv B$	Well-formed types *A* and *B* are equal
$\Gamma \vdash A\ tp$	Type A is well-formed

Example

Natural Numbers - Theory

types	constants/functions	axioms
$nat\ tp$	$0 : nat$	$\forall n, m : nat.(plus\ (suc\ m)\ n =_{nat} plus\ m\ (suc\ n))$
	$suc : nat \rightarrow nat$	$\forall n : nat.(plus\ 0\ n =_{nat} n)$
	$plus : nat \rightarrow nat \rightarrow nat$	

Natural Numbers - Judgements

- $\Gamma \vdash \forall i, j, k : nat.(plus\ i\ (plus\ j\ k) =_{nat} plus\ (plus\ i\ j)\ k)$
- $\Gamma \vdash suc\ (plus\ 0\ (suc\ 0)) : nat$

Going Beyond

Automated Reasoning & TPTP

- Started with CNF and FOL
- Grew with demand and power
- Now includes HOL, polymorphism, etc

→ Increasingly complex type systems

Going Beyond

Automated Reasoning & TPTP

- Started with CNF and FOL
- Grew with demand and power
- Now includes HOL, polymorphism, etc

→ Increasingly complex type systems

Interactive Reasoning

- Other end of Expressivity/Automation spectrum
- Often incorporates Depdendent Types
- Increasingly powerful automation through hammers

Going Beyond

Automated Reasoning
TPTP

- Started with CNF and FOL
- Grew with demand and power
- Now includes HOL, polymorphism, etc

→ Increasingly complex type systems

eractive Reasoning

- Other end of Expressivity/Automation spectrum
- Often incorporates Depdendent Types
- Increasingly powerful automation through hammers

Changes and Additions

DHOL Syntax

Now we can extend HOL to dependent types by replacing every occurrence of type-formation...

$$
\begin{array}{llll}
T & ::= & \circ \mid T, a\ tp \mid T, x : A \mid T, F & \text{theory} \\
\Gamma & ::= & \bullet \mid \Gamma, x : A \mid \Gamma, F & \text{context} \\
A, B & ::= & a \mid o \mid A \to B & \text{types} \\
t, u, v & ::= & x \mid \lambda x : A.t \mid tu \mid t \Rightarrow u \mid t =_A u \mid \bot & \text{terms}
\end{array}
$$

Changes and Additions

DHOL Syntax

Now we can extend HOL to dependent types by replacing every occurrence of type-formation...

$$
\begin{array}{llll}
T & ::= & \circ \mid T, a : (\Pi x : A.)^{*}\ tp \mid T, x : A \mid T, F & \text{theory} \\
\Gamma & ::= & \bullet \mid \Gamma, x : A \mid \Gamma, F & \text{context} \\
A, B & ::= & a t_1 \ldots t_n \mid o \mid \Pi x : A.B & \text{types} \\
t, u, v & ::= & x \mid \lambda x : A.t \mid tu \mid t \Rightarrow u \mid t =_A u \mid \bot & \text{terms}
\end{array}
$$

... with the more general, dependent variant

Rules

HOL-ND to DHOL-ND

$$\frac{\Gamma \vdash s : o \quad \Gamma \vdash t : o}{\Gamma \vdash (s \Rightarrow t) : o}\Rightarrow\text{Type} \qquad \frac{\Gamma \vdash s : o \quad \Gamma, s \vdash t}{\Gamma \vdash s \Rightarrow t}\Rightarrow$$

$$\frac{\Gamma \vdash A \equiv A' \quad \Gamma \vdash B \equiv B'}{\Gamma \vdash A \to B \equiv A' \to B'}\to\text{Cong} \qquad \frac{\Gamma \vdash A\ tp}{\Gamma \vdash A \equiv A}\text{tpRefl}$$

Rules

HOL-ND to DHOL-ND

$$\frac{\Gamma \vdash s : o \quad \Gamma, s \vdash t : o}{\Gamma \vdash (s \Rightarrow t) : o} \Rightarrow\text{Type} \qquad \frac{\Gamma \vdash s : o \quad \Gamma, s \vdash t}{\Gamma \vdash s \Rightarrow t} \Rightarrow$$

$$\frac{\Gamma \vdash A \equiv A' \quad \Gamma \vdash B \equiv B'}{\Gamma \vdash A \to B \equiv A' \to B'} \to\text{Cong} \qquad \frac{\Gamma \vdash A\ tp}{\Gamma \vdash A \equiv A} \text{tpRefl}$$

Rules

HOL-ND to DHOL-ND

$$\frac{\Gamma \vdash s : o \quad \Gamma, s \vdash t : o}{\Gamma \vdash (s \Rightarrow t) : o} \Rightarrow\text{Type} \qquad \frac{\Gamma \vdash s : o \quad \Gamma, s \vdash t}{\Gamma \vdash s \Rightarrow t} \Rightarrow$$

$$\frac{\Gamma \vdash A \equiv A' \quad \Gamma, x : A \vdash B \equiv B'}{\Gamma \vdash \Pi x : A.B \equiv \Pi x' : A'.B'} \Pi\text{Cong} \qquad \frac{\Gamma \vdash A\ tp}{\Gamma \vdash A \equiv A} \text{tpRefl}$$

Rules

HOL-ND to DHOL-ND

$$\frac{\Gamma \vdash s : o \quad \Gamma, s \vdash t : o}{\Gamma \vdash (s \Rightarrow t) : o} \Rightarrow\text{Type} \qquad \frac{\Gamma \vdash s : o \quad \Gamma, s \vdash t}{\Gamma \vdash s \Rightarrow t} \Rightarrow$$

$$\frac{\Gamma \vdash A \equiv A' \quad \Gamma, x : A \vdash B \equiv B'}{\Gamma \vdash \Pi x : A.B \equiv \Pi x' : A'.B'} \Pi\text{Cong}$$

$\Gamma \vdash$

$\Gamma \vdash A$

tpRefl

Rules

HOL-ND to DHOL-ND

$$\frac{\Gamma \vdash s : o \quad \Gamma, s \vdash t : o}{\Gamma \vdash (s \Rightarrow t) : o} \Rightarrow \mathrm{Type} \qquad \frac{\Gamma \vdash s : o \quad \Gamma, s \vdash t}{\Gamma \vdash s \Rightarrow t} \Rightarrow$$

$$\frac{\Gamma \vdash A \equiv A' \quad \Gamma, x : A \vdash B \equiv B'}{\Gamma \vdash \Pi x : A.B \equiv \Pi x' : A'.B'} \Pi\mathrm{Cong}$$

$$\frac{a : (\Pi x_1 : A_1,\ \ldots,\ \Pi x_n : A_n) \in \Gamma \quad \Gamma \vdash s_1 =_{A_1} t_1 \quad \ldots \quad \Gamma \vdash s_n =_{A_n[x_1/s_1,\ldots,x_{n-1}/s_{n-1}]} t_n}{\Gamma \vdash a\ s_1 \ldots s_n \equiv a\ t_1 \ldots t_n}$$

tpRefl

Example

Fixed Length Lists of Natural Numbers - Theory

types	constants/functions
$lst : \Pi n : nat\ tp$	$nil : lst\ 0$
	$cons : \Pi n : nat.nat \to lst\ n \to lst\ (suc\ n)$
	$app : \Pi n, m : nat.lst\ n \to lst\ m \to lst\ (plus\ n\ m)$

Fixed Length Lists of Natural Numbers - Judgements

- $\Gamma \vdash \forall n : nat.\forall x : lst\ n.(app\ 0\ n\ nil\ x =_{lst\ n} x)$

Erasure

Simplifying things by making them more complicated

- To increase usability, an erasure from DHOL to HOL exists
- Basic idea: Capture information lost during erasure in a Partial Equivalence Relation (PER)

Erasure, abridged

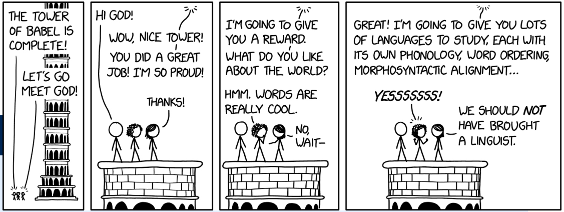
$\overline{a : \Pi x_1 : A_1, \ldots, \Pi x_n : A_n\; tp} =$

- $a\; tp$
- $a^* : \overline{A_1} \to \ldots \to \overline{A_n} \to a \to a \to o$
- Set of axioms establishing PER properties for a^*

$\overline{x : A} =$

- $x : \overline{A}$
- $A^* x x$

DHF



New TPTP dialect

Extending THF to DHF is conservative:

- THF: Allows types to depend on types (Polymorphism), e.g.

```
lst: $tType > $tType
nat_lst: lst @ nat
cons: !>[A: $tType]: ( A > (lst @ A) > (lst @ A) )
```

- DHF: Allows types to depend on terms, e.g.

```
lst: nat > $tType
empty_lst: lst @ zero
cons: !>[N: nat]: ( foo > (lst @ N) > (lst @ (suc @ N)) )
```

Transformation from DHOL to DHF - Type Declaration

$$app : \Pi n, m : nat. lst\; n \rightarrow lst\; m \rightarrow lst\; (plus\; n\; m)$$

DHF

Transformation from DHOL to DHF - Type Declaration

$$app : \Pi n, m : nat . lst\ n \to lst\ m \to lst\ (plus\ n\ m)$$

DHF

Transformation from DHOL to DHF - Type Declaration

$$\texttt{app}: \ \Pi n, m : nat.\, lst\ n \to lst\ m \to lst\ (plus\ n\ m)$$

DHF

Transformation from DHOL to DHF - Type Declaration

$$\texttt{app: !>[}n, m : nat\texttt{]:(}\ lst\ n \rightarrow lst\ m \rightarrow lst\ (plus\ n\ m)\ \texttt{)}$$

DHF

Transformation from DHOL to DHF - Type Declaration

`app: !>[N:nat, M:nat]:(` $\mathit{lst}\ n \rightarrow \mathit{lst}\ m \rightarrow \mathit{lst}\ (\mathit{plus}\ n\ m)$ `)`

DHF

Transformation from DHOL to DHF - Type Declaration

`app: !>[N:nat, M:nat]:(` *lst n* `>` *lst m* `>` *lst (plus n m)* `)`

DHF

Transformation from DHOL to DHF - Type Declaration

```
app: !>[N:nat, M:nat]:( (lst @ N) > (lst @ M) > (lst @
(plus @ N @ M)) )
```

DHF

Transformation from DHOL to DHF - Type Declaration

```
app: !>[N:nat, M:nat]:( (lst @ N) > (lst @ M) > (lst @ (plus @ N @ M)) )
```

Transformation from DHOL to DHF - Axiom/Conjecture

$$\Gamma \vdash \forall n : nat.\forall x : lst\, n.(app\; 0\; n\; nil\; x =_{lst\, n} x)$$

DHF

Transformation from DHOL to DHF - Type Declaration

```
app: !>[N:nat, M:nat]:( (lst @ N) > (lst @ M) > (lst @ (plus @ N @ M)) )
```

Transformation from DHOL to DHF - Axiom/Conjecture

$$\Gamma \vdash \forall n : nat.\forall x : lst\ n.(app\ 0\ n\ nil\ x =_{lst\ n} x)$$

DHF

Transformation from DHOL to DHF - Type Declaration

```
app: !>[N:nat, M:nat]:( (lst @ N) > (lst @ M) > (lst @ (plus @ N @ M)) )
```

Transformation from DHOL to DHF - Axiom/Conjecture

```
![N:nat, X:lst @ N]:( (app @ 0 @ N @ nil @ X) = X )
```

DHF

Supporting Infrastructure

- DLash - Extension of Lash
- Leo-III's Logic Embedding Tool extensions for
  - Type checking DHF problems
  - Erasing DHF to THF
- MMT/DHOL implementation for developing formalizations
- TPTP World support in the form of
  - TPTP4X - DHF pretty printer
  - BNFParser - creates abstract syntax tree from DHF problems
  - SystemOnTPTP prover based on Logic Embedding Tool and Vampire
  - several more

Conclusion

- DHOL is an extension of HOL with dependent types
- Reasoning is intuitive due to classical logic and extensional equality
- However, type checking is now (potentially) difficult
- DHF is the corresponding extension of THF and already has some support
- Further adoption of dependently typed reasoning could close gap between ATP and ITP
- DHF is our contribution to that goal

Conclusion

- DHOL is an extension of HOL with dependent types
- Reasoning is intuitive due to classical logic and extensional equality
- However, type checking is now (potentially) difficult
- DHF is the corresponding extension of THF and already has some support
- Further adoption of dependently typed reasoning could close gap between ATP and ITP
- DHF is our contribution to that goal

Thank you for your attention!