universität innsbruck

Dataset of Problems for Learning in Dependent Higher-Order Logic

AITP’24

Daniel Ranalter and Cezary Kaliszyk

Introducing DHOL (Rothgang et al. CADE 2023)

Introducing DHOL (Rothgang et al. CADE 2023)

DHOL Syntax

T	$::=$	$\circ \mid \Gamma, a : (\Pi x : A.)^{*}\ tp \mid \Gamma, c : A \mid \Gamma, F$	theory
Γ	$::=$	$\cdot \mid \Gamma, x : A \mid \Gamma, F$	context
A, B	$::=$	$a\ t_1 \ldots t_n \mid bool \mid \Pi x :\ A.B$	types
t, u, v	$::=$	$x \mid \lambda x : A.t \mid tu \mid t \Rightarrow u \mid\ t =_A u \mid \perp$	terms

Introducing DHOL (Rothgang et al. CADE 2023)

DHOL Syntax

T	$::=$	$\circ \mid \Gamma, a : (\Pi x : A.)^{*}\ tp \mid \Gamma, c : A \mid \Gamma, F$	theory
Γ	$::=$	$\cdot \mid \Gamma, x : A \mid \Gamma, F$	context
A, B	$::=$	$a\ t_1 \ldots t_n \mid bool \mid \Pi x : A.B$	types
t, u, v	$::=$	$x \mid \lambda x : A.t \mid tu \mid t \Rightarrow u \mid t =_A u \mid \bot$	terms

Judgments asserting ...

$\vdash T\ thy$

$\vdash_T \Gamma\ Ctx$

$\Gamma \vdash_T F$

$\Gamma \vdash_T t : A$

$\Gamma \vdash_T A\ tp$

$\Gamma \vdash_T A \equiv B$

Introducing DHOL (Rothgang et al. CADE 2023)

DHOL Syntax

T	$::=$	$\circ \mid \Gamma, a : (\Pi x : A.)^{*}\ tp \mid \Gamma, c : A \mid \Gamma, F$	theory
Γ	$::=$	$\cdot \mid \Gamma, x : A \mid \Gamma, F$	context
A, B	$::=$	$a\ t_1\ \dots\ t_n \mid bool \mid \Pi x : A.B$	types
t, u, v	$::=$	$x \mid \lambda x : A.t \mid tu \mid t \Rightarrow u \mid t =_A u \mid \perp$	terms

Judgments asserting ...

... well-formedness of the theory

... well-formedness of the context

... provability of boolean terms

... well-formedness of types

... equality of well-formed types

... typing of terms

Why DHOL?

Current Situation

- Several DTT ITPs exist (Lean, Agda, Rocq, ...)
- ... but they are intensional and/or intuitionistic
  - i.e. differing judgmental and provable equalities
- Next to no support for DTT ATPs

Why DHOL?

Current Situation

- Several DTT ITPs exist (Lean, Agda, Rocq, ...)
- ... but they are intensional and/or intuitionistic
  - i.e. differing judgmental and provable equalities
- Next to no support for DTT ATPs

Distinguishing Features

Formulation closer to HOL as used in theorem proving:

- Classical
- Extensional
  - i.e. judgemental equality is reflected into provable equality
- Equipped with translation to HOL

Why DHOL?

However...

- Translation to HOL makes problems more complicated
- Native DHOL-ATPs that exist are comparatively weak

Why DHOL?

However...

- Translation to HOL makes problems more complicated
- Native DHOL-ATPs that exist are comparatively weak

Solution

Learning!

... but to learn, we need data to learn from.

Dataset

Challenges

- How to present the problems
- The number of problems - TPTP THF consists of over 5000 problems
- The number of domains - TPTP THF problems exist in 35 domains

Our plan $\hookrightarrow$ translate existing problems!

Dataset

Challenges

- How to present the problems
- The number of problems - TPTP THF consists of over 5000 problems
- The number of domains - TPTP THF problems exist in 35 domains

Our plan $\hookrightarrow$ translate existing problems!

- Translation of dependently typed problems tends to be involved
  - Correctness of typing is undecidable
  - In general, correctness of translation itself is not obvious

Dataset

TPTP

- No officially sanctioned standard (yet)
- However, easy to adapt existing TH1:
  - The universal type quantifier `!>[]:` is promoted to type binder
  - The types of variables in the binder's list can now take the form of (fully-applied) dependent (function-)types

```
thf(elem_type,type,elem: $tType).
thf(nat_type,type,nat: $tType).
thf(suc_type,type,suc: nat > nat).
thf(list_type,type,list: $tType).
thf(cons_type,type,cons:
  (elem > list > list)).
```

Dataset

TPTP

- No officially sanctioned standard (yet)
- However, easy to adapt existing TH1:
  - The universal type quantifier `!>[]:` is promoted to type binder
  - The types of variables in the binder’s list can now take the form of (fully-applied) types

```
thf(elem_type,type,elem: $tType).
thf(nat_type,type,nat: $tType).
thf(suc_type,type,suc: nat > nat).
thf(list_type,type,list: nat > $tType).
thf(cons_type,type,cons:
   !>[N:nat]:  (elem > (list @ N) > (list @ (suc @ N)))).
```

Dataset

Sources

- At this point made mostly by hand
- But we want to have a large number of sources for problems to increase diversity
- Currently:
  - Original formulations
  - Finite Sets in the Rocq prover standard library
  - (polymorphic) Red-Black Trees in Agda
- Other interesting sources:
  - Further examples from Rocq and Agda
  - Mizar Mathematical Library
  - Lean
- Some examples:

Matrices

```
thf(matrix_type,type,matrix: nat > nat > $tType ).
thf(mempty_type,type,mempty: !>[N: nat] : (matrix @ zero @ N) ).
thf(mcons_type,type,mcons:
  !>[M:nat,N:nat]: ((list @ N) > (matrix @ M @ N) > (matrix @ (suc @ M) @ N))).
thf(ladd_type,type,ladd:
  !>[N:nat]: ((list @ N) > (list @ N) > (list @ N))).
thf(madd_type,type,madd:
  !>[M:nat,N:nat]: ((matrix @ M @ N) > (matrix @ M @ N) > (matrix @ M @ N))).
thf(madd_mempty,axiom,
 ![N:nat]: ((madd @ zero @ N @ (mempty @ N) @ (mempty @ N))
 = (mempty @ N))).
thf(madd_mcons,axiom,
 ![M:nat,N:nat,L1:list @ N,M1:matrix @ M @ N,L2:list @ N,M2:matrix @ M @ N]:
 ((madd @ (suc @ M) @ N @ (mcons @ M @ N @ L1 @ M1) @ (mcons @ M @ N @ L2 @ M2))
 = (mcons @ M @ N @ (ladd @ N @ L1 @ L2) @ (madd @ M @ N @ M1 @ M2)))).
```

Finite Sets from Rocq

```
Inductive t : nat -> Set :=
|F1 : forall {n}, t (S n)
|FS : forall {n}, t n -> t (S n).

Definition caseS (P: forall {n}, t (S n) -> Type)
  (P1: forall n, @P n F1) (PS : forall {n} (p: t n), P (FS p))
  {n} (p: t (S n)) : P p

Definition rectS (P: forall {n}, t (S n) -> Type)
  (P1: forall n, @P n F1) (PS : forall {n} (p: t (S n)), P p -> P (FS p))
  forall {n} (p: t (S n)), P p
```

Finite Sets from Rocq

```
thf(fin_type,type,fin: nat > $tType).
thf(f1_type,type,f1: !>[A:nat]: (fin @ (suc @ A))).
thf(fs_type,type,fs: !>[A:nat]: ((fin @ A) > (fin @ (suc @ A)))).
thf(fin_case,axiom,![P:(!>[N:nat]: ((fin @ (suc @ N)) > $o))]:
  (((![N:nat]: (P @ N @ (f1 @ N)))
  & (![N:nat]: (![F:(fin @ N)]: (P @ N @ (fs @ N @ F)))))
  => (![N:nat]: (![F:(fin @ (suc @ N))]: (P @ N @ F))))).
thf(fin_rec,axiom,![P:(!>[N:nat]: ((fin @ (suc @ N)) > $o))]:
  (((![N:nat]: (P @ N @ (f1 @ N)))
  & (![N:nat]: (![F:(fin @ (suc @ N))]: ((P @ N @ F)
    => (P @ (suc @ N) @ (fs @ (suc @ N) @ F))))))
  => (![N:nat]: (![F:(fin @ (suc @ N))]: (P @ N @ F))))).
```

Red-Black Trees from Agda

```
data Color : Set where
  R : Color
  B : Color

data Tree : Color → Nat → Set where
  E  : Tree B Zero
  TR : ∀ {n} → Tree B n → A → Tree B n → Tree R n
  TB : ∀ {n c1 c2} → (Tree c1 n) → A → (Tree c2 n)
    → Tree B (Suc n)
```

Red-Black Trees from Agda

```
thf(color_type,type,color: $tType).
thf(black_type,type,black: color).
thf(red_type,type,red: color).
thf(tree_type,type,tree: $tType > color > nat > $tType ).
thf(leaf_type,type,leaf:
  !>[A:$tType]: (tree @ A @ black @ zero)).
thf(redTree_type,type,rt: !>[A:$tType,N:nat]:
  ((tree @ A @ black @ N) > A > (tree @ A @ black @ N)
  > (tree @ A @ red @ N))).
thf(blackTree_type,type,bt:!>[A:$tType,N:nat,C1:color,C2:color] :
  ((tree @ A @ C1 @ N) > A > (tree @ A @ C2 @ N)
  > (tree @ A @ black @ (suc @ N)))).
```

Dependent Choice on Lists

```
thf(nat_type,type,nat: $tType).
thf(zero_type,type,zero: nat).
thf(suc_type,type,suc: nat > nat).
thf(list_type,type,list: nat > $tType).
thf(nil_type,type,nil: (list @ zero)).
thf(cons_type, type,cons:
  !>[N:nat]: (nat > (list @ N) > (list @ (suc @ N)))).
thf(hd_type,type, hd:
  !>[LENMINUSONE:nat]: ((list @ (suc @ LENMINUSONE)) > nat)).
thf(hd,axiom,![LEN:nat,H:nat,L:(list @ LEN)]:
  ((hd @ LEN @ (cons @ LEN @ H @ L)) = H)).
thf(c,conjecture,(hd @ zero @ (@+[X: (list @ (suc @ zero))]:
  ((hd @ zero @ X) = zero))) = zero).
```

Further Steps

Wishlist

- Automate translation
- Establish formal TPTP format
- Collect more problems
- Experiment with learning from these problems in Lash and others

Further Steps

Wishlist

- Automate translation
- Establish formal TPTP format
- Collect more problems
- Experiment with learning from these problems in Lash and others

Thank you for your attention!