First-Order Proof Tactics in Higher Order Logic Theorem Provers

Joe Hurd
joe.hurd@cl.cam.ac.uk

University of Cambridge
Contents

- Introduction
- Logical Interface
- First-Order Calculi
- Porting to PVS
- Conclusion
HOL already has a proof tactic for first-order logic with equality, called \texttt{MESON_TAC}.

- Based on the model elimination calculus.
- Added to HOL in 1996 by John Harrison.

Building the core distribution of HOL uses \texttt{MESON_TAC} to prove 1779 subgoals:

- Up from 1428 just five months ago.
- A further 2024 subgoals in the HOL examples.

Clearly a useful tool for interactive proof.
A typical HOL subgoal proved using \texttt{MESON_TAC}:

\[(G) \quad \forall x, y, z. \text{divides } x \ y \implies \text{divides } x \ (z * y)\]

We pass as arguments the following theorems:

\[(D) \quad \vdash \forall x, y. \text{divides } x \ y \iff \exists z. y = z * x\]

\[(C) \quad \vdash \forall x, y. x * y = y * x\]

\[(A) \quad \vdash \forall x, y, z. (x * y) * z = x * (y * z)\]

The tactic succeeds because the formula

\[(D) \land (C) \land (A) \implies (G)\]

is a tautology in first-order logic with equality.
To prove the HOL subgoal $g$

1. Convert the negation of $g$ to CNF

   $\vdash \neg g \iff \exists \bar{a}. (\forall v_1. c_1) \land \cdots \land (\forall v_n. c_n)$

2. Map each HOL term $c_i$ to a first-order logic clause.

3. The first-order prover finds a refutation for the clauses.

4. The refutation is translated to the HOL theorem

   $\vdash \{ (\forall v_1. c_1), \ldots, (\forall v_n. c_n) \}$

5. Finally, use (A) and (B) to deduce

   $\vdash g$
Contents

- Introduction
- Logical Interface
- First-Order Calculi
- Porting to PVS
- Conclusion
Logical Interface

- Can program versions of first-order calculi that work directly on HOL terms.
  - But types (and \( \lambda \)'s) add complications;
  - and then the mapping from HOL terms to first-order logic is hard-coded.

- Would like to program versions of the calculi that work on standard first-order terms, and have someone else worry about the mapping to HOL terms.
  - Then coding is simpler and the mapping is flexible;
  - but how can we keep track of first-order proofs, and automatically translate them to HOL?
First-order Logical Kernel

Use the ML type system to create an LCF-style logical kernel for clausal first-order logic:

signature Kernel = sig
  (* An ABSTRACT type for theorems *)
  eqtype thm

  (* Destruction of theorems is fine *)
  val dest_thm : thm → formula list × proof

  (* But creation is only allowed by these primitive rules *)
  val AXIOM : formula list → thm
  val REFL : term → thm
  val ASSUME : formula → thm
  val INST : subst → thm → thm
  val FACTOR : thm → thm
  val RESOLVE : formula → thm → thm → thm
  val EQUALITY : formula → int list → term → bool → thm → thm
end
Making Mappings Modular

The logical kernel keeps track of proofs, and allows the HOL mapping to first-order logic to be modular:

signature Mapping =
nig
  (* Mapping HOL goals to first-order logic *)
  val map_goal : HOL.term → FOL.formula list

  (* Translating first-order logic proofs to HOL *)
  type Axiom_map = FOL.formula list → HOL.thm
  val translate_proof : Axiom_map → Kernel.thm → HOL.thm

end

Implementations of Mapping simply provide HOL versions of the primitive inference steps in the logical kernel, and then all first-order theorems can be translated to HOL.
Type Information?

- It is not necessary to include type information in the mapping from HOL terms to first-order terms/formulas.
- Principal types can be inferred when translating first-order terms back to HOL.
- But for various reasons the untyped mapping occasionally fails.
  - We’ll see examples of this later.
Four Mappings

We have implemented four mappings from HOL to first-order logic.

Their effect is illustrated on the HOL goal $n < n + 1$:

<table>
<thead>
<tr>
<th>Mapping</th>
<th>First-order formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>first-order, untyped</td>
<td>$n &lt; n + 1$</td>
</tr>
<tr>
<td>first-order, typed</td>
<td>$(n : \mathbb{N}) &lt; ((n : \mathbb{N}) + (1 : \mathbb{N}) : \mathbb{N})$</td>
</tr>
<tr>
<td>higher-order, untyped</td>
<td>$\uparrow ((&lt; \cdot n) \cdot ((+ \cdot n) \cdot 1))$</td>
</tr>
<tr>
<td>higher-order, typed</td>
<td>$\uparrow (((&lt; : \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{B}) \cdot (n : \mathbb{N}) : \mathbb{N} \rightarrow \mathbb{B}) \cdot (((+ : \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N}) \cdot (n : \mathbb{N}) : \mathbb{N} \rightarrow \mathbb{N}) \cdot (1 : \mathbb{N}) : \mathbb{N}) : \mathbb{B})$</td>
</tr>
</tbody>
</table>
Mapping Efficiency

- Effect of the mapping on the time taken by model elimination calculus to prove a HOL version of Łoś’s ‘nonobvious’ problem:

<table>
<thead>
<tr>
<th>Mapping</th>
<th>untyped</th>
<th>typed</th>
</tr>
</thead>
<tbody>
<tr>
<td>first-order</td>
<td>1.70s</td>
<td>2.49s</td>
</tr>
<tr>
<td>higher-order</td>
<td>2.87s</td>
<td>7.89s</td>
</tr>
</tbody>
</table>

- These timing are typical, although 2% of the time higher-order, typed does beat first-order, untyped.

- We run in untyped mode, and if an error occurs during proof translation then restart search in typed mode.
  - Restarts 17+3 times over all 1779+2024 subgoals.
Mapping Coverage

```
higher-order ✓  first-order ×

⊢ ∀f, s, a, b. (∀x. f x = a) ∧ b ∈ image f s ⇒ (a = b)
   (f has different arities)

⊢ ∃x. x
   (x is a predicate variable)

⊢ ∃f. ∀x. f x = x
   (f is a function variable)

typed ✓  untyped ×

⊢ length ([ ] : ℤ^*) = 0 ∧ length ([ ] : ℜ^*) = 0 ⇒
   length ([ ] : ℜ^*) = 0
   (indistinguishable terms)

⊢ ∀x. S K x = I
   (extensionality applied too many times)

⊢ (∀x. x = c) ⇒ a = b
   (bad proof via ⊤ = ⊥)
```
Contents

- Introduction
- Logical Interface
- **First-Order Calculi**
- Porting to PVS
- Conclusion
First-Order Calculi

- Implemented ML versions of several first-order calculi.
  - Model elimination; resolution; the delta preprocessor.
  - Trivial reduction to our first-order primitive inferences.
- Can run them simultaneously using time slicing.
  - They cooperate by contributing to a central pool of unit clauses.
- Used the TPTP problem set for most of the tuning.
  - Verified correlation between performance on TPTP and performance on HOL subgoals.
Similar search strategy (but not identical!) to MESON_TAC.

Incorporated three major optimizations:
- Ancestor pruning (Loveland).
- Unit lemmaizing (Astrachan and Stickel).
- Divide & conquer searching (Harrison).

Unit lemmaizing gave a big win.
- The logical kernel made it easy to spot unit clauses.
- Surprise: divide & conquer searching can prevent useful unit clauses being found!
Resolution

- Implements ordered resolution and ordered paramodulation.

- Powerful equality calculus allows proofs way out of `MESON_TAC`'s range:

\[
\begin{align*}
\vdash & \ (\forall x, y. \ x \ast y = y \ast x) \land \\
& (\forall x, y, z. \ (x \ast y) \ast z = x \ast (y \ast z)) \Rightarrow \\
& a \ast b \ast c \ast d \ast e \ast f \ast g \ast h = h \ast g \ast f \ast e \ast d \ast c \ast b \ast a
\end{align*}
\]

- Had to tweak it for HOL in two important ways:
  - Avoid paramodulation into a typed variable.
  - Sizes of clauses shouldn’t include types.
• Schumann’s idea: perform shallow resolutions on clauses before passing them to model elimination prover.

• Our version: for each predicate $P/n$ in the goal, use model elimination to search for unit clauses of the form $P(X_1, \ldots, X_n)$ and $\neg P(Y_1, \ldots, Y_n)$.

• Doesn’t directly solve the goal, but provides help in the form of unit clauses.
TPTP Evaluation

Total “unsatisfiable” problems in TPTP v2.4.1 = 3297

<table>
<thead>
<tr>
<th></th>
<th>rmd</th>
<th>rm</th>
<th>rd</th>
<th>r</th>
<th>md</th>
<th>m</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>rmd</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1819</td>
</tr>
<tr>
<td>*</td>
<td>+20</td>
<td>+238</td>
<td>+351</td>
<td>+575</td>
<td>+591</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rm</td>
<td>+11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1811</td>
</tr>
<tr>
<td>*</td>
<td>+231</td>
<td>+338</td>
<td>+575</td>
<td>+591</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rd</td>
<td>+10</td>
<td>+12</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>1592</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>+14</td>
<td>+10</td>
<td>+5</td>
<td>*</td>
<td></td>
<td></td>
<td>1483</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>md</td>
<td>+72</td>
<td>+81</td>
<td>+283</td>
<td>+383</td>
<td></td>
<td></td>
<td>1316</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>+69</td>
<td>+78</td>
<td>+277</td>
<td>+377</td>
<td>+2</td>
<td>*</td>
<td>1297</td>
</tr>
</tbody>
</table>

First-Order Proof Tactics in Higher Order Logic Theorem Provers – Joe Hurd – p.20/24
Contents

- Introduction
- Logical Interface
- First-Order Calculi
- **Porting to PVS**
- Conclusion
Porting to PVS

- The first-order logical kernel and calculi are freely available as a Standard ML package.
- ‘All’ that remains is to implement a mapping from PVS to first-order logic.
- The mapping and proof translation would work in exactly the same way as the HOL mapping, except for one situation.
- During proof translation, it is often necessary to lift first-order terms to higher-order logic terms. In PVS, this operation would generate type correctness conditions (TCCs).
- Is it always possible to automatically prove TCCs generated in this way?
Contents

- Introduction
- Logical Interface
- First-Order Calculi
- Porting to PVS
- Conclusion
Conclusions

- We have presented a HOW-TO for integrating first-order provers as tactics in higher-order logic theorem provers.
  - The technology has proven itself in HOL.
  - Hopefully it can be transferred to PVS (and others).
- The logical interface allowed free experimentation with the first-order calculi.
- Resolution performed better than model elimination on HOL subgoals.
  - Even on the biased set of \texttt{MESON\_TAC} subgoals!
- Combining first-order calculi resulted in a much better prover, both for TPTP problems and HOL subgoals.