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

First-Order Proof Tactics: Why?

- HOL already has a proof tactic for first-order logic with equality, called `MESON_TAC`.
  - Based on the model elimination calculus.
  - Added to HOL in 1996 by John Harrison.
- Building the core distribution of HOL uses `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.

First-Order Proof Tactics: Example

A typical HOL subgoal proved using `MESON_TAC`:

$$\text{(G)} \qquad \forall x, y, z.\ \text{divides}\ x\ y \Rightarrow \text{divides}\ x\ (z * y)$$

We pass as arguments the following theorems:

$$\begin{array}{lll} \text{(D)} & \vdash & \forall x, y.\ \text{divides}\ x\ y \iff \exists z.\ y = z * x \\ \text{(C)} & \vdash & \forall x, y.\ x * y = y * x \\ \text{(A)} & \vdash & \forall x, y, z.\ (x * y) * z = x * (y * z) \end{array}$$

The tactic succeeds because the formula

$$\text{(D)} \land \text{(C)} \land \text{(A)} \Rightarrow \text{(G)}$$

is a tautology in first-order logic with equality.

First-Order Proof Tactics: How?

To prove the HOL subgoal g

1. Convert the negation of g to CNF

$$\text{(A)} \qquad \vdash \; \neg g \iff \exists\, \vec{a}.\ (\forall\, \vec{v_1}.\ c_1) \land \cdots \land (\forall\, \vec{v_n}.\ c_n)$$

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

$$\text{(B)} \qquad \{(\forall\, \vec{v_1}.\ c_1), \ldots, (\forall\, \vec{v_n}.\ c_n)\} \;\vdash\; \bot$$

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 λ'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 =
sig
  (* 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 $n < n + 1$:

Mapping	**First-order formula**
first-order, untyped	$n < n + 1$
first-order, typed	$(n : \mathbb{N}) < ((n : \mathbb{N}) + (1 : \mathbb{N}) : \mathbb{N})$
higher-order, untyped	$\uparrow ((< \,.\, n) \,.\, ((+ \,.\, n) \,.\, 1))$
higher-order, typed	

$$\uparrow (((< : \mathbb{N} \to \mathbb{N} \to \mathbb{B}) \,.\, (n : \mathbb{N}) : \mathbb{N} \to \mathbb{B}) \,.\, \\ (((+ : \mathbb{N} \to \mathbb{N} \to \mathbb{N}) \,.\, (n : \mathbb{N}) : \mathbb{N} \to \mathbb{N}) \,.\, (1 : \mathbb{N}) : \mathbb{N}) : \mathbb{B})$$

Mapping Efficiency

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

Mapping	untyped	typed
first-order	1.70s	2.49s
higher-order	2.87s	7.89s

- 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 $\surd$ first-order $\times$

$$\vdash \ \forall f, s, a, b.\ (\forall x.\ f\ x = a) \ \wedge \ b \in \mathsf{image}\ f\ s \ \Rightarrow \ (a = b)$$

(f has different arities)

$$\vdash \ \exists x.\ x$$

(x is a predicate variable)

$$\vdash \ \exists f.\ \forall x.\ f\ x = x$$

(f is a function variable)

typed $\surd$ untyped $\times$

$$\vdash \ \mathsf{length}\ ([\,] : \mathbb{N}^*) = 0 \ \wedge \ \mathsf{length}\ ([\,] : \mathbb{R}^*) = 0 \ \Rightarrow$$
$$\mathsf{length}\ ([\,] : \mathbb{R}^*) = 0$$

(indistinguishable terms)

$$\vdash \ \forall x.\ \mathsf{S}\ \mathsf{K}\ x = \mathsf{I}$$

(extensionality applied too many times)

$$\vdash \ (\forall x.\ x = c) \ \Rightarrow \ a = b$$

(bad proof via $\top = \bot$)

Contents

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.

Model Elimination

- 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:

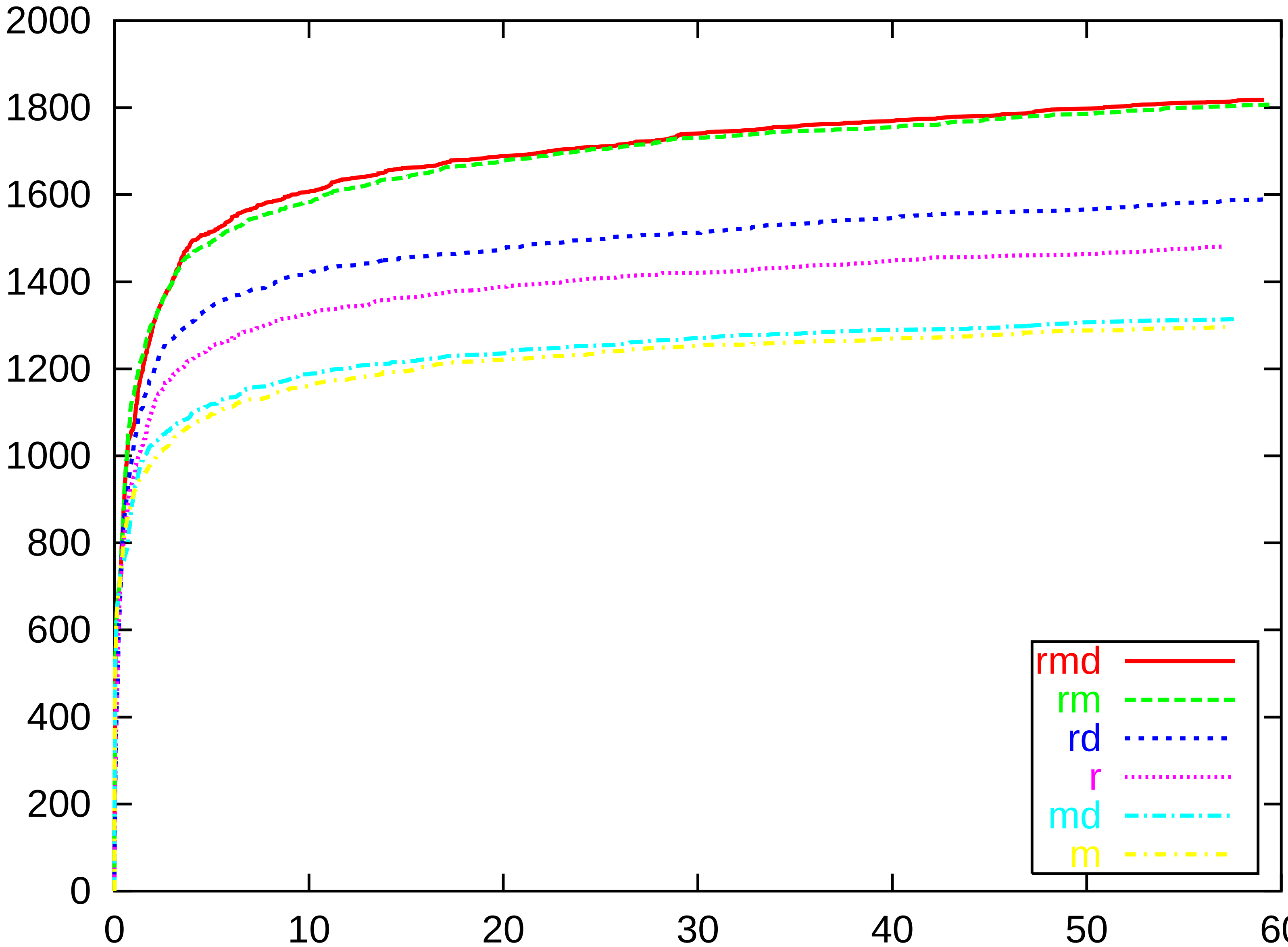
$$\begin{aligned}\vdash\quad & (\forall x, y.\ x * y = y * x) \wedge \\ & (\forall x, y, z.\ (x * y) * z = x * (y * z)) \Rightarrow \\ & a * b * c * d * e * f * g * h = h * g * f * e * d * c * b * a\end{aligned}$$

- Had to tweak it for HOL in two important ways:
  - Avoid paramodulation into a typed variable.
  - Sizes of clauses shouldn't include types.

Delta Preprocessor

- 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



TPTP Evaluation

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

	rmd	rm	rd	r	md	m	total
rmd	$*$	$\overset{+20}{95.0\%}$	$\overset{+238}{99.5\%}$	$\overset{+351}{99.5\%}$	$\overset{+575}{99.5\%}$	$\overset{+591}{99.5\%}$	1819
rm	$\underline{+11}$	$*$	$\overset{+231}{99.5\%}$	$\overset{+338}{99.5\%}$	$\overset{+575}{99.5\%}$	$\overset{+591}{99.5\%}$	1811
rd	$\underline{+10}$	$\underline{+12}$	$*$	$\overset{+114}{99.5\%}$	$\overset{+558}{99.5\%}$	$\overset{+571}{99.5\%}$	1592
r	$\underline{+14}$	$\underline{+10}$	$\underline{+5}$	$*$	$\overset{+549}{99.5\%}$	$\overset{+562}{99.5\%}$	1483
md	$\underline{+72}$	$\underline{+81}$	$\underline{+283}$	$\underline{+383}$	$*$	$\overset{+21}{99.5\%}$	1316
m	$\underline{+69}$	$\underline{+78}$	$\underline{+277}$	$\underline{+377}$	$\underline{+2}$	$*$	1297

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

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 `MESON_TAC` subgoals!
- Combining first-order calculi resulted in a much better prover, both for TPTP problems and HOL subgoals.