An LCF-Style Interface between HOL and First-Order Logic

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

University of Cambridge

Introduction

- Many HOL goals can be proved by first-order calculi.
- Can tackle them by programming versions of the calculi that work directly on HOL terms.
  - But types (and λ's) add complications;
  - and then it's not easy to change the way HOL terms are mapped to first-order logic.
- Would like to program a version of the calculi that works 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 ASSUME  : formula → thm
  val INST    : subst → thm → thm
  val FACTOR  : thm → thm
  val RESOLVE : formula → thm → 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 trans_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.
  - This wouldn’t be the case if the type system was undecidable (e.g., the PVS type system).
- 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	$B(@(@(<, n), @(@(+, n), 1)))$
higher-order, typed	

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

Mapping Efficiency

- We coded up ML versions of simple first-order calculi.
  - Model elimination; resolution; the delta preprocessor.
  - Can be used with any mapping to prove HOL goals.
  - This proof tool is released with HOL4.
- Effect of the mapping on the time taken to prove a HOL version of Łoś's 'nonobvious' problem:

Mapping	untyped	typed
first-order	3.50s	4.89s
higher-order	3.76s	17.73s

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

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)

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

(f chosen to be $(\wedge)\top$)

Conclusions

- It's possible to modularize the mapping from HOL to first-order logic.
  - This allows simpler implementation of proof tools;
  - and different mappings for different application areas.
- The untyped mapping shows that including type information is not necessary, but often advisable.
- The higher-order mapping gives surprisingly large coverage on HOL goals, but is rather slow.
- Future Work: Use the mappings to create a flexible interface to 'industrial strength' first-order provers.