Porting Proofs Between HOL Implementations

Sebastian Skalberg

skalberg@in.tum.de

Technische Universität München

Overview

- Motivation
- Recording and saving proofs
- Replaying proofs
- Conclusion

Motivation: HOL4 vs. Isabelle/HOL

HOL4:

- Industrial strength.
- Massive formalization of real analysis.
- Primitive user interface.

Isabelle/HOL:

- Massive formalization of Java.
- Very accessible user interface.
- Not many results in real analysis.

Porting Theorems

- Oracles
  - Little workload
  - Not safe

Porting Theorems

- Oracles
  - Little workload
  - Not safe
- Port proof scripts
  - New system has all functionality from original system
  - Completely safe
  - Large workload
  - Must be done for each new proof tactic

Porting Theorems

- Oracles
  - Little workload
  - Not safe
- Port proof scripts
  - New system has all functionality from original system
  - Completely safe
  - Large workload
  - Must be done for each new proof tactic
- Port proofs
  - Medium workload
  - Completely safe
  - Tools not available in new system

Design Goals

Export from HOL:

- The extension should be as unobtrusive as possible, should “clearly” not have an impact on the soundness of the system.
- Proof recording should be reasonably fast.

Import into Isabelle/HOL:

- Only import new types, terms, and theorems.
- Produce valid Isar theories.

Also, proof file format should not be specific to any given ML system.

Concrete Representation

```
datatype proof_info
  = Info of {disk_info: (string * string) option ref,
             references: int ref}

datatype proof = Proof of proof_info * proof_content
     and proof_content
       = PRefl of term
       | PInstT of proof * (hol_type,hol_type) Lib.subst
       | PSubst of proof list * term * proof
       | PAbs of proof * term
       | PDisch of proof * term
       | PMp of proof * proof
       | PHyp of term
       | PAxm of string * term
       | ...

datatype thm = THM of tag * term HOLset.set * term * proof
```

Efficiency

Due to their size, several problems present themselves once we want to write proofs to disk:

- Proofs, terms, and types shared in memory need to be shared on disk too.
- Even with sharing, efficient I/O is required.

On the other hand, we do not need to

- remove types (variables may differ in only their type).
- compress terms.
- normalize proofs (they are first-order).

The Poincaré Principle

It has been argued that proof objects for proofs in classical logics (as opposed to constructive type theories) would be infeasible due to the explicit proofs of β-convertibility. As with other practical issues, this is a question of trading the complexity of proof checkers for the size of proof objects.

	logical kernel	HOL4 kernel	proof objects
matching modulo	α	α	$\beta\eta$
inference rules	8	42	33

Importing in Isabelle/HOL

The overall strategy is to implement an HOL4 kernel in Isabelle/HOL. Some of the technical problems encountered were:

- All variables occuring in an Isabelle term must have either different names or the same type, not so in HOL4.
- Isabelle has a logical level more than HOL4.
- Printing of terms in Isabelle, so that Isabelle could parse it back.

Informing Isabelle/HOL

Many type and term constants of HOL4 are already present in Isabelle/HOL, as well as many theorems. To minimize the import, we make sure that the Isabelle variants are used whenever possible.

- For type and term constants, this means proving the relevant abstract properties of the constant, and telling Isabelle about the renaming.
- For theorems, we make an exhaustive search of the Isabelle/HOL theorem data base for “equivalent” theorems.

The Actual Import

The import has two stages:

- First, Isabelle takes configuration information and proof objects, and spits out a number of Isar theories.
- Second, the new Isar theories are processed like any other in Isabelle/HOL.

Example Configuration Section

```
import_theory sum

type_maps
  sum > "+"

const_maps
  INL      > Inl
  INR      > Inr
  ISL      > HOL4Compat.ISL
  ISR      > HOL4Compat.ISR
  OUTL     > HOL4Compat.OUTL
  OUTR     > HOL4Compat.OUTR
  sum_case > Datatype.sum.sum_case

ignore_thms sum_TY_DEF sum_ISO_DEF IS_SUM_REP INL_DEF
  INR_DEF sum_axiom sum_Axiom

end_import
```

Example Output

```
setup_theory sum

lemma ISL_OR_ISR: "ALL x. ISL x | ISR x"
  by (import sum ISL_OR_ISR)

lemma INL: "ALL x. ISL x --> Inl (OUTL x) = x"
  by (import sum INL)

lemma INR: "ALL x. ISR x --> Inr (OUTR x) = x"
  by (import sum INR)

lemma sum_case_cong: "ALL (M::'b + 'c) (M'::'b + 'c) (f::'b => 'a)
   g::'c => 'a. M = M' &
   (ALL x::'b. M' = Inl x --> f x = (f'::'b => 'a) x) &
   (ALL y::'c. M' = Inr y --> g y = (g'::'c => 'a) y) -->
   sum_case f g M = sum_case f' g' M'"
  by (import sum sum_case_cong)

end_setup
```

Empirical Data

Time taken to build the base HOL4 system:

original system	with proof recording	with proof saving
42m	44m	1h32m

The base HOL4 system comprises 5,000 named theorems. These are saved in 80,000 files, taking up 350Mb (13Mb when gzipped).
The generating of Isar theories in Isabelle/HOL takes about 42m, the replaying 1h22m.

Conclusion

- Proof recording has been added to HOL4 and HOL Light (the latter by Steven Obua).
- Replaying of proof objects in Isabelle/HOL is implemented and functional.
- Recording, saving, and replaying of proofs seems acceptably efficient.
- Data sharing is essential, probably more so than compression per se.