Portable Higher Order Logic Proofs

Joe Hurd and **Rob Arthan**

Galois, Inc. and Lemma 1 Ltd.
joe@galois.com and rda@lemma-one.com

TEITP Workshop
Wednesday 11 August 2010

Motivation

- Interactive theorem proving is growing up.
    - The FlySpeck project is driving the HOL Light theorem prover towards a formal proof of the Kepler sphere-packing conjecture.
    - The seL4 project recently completed a 20 man-year verification of an operating system kernel in the Isabelle theorem prover.
- There is a need for theory engineering techniques to support these major verification efforts.
    - Theory engineering is to proving as software engineering is to programming.
    - *"Proving in the large."*
    - *"Mixed language proving."*

OpenTheory Proof Archive

- In theory, proofs are immortal.
- In practice, proofs that depend on theorem prover implementations bit-rot at an alarming rate.
- Idea: Archive proofs as theory packages.
- The goal of the OpenTheory project is to transfer the benefits of package management to logical theories.
- Slogan: *Logic is an ABI for mathematics.*

Project Approach

- The initial case study for the project is Church's simple theory of types, extended with Hindley-Milner style type variables.
  - The logic implemented by HOL4, HOL Light and ProofPower.
- By focusing on a concrete case study we aim to investigate the issues surrounding:
  - Designing theory languages portable across theorem prover implementations.
  - Discovering design techniques for reusable theories.
  - Uploading, installing and upgrading theory packages from online repositories.
  - Building a standard theory library.

Tactic Proof Scripts

Porting theories between higher order logic theorem provers is currently a painful process of transcribing scripts that call proof tactics:

Code (Typical HOL Light tactic script proof)

```
let NEG_IS_ZERO = prove
    (`!x. neg x = Zero <=> x = Zero`,
     MATCH_MP_TAC N_INDUCT THEN
     REWRITE_TAC [neg_def] THEN
     MESON_TAC [N_DISTINCT]);;
```

Difficulty: Every theorem prover implements a subtly different set of tactics, the behaviour of which evolves across versions.

Theorem Provers in the LCF Design

- A theorem $\Gamma \vdash \phi$ states *"if all of the hypotheses* Γ *are true, then so is the conclusion* ϕ*"*.
- The novelty of Milner's Edinburgh LCF ITP was to make `theorem` an abstract ML type.
- Values of type `theorem` can only be created by a small logical kernel which implements the primitive inference rules of the logic.
- Soundness of the whole ML ITP thus reduces to soundness of the logical kernel.

$\mathsf{THM} \subseteq \mathbb{P}\{\mathsf{Blue}, \mathsf{White}, \ldots\}$

Compiling Theories

- Idea: Instead of storing the source tactic script, store a compiled version of the theory by fully expanding the tactics to a primitive inference proof.
- Benefit: The logic almost never changes, so the compiled theories will never suffer from bit rot.
  - Whereas tactic scripts can break every time the tactics change.
- Benefit: The compiled proof need only store the inferences that contribute to the proof.
  - Whereas tactic scripts often explore many dead ends before finding a valid proof.
- Drawback: Once the theory has been compiled to a proof, it is difficult to change it.
  - So theories should be compiled only when they are stable enough to be archived and shared.

OpenTheory Articles

- A theory of higher order logic consists of:
  1. A set Γ of assumption sequents.
  2. A set Δ of theorem sequents.
- For assurance, we want evidence that $\Gamma \vdash \Delta$,
  E.g., via ML type `THM` or a formal proof.
- This talk will present the OpenTheory article file format for higher order logic theories.
- This is a standards-based approach to theories:
  - Enables simple import and export between theorem prover implementations.
  - Evidence of correctness is a replayable low-level proof providing a way to independently check proofs.

Proofs are (Stack-Based) Programs

- Proof articles are represented as programs for a stack-based virtual machine.
  - There are commands for building types and terms, and performing primitive inferences.
  - The stack avoids the need to store the whole proof in memory.
- A dictionary is used to support structure sharing.
  - The article should preserve structure sharing as much as possible to avoid a space blow-up.
  - **Implementation Challenge:** Structure-sharing substitution.

Article Commands

- Article files consist of a sequence of commands, one per line.
- Commands such as `var` construct data to be used as arguments in primitive inferences.

Definition (The "var" article command)

```
var
    Pop a type ty; pop a name n; push a variable
    with name n and type ty.

    Stack:  Before:  Type ty
                     :: Name n
                     :: stack
            After:   Term (mk_var (n,ty))
                     :: stack
```

Article Primitive Inferences

- There are 8 primitive inference commands (such as `refl`).
- There is also one command for defining new constants, and one for defining new type operators.

Definition (The "refl" article command)

```
refl
    Pop a term t; push a theorem with no
    hypotheses and conclusion t = t.

    Stack:  Before:  Term t
                     :: stack
            After:   Thm ( |- t = t )
                     :: stack
```

The OpenTheory Logical Kernel

$$\frac{}{\vdash t = t}\,\text{refl } t \qquad \frac{}{\{\phi\} \vdash \phi}\,\text{assume } \phi \qquad \frac{\Gamma \vdash \phi = \psi \quad \Delta \vdash \phi}{\Gamma \cup \Delta \vdash \psi}\,\text{eqMp}$$

$$\frac{\Gamma \vdash t = u}{\Gamma \vdash (\lambda v.\ t) = (\lambda v.\ u)}\,\text{absThm } v \qquad \frac{\Gamma \vdash f = g \quad \Delta \vdash x = y}{\Gamma \cup \Delta \vdash f\ x = g\ y}\,\text{appThm}$$

$$\frac{\Gamma \vdash \phi \quad \Delta \vdash \psi}{(\Gamma - \{\psi\}) \cup (\Delta - \{\phi\}) \vdash \phi = \psi}\,\text{deductAntisym} \qquad \frac{\Gamma \vdash \phi}{\Gamma[\sigma] \vdash \phi[\sigma]}\,\text{subst } \sigma$$

$$\frac{}{\vdash (\lambda v.\ t)\ u = t[u/v]}\,\text{betaConv } ((\lambda v.\ t)\ u) \qquad \frac{}{\vdash c = t}\,\text{defineConst } c\ t$$

$$\frac{\vdash \phi\ t}{\vdash abs\ (rep\ a) = a \quad \vdash \phi\ r = (abs\ (rep\ r) = r)}\,\text{defineTypeOp } n\ abs\ rep\ vs$$

Article Assumptions

- The `axiom` command is used to import an assumption to the theory.

Definition (The "axiom" article command)

```
axiom
    Pop a term c; pop a list of terms h;
    push the new axiom h |- c and add it
    to the theory assumptions.

    Stack:  Before:  Term c
                     :: List [Term h1, ..., Term hn]
                     :: stack
            After:   Thm ( {h1, ..., hn} |- c )
                     :: stack
```

Article Theorems

- The thm command is used to export a theorem from the theory.

Definition (The "thm" article command)

```
thm
    Pop a term c; pop a list of terms h; pop a
    theorem th; check the theorem {h1, ..., hn} |- c
    is alpha-equivalent to th and (if so) add it to
    the theory theorems.

    Stack:  Before:  Term c
                     :: List [Term h1, ..., Term hn]
                     :: Thm th
                     :: stack
            After:   stack
```

Article Theories

- The result of executing a proof article is a theory $\Gamma \triangleright \Delta$.
  - Γ is the set of imported assumptions.
  - Δ is the set of exported theorems.
- The definitions made by the article manifest themselves as constants and types that appear in Δ but not in Γ.

Example Article Theory

Theory (Proof article defining the “unit” type)

```
input-types: -> bool
input-consts: ! /\ = ==> ? T select
assumed:
  |- !t. (\x. t x) = t
  |- T = ((\p. p) = \p. p)
  |- (!) = \P. P = \x. T
  |- (==>) = \p q. (p /\ q) = p
  |- !P x. P x ==> P ((select) P)
  |- (/\) = \p q. (\f. f p q) = \f. f T T
  |- (?) = \P. !q. (!x. P x ==> q) ==> q
defined-types: unit
defined-consts: one
thms:
  |- !v. v = one
```

HOL Light Experiment

- To test the article format, we instrumented HOL Light v2.20 to emit articles for all of the theory files in the distribution.
- Proofs fully expanded to primitive inferences are large.
- However, the following compression techniques are effective on proof articles:
  - The equivalent of hash-consing for types, terms and theorems.
  - Dead-inference elimination.
- Concatenating all of the articles and compressing results in an article with the following characteristics:
  - Contains 769,138 primitive inferences.
  - Applying `gzip` produces an 18Mb file.

Compressing the HOL Light Theories

HOL Light theory	article (Kb)	comp. (Kb)	comp. saving	gzip'ed article (Kb)	gzip'ed comp. (Kb)	comp. saving
num	1,820	813	56%	227	113	51%
arith	27,469	7,548	73%	2,884	1,015	65%
wf	29,277	6,330	79%	3,222	861	74%
calc_num	3,922	1,570	60%	374	203	46%
normalizer	2,845	688	76%	300	92	70%
grobner	2,417	748	70%	257	103	60%
ind-types	10,625	4,422	59%	1,274	599	53%
list	12,368	4,870	61%	1,485	673	55%
realax	23,628	7,989	67%	2,519	1,070	58%
calc_int	2,844	861	70%	314	119	63%
realarith	16,275	4,684	72%	1,326	589	56%
real	30,031	9,346	69%	3,179	1,217	62%
calc_rat	2,555	1,166	55%	289	157	46%
int	40,617	9,546	77%	3,465	1,249	64%
sets	168,586	30,315	83%	17,514	4,048	77%
iter	207,324	32,422	85%	17,557	4,199	77%
cart	20,351	3,632	83%	2,076	495	77%
define	82,185	16,409	81%	8,157	2,175	74%

Summary

- The article format for higher order logic theories is now stable.
- Looking for volunteers to build tools to import and export articles for HOL theorem provers.
- Get in touch using the project web page:

  `http://gilith.com/research/opentheory`