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
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.”
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)

```ml
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 \( \Gamma \) are true, then so is the conclusion \( \phi \).”

- The novelty of Milner’s Edinburgh LCF ITP was to make theorem an abstract ML type.

- Values of type \textit{theorem} can only be created by a small \textit{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.

\[ \text{THM} \subseteq \mathbb{P}\{\text{Blue, White, \ldots}\} \]
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.
A theory of higher order logic consists of:

1. A set $\Gamma$ of assumption sequents.
2. A set $\Delta$ 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)**

```plaintext
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
```
There are 8 primitive inference commands (such as \texttt{refl}).

There is also one command for defining new constants, and one for defining new type operators.

**Definition (The “refl” article command)**

\texttt{refl}

Pop a term \( t \); push a theorem with no hypotheses and conclusion \( t = t \).

Stack: Before: Term \( t \)

:: stack

After: Thm ( \( \vdash t = t \) )

:: stack
The OpenTheory Logical Kernel

\[
\begin{align*}
\Gamma \vdash t = u & \quad \Gamma \vdash (\lambda v. t) = (\lambda v. u) \\
\Gamma \vdash \phi & \quad \Delta \vdash \psi & \quad \Gamma \cup \Delta \vdash \phi = \psi \\
(\Gamma - \{\psi\}) \cup (\Delta - \{\phi\}) \vdash \phi = \psi & \quad \Gamma \vdash \phi & \quad \Delta \vdash x = y \\
(\lambda v. t) u & = t[u/v] & \quad \Gamma \vdash c = t \\
\Gamma \vdash \phi t & \quad \Gamma \vdash abs (rep a) = a & \quad \Gamma \vdash \phi r = (abs (rep r) = r)
\end{align*}
\]
The **axiom** command is used to import an assumption to the theory.

**Definition (The “axiom” article command)**

```plaintext
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
The \texttt{thm} command is used to export a theorem from the theory.

\begin{mdframed}
\textbf{Definition (The “thm” article command)}

\texttt{thm}

Pop a term $c$; pop a list of terms $h$; pop a theorem $th$; check the theorem \{h_1, \ldots, h_n\} $\vdash c$ is alpha-equivalent to $th$ and (if so) add it to the theory theorems.

Stack: Before: Term $c$

\begin{verbatim}
:: List [Term h_1, \ldots, Term h_n]
:: Thm th
:: stack
\end{verbatim}

After: stack
\end{mdframed}
The result of executing a proof article is a theory $\Gamma \triangleright \Delta$.
- $\Gamma$ is the set of imported assumptions.
- $\Delta$ is the set of exported theorems.

The definitions made by the article manifest themselves as constants and types that appear in $\Delta$ but not in $\Gamma$. 
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

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