Challenges in Using OpenTheory to Transport Harrison's HOL Model from HOL Light to HOL4

Ramana Kumar

Computer Laboratory, University of Cambridge, UK
Ramana.Kumar@cl.cam.ac.uk

Abstract

OpenTheory is being used for the first time (in work to be described at ITP 2013) as a tool in a larger project, as opposed to in an example demonstrating OpenTheory's capability. The tool works, demonstrating its viability. But it does not work completely smoothly, because the use case is somewhat at odds with OpenTheory's primary design goals. In this extended abstract, we explore the tensions between the goals that OpenTheory-like systems might have, and question the relative importance of various kinds of use. My hope is that describing issues arising from work in progress will stimulate fruitful discussion relevant to the development of proof exchange systems.

1 The OpenTheory Vision: Reusable Theory Packages

OpenTheory [3] is a format for representing theories in higher-order logic, inspired by an analogy: theories are software packages, which may depend on one another, and proofs are bytecode for a portable virtual machine. Just as a piece of software (especially a library) can be useful in the development of another (library or application), so can the definitions and theorems in one formal development be a useful component for another. For example, existing theories about floating-point numbers and finite words might both be used in the development a formal model of an instruction set architecture. The problem is that these theories may have been developed in different provers and may not be directly compatible.

There are multiple HOL prover implementations, analogous to different platforms, including HOL Light, HOL4, ProofPower, and Isabelle/HOL, each with a broadly similar logic and architecture, but with significant differences in their libraries of native theories and their integrated proof development (e.g. automation) tools. The aim of OpenTheory is to support theory engineering across provers, akin to the software engineering applicable to large software developments. (OpenTheory does not yet support provers using different logics like Coq or Twelf: bridging the differences between HOL-based systems is enough of a challenge already.) Large formal developments that span multiple provers require methods of exchange, and benefit when those methods promote reuse of native theories as opposed to isolated duplications.

OpenTheory's approach to exchange is based on a standard format for low-level proofs coupled with a standard library of theory packages. The standard format enables capture and replay of proof work in any prover supporting the format. The standard library factors out various core theories that are likely to be used by many developments and provides a standard interface for depending on such library theories by reusing a prover's native theories as opposed to rederiving them on import. (For example, if the standard library has some basic theorems about `foo` and HOL4 has an equivalent constant `FOO`, then we map `foo` to `FOO` on import and enable reuse of all of HOL4's `FOO` theorems, rather than importing `foo` theorems, proving $\vdash$ `foo` $=$ `FOO`, and mediating any combined reasoning through the equivalence.)

When using OpenTheory, the goal therefore is to create a package that is independent of any particular prover in the sense that its dependencies are all standard theories that are supposed to be supported by every prover. Furthermore, the package should have a clear topic and a clean interface (the theorems and definitions it exports should be meaningful, useful results, as opposed to auxiliary constants or intermediate lemmas), because these criteria promote reusability in different environments: if intermediate steps can be handled entirely within OpenTheory, they never need to be proved at the endpoints.

2 The Use Case: Proof Transport, with Modifications

John Harrison [2] created a formal model of HOL in HOL Light and proved it sound[1]. We (Myreen et al. [5]) are working on tools in HOL4 for verifying implementations of code derived from shallow embeddings. Harrison's soundness proof for HOL supports an excellent case study for our tools wherein we will verify an implementation of HOL Light, but it lives in a different prover. Our options to use his work are:

- Manually port the high-level script files from HOL Light to HOL4.
- Automatically port the low-level proofs from HOL Light to HOL4.
- (Manually port our verification tools from HOL4 to HOL Light: too much work.)
- (Automatically port the high-level script files: we are unaware of any tools for that.)

A manual port of the high-level scripts would have been feasible, although more work than using the OpenTheory link. It would also have been tedious, and would not scale to larger developments, so we went for the automatic option. OpenTheory is currently the only tool for low-level proof transport with an exporter for HOL Light and an importer for HOL4, so it was the obvious choice.

Transport Overview

We used Joe Hurd's proof-logging fork of HOL Light[2], rather than mainline HOL Light, because that is the only version with an OpenTheory exporter. The key steps to transport the contents of a single HOL Light script file are:

1. Modify the HOL Light script file so that it builds in the proof-logging fork.
2. Mark each theorem that is desired in the exported output.
3. Run the exporter to create an OpenTheory article.
4. Run the OpenTheory tool on the article to compress it and clean it of artefacts[3] produced by the HOL Light exporter.
5. Run the HOL4 importer on the article, and find which constants and theorems the article requires as assumptions. (These should only be standard library theorems, but HOL4 does not yet automatically support everything in the standard library.)

[1] Specifically, his theories define the syntactic inference rules of HOL and their intended semantics, and prove soundness (statements derived by the rules are true in the semantics) and consistency (some statement cannot be derived).

[2] `http://src.gilith.com/hol-light.html`

[3] For example, the identity function `NUMERAL` wrapped around numerals to aid parsing and printing.

6. Create a HOL4 script that proves the article's assumptions, imports the article, and saves the theorems. The result is a HOL4 theory suitable for native use.

We only transported a single script, the one defining Harrison's inference rules for HOL. We made little use of OpenTheory's facilities for composing theory packages, since our focus was getting all the results rather than packaging them nicely for others.

The Need for Modifiability

Harrison's model of HOL does not include rules for defining new constants, but the implementation we plan to verify will support definitions. Therefore, we would like to extend his model and soundness proof with support for definitions. The problem is that low-level proofs are effectively unmodifiable.

After following the transport process above, the original HOL Light script files remain the only human-modifiable sources for the theory. We must work with them to make our extensions, and then transport the results. The transport process thus becomes part of our development chain.

We contend that most uses of proof transport will share this feature of wanting access to modifiable sources. Even when no major extensions need to be made to the transported theory, it is often necessary to tweak definitions or expose internal results to make downstream development possible or easier. Harrison's proofs could be packaged, as they are, as a reusable theory, but we would still want the option for an easy re-export of the proofs after—and while—we modify and extend them to support definitions.

Specific Experiences

In this section, I describe some of the particular issues that affected our use of OpenTheory. At present, we have successfully transported Harrison's definition of the syntactic inference rules of HOL (130 theorems/definitions in total). We also updated his definition to support constant definition (and transported it again). We have not yet updated or transported the soundness proof.

The main result of transport is the definition of the provability relation, `|-`, and all the constants it depends on (such as `ACONV`). The theorem in HOL4 after transport, shown partially below, looks the same as the original[4] in HOL Light.

$$
\begin{aligned}
\vdash\ & (\forall t\, defs.\ \texttt{welltyped\_in}\ t\ defs \implies ((defs, [])\ \texttt{|-}\ t\ \texttt{===}\ t)) \wedge \\
& (\forall asl_1\ asl_2\ l\ m_1\ m_2\ r\ defs. \\
& \quad ((defs, asl_1)\ \texttt{|-}\ l\ \texttt{===}\ m_1) \wedge ((defs, asl_2)\ \texttt{|-}\ m_2\ \texttt{===}\ r) \wedge \\
& \quad \texttt{ACONV}\ m_1\ m_2 \implies \\
& \quad ((defs, \texttt{TERM\_UNION}\ asl_1\ asl_2)\ \texttt{|-}\ l\ \texttt{===}\ r)) \wedge \\
& (\forall asl_1\ l_1\ r_1\ asl_2\ l_2\ r_2\ defs. \\
& \quad ((defs, asl_1)\ \texttt{|-}\ l_1\ \texttt{===}\ r_1) \wedge ((defs, asl_2)\ \texttt{|-}\ l_2\ \texttt{===}\ r_2) \wedge \\
& \quad \texttt{welltyped}\ (\texttt{Comb}\ l_1\ l_2) \implies \\
& \quad ((defs, \texttt{TERM\_UNION}\ asl_1\ asl_2)\ \texttt{|-}\ \texttt{Comb}\ l_1\ l_2\ \texttt{===}\ \texttt{Comb}\ r_1\ r_2)) \wedge \\
& \ldots
\end{aligned}
$$

[4]Not Harrison's original, but the extended version with support for definitions; the extension was made to the HOL Light script file before transport.

Some of the issues described below arise in part because the tools (the HOL Light exporter, the OpenTheory tool, and the HOL4 importer) and the OpenTheory standard library are all still young. I expect that if OpenTheory is used more, and the standard library gains traction, proof transport will enjoy much better support.

Reuse of native constants and theorems On the whole, the standard ontology provided by OpenTheory was a useful intermediary for mapping constants in HOL Light to their moral equivalents in HOL4. For our work a direct mapping of constants would also have been fine, but we would have had to write it. The advantage of using standard library constants is that both the HOL Light exporter and the HOL4 importer contained (most of) the mappings we needed before we started. We needed to manually provide only six additional mappings, mostly for constants in the standard library that don't exist (and hence also had to be defined) in HOL4, for example the standard library's `BIT0` constant for numerals (since HOL4 uses `BIT1` and `BIT2` instead).

It was also good that many of the assumptions of the transported theory were satisfied by corresponding native theorems in HOL4. We needed, however, to state and prove sixty additional theorems in order to match the article's assumptions exactly. All of these had trivial (less than two lines) proofs, and were often rephrasings of native theorems.

For example, the standard library asks for both

$$\vdash \forall p.\, \texttt{EVERY}\ p\ [\,] \text{ and } \vdash \forall p\, h\, t.\, \texttt{EVERY}\ p\ (h{::}t) = p\ h \wedge \texttt{EVERY}\ p\ t,$$

whereas HOL4 has a single theorem

$$\vdash (\forall P.\, \texttt{EVERY}\ P\ [\,] = \texttt{T}) \wedge (\forall P\, h\, t.\, \texttt{EVERY}\ P\ (h{::}t) = P\ h \wedge \texttt{EVERY}\ P\ t).$$

Slightly less trivially, the standard library asks for

$$\vdash \forall p\, g\, h.\, \exists f.\, \forall x.\, f\ x = \texttt{if}\ p\ x\ \texttt{then}\ f\ (g\ x)\ \texttt{else}\ h\ x,$$

whereas HOL4 has a constant, `WHILE`, and a theorem saying it is a suitable witness for f.

The HOL4 importer might be augmented to try strategies for automatically proving such assumptions on import. Alternatively, since all the assumptions are supposed to be in the standard library, HOL4 might want a native copy of every theorem in the standard library. It is unclear which of these approaches is better.

Representation of sets OpenTheory represents sets as an abstract type (A `set`), which leads to a clean interface and easy import, but can make exporting scripts that represent sets as predicates ($A \to$ `bool`) difficult. Harrison's work, especially the soundness proof, makes considerable use of sets as predicates[5]. We hope to port his proof, but doing so would be non-trivial because it has not been made to work with the proof-logging fork of HOL Light and makes extensive use of sets represented by predicates.

Source annotations The proof-logging fork of HOL Light was designed primarily for creating theory packages for the standard library. As such, many of HOL Light's native theories have been modified so they can be exported as the standard library theories, and the exporter is not designed to accept native HOL Light theories without modifications.

[5]His use of predicates is in addition to his definition of a new type for representing sets in the HOL semantics; being of a separate type, the latter "sets" pose no problem.

At best, these modifications include marking each theorem in the script that is desired as an output of the final theory package, which for us meant marking essentially every theorem proved in the file.

At worst, proof scripts need to be changed substantially to accommodate the standard library's representation of sets and the fact that some of HOL Light's automation has not yet been made to work with the rest of the fork. We ran into this problem when one of the definitions required some of HOL Light's advanced recursive function definition tools, which did not work in the proof-logging fork. After failing to devise an alternative proof, we called on Joe Hurd to upgrade the fork. Thankfully he was able to do so.

Article file size, and transport efficiency The transport process centres around an article file that contains the log of primitive inferences comprising the theory. The generated article for Harrison's definition of HOL's inference rules is around twelve megabytes in size. The original script file is only thirty kilobytes in size. Having such a large file in the development chain is unwieldy, especially when using version control since it increases our repository size significantly. We opt to keep the article in our repository because we don't expect everyone using the repository to have the tools necessary to generate it (e.g. an installation of proof-logging HOL Light). Processing (exporting, compressing, importing) the article also takes several minutes.

Kaliszyk and Krauss [4] will present techniques for efficient proof transport, in terms of both speed and size, at ITP 2013. We hope these will be adopted by OpenTheory, where efficiency has been an important concern but not the highest priority.

Theorem names OpenTheory avoids names wherever possible, and refers to theorems by their statements. This policy has the benefit of reducing opportunities for conflict. But nicely named theorems are useful in most provers, including HOL4. A principled approach, which is currently unimplemented, would be to pass names and other metadata (e.g. whether a theorem should be an automatic rewrite rule) on a separate channel alongside the OpenTheory proof data. We used the simpler hack of encoding the theorem name in the theorem itself (as the name of an extra variable).

3 The Impossible Wish: Script Portability

In light of the need for modifiability, the holy grail of theory exchange would be automatic methods for porting high-level script files. This corresponds to porting software between high-level languages as opposed to compiling to a common bytecode that has an interpreter on each platform. But just as software ports are usually done manually, I expect the variety and complexity of script files to make automatic ports unlikely: HOL Light/HOL4 script files may contain arbitrary OCaml/SML code.

Another approach would be to generate script files by "decompiling" low-level proofs. This idea is relatively unexplored in the context of proofs, but the analogous field of decompilation for software binaries may offer some ideas. Recovering an invocation of a high-level tactic from the trace of its primitive inferences sounds like a difficult problem, but perhaps the traces are sufficiently idiosyncratic.

The Common HOL platform, as used by HOL Zero [1], is aimed in the direction of script file portability. It defines an API of high-level functions that a HOL-based prover might use

in a script file. Beyond such an API, one could use a standard, purpose-built script file language, such as the Isar formal proof language [6] used by Isabelle. Or, going further, we might standardise on a single prover and eliminate the need for exchange at all.

Of course, diversity has its advantages: there are various factors affecting the experience of using a prover including the interface, the speed, the automation libraries, the theory libraries, and the degree to which hacking the prover is encouraged. There are different trade-offs to make between these; we don't know the single best answer and there probably isn't one.

4 Summary of Tensions

In the table below, we characterise the two ways to consider a proof exchange system that we have explored in our use of OpenTheory.

For Making Reusable Packages	For Proof Transport
export a clean interface	export everything
depend on and produce standard results	depend on standard results
export once and archive	export and import with every modification
export can be expensive so import is cheap	export and import must be cheap
can expect script file preparation	should accept script files untouched

In some ways, the differences are in degree rather than in kind. For example, reusable packages may have a clean interface that happens to include everything, or may need occasional modification for maintenance. But when modifications are being made in a development cycle, where one is experimenting with the effects of changes in imported definitions on further theories in the target system, cheap proof transport becomes more important.

Should more work be done by the importer or the exporter? OpenTheory tries to find a balance in the middle, but perhaps biases against the exporter since the work to make a native theory into an OpenTheory package is slightly different for each theory, and must be done for each one, whereas the work required to import any OpenTheory package (which respects the standard library) as a native theory can be done once and for all.

Two Modes of Use

Direct proof transport is likely to be a commonly desired function of proof exchange systems like OpenTheory. Such usage benefits from the existence of a well-supported (by provers) well-designed standard library of theories. However, the concerns when creating such a library can get in the way of making the transport process smooth. Efficient transport is necessary as long as the original sources remain the only modifiable ones.

It might be possible, however, for OpenTheory to support two modes of use. One mode would be for creating theory packages fit for public consumption, and conforming to the standards required by a high-quality package repository. The other mode would be for porting a theory from one prover to another wholesale, without modifications. Implementing the second (proof transport) mode would require more ingenuity, since the source theory may depend on particular representations and theories that are outside the standard library. However, one could always fall back on exporting a larger, less reusable package that derives its results from (at worst) the axioms.

Acknowledgements I am grateful to Magnus Myreen for discussion about his experiences with OpenTheory, and for advice and comments on this paper. I appreciate feedback on this paper from Mike Gordon, and from the anonymous reviewers. I thank Joe Hurd for introducing me to OpenTheory and keeping me involved in its development. My work is supported financially by the Gates Cambridge Trust.

References

[1] Mark Adams. Introducing HOL Zero - (extended abstract). In Komei Fukuda, Joris van der Hoeven, Michael Joswig, and Nobuki Takayama, editors, *ICMS*, volume 6327 of *Lecture Notes in Computer Science*, pages 142–143. Springer, 2010.

[2] John Harrison. Towards self-verification of HOL Light. In Ulrich Furbach and Natarajan Shankar, editors, *IJCAR*, volume 4130 of *Lecture Notes in Computer Science*, pages 177–191. Springer, 2006.

[3] Joe Hurd. The OpenTheory standard theory library. In Mihaela Bobaru, Klaus Havelund, Gerard J. Holzmann, and Rajeev Joshi, editors, *Third International Symposium on NASA Formal Methods (NFM 2011)*, volume 6617 of *Lecture Notes in Computer Science*, pages 177–191. Springer, April 2011.

[4] Cezary Kaliszyk and Alexander Krauss. Scalable LCF-style proof translation. To appear in ITP 2013.

[5] Magnus Myreen, Scott Owens, and Ramana Kumar. Steps towards verified implementations of HOL Light. To appear in ITP 2013.

[6] Markus Wenzel. Isar - a generic interpretative approach to readable formal proof documents. In Yves Bertot, Gilles Dowek, André Hirschowitz, C. Paulin, and Laurent Théry, editors, *TPHOLs*, volume 1690 of *Lecture Notes in Computer Science*, pages 167–184. Springer, 1999.