# Standalone Tactics using OpenTheory

Ramana Kumar*[1] and Joe Hurd[2]

[1] University of Cambridge
`Ramana.Kumar@cl.cam.ac.uk`
[2] Galois, Inc.
`joe@gilith.com`

**Abstract.** Proof tools in interactive theorem provers are usually developed within and tied to a specific system, which leads to a duplication of effort to make the functionality available in different systems. Many verification projects would benefit from access to proof tools developed in other systems. Using OpenTheory as a language for communicating between systems, we show how to turn a proof tool implemented for one system into a standalone tactic available to many systems via the internet. This enables, for example, LCF-style proof reconstruction efforts to be shared by users of different interactive theorem provers and removes the need for each user to install the external tool being integrated.

## 1 Introduction

There are many LCF-style systems for interactively developing machine-checked formal theories, including HOL4 [1], HOL Light [2], ProofPower [3] and Isabelle/HOL [4]. The logic implemented by these systems is essentially the same, but the collections of theory libraries and proof tools built on top of the logical kernels differ. Where similar proof tools exist in multiple systems it is usually the result of duplicated effort.

Examples of duplicated effort on tactics include the integration of external tools into HOL-based provers. For instance, Kumar and Weber [5] and Kunčar [6] give independent integrations of a quantified boolean formula solver into HOL4 and HOL Light. Weber and Amjad [7] give high-performance integrations of SAT solvers into three HOL-based systems; each integration requires a separate implementation. Sledgehammer [8] is only available for Isabelle/HOL, but its functionality would also be useful in other systems.

In addition to the development effort, the cost of maintenance can also be multiplied over different systems, and improvements in functionality can become restricted to a single system unnecessarily. For instance, the Metis first order logic prover [9] is integrated in multiple systems in the HOL family, but the HOL4 version is very old compared to the latest version in Isabelle/HOL. Slind's TFL package for defining recursive functions [10], originally implemented for both Isabelle/HOL and HOL4, was superseded in Isabelle/HOL by Krauss's function

* supported by the Gates Cambridge Trust

definition package [11]. The improvements of Krauss's method over TFL ought to be applicable to other HOL-based provers, but a direct reimplementation would require substantial effort.

It makes sense to speak of similar proof tools in different interactive theorem provers not just because they implement essentially the same logic, but also because there is a shared base of concepts: booleans, inductive datatypes, recursive functions, natural numbers, lists, sets, etc. The OpenTheory standard library [12] formalises this shared base as a collection of theory packages containing proofs written in the simple *article* format designed for storing and sharing higher order logic theories [13]. We use OpenTheory as a language for interactive theorem provers to communicate with proof tools on a remote server, and thereby obtain the following two benefits:

1. **Proof Articles:** A standard format to encode the goals that will be sent to the remote proof tools and the proofs that will be received in response.
2. **Standard Library:** An extensible way to fix the meaning of constants and type definitions between systems.

We contend that proof tools for interactive theorem provers need only be written and maintained in one place rather than once per system, using *standalone tactics* that are available online and communicate using OpenTheory. An added advantage when the tactic is an integration of an external tool is that a user of the interactive theorem prover need not install the external tool: it only needs to be available on the server hosting the standalone tactic.

The contributions of this rough diamond are:

1. A general method for turning existing proof tools implemented in interactive theorem provers into standalone tactics (Section 2).
2. Preliminary results profiling the performance of working examples of standalone tactics (Section 3).

## 2 Lifting Proof Tools into the Cloud

### 2.1 OpenTheory for Tactic Communication

An example: the user of an interactive theorem prover faced with the goal

$$\forall n.\ 8 \leq n \Rightarrow \exists s, t.\ n = 3s + 5t$$

decides to pass it off to a standalone tactic for linear arithmetic.

The input for the standalone tactic is the goal term, and the output is a proof of the theorem. Standalone tactics use the OpenTheory article format for communicating terms and proofs. The interactive theorem prover serializes the goal term from its local internal format to an article file, and sends the article over the internet to the standalone tactic. If successful, the standalone tactic sends back another article file encoding a proof of the goal, which the interactive theorem prover replays through its logical kernel to create the desired theorem.

This example illustrates the key requirements for an interactive theorem prover to use standalone tactics:

1. Ability to replay proofs by reading OpenTheory articles.
2. Ability to write terms as OpenTheory articles.
3. Ability to communicate with external programs.

Requirements 1 and 2 can be satisfied for an interactive theorem prover by implementing an OpenTheory interface that can interpret and construct articles. The central concept in OpenTheory is that of a *theory package*, $\Gamma \rhd \Delta$, which proves that the set of theorems $\Delta$ logically derive from the set of assumptions $\Gamma$. An article is a concrete representation of a theory package, consisting of instructions for a virtual machine whose operations include construction of types and terms, and the primitive inference rules of higher order logic. To *read* an article, an interactive theorem prover performs the primitive inferences and other instructions listed in the file. The OpenTheory logical kernel is based on HOL Light's logical kernel, and the instructions are chosen to make it easy to read articles into any system that can prove theorems of higher order logic.

An article file represents a theory $\Gamma \rhd \Delta$. By taking $\Delta$ to be the set of theorems proved by a proof tool and $\Gamma$ to be the set of theorems used by the proof tool, we can view the result of executing a proof tool as a logical theory. In our example above of using a linear arithmetic standalone tactic on the given goal, this theory might be

$$\left\{ \begin{array}{l} \vdash \forall n.\ n + 0 = n \\ \vdash \forall m, n.\ mn = nm \\ \cdots \end{array} \right\} \rhd \left\{ \vdash \forall n.\ 8 \leq n \Rightarrow \exists s, t.\ n = 3s + 5t \right\}$$

where the assumptions consist of a collection of standard arithmetic facts.

The main benefit of using OpenTheory for communication is that it provides a standard ontology for fixing the meanings of constants and type operators between different systems. For example, the numerals 3, 5 and 8 in the example goal term can be encoded in binary using the standard constants `Number.Natural.bit0` and `Number.Natural.bit1`. The full names and properties of these constants are indicated in the OpenTheory standard library, and interactive theorem provers can maintain translations to and from their local names and theorems. A system using a different encoding for numbers (say unary, with `Number.Natural.suc` and `Number.Natural.zero`) could use additional standalone tactics to translate between encodings.

Implementing an OpenTheory interface to satisfy Requirements 1 and 2 above carries the additional benefit of giving the interactive theorem prover access to all logical theories stored as OpenTheory packages, not just those that are the output of standalone tactics.

### 2.2 Extracting Tactics from Interactive Theorem Provers

There are two approaches to obtaining a standalone tactic: either write one directly; or extract an existing tactic from an interactive theorem prover. We have experimented with the second approach, extracting tactics from HOL4 and from HOL Light. The procedure is reasonably lightweight, but less flexible than writing a standalone tactic directly. For tactic extraction to succeed, the key requirements on the interactive theorem prover are:

1. Ability to read and write OpenTheory article files.
2. Ability to record proofs and reduce them to the OpenTheory kernel.
3. Ability to make a standalone executable encompassing the tactic functionality separated from the usual interface to the interactive theorem prover.

Just as the requirements for an interactive theorem prover to use standalone tactics also enable it to import OpenTheory packages in general, the first two requirements to create standalone tactics also enable a system to create and export OpenTheory packages. (The last requirement enables running on a server.)

Requirement 2 poses the highest barrier: a standalone tactic must record each proof at a level of detail sufficient to prove the same theorem using the OpenTheory kernel. The following method can be used if the system implementing the tactic has an LCF-style design, that is, theorems can only be created by a small number of primitive inference rules: (i) augment the internal theorem type with a type of proofs to track the system primitive inferences used; and (ii) express each system primitive inference as a derived rule of the OpenTheory kernel. We applied this method to meet Requirement 2 for both HOL4 and HOL Light. For some primitive inferences (such as reflexivity of equality) there is a direct translation to the OpenTheory logical kernel. But, for example, HOL4's primitive rule for definition by specification must be emulated in the OpenTheory kernel, for example by using Hilbert choice. Although Isabelle/HOL uses the LCF architecture, its logical kernel is quite different from OpenTheory; we therefore expect translating Isabelle/HOL proofs to be more difficult than HOL4 and HOL Light proofs.

To satisfy Requirement 3 in HOL4, we used the 'export' facility of the Poly/ML compiler, which creates a standalone executable that runs an ML function. For each tactic, we captured a function that reads an article representing the input term, runs the tactic (recording the proof), and writes an article containing the proof. The situation for HOL Light is more complicated because OCaml does not provide such an 'export' facility, and a HOL Light session typically starts by proving the standard library. We used a general-purpose checkpointing facility to capture a HOL Light session at the point where it is ready to read an article and run a tactic.

## 3 Preliminary Performance Results

We collected preliminary performance data for two test standalone tactics extracted from HOL4, and called from HOL Light. They are QBF [5], which proves quantified boolean formulas, and SKICo, which rewrites terms to combinatory form like so: $\vdash (\forall x.\ x \vee \neg x) = (\forall)\ (\mathtt{S}\ (\vee)\ (\neg))$. We used the following three test goals:

1. $\forall x.\ x \vee \neg x$

| Tactic-Problem | Goal Size (b) | Goal Time (s) | Remote Time (s) | Proof Size (b) | Proof Time (s) | Total Time (s) | Local Time (s) |
|---|---|---|---|---|---|---|---|
| QBF-1 | 927 | 0.001 | 1.064 | 10,991 | 0.022 | 1.088 | 0.002 |
| QBF-2 | 1,474 | 0.001 | 1.892 | 79,944 | 0.139 | 2.034 | 0.024 |
| QBF-3 | 1,546 | 0.001 | 1.821 | 91,639 | 0.172 | 1.996 | 0.024 |
| SKICo-1 | 927 | 0.001 | 1.212 | 20,047 | 0.041 | 1.255 | 0.000 |
| SKICo-2 | 1,474 | 0.002 | 1.557 | 52,249 | 0.113 | 1.673 | 0.001 |
| SKICo-3 | 1,546 | 0.002 | 1.716 | 60,642 | 0.125 | 1.844 | 0.005 |

**Table 1.** Performance profiling for the test standalone tactics.

2. $\exists p.\ (\forall q.\ p \vee \neg q) \wedge \forall q.\ \exists r.\ r$
3. $\exists x.\ \forall y.\ \exists z.\ (\neg x \vee \neg y) \wedge (\neg z \vee \neg y)$

For each invocation of a standalone tactic on a test goal, Table 1 profiles the time and space requirements of the three phases of execution: encoding the goal as an article; communicating with and executing the standalone tactic remotely; and replaying the proof article received. For comparison, the time to run the tactic locally within HOL4 is given in the rightmost column.

The sizes of the articles for the test goals and the resulting proofs are comparable to the typical size of web requests and the resulting pages, so we can be confident that we are within the normal operating range of the web tools we use (`curl` on the client and CGI scripts on the server). For problems involving larger articles (bigger goals or longer proofs) we may wish to compress them using `gzip` before sending them over the network—previous experiments showed a compression ratio of 90% is typical for article files [13].

Turning now to execution time, we can see that it is significantly more expensive to call a standalone tactic over the internet compared to executing it locally. However, most of the time is spent on 'Remote Time', which includes communicating with the remote standalone tactic and waiting for it to read the goal article, run the proof tool, and write the resulting proof article. Using `traceroute` we see a 0.173s network delay between the test client in Portland, OR, USA and the test server in Cambridge, UK, which accounts for at least 0.346s of delay. The overall time is in the 1–2s range, which is very slow for workaday tactics but may well be tolerated by the user to gain access to the functionality of a proof tool on another system.

## 4 Related Work

The PROSPER project [14] pioneered the technique of sending goals over the internet to a remote solver, and packaging such procedures as tactics in the HOL4 theorem prover. The standalone tactics described in this paper further systematize this by using OpenTheory as a standard language and ontology to make it easier for interactive theorem provers and remote solvers to communicate.

An impressive example of providing reasoning infrastructure over the internet is System on TPTP [15], which enables a user to remotely execute a collection of automatic theorem provers on a problem expressed in a standard format. The usual scenario is to decide the validity of first order logic formulas (TPTP format), but there is also support for higher order terms (THF format), and for returning proofs expressed in a standard language (TSTP format).

The idea of separate reasoning tools communicating to enhance a proof development environment is also being pursued in the Evidential Tool Bus [16] and the MathServe System [17]. This idea is a natural extension of the integration of automatic tools with interactive theorem provers.

## 5 Conclusion

We have shown how, using OpenTheory for communication, we can write tools for higher order logic reasoning tasks as standalone tactics, making them available to multiple interactive theorem provers and independently maintainable. Existing proof tools can be extracted from their home systems for reuse. There is a substantial but hopefully tolerable overhead of communicating goals and proofs in article files over the internet.

## References

1. Slind, K., Norrish, M.: A brief overview of HOL4. [18] 28–32
2. Harrison, J.: HOL Light: An overview. In Berghofer, S., Nipkow, T., Urban, C., Wenzel, M., eds.: TPHOLs. Volume 5674 of LNCS., Springer (2009) 60–66
3. Arthan, R.: ProofPower manuals. (2004) `http://lemma-one.com/ProofPower`.
4. Wenzel, M., Paulson, L.C., Nipkow, T.: The Isabelle framework. [18] 33–38
5. Kumar, R., Weber, T.: Validating QBF validity in HOL4. [19] 168–183
6. Kunčar, O.: Proving valid quantified boolean formulas in HOL Light. [19] 184–199
7. Weber, T., Amjad, H.: Efficiently checking propositional refutations in HOL theorem provers. Journal of Applied Logic **7**(1) (2009) 26–40
8. Blanchette, J.C., Bulwahn, L., Nipkow, T.: Automatic proof and disproof in Isabelle/HOL. In Tinelli, C., Sofronie-Stokkermans, V., eds.: FroCos. Volume 6989 of LNCS., Springer (2011) 12–27
9. Hurd, J.: First-order proof tactics in higher-order logic theorem provers. In Archer, M., Vito, B.D., Muñoz, C., eds.: STRATA 2003. Number NASA/CP-2003-212448 in NASA Technical Reports (September 2003) 56–68
10. Slind, K.: Reasoning about Terminating Functional Programs. PhD thesis, Technischen Universität München (1999)
11. Krauss, A.: Partial and nested recursive function definitions in higher-order logic. J. Autom. Reasoning **44**(4) (2010) 303–336
12. Hurd, J.: The OpenTheory standard theory library. In Bobaru, M., Havelund, K., Holzmann, G.J., Joshi, R., eds.: NFM 2011. Volume 6617 of LNCS., Springer (April 2011) 177–191
13. Hurd, J.: OpenTheory: Package management for higher order logic theories. In Reis, G.D., Théry, L., eds.: PLMMS, ACM (August 2009) 31–37

14. Dennis, L.A., Collins, G., Norrish, M., Boulton, R., Slind, K., Robinson, G., Gordon, M., Melham, T.: The PROSPER toolkit. In Graf, S., Schwartzbach, M., eds.: TACAS. Volume 1785 of LNCS., Springer (2000) 78–92
15. Sutcliffe, G.: The TPTP world: Infrastructure for automated reasoning. In Clarke, E.M., Voronkov, A., eds.: LPAR. Volume 6355 of LNCS., Springer (2010) 1–12
16. Rushby, J.M.: An evidential tool bus. In Lau, K.K., Banach, R., eds.: ICFEM. Volume 3785 of LNCS., Springer (2005) 36–36
17. Zimmer, J., Autexier, S.: The MathServe system for semantic web reasoning services. In Furbach, U., Shankar, N., eds.: IJCAR. Volume 4130 of LNCS., Springer (2006) 140–144
18. Mohamed, O.A., Muñoz, C., Tahar, S., eds.: TPHOLs. Volume 5170 of LNCS., Springer (2008)
19. van Eekelen, M.C.J.D., Geuvers, H., Schmaltz, J., Wiedijk, F., eds.: ITP. Volume 6898 of LNCS., Springer (2011)