The OpenTheory Standard Theory Library

Joe Hurd

Galois, Inc.
joe@gilith.com
http://www.gilith.com

Abstract. Interactive theorem proving is tackling ever larger formalization and verification projects, and there is a critical need for theory engineering techniques to support these efforts. One such technique is cross-prover package management, which has the potential to simplify the development of logical theories and effectively share theories between different theorem prover implementations. The OpenTheory project has developed standards for packaging theories of the higher order logic implemented by the HOL family of theorem provers. What is currently missing is a standard theory library that can serve as a published contract of interoperability and contain proofs of basic properties that would otherwise appear in many theory packages. The core contribution of this paper is the presentation of a standard theory library for higher order logic represented as an OpenTheory package. We identify the core theory set of the HOL family of theorem provers, and describe the process of instrumenting the HOL Light theorem prover to extract a standardized version of its core theory development. We profile the axioms and theorems of our standard theory library and investigate the performance cost of separating the standard theory library into coherent hierarchical theory packages.

1 Introduction

Interactive theorem proving has grown from toy examples to major formalization and verification projects in mathematics and computer science. Recent examples include: the 20 man-year verification of the seL4 operating system kernel [24]; the CompCert project, which verified an optimizing compiler from a large subset of C to PowerPC assembly code [25]; and the Flyspeck project, which aims to mechanize a proof of the Kepler sphere-packing conjecture [14].

Just as the term software engineering was coined in 1968 [26] to give a name to techniques for developing increasingly large programs, there is now a need for *theory engineering* techniques to develop increasingly large proofs (*"proving in the large"*). One software engineering technique that can be applied to proof development is effective package management. Modern operating systems [8] and programming languages [6] bundle software into packages that carry their dependencies, supporting easy distribution and automatic checking at installation time to ensure that the system can properly support the package. The goal of

M. Bobaru et al. (Eds.): NFM 2011, LNCS 6617, pp. 177–191, 2011.

the OpenTheory project is to transfer the benefits of package management to aid the development of logical theories[1].

The initial case study of the OpenTheory project is to develop the infrastructure necessary to port theories between three related interactive theorem provers: HOL Light [15], HOL4 [28] and ProofPower [23]. These three theorem provers implement the same higher order logic, namely Church's simple theory of types extended with Hindley-Milner style type variables [10]. They also have a similar design of an interactive interface where the user invokes proof tools to prove subgoals, built on top of a small logical kernel that enforces soundness. The logical kernel design is inherited from Milner's pioneering work on the LCF theorem prover [11], which Gordon reused to implement higher order logic in the HOL theorem prover [12], and from which the three chosen theorem provers are all descended [13].

Even though HOL Light, HOL4 and ProofPower implement the same logic using the same conceptual design, they each contain significant theory formalizations that are not accessible to each other. For example, HOL Light has a formalization of complex analysis [16], HOL4 has a formalization of probability theory [18], and ProofPower has a formalization of the Z specification language [2]. The reason that these useful theories are not available in all of the theorem provers is that it requires significant human effort to port a theory to a new environment, due to differences in the native theories and proof tools[2].

To overcome the differences between the name and behavior of proof tools between the theorem provers, the OpenTheory project has developed a standard *article* file format for serializing proofs of higher order logic [21]. Proofs are reduced to a standard set of primitive inferences that are precisely specified and can be simulated by any theorem prover in the HOL family. This bypasses the differences in the proof tools, at the cost of archiving proofs in a format that is hard to modify.

Once the differences between the proof tools have been removed as an obstacle, the challenge that remains is to reconcile the differences between the native theories available in each theorem prover. To illustrate the need for this, suppose we desire to port the theory of complex numbers from HOL Light to HOL4. One way to do this is to export every theory that the HOL Light complex numbers depend on as proof articles, and then import these into HOL4. However, now we have two copies of the theory of real numbers inside HOL4: the original real number theory of HOL4 and the real number theory imported from HOL Light that the complex numbers depend on. Because of this, we cannot easily combine the new theory of complex numbers with other HOL4 theories that depend on the original real number theory, such as the theory of probability.

To avoid this *duplicate theory* problem, when we speak of porting theories between theorem provers we usually have in mind the following procedure:

[1] The OpenTheory project homepage is `http://gilith.com/research/opentheory`

[2] The author has first-hand experience of this: his introduction to theorem proving was porting a theory of real numbers from HOL Light to HOL4.

1. Export the theory of complex numbers from HOL Light, leaving the references to native theories as uninterpreted type operators, constants, and assumptions.
2. Import the theory of complex numbers into HOL4, binding the references to native theories to native type operators, constants and theorems.

Note that the success of this porting procedure depends on there being a degree of alignment between the native theories of HOL Light and HOL4. The native theories do not have to be identical: type operators and constants may have different names in the two theorem provers; and HOL4 may contain additional theorems beyond the set required for the import of the HOL Light theory to succeed. But beyond these superficial differences, to port a theory from the theorem prover context A to B there must be a semantic embedding $A \rightarrow B$ mapping type operators and constants from A to ones in B with properties that are at least as logically strong. This notion of semantic embeddings between theorem prover contexts has been formalized in category theory as *theory morphisms* [32], and this provides a theoretical foundation for the OpenTheory project.

To support the use case of porting theories between the HOL Light, HOL4 and ProofPower, we will need semantic embeddings from the core theories of each theorem prover to the core theories of the others. As an alternative to explicitly maintaining these semantic embeddings, we instead take the set of core theories that the theorem provers share and release a *standard theory library* of them in OpenTheory format. The advantage of a standard theory library are as follows:

- Each theorem prover is responsible for maintaining mappings between its core theories and the OpenTheory standard library, reducing the number of semantic embeddings that must be maintained from $O(n^2)$ to $O(n)$ (where n is the number of theorem provers that wish to share theories).
- The standard theory library is a published contract of interoperability: *"If your theory uses only the standard theory library, we promise it will work on all of the supported theorem provers."*
- If a property such as associativity of addition is in the standard theory library, it does not need to be proved in every theory that relies on it. This is analogous to dynamic linking of programs to standard libraries.
- Constructions in the standard library can serve as standard specifications. A formal proof of Fermat's Last Theorem that uses the version of the natural numbers in the standard theory library is much easier to check than one that uses a custom version.

This paper presents the OpenTheory standard theory library and describes the process of identifying and extracting the core theory set of the HOL family of theorem provers. The proof articles that result from this process are combined to form higher level theories such as *natural numbers* or *lists*, and the final step is to combine these to form the standard theory library.

The remainder of the paper is structured as follows: Section 2 reviews the OpenTheory formats and infrastructure that we used and extended to support this work; Section 3 identifies the core theories that are included in the standard theory library; Section 4 describes the process of extracting standard proof

articles by instrumenting an existing theorem prover; Section 5 profiles the result of combining proof articles into the standard theory library; and finally Sections 6–8 examine related work, summarize and consider future directions.

2 The OpenTheory Proof Archive

In this section we review the OpenTheory proof article [19] and theory package [20] formats, which are used to represent the standard theory library. These formats are now stable, and tools for processing theory packages are included with the OpenTheory toolset[3]. Tools exist for displaying meta-information, querying dependencies, pretty-printing assumptions and theorems, and compiling theory packages to proof articles.

2.1 Articles of Proof

The unit of composition in OpenTheory is a higher order logic theory $\Gamma \triangleright \Delta$, which consists of:

1. A set Γ of assumption sequents.
2. A set Δ of theorem sequents.
3. A proof that the theorems in Δ logically derive from the assumptions in Γ.

An article is a compact representation of a higher order logic theory, encoded as instructions for a stack-based virtual machine. The format was designed to simplify the process of importing theories into theorem prover implementations: all that is required is to execute the article instructions in the desired context.

The initial version of the proof article format [19] contained instructions for constructing types and terms, but the inference rules were system dependent. However, after receiving comments from the interactive theorem proving community, this system dependence was replaced with a set of 10 article instructions for executing precisely specified primitive inferences. These new instructions are shown in Figure 1.

2.2 Theory Packages

The proof article format supports a theorem prover independent representation of theories. The theory package format is a domain-specific language for combining theories, supporting the following operations:

1. Renaming type operators and constants in theories, either to avoid namespace clashes or to bind the arguments of a *parametric theory*.
2. Forming compound theories by satisfying the assumptions of one theory with the theorems of others.

[3] The OpenTheory toolset is available for download at `http://gilith.com/software/opentheory`

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

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

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

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

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

Fig. 1. The OpenTheory logical kernel

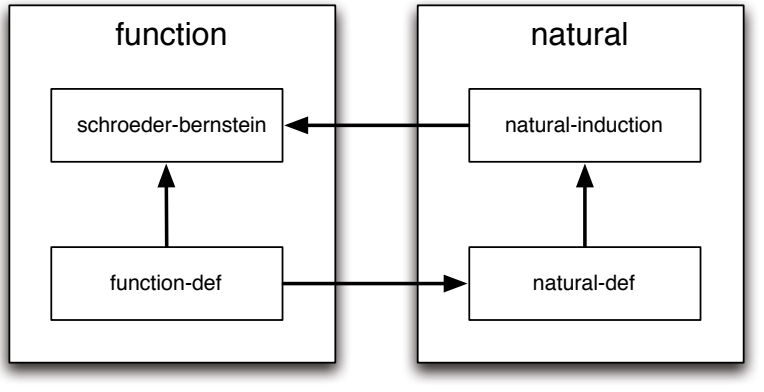

Fig. 2. Example theory dependency graph

Theory packages are hierarchical, using the above operations to build up from basic theory packages containing proof articles to more complex theories. An important concept for a standard theory library is the *compilation theory package*, which is designed to help construct coherent theory packages in the face of the complex dependency structures that often arise in theory development.

An example of compilation theories is shown in Figure 2, where four theory packages are contained in two compilation theory packages, and the arrows indicate package dependencies. The statement of the Schroeder-Bernstein theorem depends only on the function theory definitions, but the proof also depends on natural number induction. Natural numbers in turn are constructed using function theory definitions. The most coherent function theory package would contain both the function theory definitions and the Schroeder-Bernstein theorem, but this package would then have a cyclic dependency with the natural

number theory package. Defining the function theory package as a compilation of two theory packages allows finer grained theory package dependencies, which removes the offending cycle.

Early experimentation with the theory package language revealed some desirable properties of a reusable theory package:

1. a clear topic (e.g., trigonometric functions);
2. assumptions that are satisfied by the theorems of other reusable theory packages;
3. a carefully chosen set of theorems, presenting an abstract interface to the theory (hiding construction details).

We will refer to these guiding principles when describing the construction details of the OpenTheory standard theory library.

3 Identifying Core Theories

The first step in the construction of the OpenTheory standard theory library is to identify the core theories shared by the HOL family of theorem provers. Looking at the system documentation and source code for HOL Light, HOL4 and ProofPower turns up the following set of core theories, sorted into the OpenTheory standard namespace:

- Data.Bool – A theory of the boolean type
- Data.List – A theory of list types
- Data.Option – A theory of option types
- Data.Pair – A theory of product types
- Data.Sum – A theory of sum types
- Data.Unit – A theory of the unit type
- Function – A theory of functions
- Number.Natural – A theory of natural numbers
- Number.Numeral – A theory of natural number numerals
- Relation – A theory of relations

This is not intended to be a complete list, but sufficient to demonstrate the practicality of building an OpenTheory standard theory library, and full-featured enough upon which to build some non-trivial theories. The above theories are all present in version 1.0 of the standard theory library, and future versions can standardize other shared theories such as integers, reals, sets, characters and strings.

4 Extracting Standard Articles

The next step in the construction of the standard theory library is to represent the core theories as a set of proof articles that can be turned into basic theory packages. One approach to this would be to create an OpenTheory version of

the standard theory library from scratch, proving everything using the standard inference rules. However, since the standard theory library is (by definition) shared by each member of the HOL family of theorem provers, an alternative to this is to instrument one of the theorem provers to emit its version of the standard theory library in proof article format. We take this latter approach and choose the HOL Light theorem prover as having the simplest logical kernel to instrument. The remainder of this section describes the experience of extracting standard proof articles from HOL Light theories.

4.1 Granularity

With the primitive inferences of HOL Light instrumented to emit proof articles, the next choice to be made is the granularity of the proof articles. At the coarsest extreme of the granularity spectrum, the whole standard theory library could be emitted as one big proof article. However, this would violate Guideline 1 of constructing reusable theories from Section 2.2, because the resulting theory would not have a clear topic. At the finest extreme, we could put every exported theorem into its own proof article file, with the caveat that proof articles that make definitions need to export enough theorems to form a minimal abstract interface. This would result in a set of theories that score well according to the reusability guidelines (except possibly for Guideline 3 that asks for a carefully chosen set of theorems), but introduces myriad theory packages to be stored and processed.

We choose an intermediate point on the granularity spectrum where theory packages that make definitions export enough theorems to form a minimal abstract interface, and theory packages that make no definitions can export any number of theorems so long as they form a coherent topic. This design choice is made to maximize the reusability of the resulting theory packages while respecting performance goals.

Another issue is that there are two kinds of proved theorems in HOL Light: visually appealing theorems designed for the user to apply as lemmas in future proofs; and auxiliary pro-forma theorems designed to be used internally by proof tools. The reusable theory guidelines dictate that only the visually appealing theorems should appear in the standard theory library. This is achieved by collecting together the auxiliary theorems as they are proved and storing them in a separate proof article, which is 'statically linked' to standard proof articles as they are generated. When the whole standard theory library has been harvested, the auxiliary proof article is packaged as a special theory to support theory development building on the standard theory library using the HOL Light proof tools.

4.2 Standardization

In addition to statically linking auxiliary theorems, we used other methods to standardize the proof articles generated from HOL Light. As a simple example, the names of HOL Light type operators and constants are mapped into the OpenTheory standard namespace (as described in Section 3).

Another source of system dependence is the presence of 'tags' in terms. For example, in HOL Light every natural number numeral is a term of the form NUMERAL t, where the constant NUMERAL is defined as a synonym for the identity function. The presence of NUMERAL has no logical significance, but is a tag to help proof tools and other theorem prover infrastructure. Different theorem prover implementations may have different tagging schemes, so we remove tags from theorems that we add to the standard theory library. The scheme we use to do this is to rename the tag constant NUMERAL to be called Unwanted.id, and then rewrite all generated proof articles to remove all type operators and constants in the Unwanted namespace.

Finally, during the process of extracting proof articles from HOL Light we discovered many improvements to HOL Light that would simplify the extraction process, including: removing duplicate theorems; simplifying the definition of numerals; universally quantifying theorems with free variables. We submitted these as patches to the HOL Light developer, and several have already been incorporated into the upstream version.

4.3 Partial Functions

Partial functions require special handling in a classical two-valued logic such as higher order logic. For example, the natural number div and mod functions are not mathematically defined when the denominator is zero, but since every function in higher order logic is total the term 1 div 0 must be some natural number. In this case the solution we adopt is for the theory defining div and mod to export the single theorem

$$\vdash\ \forall m, n.\ n \neq 0\ \Longrightarrow\ m = (m\ \mathsf{div}\ n) * n + m\ \mathsf{mod}\ n\ \wedge\ m\ \mathsf{mod}\ n < n\ ,$$

preventing client theories from deducing anything about the value 1 div 0 (that could not be deduced about every natural number).

There are a few situations when this information-hiding approach cannot be used. Both HOL Light and HOL4 (but not ProofPower) define a predecessor function pre as an inverse to the successor function, and set pre $0 = 0$ even though the inverse of successor is not mathematically defined for zero. The value of pre 0 is subsequently relied on, among other things to define cut-off subtraction, and so we choose to 'grandfather' the value of pre 0 into the standard library. However, in the theory that defines the predecessor function we separate the definition into the two theorems

$$\vdash\ \mathsf{pre}\ 0 = 0 \quad \text{and} \quad \vdash\ \forall n.\ \mathsf{pre}\ (\mathsf{suc}\ n) = n\ ,$$

to encourage client theories to rely only on the standard domain.

5 The Standard Theory Library

After identifying the core theories of the HOL family of theorem provers and extracting them as proof articles, the final step is to use them to construct the standard theory library.

5.1 Construction

This is the procedure for converting the proof articles extracted from HOL Light into the standard theory library:

1. Create a basic theory package for each proof article.
2. Create theory packages for higher-level topics, such as `bool` or `list`, which are compilations of lower-level theory packages.
3. Create a theory package called `base`, which is a compilation of the highest-level theory packages.

Although the standard theory library consists of the whole collection of these theory packages, the `base` theory package exports all the theorems needed to build client theories on top of the standard theory library. The other theory packages can be regarded as scaffolding by most users of the standard theory library, and safely ignored.

As we expected, it was straightforward to carry out Steps 1 and 2 of the above procedure. We expected the difficulty to appear in Step 3, when compiling highest level theories with potentially complex dependencies between them. Surprisingly, it turned out that there was a natural order to arrange the highest-level theories where each one only depended on the previous ones: `bool`; `unit`; `function`; `pair`; `natural`; `relation`; `sum`; `option`; and `list`. Because of this, there was no need to unpack the compilation theories to eliminate cyclic dependencies as described in Section 2.2. The real-life example shown in Figure 2 demonstrates that this functionality will be required for some theories, but the current version of the standard theory library is naturally acyclic.

5.2 Axioms

Before looking at the theorems and proofs of the standard theory library, it is worth examining what it depends on. The OpenTheory primitive inference rules, shown in Figure 1, were taken from HOL Light and refer only to:

- the type operator bool;
- the function space type operator $\cdot \to \cdot$; and
- the equality constant $= : \alpha \to \alpha \to \mathsf{bool}$.

These two type operators and one constant can be considered to be implicitly axiomatized by the primitive inferences. In addition, the standard theory library explicitly asserts the following three axioms, each of which is contained in its own theory package:

$$\vdash\ \forall t.\ (\lambda x.\ t\ x) = t \qquad (\texttt{axiom-extensionality})$$

$$\vdash\ \forall P, x.\ P\ x \implies P\ (\mathsf{select}\ P) \qquad (\texttt{axiom-choice})$$

$$\vdash\ \exists f : \mathsf{ind} \to \mathsf{ind}.\ \mathsf{injective}\ f \wedge \neg \mathsf{surjective}\ f \qquad (\texttt{axiom-infinity})$$

The formulation of these three axioms is taken from HOL Light, but can be proved as theorems in HOL4 and ProofPower.

The standard theory library offers a simple way to check that a theory package developed on a HOL family theorem prover will easily port to other theorem

provers. Just statically link the new theory package to the `base` theory package, and any system dependent behavior will appear as extra axioms (beyond the standard three). This static linking procedure could also be used to develop theories that avoid the axiom of choice, while still making use of theorems from the standard theory library that do not use choice in their proof.

5.3 Theorems

The current version of the standard theory library exports 450 theorems, containing 64 defined constants and 6 defined type operators. For reasons of space the theorems cannot all be shown here, but an HTML version of the `base` package can be viewed at the following URL:

`http://opentheory.gilith.com/?pkg=base-1.0`

The standard theory library comprises 139 packages: 102 of which are basic theory packages wrapping proof articles; 36 of which are higher-level theory packages; and one is the `base` package. The left side of Table 1 shows the primitive inference count of replaying all the proofs in the standard theory library, for a total of 211,058 inferences. The cost of separating the standard theory library into 102 basic theory packages is highlighted by the axiom count of 1,672, which is increased whenever one basic theory uses a theorem proved by an imported theory.

It is possible to compile the whole of the standard theory library into a single proof article (with 965,433 commands), and then repeat the experiment of replaying all of the proofs and counting the primitive inferences: the results are shown on the right side of Table 1. The total number of primitive inferences drops by 40%, and the axiom count is only three: one for each of the standard axioms. It is also remarkable that the number of auxiliary defined type operators and constants (used in proofs but that do not appear in any theorems)

Table 1. The primitive inference count of replaying all the proofs in the standard theory library, when split into theories (left) and compiled into a single article (right)

Primitive Inference	Count
eqMp	55,209
subst	45,651
appThm	44,130
deductAntisym	28,625
refl	17,388
betaConv	8,035
absThm	7,765
assume	2,455
axiom	1,672
defineConst	119
defineTypeOp	9
Total	**211,058**

Primitive Inference	Count
eqMp	32,386
subst	27,949
appThm	27,796
deductAntisym	17,300
refl	9,332
absThm	6,313
betaConv	3,646
assume	1,169
defineConst	85
defineTypeOp	7
axiom	3
Total	**125,986**

drops from 55 and 3 to 21 and 1 (respectively). This provides some concrete data quantifying the performance cost of splitting the standard theory library into coherent hierarchical theory packages.

6 Related Work

Extracting proofs from LCF theorem provers is not new: Wong's pioneering *Recording and checking HOL proofs* in 1995 appears to be the first [35]. More recently, Obua and Skalberg [29] instrumented HOL4 and HOL Light to export theories in XML format that could be imported into the Isabelle/HOL theorem prover. The present work differs from this line of proof recording work by its focus on the theory as the central concept, independent of any particular theorem prover implementation.

From this point of view, the most related work is the AWE project [5], which builds on the explicit proof terms in Isabelle [4]. Though tied to one theorem prover, it nevertheless focuses on the theory as the central concept, and has developed sophisticated mechanisms for theory interpretation based on rewriting proof terms. The present work differs from AWE by being theorem prover independent, and also by its technique of processing proofs one step at a time rather than requiring the whole proof to be in memory, which may allow it to scale up more effectively.

The HOL Zero project [1] has aims similar to OpenTheory of making proofs portable between different implementations of the HOL family of theorem provers, by creating a minimal theorem prover "for checking and/or consolidating proofs created on other theorem provers", "designed with trustworthiness as its top priority". OpenTheory differs from HOL Zero by its focus on proofs that have been reduced to the object format of primitive inferences, and the theory packaging mechanisms that can be built on top of this starting point. HOL Zero complements OpenTheory by encouraging portability at the earlier stage of proof source files.

Many theorem provers implement a theory infrastructure that offers functionality similar to the theory operations in the OpenTheory package format. ProofPower has a sophisticated system for building and navigating a hierarchy of theories which contain both logical data and information for tools such as parsers, pretty printers and proof tools [23]. Theory interpretations are implemented in the EVES [7], IMPS [9], PVS [31] and Specware [34] theorem provers. Called locales in the Isabelle theorem prover [22], they are integrated with its declarative proof language [3]. The present work differs from these efforts by pursuing a theorem prover independent approach to theory combination and interpretation.

Another approach to higher order logic theory operations is to extend the logic so that theories can be directly represented with theorems [33,17]. The goal of the present work is to implement a theory infrastructure on top of the existing logic, but extending the logic has the significant advantage of supporting theory operations without replaying proofs.

7 Summary

In this paper we motivated the need for a standard library of higher order logic theories to support large-scale logical theory development and increase portability of theories between the HOL family of theorem provers.

The core contribution of this paper is the presentation of a theorem prover independent standard theory library, represented as an OpenTheory package. We identified the core theory set of the HOL family of theorem provers, and described the process of instrumenting the HOL Light theorem prover to extract a standardized version of its core theory development.

The OpenTheory package language is suitable to package proof articles extracted from HOL Light, and we showed how to combine these first into higher level theory packages, and then into a single package representing the user interface to the whole standard theory library. Finally, we profiled the axioms and theorems of the standard theory library, and investigated the performance cost of separating the standard theory library into coherent hierarchical theory packages.

8 Future Work

The current version of the standard theory library is not fixed, and in fact is expected to evolve as more theories are standardized between the HOL family of theorem provers. One desirable goal would be keep later versions of the standard theory library backwards compatible with earlier versions, which implies that we should exercise caution when adding theorems, because they might be hard to remove later.

The current version of the standard theory library does not make any use of parametric theories containing assumptions about uninterpreted type operators and constants, which users are expected to interpret to defined type operators and constants in their proof context. The advantage of using parametric theories is that proving a theorem once in a parametric theory makes it available 'for free' in every context in which it is used. The use of parametric theories has the potential to reduce the effort required to extend the standard theory library, while giving users more tools to use in their theory developments.

The standard theory library is based on the simple version of higher order logic implemented by the HOL family of theorem provers. There is a straightforward semantic embedding from this logic to the more complex versions of higher order logic implemented by the Isabelle/HOL [27] and PVS [30] theorem provers, making it technically possible to import OpenTheory packages into these systems. However, there is an interesting line of research in designing importers that result in 'natural-looking' theories in the target system. Such an importer could modify the theories as they were processed (similar to the de-tagging rewriting described in Section 4.2) to use logical features of the target system such as Isabelle/HOL type classes or PVS subtypes.

Acknowledgements

The OpenTheory project was initiated in 2004 as a result of discussions between Rob Arthan and the author, and the work since then has been guided by feedback from many other people, including John Harrison, Rebekah Leslie, John Matthews, Michael Norrish and Konrad Slind. This paper was greatly improved by comments from Rob Arthan, Ramana Kumar, Lee Pike and the anonymous referees.

References

1. Adams, M.: Introducing HOL zero. In: Fukuda, K., van der Hoeven, J., Joswig, M., Takayama, N. (eds.) ICMS 2010. LNCS, vol. 6327, pp. 142–143. Springer, Heidelberg (2010)
2. Arthan, R.D., Jones, R.B.: Z in HOL in ProofPower. In: BCS FACS FACTS (January 2005)
3. Ballarin, C.: Locales and locale expressions in isabelle/Isar. In: Berardi, S., Coppo, M., Damiani, F. (eds.) TYPES 2003. LNCS, vol. 3085, pp. 34–50. Springer, Heidelberg (2004)
4. Berghofer, S., Nipkow, T.: Proof terms for simply typed higher order logic. In: Aagaard, M.D., Harrison, J. (eds.) TPHOLs 2000. LNCS, vol. 1869, pp. 38–52. Springer, Heidelberg (2000)
5. Bortin, M., Johnsen, E.B., Lüth, C.: Structured formal development in Isabelle. Nordic Journal of Computing 13, 1–20 (2006)
6. Coutts, D., Potoczny-Jones, I., Stewart, D.: Haskell: Batteries included. In: Gill, A. (ed.) Haskell 2008: Proceedings of the first ACM SIGPLAN symposium on Haskell, pp. 125–126. ACM, New York (2008)
7. Craigen, D., Kromodimoeljo, S., Meisels, I., Pase, B., Saaltink, M.: EVES: An overview. Technical Report CP-91-5402-43, ORA Corporation (1991)
8. Dolstra, E., Löh, A.: Nixos: a purely functional linux distribution. In: Hook, J., Thiemann, P. (eds.) Proceedings of the 13th ACM SIGPLAN International Conference on Functional Programming (ICFP 2008), pp. 367–378. ACM, New York (2008)
9. Farmer, W.M.: Theory interpretation in simple type theory. In: Heering, J., Meinke, K., Möller, B., Nipkow, T. (eds.) HOA 1993. LNCS, vol. 816, pp. 96–123. Springer, Heidelberg (1994)
10. Farmer, W.M.: The seven virtues of simple type theory. Journal of Applied Logic 6, 267–286 (2008)
11. Gordon, M., Milner, R., Wadsworth, C.: Edinburgh LCF. LNCS, vol. 78. Springer, Heidelberg (1979)
12. Gordon, M.J.C., Melham, T.F. (eds.): Introduction to HOL (A theorem-proving environment for higher order logic). Cambridge University Press, Cambridge (1993)
13. Gordon, M.J.C.: Proof, Language, and Interaction: Essays in Honour of Robin Milner. From LCF to HOL: A Short History, ch. 6. MIT Press, Cambridge (2000)
14. Hales, T.C.: Introduction to the Flyspeck project. In: Coquand, T., Lombardi, H., Roy, M.-F. (eds.) Mathematics, Algorithms, Proofs. Dagstuhl Seminar Proceedings, vol. 05021, Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl (2006)

15. Harrison, J.: HOL light: A tutorial introduction. In: Srivas, M., Camilleri, A. (eds.) FMCAD 1996. LNCS, vol. 1166, pp. 265–269. Springer, Heidelberg (1996)
16. Harrison, J.: Formalizing basic complex analysis. In: Matuszewski, R., Zalewska, A. (eds.) From Insight to Proof: Festschrift in Honour of Andrzej Trybulec, Studies in Logic, Grammar and Rhetoric, vol. 10(23), pp. 151–165. University of Białystok (2007)
17. Homeier, P.V.: The HOL-Omega Logic. In: Berghofer, S., Nipkow, T., Urban, C., Wenzel, M. (eds.) TPHOLs 2009. LNCS, vol. 5674, pp. 244–259. Springer, Heidelberg (2009)
18. Hurd, J.: A formal approach to probabilistic termination. In: Carreño, V.A., Muñoz, C.A., Tahar, S. (eds.) TPHOLs 2002. LNCS, vol. 2410, p. 230-245. Springer, Heidelberg (2002)
19. Hurd, J.: OpenTheory: Package management for higher order logic theories. In: Reis, G.D., Théry, L. (eds.) PLMMS 2009: Proceedings of the ACM SIGSAM 2009 International Workshop on Programming Languages for Mechanized Mathematics Systems, pp. 31–37. ACM, New York (2009)
20. Hurd, J.: Composable packages for higher order logic theories. In: Aderhold, M., Autexier, S., Mantel, H. (eds.) Proceedings of the 6th International Verification Workshop (VERIFY 2010) (July 2010)
21. Hurd, J.: OpenTheory Article Format (August 2010), Available for download at `http://gilith.com/research/opentheory/article.html`
22. Kammüller, F.: Modular reasoning in Isabelle. In: McAllester, D.A. (ed.) CADE 2000. LNCS, vol. 1831. Springer, Heidelberg (2000)
23. King, D.J., Arthan, R.D.: Development of practical verification tools. ICL Systems Journal 11(1) (May 1996)
24. Klein, G., Elphinstone, K., Heiser, G., Andronick, J., Cock, D., Derrin, P., Elkaduwe, D., Engelhardt, K., Kolanski, R., Norrish, M., Sewell, T., Tuch, H., Winwood, S.: seL4: Formal verification of an OS kernel. In: Matthews, J.N., Anderson, T.E. (eds.) Proceedings of the 22nd ACM Symposium on Operating Systems Principles, pp. 207–220. ACM, New York (2009)
25. Leroy, X.: Formal certification of a compiler back-end or: programming a compiler with a proof assistant. In: Morrisett, J.G., Jones, S.L.P. (eds.) Proceedings of the 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2006), pp. 42–54. ACM, New York (2006)
26. Naur, P., Randell, B. (eds.): Software Engineering. Scientific Affairs Division, NATO (October 1968)
27. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL. LNCS, vol. 2283. Springer, Heidelberg (2002)
28. Norrish, M., Slind, K.: A thread of HOL development. The Computer Journal 41(1), 37–45 (2002)
29. Obua, S., Skalberg, S.: Importing HOL into Isabelle/HOL. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 298–302. Springer, Heidelberg (2006)
30. Owre, S., Shankar, N., Rushby, J.M., Stringer-Calvert, D.W.J.: PVS System Guide. Computer Science Laboratory, SRI International, Menlo Park, CA (September 1999)
31. Owre, S., Shankar, N.: Theory interpretations in PVS. Technical Report SRI-CSL-01-01, SRI International (April 2001)
32. Rabe, F.: Representing Logics and Logic Translations. PhD thesis, Jacobs University Bremen (May 2008)

33. Völker, N.: HOL2P - A System of Classical Higher Order Logic with Second Order Polymorphism. In: Schneider, K., Brandt, J. (eds.) TPHOLs 2007. LNCS, vol. 4732, pp. 334–351. Springer, Heidelberg (2007)
34. Westfold, S.: Integrating Isabelle/HOL with Specware. In: Schneider, K., Brandt, J. (eds.) Theorem Proving in Higher Order Logics: Emerging Trends Proceedings. Department of Computer Science, University of Kaiserslautern Technical Reports, vol. 364/07 (August 2007)
35. Wong, W.: Recording and checking HOL proofs. In: Schubert, E.T., Windley, P.J., Alves-Foss, J. (eds.) HUG 1995. LNCS, vol. 971, pp. 353–368. Springer, Heidelberg (1995)