Theory Engineering Using Composable Packages

Joe Leslie-Hurd

Intel Corp.
joe@gilith.com

SVARM & VMCAI
21 January 2013

Theory Engineering

- 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.
  - **Slogan:** *"Proving in the large."*

Software Engineering for Theories

An incomplete list of software engineering techniques applicable to the world of theories:

- **Standards:** Programming languages, basis libraries.
- **Abstraction:** Module systems to manage the namespace and promote reuse.
- **Multi-Language:** Tight/efficient (e.g., FFIs) to loose/flexible (e.g., SOAs).
- **Distribution:** Package repos with dependency tracking and automatic installation.

OpenTheory Project

- In theory, mathematical 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.[1]
- **Slogan:** *"Logic is an ABI for mathematics."*

[1] OpenTheory was initiated in 2004 with Rob Arthan.

Theory Definition

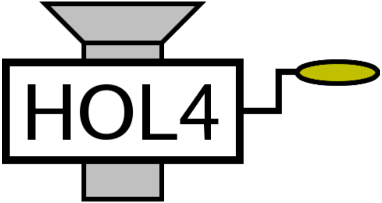
- A theory $\Gamma \rhd \Delta$ of higher order logic consists of:
  1. A set Γ of assumption sequents.
  2. A set Δ of theorem sequents.
  3. A formal proof that the theorems in Δ logically derive from the assumptions in Γ.
- **This Talk:** A common standard for packaging higher order logic theories that allows us to:
  - Liberate theories from the theorem proving system in which they were created.
  - Compose theories from different origins.
  - Process theories with a diverse array of tools.

Talk Plan

1. Proof Articles
2. Application: Sharing Proofs between Theorem Provers
3. Theory Packages
4. Application: Community Theory Development
5. Application: Synthesizing Verified Programs
6. Summary

Anatomy of an Interactive Theorem Prover

Goals



*Theorems**

- Interactive theorem provers are really high assurance proof checkers.
- Users set goals and invoke automatic tactics to break goals into subgoals.
- Tactics generate pieces of proof as a by-product of breaking down goals.

*Made with mechanically extracted proof.

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 theorem prover 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 theorem prover thus reduces to soundness of the logical kernel.

HOL4 theorem prover $\sim$ the elephant
logical kernel $\sim$ the ball

The OpenTheory Logical Kernel

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

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

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

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

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

Proofs are (Stack-Based) Programs

- The proof of theorems constructed using the OpenTheory logical kernel can be represented by an article.
- A proof article takes the form of a program for a stack-based virtual machine.
  - The program consists of a sequence of 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.
- Some 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 term variable *v* with name *n* and type *ty* onto the stack.

Stack:	Before:	Type *ty*
		:: Name *n*
		:: stack
	After:	Var *v*
		:: stack

Article Primitive Inferences

- There are commands implementing each primitive inference in the OpenTheory logical kernel (e.g., refl).
- Constants and type operators contain pointers to their definitions, eliminating the need for a global symbol table.[2]

Definition (The "refl" article command)

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

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

[2] An idea of Freek Wiedijk.

Article Assumptions

- The axiom command imports an assumption into the article.
- The context supplies the assumption theorem (e.g., by creating a new axiom).

Definition (The "axiom" article command)

```
axiom
    Pop a term c; pop a list of terms [h1, ..., hn];
    push the theorem {h1, ..., hn} ⊢ c onto the stack
    and add it to the article assumptions.

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

Article Theorems

- The thm command exports a theorem from the article.
- The particular form eliminates any differences caused by capture-avoiding substitution implementations.

Definition (The "thm" article command)

thm

Pop a term c; pop a list of terms $[h_1, \ldots, h_n]$; pop a theorem th; alpha-convert the theorem th to $\{h_1, \ldots, h_n\} \vdash c$ and add it to the article theorems.

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

Example Proof Article

```
# TINY EXAMPLE ARTICLE
#
# Construct the hypothesis list
nil
# Construct the conclusion term
"T"
const
"bool"
typeOp
nil
opType
constTerm
1
def
# Import an assumption: ⊢ T
axiom
# Export a theorem: ⊢ T
nil
1
remove
thm
```

- Article commands are executed by a stack-based virtual machine.
- The result is a theory $\Gamma \triangleright \Delta$:
  - Γ is the set of imported assumptions.
  - Δ is the set of exported theorems.

Theory (Tiny example result)

```
1 input type operator: bool
1 input constant: T
1 assumption:
  ⊢ T
1 theorem:
  ⊢ T
```

Sharing Proofs between Theorem Provers

- **Aim:** Share proofs between three interactive theorem provers in the HOL family:
  - HOL4, HOL Light and ProofPower.
- What do they have **in common?**
  - Theorem provers in the LCF design.
  - They implement the same higher order logic as the OpenTheory logical kernel.[3]
- What is **different?**
  - Contain different theories.
  - Implement different proof tools.

[3] The particular higher order logic is Church's simple theory of types, extended with Hindley-Milner style type variables.

Current Practice: Porting Proof Scripts

Porting theories between theorem provers is typically carried out by manually porting proof scripts:

Code (Example HOL Light proof script)

```
let MODULAR_TO_NUM_DIV_BOUND = prove
  (`!x. modular_to_num x DIV modulus = 0`,
   GEN_TAC THEN
   MATCH_MP_TAC DIV_LT THEN
   REWRITE_TAC [MODULAR_TO_NUM_BOUND]);;
```

This is a labor-intensive process, and its success relies on the target system containing similar proof tools and dependent theories.

Alternative: Proof Articles

- **Idea:** Instead of porting the source proof script, execute the script and record the generated primitive inference rules in the form of proof articles.
- Separates the concerns of proof search and proof storage:
  - Proof scripts often call proof tools that explore a search space.
  - Primitive inference proofs simply store the result of the search.
- **Benefit:** Primitive inference proofs do not rely on any proof tools, so are immune to bit-rot and can be read by any HOL theorem prover.
- **Drawback:** Primitive inference proofs are not human readable, so theories should be packaged only when they are stable enough to be archived and shared.

Proof Standardization

To share proof articles extracted from a theorem prover, we must standardize them to remove implementation-dependent data.

We used the following techniques to standardize HOL Light proofs:

1. Mapping HOL Light names of type operators and constants into the OpenTheory standard namespace.
2. Compiling HOL Light primitive inference rules to OpenTheory.
    - e.g., expressing TRANS in terms of refl, appThm and eqMp.
3. Removing HOL Light term tags.
    - e.g., post-processing proofs to rewrite NUMERAL $t \to t$.

Such techniques need to be invertible to import standardized proofs into HOL Light.

Compressing Articles

- To test the article format, we instrumented HOL Light v2.20 to emit articles for all of the theory files in the distribution.
- **Challenge:** Proofs fully expanded to primitive inferences result in large article files.
- **Good News:** Automatic compression techniques are effective on proof articles:
  - The equivalent of hash-consing for types, terms and theorems.
  - Dead-inference elimination (garbage collector trick).
- **Bonus:** These compression techniques have little effect on the compression ratio ($\sim 90\%$) of standard tools such as `gzip`.
- **Upshot:** A compressed article storing all the HOL Light theories contains 769,138 primitive inferences.
  - Further compressing with `gzip` results in a 18Mb file.

Cloud Tactics

- The proof article format has been used to manually port theories from HOL Light to HOL4.
- The format can also be used to share proof tools between theorem provers.[4]
- Wrapping a theorem prover in a CGI script creates cloud tactics available to any theorem prover in the HOL family.
- In fact, the proof article format is simple enough that the CGI script need not even contain a theorem prover.
- Kumar wrote a standalone Haskell program to prove equivalences between different number representations.

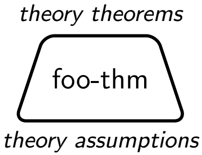
[4] Kumar and Hurd, *Standalone Tactics Using OpenTheory*, ITP 2012.

Basic Theory Packages

- A basic theory package just wraps a proof article with some meta-data.
- We depict theory packages $\Gamma \rhd \Delta$ as named proof boxes that build up from an assumption set Γ to a theorem set Δ.

Theory (Basic theory package)

```
name: foo-thm
version: 1.0
author: Joe Leslie-Hurd <joe@gilith.com>

main {
  article: "foo-thm.art"
}
```

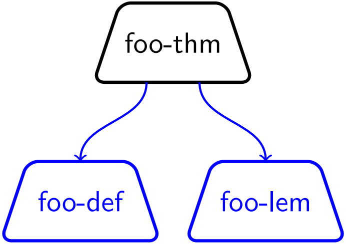


Required Theory Packages

- Theorems of required theories listed in a package must collectively satisfy all theory assumptions.
- In this way we can specify and check logical dependencies between a collection of theory packages.

Theory (Required theories)

```
name: foo-thm
version: 1.0
author: JLH <joe@gilith.com>
requires: foo-def
requires: foo-lem

main {
  article: "foo-thm.art"
}
```



Checking Theory Dependencies

- A theory package $\Gamma \rhd \Delta$ in a collection is up-to-date if it is possible to prove all of its theorems *'from scratch'*.
- This boils down to the following two conditions:
  1. Every required theory package $\Gamma_i \rhd \Delta_i$ is up-to-date and proves the theorem set Θ_i.
  2. The theory $\Gamma \rhd \Delta$ can be imported into $\bigcup_i \Theta_i$, proving the theorem set Θ.
- Importing a theory $\Gamma \rhd \Delta$ into a theorem set Θ means:
  - replacing input symbols with defined symbols in Θ; and
  - satisfying all assumptions with theorems in Θ.
- Proof articles can be imported into Θ while executing them.
  - Modify the typeOp, const and axiom commands to use Θ.

What Can Go Wrong?

- **Circular Reasoning:** Theory package dependency graphs must not contain any loops!
  - Theory packages are representations of proofs, which are directed acyclic graphs.
- **Inconsistent Definitions:** The same constant or type operator must not be defined in multiple required theory packages.
  - **Example:** The two theories

    $$\emptyset \rhd \{\vdash \mathsf{c} = 0\} \qquad \text{and} \qquad \emptyset \rhd \{\vdash \mathsf{c} = 1\}$$

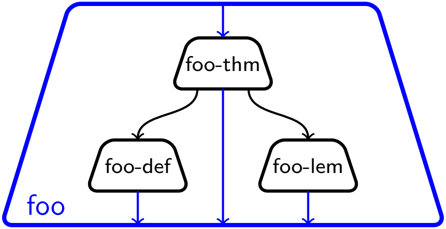
    are individually fine, but must never be required by the same theory package.

Nested Theory Packages

Theory (Nested theories)

```
name: foo
version: 1.0
author: JLH <joe@gilith.com>

def {
  package: foo-def-1.0
}

lem {
  package: foo-lem-1.0
}

thm {
  import: def
  import: lem
  package: foo-thm-1.0
}

main {
  import: thm
}
```

- Theory packages can contain nested theories.
- Proofs of nested theories are replayed, with optional renaming of symbols.



Semantic Embeddings

- Packaging theories as primitive inference rules solves the problem of differences in theorem prover proof tools.
- But how to deal with differences in the available theories?
- To successfully port a theory from theorem prover context A to B, we must find a semantic embedding $A \to B$ mapping type operators and constants in A to ones in B with properties that are at least as logically strong.
- We will need semantic embeddings from the core theories of each theorem prover in the HOL family to the core theories of the others.

Standard Theory Library

- Instead of maintaining pairwise semantic embeddings, we take the core theories and release a standard theory library of them in OpenTheory format.
- Distributes responsibility: each theorem prover maintains the semantic embeddings to and from the standard theory library.
- Serves as 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."*
- Permits dynamic linking of proofs: theorems proved in the standard theory library can be used by any theory.

Identifying Core Theories

- By looking at the system documentation and source code for HOL Light, HOL4 and ProofPower, we can identify a core set of theories present in each theorem prover.
- For the core theories, the semantic embeddings between the theorem provers are just renamings of the type operators and constants.
- OpenTheory implements hierarchical namespaces for type operators and constants to help avoid name clashes.

Standard Theories

The standard theory library lives inside the following namespace:

- Data
  - Bool – The boolean type
  - List – List types
  - Option – Option types
  - Pair – Product types
  - Sum – Sum types
  - Unit – The unit type
- Function – Theory of functions
- Number
  - Natural – Natural numbers
  - Real – Real numbers
- Set – Theory of sets
- Relation – Theory of relations

Construction Technique

We use the following procedure for converting standardized proofs extracted from HOL Light into the standard theory library:

1. Create a basic theory package wrapping each emitted proof.
2. Create nested theory packages for higher-level topics, such as `bool` or `list`.
3. Create a theory package called `base` which is a nesting of the highest-level theory packages.

Axioms

- It is standard practice in the higher order logic theorem proving community to avoid axioms.
- An exception is made for a small set of standard axioms that are used to set up the basic theories of higher order logic.
- The OpenTheory standard theory library is built on top of the following three axioms:
  1. **Extensionality:** $\vdash \forall t.\ (\lambda x.\ t\ x) = t$
  2. **Choice:** $\vdash \forall p, x.\ p\ x \implies p\ (\text{select}\ p)$
  3. **Infinity:** $\vdash \exists f : \text{ind} \to \text{ind}.\ \text{injective}\ f \land \neg \text{surjective}\ f$

Profiling the Standard Theory Library

The standard theory library consists of:

- 139 theory packages
  - = 102 basic
  - + 36 higher-level
  - + 1 top-level base
- 3 axioms
- 6 defined type operators
- 64 defined constants
- 450 theorems

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**

Community Theory Development

- The standard theory library is designed to be supported by all theorem provers in the HOL family.
- Therefore any theory that is built on top of the standard theory library can be shared between HOL theorem provers.
- We have implemented a web-based repository to support community theory development.
  - **Now:** Allowing developers to upload packages and share them with the community.
  - **Soon:** Automatically track logical dependencies between theory versions.
  - **Future:** Searching through theories for relevant theorems.

Theory Repository Demo

Gilith OpenTheory Repo

opentheory.gilith.com

Gilith OpenTheory Repo

packages • recent • upload

Log in

Welcome to the Gilith OpenTheory repo, which is currently storing 35 theory packages. Each theory package contains a collection of theorems together with their proofs. The proofs have been broken down into the primitive inferences of higher order logic, allowing them to be checked by computer.

This web interface is provided to help browse through the available packages, but the recommended way of downloading and processing theory packages is to use the opentheory package management tool. For more information on OpenTheory please refer to the project homepage.

Recently Uploaded Packages [more]

haskell-prime-1.25 — Prime numbers
Uploaded 7 weeks ago by Joe Leslie-Hurd

haskell-char-1.43 — Unicode characters
Uploaded 7 weeks ago by Joe Leslie-Hurd

haskell-parser-1.119 — Stream parsers
Uploaded 7 weeks ago by Joe Leslie-Hurd

OpenTheory twitter feed:

- haskell-prime-1.25 uploaded by Joe Leslie-Hurd *http://t.co/y6QZ0MOJ 50 days ago*
- haskell-char-1.43 uploaded by Joe Leslie-Hurd *http://t.co/ohnibphF 50 days ago*
- haskell-parser-1.119 uploaded by Joe Leslie-Hurd *http://t.co/il0gIGWP 50 days ago*

Gilith OpenTheory Repo, maintained by Joe Leslie-Hurd. ✓

Theory Package Design

- What makes a well-designed theory package contribution?
  1. A clear topic (e.g., trigonometric functions).
  2. All assumptions are satisfied by well-designed theory packages.
  3. Any defined symbols are generally useful and occupy descriptive slots in the hierarchical namespace.
  4. A carefully chosen set of theorems (no junk, no free vars).
- **Note:** None of these conditions can be automatically checked—*being well-designed is a matter of taste*.

Packaging Verified Software

Using theory packages for verified software addresses many of the logistical needs:

- **Distribution:** Download software from repos, check the proofs, and install on your local machine.
- **Versioning:** Developers can release new versions of software, obsolete packages can be marked.
- **Upgrade:** Can statically guarantee that an upgrade will be safe, so long as the required properties still hold of the new version.

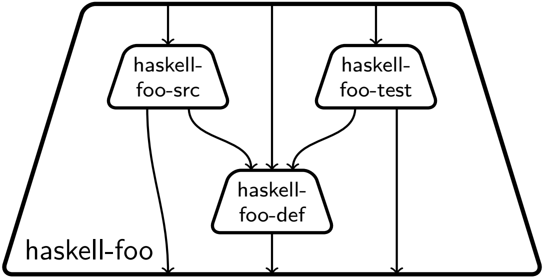
Formal Verification of Haskell Packages

- Haskell is a functional programming language that is rapidly growing in popularity.
  - Its package system makes it easy to reuse code.
  - There are 4,609 unique Haskell packages available at the Hackage repo.
- There is a well-known correspondance between higher order logic functions and a pure subset of the Haskell language.[5]
- **Case Study:** Verify higher order logic functions, then automatically generate Haskell programs.
  - The synthesis tool operates at the package level: OpenTheory packages to Haskell packages.

[5] Haftmann, *From Higher-Order Logic to Haskell*, PEPM 2010.

Analyzing the Source OpenTheory Package

Consider a package `haskell-foo` with three nested theories:



- `def`: Defining Haskell structures in terms of verified functions.
- `src`: Deriving computational forms for the Haskell structures.
- `test`: Deriving executable properties of the Haskell structures.

Synthesizing a Haskell Package

The target Haskell package is synthesized from the source OpenTheory package as follows:

1. The source code is generated by pretty-printing the computational forms in the `src` nested package.
2. A QuickCheck test suite is generated from the executable properties in the `test` nested package.
3. Most of the package meta-data is derived from the OpenTheory package meta-data.

Synthesizing Haskell Package Build Dependencies

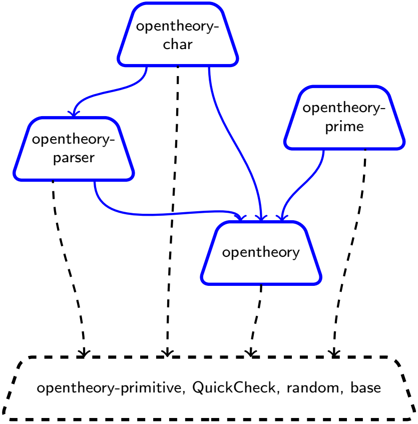
- **Problem:** Generating the meta-data that describes the acceptable version ranges of required Haskell packages.
- **Solution:** Analyze the corresponding OpenTheory packages, and select a set of version ranges for which the source package is up-to-date.
- *"Bringing the benefits of logical theories back to software engineering!"*

Verified Haskell Packages

- The synthesis scheme was tested on some example packages.
- They are all available on Hackage.

Code (opentheory-prime)

```
build-depends:
  base >= 4.0 && < 5.0,
  random >= 1.0.1.1 && < 2.0,
  QuickCheck
    >= 2.4.0.1 && < 3.0,
  opentheory-primitive
    >= 1.0 && < 2.0,
  opentheory >= 1.73 && <= 1.74
```



Summary

- We presented a common standard for packaging higher order logic theories, allowing them to be processed by diverse tools.
- This capability was first used by theorem provers to share theories and support community theory development.
- But new proof-of-concept tools are being developed too: standalone cloud tactics and verified program synthesizers.
- The current challenge is to make theories easier to work with, for example by automatically tracking their logical dependencies and making their theorems searchable.

Any Questions?

`joe@gilith.com` `@gilith`

`gilith.com/research/opentheory`

Compressing the HOL Light Articles

HOL Light theory	article (Kb)	gzip'ed ratio	compress (Kb)	gzip'ed ratio
num	1,820	12%	813	13%
arith	27,469	10%	7,548	13%
wf	29,277	11%	6,330	13%
calc_num	3,922	9%	1,570	12%
normalizer	2,845	10%	688	13%
grobner	2,417	10%	748	13%
ind-types	10,625	11%	4,422	13%
list	12,368	12%	4,870	13%
realax	23,628	10%	7,989	13%
calc_int	2,844	11%	861	13%
realarith	16,275	8%	4,684	12%
real	30,031	10%	9,346	13%
calc_rat	2,555	11%	1,166	13%
int	40,617	8%	9,546	13%
sets	168,586	10%	30,315	13%
iter	207,324	8%	32,422	12%
cart	20,351	10%	3,632	13%
define	82,185	9%	16,409	13%

Profiling the Standard Theory Library

What if we compress the 139 theory packages into one giant proof?

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**