HOL with Definitions: Semantics, Soundness, and a Verified Implementation

Ramana Kumar[1] Rob Arthan[2]
Magnus O. Myreen[1] Scott Owens[3]

[1]Computer Laboratory, University of Cambridge

[2]School of EECS, Queen Mary, University of London

[3]School of Computing, University of Kent

Interactive Theorem Proving, 2014
Vienna Summer of Logic

Verified HOL: The Goal

Produce a useful theorem proving system together with a proof that every theorem obtained by running the system (according to the semantics of the machine-code) is true according to the semantics of higher-order logic.

Verified HOL: The Goal

Produce a useful theorem proving system together with a proof that every theorem obtained by running the system (according to the semantics of the machine-code) is true according to the semantics of higher-order logic.

Achieved: formal **semantics** for HOL, **soundness** of the inference system and principles of definition, **verified** high-level **implementation**

Remaining: **interface** to proved theorems (printing), verification of **LCF architecture**

Why Verify a Theorem Prover?

For Leverage

The theorem prover sits at centre of the trusted code base.

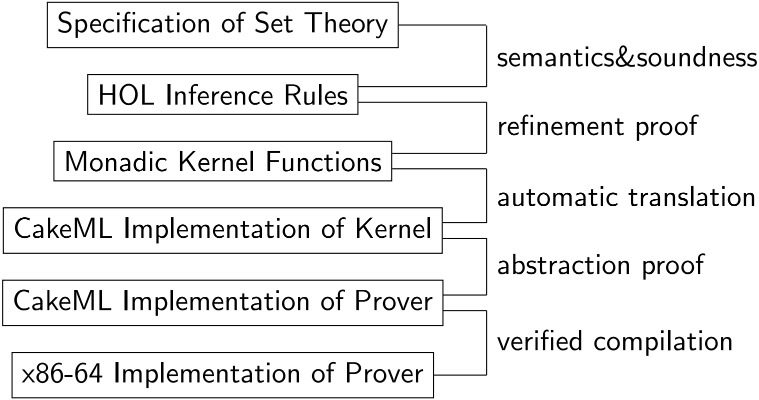
For Understanding

Formalisation clarifies details of the logic and the implementation.

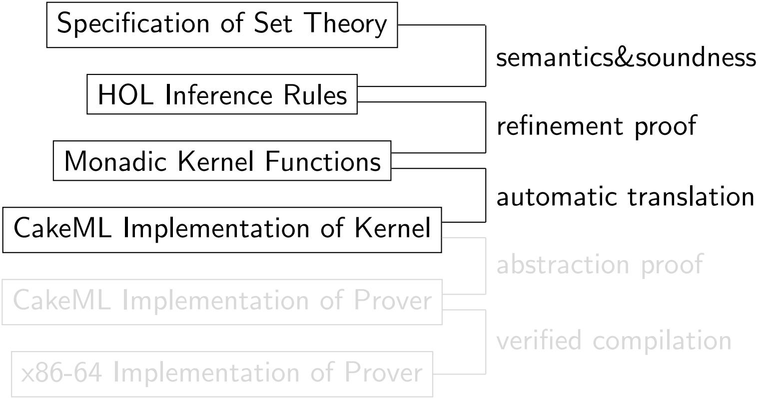
As a Catalyst

Being medium-sized, with a clear specification, a verified theorem prover is a good testing ground for application-verification tools.

Verified HOL: The Approach



Verified HOL: The Approach



Outline

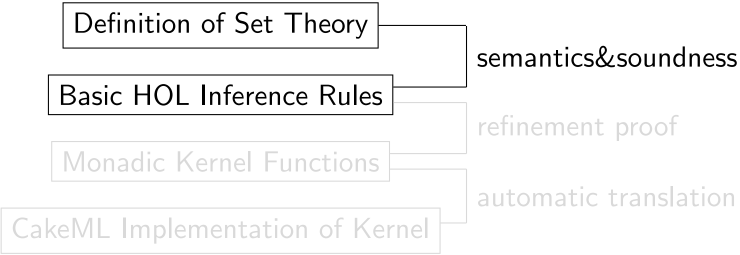
Motivation
- Verified Theorem Provers
- Previous Work and Context

Formalising all of HOL
- Specification of Set Theory
- Basic HOL Semantics and Soundness
- Supporting a Context of Definitions
- Consistency of HOL's Axioms

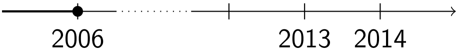
(Towards) Verifying HOL Light
- Monadic Implementation in HOL
- Producing CakeML for Compilation

Towards Self-Verification of HOL Light

Harrison, IJCAR 2006:

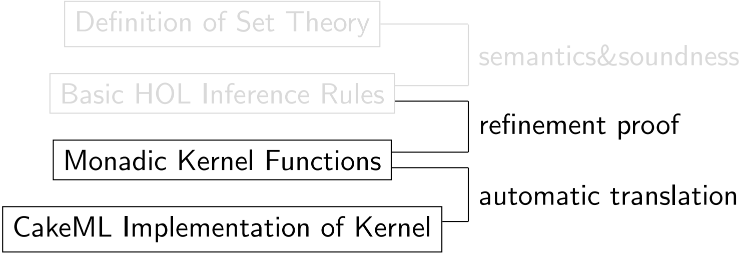


Does not include rules for making definitions.

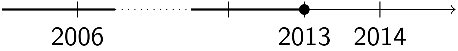


Towards Self-Verification of HOL Light

Myreen et al, ITP 2013:

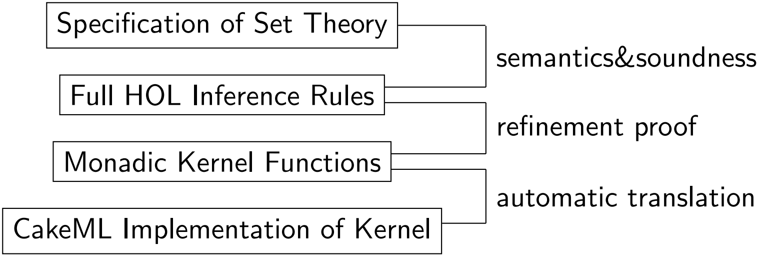


Does not connect to the semantics.

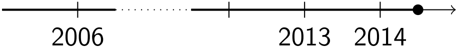


Towards Self-Verification of HOL Light

This work:

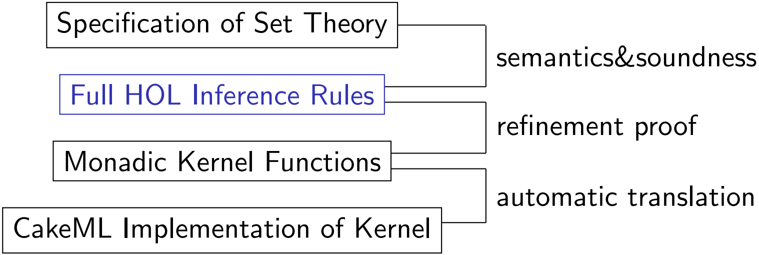


Includes both semantics and rules for definitions.

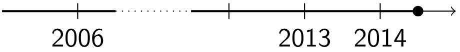


Towards Self-Verification of HOL Light

This work (after the paper):

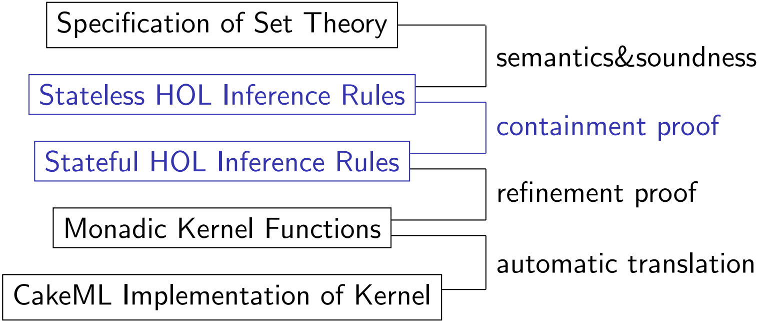


Includes both semantics and rules for definitions.

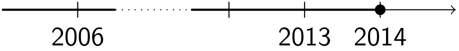


Towards Self-Verification of HOL Light

This work (in the paper):

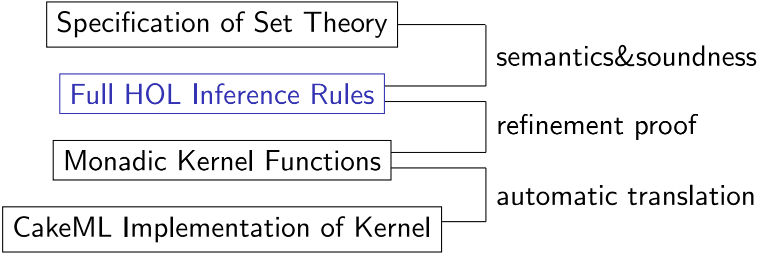


Includes both semantics and rules for definitions.

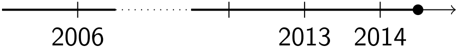


Towards Self-Verification of HOL Light

This work (after the paper):



Includes both semantics and rules for definitions.



Outline

Motivation
- Verified Theorem Provers
- Previous Work and Context

Formalising all of HOL
- Specification of Set Theory
- Basic HOL Semantics and Soundness
- Supporting a Context of Definitions
- Consistency of HOL's Axioms

(Towards) Verifying HOL Light
- Monadic Implementation in HOL
- Producing CakeML for Compilation

Specifying the Semantic Domain

Basic Idea

$$\text{is\_set\_theory}\ (mem : \mathcal{U} \rightarrow \mathcal{U} \rightarrow \textit{bool})$$

Specifying Axioms

- Extensionality
  $\forall\, x\ y.\ x = y \iff \forall\, a.\ mem\ a\ x \iff mem\ a\ y$
- Separation
  $\forall\, a\ x\ P.\ mem\ a\ (sep\ x\ P) \iff mem\ a\ x\ \wedge\ P\ a$
- etc.

Compared to Defining the Universe

Harrison's Original Approach

$\mathsf{mem} : V \rightarrow V \rightarrow \mathit{bool}$

$\mathsf{level}\ (\mathsf{sep}\ x\ P) = \mathsf{level}\ x$

$\mathsf{mem}\ a\ (\mathsf{sep}\ x\ P) \iff \mathsf{mem}\ a\ x \land P\ a$

Advantages of New Approach

- Avoid stratifying sets into levels, get extensionality.
- Isolate the assumption required for the axiom of infinity.

Derived Operations

Define Useful Sets

Empty set, Cartesian products, functions-as-graphs, etc.

Prove Interface Theorems

$$\begin{array}{l}\vdash \ \textsf{is\_set\_theory}\ mem \ \Rightarrow \\ \quad \forall f\ x\ s\ t. \\ \qquad mem\ x\ s \ \wedge\ mem\ (f\ x)\ t \ \Rightarrow \\ \qquad \textsf{apply}\ mem\ (\textsf{abstract}\ mem\ s\ t\ f)\ x \ =\ f\ x\end{array}$$

A layer of such theorems, supported by the set theory axioms, is what supports the HOL soundness proof.

Outline

Motivation
- Verified Theorem Provers
- Previous Work and Context

Formalising all of HOL
- Specification of Set Theory
- Basic HOL Semantics and Soundness
- Supporting a Context of Definitions
- Consistency of HOL's Axioms

(Towards) Verifying HOL Light
- Monadic Implementation in HOL
- Producing CakeML for Compilation

Formalising HOL Syntax

Define Types and Terms

type = Tyvar *string* | Tyapp *string* (*type list*)
term = Var *string type* | Const *string type* |
Comb *term term* | Abs *string type term*

Define Inference System

$$\frac{\text{theory\_ok } thy \quad p \text{ has\_type Bool} \quad \text{term\_ok (sigof } thy\text{) } p}{(thy, [p]) \ \Vdash \ p} \ \text{ASSUME} \qquad \frac{\text{theory\_ok } thy \quad \text{term\_ok (sigof } thy\text{) } t}{(thy, []) \ \Vdash \ t \ == \ t} \ \text{REFL}$$

etc.

Semantics of Types and Terms

Types are Inhabited Sets

typesem δ τ (Tyvar s) $=$ τ s
typesem δ τ (Tyapp $name$ $args$) $=$
  δ $name$ (map (typesem δ τ) $args$)

Terms are Elements of Their Types

termsem mem Θ (δ, γ) (τ, σ) (Abs x ty b) $=$
  abstract mem (typesem δ τ ty) (typesem δ τ (typeof b))
   ($\lambda\, m.$ termsem mem Θ (δ, γ) $(\tau, ((x, ty) \mapsto m)\ \sigma)$ b)
etc.

(In Stateless HOL, not shown, these need to be in mutual recursion and are rather more complicated.)

Soundness in a Fixed Context

Entailment

$(thy, h) \models c$ holds if:
every interpretation (δ, γ) that models thy also satisfies $h \models c$.

Soundness Theorem

$\vdash$ is_set_theory $mem \Rightarrow$
$\forall\, thy\ h\ c.\ (thy, h) \Vdash c \Rightarrow (thy, h) \models c$

Proved by induction on the inference system.
(mem is used by the term semantics inside $(thy, h) \models c$.)

Outline

Motivation
- Verified Theorem Provers
- Previous Work and Context

Formalising all of HOL
- Specification of Set Theory
- Basic HOL Semantics and Soundness
- Supporting a Context of Definitions
- Consistency of HOL's Axioms

(Towards) Verifying HOL Light
- Monadic Implementation in HOL
- Producing CakeML for Compilation

Theory Updates

Signatures

- Sequents carry a context: $(thy, h) \Vdash c$.
- thy says which constants are defined and their arity/type.
- thy also carries the set of axioms.

Extension Principles

- Basic idea: extend the theory with new constants or axioms.
- The sound rules for doing so have many side-conditions (hence skipped in previous formalisations).
- Simply adding new type operators, constants, or axioms to the theory is also possible (the latter may not be sound).

Soundness of Updates

Each update

receives some input data, then

- introduces axioms,
- introduces constants or type operators, and,
- has side-conditions.

An update is sound if

- whenever there is a model of the theory before the update,
- and the side conditions hold, then
- there is a model of the theory after the update.

Mainly: the introduced axioms (which mention the introduced constants) are consistent.

Type Definition

Data and Side-Conditions

- TypeDefn $name$ $pred$ abs rep,
- $(thy, []) \Vdash$ Comb $pred$ $witness$,
- $pred$ is closed, and all names are fresh.

Introduced Constants and Axioms

- Type operator $name$ with type variables in $pred$ as arguments.
- Constants abs and rep, functions between the new type and subset of the type of $witness$ where $pred$ holds.
- Axioms asserting abs and rep form a bijection.

Soundness

(For full details: see code at `https://cakeml.org`.)

Constant Specification

Data and Side-Conditions

- ConstSpec $(\bar{x} = \bar{t})$ $prop$,
- $(thy, \bar{x} = \bar{t}) \Vdash prop$,
- FV $prop \subseteq \bar{x}$, $\bar{t}$ all closed, and all type variables in type,
- $\bar{x}$ all distinct and fresh names.

For details, see Rob Arthan's talk tomorrow.

Introduced Constants and Axioms

- New constants $\bar{c}$ for each $\bar{x}$.
- New axiom: $prop[\bar{c}/\bar{x}]$.

Soundness

Outline

Motivation
- Verified Theorem Provers
- Previous Work and Context

Formalising all of HOL
- Specification of Set Theory
- Basic HOL Semantics and Soundness
- Supporting a Context of Definitions
- Consistency of HOL's Axioms

(Towards) Verifying HOL Light
- Monadic Implementation in HOL
- Producing CakeML for Compilation

The Three Mathematical Axioms

The Axioms

1. Extensionality: $(\lambda\, x.\, f\ x)\ =\ f$
2. Choice: $P\ x\ \Rightarrow\ P\ ((\varepsilon)\ P)$
3. Infinity: $\exists\, f.$ ONE_ONE $f\ \wedge$ ONTO f

Formalised as Updates

Choice: NewAxiom (Implies (Comb (Var "P" . . .) . . .) . . .) ::
NewConst "ε" (Fun (Fun A Bool) A) :: $ctxt$

The same framework can handle user-supplied axioms.

Consistency, Avoiding Self-Consistency

Main Theorem

$\vdash$ is_set_theory mem $\wedge$ $(\exists\, inf.$ INFINITE $\{\, a \mid mem\ a\ inf \,\})$ $\Rightarrow$
$\forall\, ctxt.$
$ctxt$ extends hol_ctxt $\wedge$
$(\forall\, p.$ NewAxiom $p \in ctxt \Rightarrow$ NewAxiom $p \in$ hol_ctxt$)$ $\Rightarrow$
$\exists\, p_1\ p_2.$ (thyof $ctxt$, []) $\Vdash p_1 \wedge \neg$((thyof $ctxt$, []) $\Vdash p_2)$

Explanation

Assuming we have a set-theory satisfying the axiom of infinity, every extension of HOL's initial theory context that does not introduce new axioms has both provable and unprovable sequents.

Outline

Motivation
- Verified Theorem Provers
- Previous Work and Context

Formalising all of HOL
- Specification of Set Theory
- Basic HOL Semantics and Soundness
- Supporting a Context of Definitions
- Consistency of HOL's Axioms

(Towards) Verifying HOL Light
- Monadic Implementation in HOL
- Producing CakeML for Compilation

HOL Light Kernel as Monadic Functions

Inference Rules

- Define theorem datatype:
  Sequent (h : *hol_term list*) (c : *hol_term*).
- For each clause of the $(thy, h) \Vdash c$ relation, define a monadic function that returns its conclusion.
- For example:

$$\frac{\text{every (type\_ok (tysof } thy\text{)) (map fst } tyin\text{)} \qquad (thy, h) \Vdash c}{(thy, \text{map (INST } tyin\text{) } h) \Vdash \text{INST } tyin\ c}\ \text{INST\_TYPE}$$

becomes

INST_TYPE $tyin$ (Sequent h c) $=$
  bind (map (inst $tyin$) h)
    ($\lambda\, l.$ bind (inst $tyin$ c) ($\lambda\, x.$ return (Sequent l x)))

HOL Light Kernel as Monadic Functions

Inference Rules

- Define theorem datatype:
  Sequent $(h$: *hol_term list*$)$ $(c$: *hol_term*$)$.
- For each clause of the $(thy, h) \Vdash c$ relation, define a monadic function that returns its conclusion.
- For example:

$$\frac{\begin{array}{c}\text{every (type\_ok (tysof } thy\text{)) (map fst } tyin\text{)} \\ (thy, h) \Vdash c\end{array}}{(thy, \text{map (INST } tyin\text{) } h) \Vdash \text{INST } tyin\ c}\ \text{INST\_TYPE}$$

  becomes

$$\begin{array}{l}\text{INST\_TYPE } tyin \text{ (Sequent } h\ c) = \\ \quad \text{bind (map (inst } tyin\text{) } h) \\ \qquad (\lambda\, l.\ \text{bind (inst } tyin\ c)\ (\lambda\, x.\ \text{return (Sequent } l\ x)))\end{array}$$

HOL Light Kernel as Monadic Functions

Principles of Definition

Monadic functions are in a state-exception monad.
The state includes:

- the term and type constants,
- the axioms, and,
- a log of the definitions.

For each theory-extension principle, define a monadic function.
This function:

- takes the data as input,
- checks the side-conditions, and,
- updates the references above.

HOL Light Kernel as Monadic Functions

Principles of Definition

Monadic functions are in a state-exception monad.
The state includes:

- the term and type constants,
- the axioms, and,
- a log of the definitions.

For each theory-extension principle, define a monadic function.
This function:

- takes the data as input,
- checks the side-conditions, and,
- updates the references above.

Verifying the Monadic Functions

Basic Idea

Prove: whenever a monadic function produces Sequent $h\ c$ in some good context thy on good arguments, then $(thy, h) \Vdash c$ holds.

Why Log Definitions?

- The semantics of theorem values is in context of the log.
- In real HOL Light the log is not stored (ephemeral).
- We could avoid the log in our state monad, at the expense of an existential quantifier on the verification theorems.

Outline

Motivation
- Verified Theorem Provers
- Previous Work and Context

Formalising all of HOL
- Specification of Set Theory
- Basic HOL Semantics and Soundness
- Supporting a Context of Definitions
- Consistency of HOL's Axioms

(Towards) Verifying HOL Light
- Monadic Implementation in HOL
- Producing CakeML for Compilation

Automatic Proof-Producing Translation

Shallow to Deep

INST_TYPE $tyin$ (Sequent h c) $=$
bind (map (inst $tyin$) h)
($\lambda\, l.$ bind (inst $tyin$ c) ($\lambda\, x.$ return (Sequent l x)))

becomes

```
fun inst_type tyin (Sequent (h,c)) =
  let val l = map (inst tyin) h
      val x = inst tyin c
  in Sequent (l,x) end
```

Certificate Theorem

Generated theorem relates above syntax via the operational semantics of CakeML to the monadic function INST_TYPE.

Proof Effort

Breakdown of Lines of Proof Script

Set-Theory Specification	319
HOL Syntax	347
Syntax Lemmas	1852
HOL Semantics	693
HOL Soundness & Consistency	2368
Monadic Kernel Functions	628
Kernel Verification	2644
Verified CakeML Production	1429
	10280

Builds on Existing Infrastructure

Namely: HOL4 and CakeML

Summary

Achievements

- The semantics and soundness of all of HOL (including definitions and axioms) has now been formalised in HOL.
- We have produced an implementation of the HOL Light kernel in CakeML, and verified it against the above semantics.

Outlook

- Next step: package the verified kernel as a module in a verified theorem prover.
- Self-verifying theorem provers raise interesting opportunities for logical reflection.