Saarland University
Faculty of Natural Sciences and Technology I
Department of Computer Science
Bachelor's Program in Computer Science

Bachelor's Thesis

Using LEO-II to Prove Properties of an Explicit Substitution M-set Model

submitted by

Xin Zhang

on October 22, 2008

Supervisor

Prof. Dr. Gert Smolka

Advisor

Dr. Chad E. Brown

Reviewers

Prof. Dr. Gert Smolka
Dr. Chad E. Brown

Statement

Hereby I confirm that this thesis is my own work and that I have documented all sources used.

Saarbrücken, <day of statement>

<Signature>

Declaration of Consent

Herewith I agree that my thesis will be made available through the library of the Computer Science Department.

Saarbrücken, <day of statement>

<Signature>

Acknowledgments

I wish to thank all those who have helped and encouraged me during my studies at Saarland University.

Above all, I am grateful to my supervisor Dr. Chad E Brown for numerous fruitful discussions during the development of my thesis. With his enthusiasm, his inspiration, and his great efforts to explain things clearly and simply, he helped to make theorem proving fun for me. I want to thank Prof. Dr. Gert Smolka for examining my thesis.

A final special word of thanks to my parents for their understanding, endless patience, love and support when it was most required.

Abstract

LEO-II is an extensional higher-order theorem prover based on resolution. It uses a higher-order logic based upon Church's simply typed-calculus, so that the comprehension axioms are implicitly handled. LEO-II employs a higher-order resolution calculus, where the search for an empty clause and higher-order pre-unification are interleaved.

In this project, we considered a particular mathematical domain: an M-set model of elementary type theory. We experimented with using the higher-order theorem prover LEO-II to verify properties of the model. In particular, we have isolated 14 theorems we would like LEO to prove automatically. Some of the theorems required intermediate lemmas to be formulated before the theorem can be proven automatically. These lemmas also needed to be automatically proven. We evaluated LEO based on how many intermediate lemmas are required in each case.

Contents

Chapter 1

Introduction

In this project, we have used an automated theorem prover LEO-II [3] to verify properties of a particular M-set model [4]. LEO-II is an extensional higher-order theorem prover based on resolution. In principle one should be able to formulate mathematical results in higher-order logic and have LEO help find proofs of the results.

We consider a particular mathematical domain: an M-set model [4] of elementary type theory. The $\lambda\sigma$-calculus of explicit substitutions induces a monoid M of explicit substitutions and an M-set of terms. This can be used to construct a model of elementary type theory in which one can interpret higher-order abstract syntax. Within this domain, we explore the effectiveness of using LEO to help find proofs. For example, we will attempt to use LEO to prove soundness of axioms for higher-order abstract syntax in the M-set model.

In particular, we have isolated 14 theorems we would like LEO to prove automatically. We have already proven these theorems by hand, but we would like to know how realistic it is to use LEO to prove these theorems automatically. This provides an evaluation of LEO, as well as an extra level of confidence that our proofs by hand are correct.

Several constants, definitions, axioms, and theorems were formulated in thf [2] syntax. Some of these are theorems that we would like to prove using LEO. Each of the theorems can either be proven assuming every fact that has already been given or using only some of the human-selected facts that have been given. Part of the project involves selecting the appropriate facts for a local version of each theorem.

LEO could prove 9 theorems without any help. We tried to solve the other 5

problems. For example, we split the theorem into smaller pieces by creating intermediate lemmas. LEO could sometimes prove the intermediate lemmas and sometimes prove the theorem from the lemmas. We have evaluated LEO based on how many intermediate lemmas are required in each case.

Chapter 2

The 14 Problems

2.1 M-sets

A monoid is a triple $< M, op, e >$ where

- M is set
- $op : M \times M \to M$ is a function (a binary operation on M)
- $e \in M$

satisfying

- $op(m, op(n,k)) = op(op(m,n),k)$ for all $m,n,k \in M$ (**associative law**)
- $op(m,e) = m = op(e,m)$ for $m \in M$ (**identity laws**)

We will refer to the monoid $< M, op, e >$ as M. We will leave the operation op implicit and write mn for $op(m,n)$. Using this convention, we write the associative and identity laws as follows:

- $(m(nk)) = ((mn)k)$
- $me = m = em$

Since the operation is associative we can omit parenthesis and write mnk instead of $m(nk)$ or $(mn)k$.

Let M be a monoid with identity e. An M-set is a pair $< A, \alpha >$ where

- A is a set
- α: $A \times M \longrightarrow A$ is a function (an action)

satisfying

- $\alpha(\alpha(a,m),n) = \alpha(a,op(m,n))$
- $\alpha(a,e) = a$

We will also leave the action α implicit and simply write am instead of $\alpha(a,m)$. Then, the defining properties of an M-set are written

- $(am)n = a(mn)$
- $ae = a$

For each base type β including the type o of truth values, choose an M-set $\mathcal{D}_\beta$. We can extend this to an M-set applicative structure by defining an M-set $\mathcal{D}_{\alpha\to\beta}$ for function types and defining functions @: $\mathcal{D}_{\alpha\to\beta} \times \mathcal{D}_\alpha \to \mathcal{D}_\beta$:

- Set: $\mathcal{D}_{\alpha\to\beta} := \{f : M \times \mathcal{D}_\alpha \to \mathcal{D}_\beta | \forall k,m \in M \forall a \in \mathcal{D}_\alpha . f(k,a)m = f(km,am)\}$
- Action: For $f \in \mathcal{D}_{\alpha\to\beta}$ and $m \in M$, define fm: $M \times \mathcal{D}_{\alpha\to\mathcal{D}_\beta}$ by $fm(k,a) := f(mk,a)$
- Application: $f@a := f(e,a)$ where $f \in \mathcal{D}_{\alpha\to\beta}, a \in \mathcal{D}_\alpha$, and $e \in M$ is the identity of M.

See [4] for details.

2.2 λσ-calculus

We briefly describe the untyped λσ-calculus, using explicit substitutions [1, 5] and deBruijn indices.

We inductively define the set of terms and the set of explicit substitutions to be the least freely generated sets satisfying:

- 1 is a term.

- $(a\,b)$ is a term if a and b are terms.
- (λa) is a term if a is a term.
- $(a[s])$ is a term if a is a term and s is an explicit substitution.
- id is an explicit substitution.
- $\uparrow$ is an explicit substitution.
- $(a.s)$ is an explicit substitution if a is a term and s is an explicit substitution.
- $(s \circ t)$ is an explicit substitution if s and t are explicit substitutions.

We often leave out parentheses. Application associates to the left: $(a\,b^1 \cdots b^n)$ means $(\cdots(a\,b^1)\cdots b^n)$. Composition associates to the right: $(s^1 \circ s^2 \circ \cdots \circ s^n)$ means $(s^1 \circ (s^2 \circ \cdots \circ s^n)\cdots)$.

We have the following rewrite system for computing σ-normal forms [1, 5] :

- **App**: $(a\,b)[s] \to (a[s]\,b[s])$
- **VarCons**: $1[a.s] \to a$
- **Id**: $a[id] \to a$
- **Abs**: $(\lambda a)[s] \to (\lambda a[1.(s \circ \uparrow)])$
- **Clos**: $(a[s])[t] \to a[s \circ t]$
- **IdL**: $id \circ s \to s$
- **ShiftCons**: $\uparrow \circ (a.s) \to s$
- **AssEnv**: $(s_1 \circ s_2) \circ s_3 \to s_1 \circ (s_2 \circ s_3)$
- **MapEnv**: $(a.s) \circ t \to a[t].(s \circ t)$
- **IdR**: $s \circ id \to s$
- **VarShift**: $1.\uparrow \to id$
- **Scons**: $1[s].(\uparrow \circ s) \to s$

A σ-normal term [explicit substitution] is a term [explicit substitution] which cannot be reduced using one of the rules above. We denote the σ-normal form of a term or explicit substitution a by $a^{\downarrow\sigma}$.

2.3 M-set Model of HOAS

We now combine the previous two sections to obtain an interesting M-set model. This M-set model and the 14 theorems in the next section were the work of Chad E. Brown (private communication).

There is an M-set model $\mathcal{M}$ with

- $\mathcal{D}_\iota$ is the set of σ-normal terms:

  $$1|1[\uparrow \circ \cdots \circ \uparrow]|(ab)|(\lambda a)$$

  where a and b are σ-normal terms.
  Action: For $a \in \mathcal{D}_\iota$ and $m \in M$,

  $$am := (a[m])^{\downarrow\sigma}$$

- $\mathcal{D}_o$ is set of all subsets of M ($X \in \mathcal{D}_o$ iff $X \subseteq M$)
  Action: For $X \in \mathcal{D}_o$ and $m \in M$

  $$Xm := \{n \in M | mn \in X\}$$

- $\mathcal{D}_{\alpha\to\beta}$ is the set of functions $f : M \times \mathcal{D}_\alpha \to \mathcal{D}_\beta$ respecting the M-set actions.
- The application operator takes $f \in \mathcal{D}_{\alpha\to\beta}$ and $x \in \mathcal{D}_\alpha$ to

  $$f@x := f(id, x)$$

Let T be the set of σ-normal terms and M be the set of σ-normal substitutions. We create an M-set model by choosing $\mathcal{D}_\iota := T$. The following properties follow easily from the definitions above.

`axapp` $((ab)[s])^{\downarrow\sigma} = ((a[s])^{\downarrow\sigma}(b[s])^{\downarrow\sigma})$ for $a, b \in T$ and $s \in M$.

`axvarcons` $(1[a.s])^{\downarrow\sigma} = a$ for $a \in T$ and $s \in M$.

`axvarid` $(a[id])^{\downarrow\sigma} = a$ for $a \in T$.

`axclos` $(a[s][t])^{\downarrow\sigma} = (a[s \circ t])^{\downarrow\sigma}$ for $a \in T$ and $s, t \in M$.

`axidl` $(id \circ s)^{\downarrow\sigma} = s$ for $s \in M$.

`axshiftcons` $(\uparrow \circ (a.s))^{\downarrow\sigma} = s$ for $a \in T$ and $s \in M$.

`axassoc` $((s_1 \circ s_2) \circ s_3)^{\downarrow\sigma} = (s_1 \circ (s_2 \circ s_3))^{\downarrow\sigma}$ for $s_1, s_2, s_3 \in M$.

`axmap` $((a.s) \circ t)^{\downarrow\sigma} = (a[t].(s \circ t))^{\downarrow\sigma}$ for $a \in T$ and $s, t \in M$.

`axidr` $(s \circ id)^{\downarrow\sigma} = s$ for $s \in M$.

`axvarshift` $(1.\uparrow)^{\downarrow\sigma} = id$.

`axscons` $(1[s].(\uparrow \circ s))^{\downarrow\sigma} = s$ for $s \in M$.

Let *Var* be the subset of T consisting of 1 and all the terms of the form $1[\uparrow \circ \cdots \circ \uparrow]$. The following properties clearly hold:

`ulamvar1` $1 \in Var$

`ulamvarsh` If $x \in Var$, then $(x[\uparrow])^{\downarrow\sigma} \in Var$.

`ulamvarind` Let Φ be a property. Suppose 1 satisfies Φ. Suppose for all $x \in Var$, if x satisfies Φ, then $(x[\uparrow])^{\downarrow\sigma}$ satisfies Φ. Then for all $x \in Var$, x satisfies Φ.

Since terms and substitutions are freely generated and σ-reduction is confluent, we have the following properties:

`apinj1` For $a, b, c, d \in T$, if $(a\,c)^{\downarrow\sigma} = (b\,d)^{\downarrow\sigma}$, then $a = b$.

`apinj2` For $a, b, c, d \in T$, if $(a\,c)^{\downarrow\sigma} = (b\,d)^{\downarrow\sigma}$, then $c = d$.

`laminj` For $a, b \in T$, if $(\lambda a)^{\downarrow\sigma} = (\lambda b)^{\downarrow\sigma}$, then $a = b$.

`shinj` For $a, b \in T$, if $(a[\uparrow])^{\downarrow\sigma} = (b[\uparrow])^{\downarrow\sigma}$, then $a = b$.

`lamnotap` For $a, b, c \in T$, $(\lambda a)^{\downarrow\sigma} \neq (b\,c)^{\downarrow\sigma}$.

`apnotvar` For $a, b \in T$, $(a\,b)^{\downarrow\sigma} \notin Var$.

`lamnotvar` For $a \in T$, $(\lambda a)^{\downarrow\sigma} \notin Var$.

`induction` Let Φ be a property. Suppose for all $x \in Var$, x satisfies Φ. Suppose for all $a, b \in T$, if a and b satisfy Φ, then $(a\,b)^{\downarrow\sigma}$ satisfies Φ. Suppose for all $a \in T$, if a satisfies Φ, then $(\lambda a)^{\downarrow\sigma}$ satisfies Φ. Then for all $a \in T$, a satisfies Φ.

The model $\mathcal{M}$ should provide an interpretation of a theory of untyped λ-calculus using *higher-order abstract syntax*(HOAS) [6].

To interpret higher-order abstract syntax in the M-set model. We define the following functions:

- `hoasap`:
$$\texttt{hoasap}(m,a)(n,b) := (a[n]\,b)^{\downarrow\sigma}$$
where $a,b \in T$ and $m,n \in M$.

- `hoaslam`:
$$\texttt{hoaslam}(m,f) := (\lambda(f(\uparrow,1)))^{\downarrow\sigma}$$
where $f : M \times T \to T$ is such that $f(mn,an) = f(m,a)n$ (as defined by the monoid operation and M-set action).

- $\texttt{hoasvar} : M \to T \to \mathcal{P}(M)$:
$$n \in \texttt{hoasvar}(m,a) \text{ iff } ((a[n])^{\downarrow\sigma} \in \texttt{Var})$$

2.4 The 14 Problems

In this project we experimented with using the higher-order theorem prover LEO-II to verify properties of this M-set model. In particular, we wanted to prove 14 theorems using LEO. We give the 14 theorems informally below.

Theorem 1 (`pushprop`) Let Φ be a property, $a \in T$ and $m \in M$. Assume for all $x \in Var$, $(x[m])^{\downarrow\sigma}$ satisfies Φ. Assume a satisfies Φ. Then $(x[a.m])^{\downarrow\sigma}$ satisfies Φ for all $x \in Var$.

Proof: Use `ulamvarind`, `axvarcons`, `axclos`, `axshiftcons`. Let Φ' be the following property: x satisfies Φ' iff $(x[a.m])^{\downarrow\sigma}$ satisfies Φ. We apply `ulamvarind` with the property Φ'. It remains to prove the two cases:

- We prove 1 satisfies Φ'. By `axvarcons`, $(1[a.m])^{\downarrow\sigma} = a$. Since a satisfies Φ, 1 satisfies Φ', as desired.

- Assume $x \in Var$ satisfies Φ'. That is, $(x[a.m])^{\downarrow\sigma}$ satisfies Φ. We prove $(x[\uparrow])^{\downarrow\sigma}$ satisfies Φ'. That is, we prove $(x[\uparrow][a.m])^{\downarrow\sigma}$ satisfies Φ. By `axclos` and `axshiftcons`,

  $$(x[\uparrow][a.m])^{\downarrow\sigma} = (x[\uparrow \circ (a.m)])^{\downarrow\sigma} = (x[m])^{\downarrow\sigma}.$$

  We are done since $(x[m])^{\downarrow\sigma}$ satisfies Φ.

□

Theorem 2 (`induction2lem`) Let Φ be a property such that the following hold:

1. For all $a, b \in T$, if a and b satisfy Φ, then $(a\,b)^{\downarrow\sigma}$ satisfies Φ.
2. For all $a \in T$, if $(a[b.id])^{\downarrow\sigma}$ satisfies Φ whenever $b \in T$ satisfies Φ, then $(\lambda a)^{\downarrow\sigma}$ satisfies Φ.

Let $a \in T$ and $m \in M$ be such that $(x[m])^{\downarrow\sigma}$ satisfies Φ for all $x \in Var$. Then $(a[m])^{\downarrow\sigma}$ satisfies Φ.

Proof: Induction over σ - normal terms a.

- Case 1 : $a \in$ Var
  We prove $(a[m])^{\downarrow\sigma}$ satisfies Φ. Since $(x[m])^{\downarrow\sigma}$ satisfies Φ for all $x \in Var$, $(a[m])^{\downarrow\sigma}$ satisfies Φ.
- Case 2 : $a = bc$
  We prove $((bc)[m])^{\downarrow\sigma}$ satisfies Φ.
  $((bc)[m])^{\downarrow\sigma} = ((b[m])^{\downarrow\sigma}(c[m])^{\downarrow\sigma})$ (`axapp`)
  That is, we prove $((b[m])^{\downarrow\sigma}(c[m])^{\downarrow\sigma})$ satisfies Φ. This follows from condition 1, if we prove $(b[m])^{\downarrow\sigma}$and $(c[m])^{\downarrow\sigma}$ satisfy Φ. By induction hypothesis, $(b[m])^{\downarrow\sigma}$and $(c[m])^{\downarrow\sigma}$ satisfy Φ. We are done.
- Case 3 : $a = \lambda b$
  Induction hypothesis: If $b \in T$ and $m \in M$ are such that $(x[m])^{\downarrow\sigma}$ satisfies Φ for all $x \in Var$, then $(b[m])^{\downarrow\sigma}$ satisfies Φ.
  We prove $((\lambda b)[m])^{\downarrow\sigma}$ satisfies Φ.
  $((\lambda b)[m])^{\downarrow\sigma} = (\lambda b[1.(m \circ \uparrow)])^{\downarrow\sigma}$ (`axabs`)
  That is, we prove $(\lambda b[1.(m \circ \uparrow)])^{\downarrow\sigma}$ satisfies Φ. We will use condition 2. Assume $c \in$ T satisfies Φ, we prove $((b[1.(m \circ \uparrow)])[c.id])^{\downarrow\sigma}$ satisfies Φ.

We verify this fact by computing

$$\begin{array}{lcll}((b[1.(m\circ\uparrow)])[c.id])^{\downarrow\sigma} & = & (b[(1.(m\circ\uparrow))\circ(c.id)])^{\downarrow\sigma} & (\texttt{axclos})\\ & = & (b[1[c.id].((m\circ\uparrow)\circ(c.id))])^{\downarrow\sigma} & (\texttt{axmap})\\ & = & (b[c.((m\circ\uparrow)\circ(c.id))])^{\downarrow\sigma} & (\texttt{axvarcons})\\ & = & (b[c.(m\circ(\uparrow\circ((c.id))])^{\downarrow\sigma} & (\texttt{axassoc})\\ & = & (b[c.(m\circ id)])^{\downarrow\sigma} & (\texttt{axshiftcons})\\ & = & (b[c.m])^{\downarrow\sigma} & (\texttt{axidr})\end{array}$$

By the induction hypothesis, it is enough to show, $(x[c.m])^{\downarrow\sigma}$ satisfies Φ. By `pushprop` we know assume for all $x \in Var$, $(x[m])^{\downarrow\sigma}$ satisfies Φ. Assume c satisfies Φ. Then $(x[c.m])^{\downarrow\sigma}$ satisfies Φ for all $x \in Var$. By assumption we know c satisfies Φ, so $(x[c.m])^{\downarrow\sigma}$ satisfies Φ, we are done.

□

Theorem 3 (`induction2`) Let Φ be a property such that the following hold:

1. For all $x \in Var$, x satisfies Φ.
2. For all $a, b \in T$, if a and b satisfy Φ, then $(a\,b)^{\downarrow\sigma}$ satisfies Φ.
3. For all $a \in T$, if $(a[b.id])^{\downarrow\sigma}$ satisfies Φ whenever $b \in T$ satisfies Φ, then $(\lambda a)^{\downarrow\sigma}$ satisfies Φ.

Then for all $a \in T$, a satisfies Φ.

Proof: By `induction2lem` and `axvarid` we know $(a[id])^{\downarrow\sigma}$ satisfies Φ and $(a[id])^{\downarrow\sigma} = a$, then a satisfies Φ □

We would like to verify that M is a monoid and T is an M-set. By a monoid, we mean a triple $(X, \cdot, e)$ where X is a set, $\cdot$ is an associative binary operation on X, and e is a two sided identity for $\cdot$. To be more precise, we would like to prove the following:

Theorem 4 (`substmonoid`) $(M, \cdot, id)$ is a monoid where $s \cdot t$ is $(s \circ t)^{\downarrow\sigma}$.

Proof:

- $(s \circ (t \circ m))^{\downarrow\sigma} = ((s \circ t) \circ m)^{\downarrow\sigma}$ (axassoc)
- $(id \circ s)^{\downarrow\sigma} = \mathrm{s}$ (axidl)
  $(s \circ id)^{\downarrow\sigma} = \mathrm{s}$ (axidr)

□

Given that $(M, \cdot, id)$ is a monoid, we would like to show T is an M-set. More precisely, we have:

Theorem 5 (termmset) $(T, *)$ is an $(M, \cdot, id)$-set where $a * s$ is $(a[s])^{\downarrow\sigma}$ for $a \in T$ and $s \in M$.

Proof:

- $(a[id])^{\downarrow\sigma} = a$ (axvarid)
- $(a[s][t])^{\downarrow\sigma} = (a[s \circ t])^{\downarrow\sigma}$ (axclos)

□

From now on we will omit the binary operators $\cdot$ and $*$. That is, we will write mn for $m \cdot n$ (i.e., $(m \circ n)^{\downarrow\sigma}$) when $m, n \in M$ and an for $a * n$ (i.e., $(a[n])^{\downarrow\sigma}$) when $a \in T$ and $n \in M$.

To ensure hoasap and hoaslam are well defined, we must ensure that the given values are in T. In both cases this fact is trivial, but we include these as theorems to ensure that the formal versions of the definitions in LEO make sense.

Theorem 6 (hoasap) For $m, n \in M$ and $a, b \in T$, we have $\mathtt{hoasap}(m, a)(n, b) \in T$.

Proof: Trivial □

Theorem 7 (hoaslam) Let $f : M \times T \to T$ be a function such that

$$f(m, a)n = f(mn, an)$$

for all $a \in T$ and $m, n \in M$. Then $\mathtt{hoaslam}(id, f) \in T$.

Proof: Trivial □

To verify soundness of "axioms" of a theory of higher-order abstract syntax, we need the following:

Theorem 8 (hoasapinj1) For $a,b,c,d \in T$, if

$$\texttt{hoasap}(id,a)(id,c) = \texttt{hoasap}(id,b)(id,d),\ \textit{then}\ a = b.$$

Proof:

$$\begin{array}{rcl}(ac)^{\downarrow\sigma} & = & (a[id]c)^{\downarrow\sigma}\ (\texttt{axvarid}) \\ & = & \texttt{hoasap}(id,a)(id,c)\ \text{(definition)} \\ & = & \texttt{hoasap}(id,b)(id,d)\ \text{(condition)} \\ & = & (b[id]d)^{\downarrow\sigma}\ \text{(definition)} \\ & = & (bd)^{\downarrow\sigma}\ (\texttt{axvarid})\end{array}$$

Then $(ac)^{\downarrow\sigma} = (bd)^{\downarrow\sigma} \overset{apinj1}{\rightarrow} a = b$

Theorem 9 (hoasapinj2) For $a,b,c,d \in T$, if

$$\texttt{hoasap}(id,a)(id,c) = \texttt{hoasap}(id,b)(id,d),\ \textit{then}\ c = d.$$

Proof: Same as theorem 8 (except with apinj2). □

Theorem 10 (hoaslaminj) Let $f,g : M \times T \rightarrow T$ be functions such that

$$f(m,a)n = f(mn,an)$$

and

$$g(m,a)n = g(mn,an)$$

for all $a \in T$ and $m,n \in M$. If $\texttt{hoaslam}(id,f) = \texttt{hoaslam}(id,g)$, then $f = g$.

Proof:

$$\texttt{hoaslam}(id,f) := (\lambda(f(\uparrow,1)))^{\downarrow\sigma}$$

$$\texttt{hoaslam}(id,g) := (\lambda(g(\uparrow,1)))^{\downarrow\sigma}$$

Since $\texttt{hoaslam}(id,f) = \texttt{hoaslam}(id,g)$, $(\lambda(f(\uparrow,1)))^{\downarrow\sigma} = (\lambda(g(\uparrow,1)))^{\downarrow\sigma}$
By laminj we know $f(\uparrow,1) = g(\uparrow,1)$.
We prove $f(m,a) = g(m,a)$ for $m \in M$ and $a \in T$. Let n be $a.m$, we compute:
$\uparrow \circ [a.m] = m$(axshiftcons)
$(1[a.m])^{\downarrow\sigma} = a$ (axvarcons)
$f(m,a) = f(\uparrow n, 1n) = f(\uparrow,1)n = g(\uparrow,1)n = g(\uparrow n, 1n) = g(m,a)$

□

Theorem 11 (hoaslamnotap) Let $f : M \times T \to T$ be a function such that

$$f(m,a)n = f(mn,an)$$

for all $a \in T$ and $m,n \in M$. For $a,b \in T$, $\texttt{hoaslam}(id,f) \neq \texttt{hoasap}(id,a)(id,b)$.

Proof: By the definition of hoaslam and hoasap

- $\texttt{hoaslam}(id,f) = (\lambda(f(\uparrow,1)))^{\downarrow\sigma}$
- $\texttt{hoasap}(id,a)(id,b) = (a[id]b)^{\downarrow\sigma} = (ab)^{\downarrow\sigma}$

Use lamnotap, we know $(\lambda(f(\uparrow,1)))^{\downarrow\sigma} \neq (ab)^{\downarrow\sigma}$. □

Theorem 12 (hoaslamnotvar) Let $f : M \times T \to T$ be a function such that

$$f(m,a)n = f(mn,an)$$

for all $a \in T$ and $m,n \in M$. Then $id \notin \texttt{hoasvar}(id,\texttt{hoaslam}(id,f))$

Proof: Use the definition of hoaslam and hoasvar.

$$\begin{array}{lcll} id \notin \texttt{hoasvar}(id,\texttt{hoaslam}(id,f)) & \Leftrightarrow & id \notin \texttt{hoasvar}(id,(\lambda(f(\uparrow,1)))^{\downarrow\sigma}) & (\texttt{hoaslam}) \\ & \Leftrightarrow & (((\lambda(f(\uparrow,1)))[id])^{\downarrow\sigma} \notin Var & (\texttt{hoasvar}) \\ & \Leftrightarrow & (\lambda(f(\uparrow,1)))^{\downarrow\sigma} \notin Var & (\texttt{axvarid}) \end{array}$$

By lamnotvar we know $(\lambda(f(\uparrow,1)))^{\downarrow\sigma} \notin Var$.

□

Theorem 13 (hoasapnotvar) For $a,b \in T$, $id \notin \texttt{hoasvar}(id,\texttt{hoasap}(id,a),(id,b))$

Proof: Use the definition of hoasvar and hoasap.

$$\begin{array}{lcll} id \notin \texttt{hoasvar}(id,\texttt{hoasap}(id,a),(id,b)) & \Leftrightarrow & id \notin \texttt{hoasvar}(id,(a[id]b)^{\downarrow\sigma}) & (\text{hoasap}) \\ & \Leftrightarrow & id \notin \texttt{hoasvar}(id,(ab)^{\downarrow\sigma}) & (\text{axvarid}) \\ & \Leftrightarrow & ((ab)[id])^{\downarrow\sigma} \notin Var & (\text{hoasvar}) \\ & \Leftrightarrow & (ab)^{\downarrow\sigma} \notin Var & (\text{axvarid}) \end{array}$$

By apnotvar we know $(ab)^{\downarrow\sigma} \notin Var$.

□

Theorem 14 (hoasinduction) Let $\Psi : M \times T \to \mathcal{P}(M)$ be a function such that

$$kn \in \Psi(m,a) \text{ iff } n \in \Psi(mk,ak)$$

for all $a \in T$ and $m,n,k \in M$. Suppose we have the following:

1. For all $x \in T$, if $id \in \mathtt{hoasvar}(id,x)$, then $id \in \Psi(id,x)$.
2. For all $a,b \in T$, if $id \in \Psi(id,a)$ and $id \in \Psi(id,b)$, then $id \in \Psi(id,\mathtt{hoasap}(id,a)(id,b))$
3. For all $f : M \times T \to T$ such that $f(m,a)n = f(mn,an)$ for all $a \in T$ and $m,n \in M$, if $id \in \Psi(id,a)$ implies $id \in \Psi(id,f(id,a))$ for all $a \in T$, then $id \in \Psi(id,\mathtt{hoaslam}(id,f))$.

Then for all $a \in T$, $id \in \Psi(id,a)$.

Proof: Use induction2. Define Φ as follows: for all $a \in T$, a satisfies Φ, if and only if $id \in \Psi(id,a)$.

- Case 1 : $a \in Var$
  We prove $id \in \Psi(id,a)$. That is, we prove $id \in$hoasvar(id,a), since condition 1.
  By the definition of hoasvar, we know $id \in$ hoasvar(id,a) iff $(a[id])^{\downarrow\sigma} \in$ Var. Since $(a[id])^{\downarrow\sigma} = a \in$ Var, we are done.
- Case 2 : $a = bc$
  We prove $id \in \Psi(id,bc)$. Induction hypothesis: b and c satisfy Φ, that is, $id \in \Psi(id,b)$ and $id \in \Psi(id,c)$.
  $(bc)^{\downarrow\sigma} = (b[id]c)^{\downarrow\sigma} =$ hoasap$((id,b),(id,c))$.
  We prove $id \in \Psi($$id$,hoasap$((id,b),(id,c))$. By condition 2 and the inductive hypothesis, we are done.
- Case 3 : $a = \lambda b$
  We prove $id \in \Psi(id,\lambda b)$. Induction hypothesis: $(b[c.id])^{\downarrow\sigma}$ satisfies Φ whenever $c \in T$ satisfies Φ, that is, $id \in \Psi(id,(b[c.id])^{\downarrow\sigma})$ whenever $id \in$

$\Psi(id,c)$.
By definition we know

$$\texttt{hoaslam}(id,f) := (\lambda(f(\uparrow,1)))^{\downarrow\sigma}$$

By condition 3 we know for all $f : M \times T \to T$ such that

$$f(m,a)n = f(mn,an).$$

For all $a \in T$ and $m,n \in M$, if $id \in \Psi(id,a)$ implies $id \in \Psi(id,f(id,a))$ for all $a \in T$, then $id \in \Psi(id,\texttt{hoaslam}(id,f))$. So we can choose an f to prove it. Define

$$f(m,a) = b[a.m]$$

We check $b = f(\uparrow,1)$. By `axvarshift` and `axvarid` we know

$$f(\uparrow,1) = b[1.\ \uparrow] = b[id] = b$$

Now we check $f(m,a)n = f(mn,an)$.

$$f(m,a)n = (b[a.m])n = b[an.mn] = f(mn,an)$$

Finally we check if $id \in \Psi(id,c)$ implies $id \in \Psi(id,f(id,c))$ for all $c \in T$, that is if $id \in \Psi(id,c)$ implies $id \in \Psi(id,(b[c.id])^{\downarrow\sigma})$ for all $c \in T$. By induction hypothesis we are done.

□

Chapter 3

Representation in Higher Order Logic

At several places there are different options for how to formally represent mathematical objects. For instance, should the set of terms be a base type, a predicate on a base type, or an element of a base type? What affect does the representation have on the behavior of the prover?

In this project we developed two representations.

3.1 Representation I

Let T be the set of σ-normal terms and M be the set of σ-normal substitutions. In thf syntax [2] we can represent these sets as constants of type ι (using a membership relation to represent set membership)

`in` is a constant of type $\iota \to \iota \to o$.

`term` is a constant of type ι.

`subst` is a constant of type ι.

For each constructor above, we can define a corresponding operator on T and M where we σ- normalize after applying the operator:

- one := 1
- (ap a b) := $(ab)^{\downarrow\sigma}$

- (lam a) := $\lambda a^{\downarrow\sigma}$
- (sub a m) := $a[m]^{\downarrow\sigma}$ where $a \in T$ and $m \in M$
- id := id
- sh := $\uparrow$
- (push a m) := $(a.m)^{\downarrow\sigma}$
- (comp m n) := $(m \circ n)^{\downarrow\sigma}$

In thf [2] syntax we can declare these constants, along with the information about the constants, as follows:

`one` is a constant of type ι

`one_p` is an abbreviation defined by

$$\mathtt{in}\,\mathtt{one}\,\mathtt{term}$$

`ap` is a constant of type $\iota \to \iota \to \iota$

`ap_p` is an abbreviation defined by

$$\forall A_\iota.\mathtt{in}\,A\,\mathtt{term} \Rightarrow \forall B_\iota.\mathtt{in}\,B\,\mathtt{term} \Rightarrow \mathtt{in}\,(\mathtt{ap}\,A\,B)\,\mathtt{term}$$

`lam` is a constant of type $\iota \to \iota$

`lam_p` is an abbreviation defined by

$$\forall A_\iota.\mathtt{in}\,A\,\mathtt{term} \Rightarrow \mathtt{in}\,(\mathtt{lam}\,A)\,\mathtt{term}$$

`sub` is a constant of type $\iota \to \iota \to \iota$

`sub_p` is an abbreviation defined by

$$\forall A_\iota.\mathtt{in}\,A\,\mathtt{term} \Rightarrow \forall M_\iota.\mathtt{in}\,M\,\mathtt{subst} \Rightarrow \mathtt{in}\,(\mathtt{sub}\,A\,M)\,\mathtt{term}$$

`id` is a constant of type ι

`id_p` is an abbreviation defined by

$$\mathtt{in}\,\mathtt{id}\,\mathtt{subst}$$

sh is a constant of type ι

sh_p is an abbreviation defined by

$$\mathtt{in}\,\mathtt{sh}\,\mathtt{subst}$$

push is a constant of type $\iota \to \iota \to \iota$

push_p is an abbreviation defined by

$$\forall A_\iota.\mathtt{in}\,A\,\mathtt{term} \Rightarrow \forall M_\iota.\mathtt{in}\,M\,\mathtt{subst} \Rightarrow \mathtt{in}\,(\mathtt{push}\,A\,M)\,\mathtt{subst}$$

comp is a constant of type $\iota \to \iota \to \iota$

comp_p is an abbreviation defined by

$$\forall M_\iota.\mathtt{in}\,M\,\mathtt{subst} \Rightarrow \forall N_\iota.\mathtt{in}\,N\,\mathtt{subst} \Rightarrow \mathtt{in}\,(\mathtt{comp}\,M\,N)\,\mathtt{subst}$$

Several properties clearly hold. We will informally take these as axioms. Below we give them as definitions. Whenever we want to prove a theorem we must take each one as an assumption explicitly.

axapp

$$\forall A_\iota.\mathtt{in}\,A\,\mathtt{term} \Rightarrow \forall B_\iota.\mathtt{in}\,B\,\mathtt{term} \Rightarrow \forall M_\iota.\mathtt{in}\,M\,\mathtt{subst} \Rightarrow \mathtt{sub}\,(\mathtt{ap}\,A\,B)\,M = \mathtt{ap}\,(\mathtt{sub}\,A\,M)\,(\mathtt{sub}\,B\,M)$$

axvarcons

$$\forall A_\iota.\mathtt{in}\,A\,\mathtt{term} \Rightarrow \forall M_\iota.\mathtt{in}\,M\,\mathtt{subst} \Rightarrow \mathtt{sub}\,\mathtt{one}\,(\mathtt{push}\,A\,M) = A$$

axvarid

$$\forall A_\iota.\mathtt{in}\,A\,\mathtt{term} \Rightarrow \mathtt{sub}\,A\,\mathtt{id} = A$$

axabs

$$\forall A_\iota.\mathtt{in}\,A\,\mathtt{term} \Rightarrow \forall M_\iota.\mathtt{in}\,M\,\mathtt{subst} \Rightarrow \mathtt{sub}\,(\mathtt{lam}\,A)\,M = \mathtt{lam}\,(\mathtt{sub}\,A\,(\mathtt{push}\,\mathtt{one}\,(\mathtt{comp}\,M\,\mathtt{sh})))$$

axclos

$$\forall A_\iota.\mathtt{in}\,A\,\mathtt{term} \Rightarrow \forall M_\iota.\mathtt{in}\,M\,\mathtt{subst} \Rightarrow \forall N_\iota.\mathtt{in}\,N\,\mathtt{subst} \Rightarrow \mathtt{sub}\,(\mathtt{sub}\,A\,M)\,N = \mathtt{sub}\,A\,(\mathtt{comp}\,M\,N)$$

axshiftcons

$$\forall A_{\mathfrak{l}}.\mathtt{in}\,A\,\mathtt{term}\Rightarrow\forall M_{\mathfrak{l}}.\mathtt{in}\,M\,\mathtt{subst}\Rightarrow\mathtt{comp}\,\mathtt{sh}\,(\mathtt{push}\,A\,M)=M$$

axassoc

$$\forall M_{\mathfrak{l}}.\mathtt{in}\,M\,\mathtt{subst}\Rightarrow\forall N_{\mathfrak{l}}.\mathtt{in}\,N\,\mathtt{subst}\Rightarrow\forall K_{\mathfrak{l}}.\mathtt{in}\,K\,\mathtt{subst}\Rightarrow$$
$$\mathtt{comp}\,(\mathtt{comp}\,M\,N)\,K=\mathtt{comp}\,M\,(\mathtt{comp}\,N\,K)$$

axmap

$$\forall A_{\mathfrak{l}}.\mathtt{in}\,A\,\mathtt{term}\Rightarrow\forall M_{\mathfrak{l}}.\mathtt{in}\,M\,\mathtt{subst}\Rightarrow\forall N_{\mathfrak{l}}.\mathtt{in}\,N\,\mathtt{subst}\Rightarrow$$
$$\mathtt{comp}\,(\mathtt{push}\,A\,M)\,N=\mathtt{push}\,(\mathtt{sub}\,A\,N)\,(\mathtt{comp}\,M\,N)$$

axidr

$$\forall M_{\mathfrak{l}}.\mathtt{in}\,M\,\mathtt{subst}\Rightarrow\mathtt{comp}\,M\,\mathtt{id}=M$$

axvarshift

$$\mathtt{push}\,\mathtt{one}\,\mathtt{sh}=\mathtt{id}$$

axscons

$$\forall M_{\mathfrak{l}}.\mathtt{in}\,M\,\mathtt{subst}\Rightarrow\mathtt{push}\,(\mathtt{sub}\,\mathtt{one}\,M)\,(\mathtt{comp}\,\mathtt{sh}\,M)=M$$

3.2 Representation II

We declared the base type `term` and `subst`.

We can also define corresponding operators on T and M:

- `one` is a constant of type `term`.
- `ap` is a constant of type `term` $\rightarrow$ `term` $\rightarrow$ `term`.
- `lam` is a constant of type `term` $\rightarrow$ `term`.
- `sub` is a constant of type `term` $\rightarrow$ `subst` $\rightarrow$ `term`.
- `id` is a constant of type `subst`.
- `sh` is a constant of type `subst`.
- `push` is a constant of type `term` $\rightarrow$ `subst` $\rightarrow$ `subst`.

- comp is a constant of type $\mathtt{subst} \to \mathtt{subst} \to \mathtt{subst}$.

Axioms became:

axapp

$$\forall A_{\mathtt{term}} B_{\mathtt{term}} M_{\mathtt{subst}}.\mathtt{sub}\,(\mathtt{ap}\,A\,B)\,M = \mathtt{ap}\,(\mathtt{sub}\,A\,M)\,(\mathtt{sub}\,B\,M)$$

axvarcons

$$\forall A_{\mathtt{term}} M_{\mathtt{subst}}.\mathtt{sub}\,\mathtt{one}\,(\mathtt{push}\,A\,M) = A$$

axvarid

$$\forall A_{\mathtt{term}}.\mathtt{sub}\,A\,\mathtt{id} = A$$

axabs

$$\forall A_{\mathtt{term}} M_{\mathtt{subst}}.\mathtt{sub}\,(\mathtt{lam}\,A)\,M = \mathtt{lam}\,(\mathtt{sub}\,A\,(\mathtt{push}\,\mathtt{one}\,(\mathtt{comp}\,M\,\mathtt{sh})))$$

axclos

$$\forall A_{\mathtt{term}} M_{\mathtt{subst}} N_{\mathtt{subst}}.\mathtt{sub}\,(\mathtt{sub}\,A\,M)\,N = \mathtt{sub}\,A\,(\mathtt{comp}\,M\,N)$$

axidl

$$\forall M_{\mathtt{subst}}.\mathtt{comp}\,\mathtt{id}\,M = M$$

axshiftcons

$$\forall A_{\mathtt{term}} M_{\mathtt{subst}}.\mathtt{comp}\,\mathtt{sh}\,(\mathtt{push}\,A\,M) = M$$

axassoc

$$\forall M_{\mathtt{subst}} N_{\mathtt{subst}} K_{\mathtt{subst}}.\mathtt{comp}\,(\mathtt{comp}\,M\,N)\,K = \mathtt{comp}\,M\,(\mathtt{comp}\,N\,K)$$

axmap

$$\forall A_{\mathtt{term}} M_{\mathtt{subst}} N_{\mathtt{subst}}.\mathtt{comp}\,(\mathtt{push}\,A\,M)\,N = \mathtt{push}\,(\mathtt{sub}\,A\,N)\,(\mathtt{comp}\,M\,N)$$

axidr

$$\forall M_{\mathtt{subst}}.\mathtt{comp}\,M\,\mathtt{id} = M$$

axvarshift

$$\mathtt{push}\,\mathtt{one}\,\mathtt{sh} = \mathtt{id}$$

axscons

$$\forall M_{\mathtt{subst}}.\mathtt{push}\,(\mathtt{sub}\,\mathtt{one}\,M)\,(\mathtt{comp}\,\mathtt{sh}\,M) = M$$

3.3 Hoasap and Hoaslam

In chapter 2 we give 14 theorems. Two representations are developed. With the Representation I all the 14 theorems are encoded and with the Representation II 12 theorems are encoded. The 2 theorems we lost (hoasap and hoaslam) are simple type checking.

Theorem 6 (hoasap) For $m,n \in M$ and $a,b \in T$, we have $\mathtt{hoasap}(m,a)(n,b) \in T$.

We encoded this theorem in LEO with Representation I as follow:

- hoasap
$$\lambda M_\iota A_\iota N_\iota B_\iota.\mathtt{ap}\,(\mathtt{sub}\,A\,N)\,B$$
- hoasap_p
$$\forall M_\iota.\mathtt{in}\,M\,\mathtt{subst}\Rightarrow\forall A_\iota.\mathtt{in}\,A\,\mathtt{term}\Rightarrow\forall N_\iota.\mathtt{in}\,N\,\mathtt{subst}\Rightarrow\forall B_\iota.\mathtt{in}\,B\,\mathtt{term}\Rightarrow$$
$$\mathtt{in}\,(\mathtt{hoasap}\,M\,A\,N\,B)\,\mathtt{term}$$

We encoded the definition of hoasap in LEO with Representation II as follow:

$$\lambda M_{\mathtt{subst}} A_{\mathtt{term}} N_{\mathtt{subst}} B_{\mathtt{term}}.\mathtt{ap}\,(\mathtt{sub}\,A\,N)\,B$$

Theorem 7 (hoaslam) Let $f : M \times T \to T$ be a function such that

$$f(m,a)n = f(mn,an)$$

for all $a \in T$ and $m,n \in M$. Then $\mathtt{hoaslam}(id,f) \in T$.

We encoded this theorem in LEO with Representation I as follow:

- hoaslam
$$\lambda M_\iota F_{\iota\to\iota\to\iota}.\mathtt{lam}\,(F\,\mathtt{sh}\,\mathtt{one})$$
- hoaslam_p
$$\forall M_\iota.\mathtt{in}\,M\,\mathtt{subst}\Rightarrow\forall F_{\iota\to\iota\to\iota}.(\forall N_\iota.\mathtt{in}\,N\,\mathtt{subst}\Rightarrow\forall A_\iota.\mathtt{in}\,A\,\mathtt{term}\Rightarrow$$
$$\mathtt{in}\,(F\,N\,A)\,\mathtt{term})\Rightarrow\mathtt{in}\,(\mathtt{hoaslam}\,M\,\lambda N_\iota A_\iota.F\,N\,A)\,\mathtt{term}$$

We encoded the definition of hoaslam in LEO with Representation II as follow:

$$\lambda M_{\mathtt{subst}} F_{\mathtt{subst}\to\mathtt{term}\to\mathtt{term}}.\mathtt{lam}\,(F\,\mathtt{sh}\,\mathtt{one})$$

There is no formula in Representation II corresponding to hoasap_p from Representation I and no formula in Representation II corresponding to hoaslam_p from Representation I , therefore we lost the both theorems.

3.4 Global and Local Theorems

When we want to prove a theorem C, we will actually prove a theorem of the form $Axioms \Rightarrow C$. We will also be interested in localizing the theorems by finding a subset A of the axioms such that $A \Rightarrow C$ can be proven automatically.

Several constants, definitions, axioms, and theorems have already been formulated in thf syntax [2]. Each of the theorems can either be proven assuming every fact that has already been given (a `global` theorem) or using only some of the human-selected facts that have been given (a `local` theorem). We selected the appropriate facts for a local version of each theorem in Representation I.

For example: `hoaslamnotap`

Theorem 11 (`hoaslamnotap`) Let $f : M \times T \to T$ be a function such that

$$f(m,a)n = f(mn,an)$$

for all $a \in T$ and $m,n \in M$. For $a,b \in T$, $\texttt{hoaslam}(id,f) \neq \texttt{hoasap}(id,a)(id,b)$.

The `gthm` is:

`one_p`⇒`ap_p`⇒`lam_p`⇒`sub_p`⇒`id_p`⇒`sh_p`⇒`push_p`⇒
`comp_p`⇒`axapp`⇒`axvarcons`⇒`axvarid`⇒`axabs`⇒`axclos`
⇒`axidl`⇒`axshiftcons`⇒`axassoc`⇒`axmap`⇒`axidr`⇒`axvarshift`
⇒`axscons`⇒`ulamvar1`⇒`ulamvarsh`⇒`ulamvarind`⇒`apinj1`⇒`apinj2`
⇒`laminj`⇒`shinj`⇒`lamnotap`⇒`apnotvar`⇒`lamnotvar`⇒`induction`
⇒`pushprop`⇒`induction2lem`⇒`induction2`⇒`substmonoid`
⇒`termmset`⇒`hoasap_p`⇒`hoaslam_p`⇒`hoasapinj1`⇒`hoasapinj2`
⇒`hoaslaminj`⇒`hoaslamnotap`

The `lthm` is:

`lamnotap`⇒`hoaslamnotap`

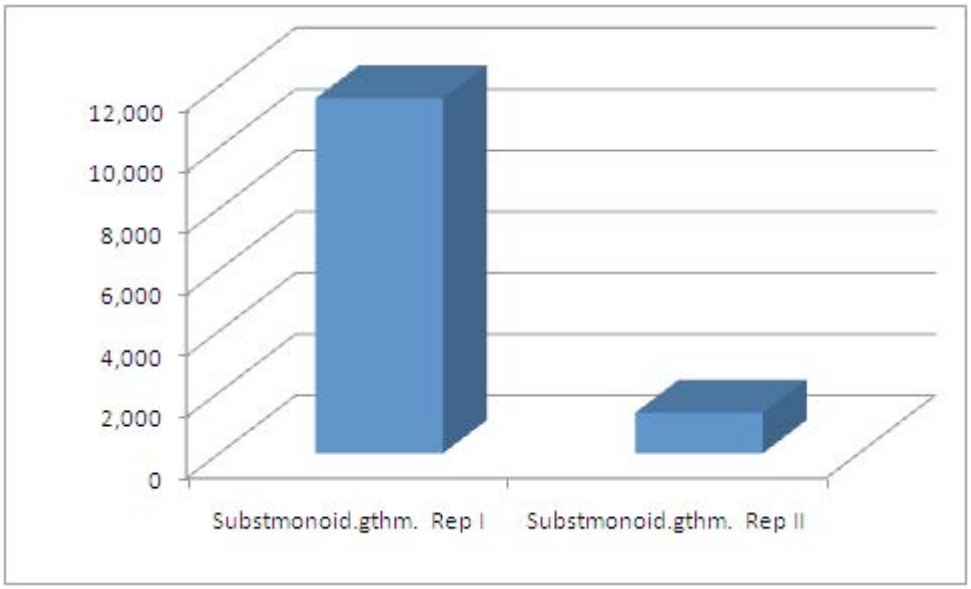
Chapter 4

LEO/Results

4.1 Basic Results

There are 14 theorems in Representation I and 12 theorems in Representation II. LEO could prove 9 theorems of 14 in Rep I and 9 theorems of 12 in Rep II. The details are in the table below.

Name	***Rep I***		***Rep II***		
	`gthm`	`lthm`	`gthm`	`lthm`	`lthm` with lemmas
Substmonoid	11.589s	5.165s	1.324s	0.521s	NA
Termmset	3.299s	0.564s	1.354s	0.505s	NA
Hoasapinj1	3.573s	0.481s	1.411s	0.515s	NA
Hoasapinj2	3.680s	0.479s	1.452s	0.509s	NA
Hoaslamnotap	6.194s	0.778s	1.622s	0.508s	NA
Hoaslamnotvar	6.495s	0.760s	1.685s	0.509s	NA
Hoasapnotvar	6.671s	0.575s	1.762s	0.503s	NA
Hoaslap	3.317s	0.437s	NA	NA	NA
Hoaslam	3.343s	0.636s	NA	NA	NA
Hoaslaminj	-	-	1.556s	0.533s	NA
Induction2	-	-	-	0.581s	NA
Pushprop	-	-	-	-	0.655s
Hoasinduction	-	-	-	-	0.807s
Induction2lem	-	-	-	-	-



LEO could prove theorems much quickly with Rep II.

Table for `hoasinduction` + lemmas (Rep II lthm)

Name	`lthm`
`hoasinduction_lem1`	1.843s
`hoasinduction_lem2`	1.877s
`hoasinduction_lem3`	0.701s
`hoasinduction_lemm3a`	0.680s
`hoasinduction_lem3b`	1.793s
`hoasinduction_lem3aa`	-
`hoasinduction_lem0`	12.359s
`hoasinduction_lem1v2`	1.873s
`hoasinduction_lem2v2`	1.903s
`hoasinduction_lem3v2`	-
`hoasinduction_lem3v2a`	-
`hoasinduction`	0.807s

4.2 Hoaslaminj

LEO could not prove the theorem `hoaslaminj` in Representation I, but could prove it in Representation II, i.e. proving the theorem is easier in Representation II.

We want our model to satisfy this axiom:

$$\forall f \forall g(((Lam f) = (Lam g)) \Rightarrow (f = g))$$

We interpret this axiom in our model as:

Theorem 10 (hoaslaminj) Let $f, g : M \times T \to T$ be functions such that

$$f(m,a)n = f(mn,an)$$

and

$$g(m,a)n = g(mn,an)$$

for all $a \in T$ and $m,n \in M$. If $\mathtt{hoaslam}(id,f) = \mathtt{hoaslam}(id,g)$, then $f = g$.

We encoded this theorem in LEO with Representation I as follows:

$$\forall F_{\iota\to\iota\to\iota}.(\forall M_\iota.\mathtt{in}M\,\mathtt{subst}\Rightarrow\forall A_\iota.\mathtt{in}A\,\mathtt{term}\Rightarrow\mathtt{in}\,(F\,M\,A)\,\mathtt{term})\Rightarrow$$
$$(\forall M_\iota.\mathtt{in}M\,\mathtt{subst}\Rightarrow\forall A_\iota.\mathtt{in}A\,\mathtt{term}\Rightarrow\forall N_\iota.\mathtt{in}N\,\mathtt{subst}\Rightarrow$$
$$\mathtt{sub}\,(F\,M\,A)\,N = F\,(\mathtt{comp}\,M\,N)\,(\mathtt{sub}\,A\,N))\Rightarrow$$
$$\forall G_{\iota\to\iota\to\iota}.(\forall M_\iota.\mathtt{in}M\,\mathtt{subst}\Rightarrow\forall A_\iota.\mathtt{in}A\,\mathtt{term}\Rightarrow\mathtt{in}\,(G\,M\,A)\,\mathtt{term})\Rightarrow$$
$$(\forall M_\iota.\mathtt{in}M\,\mathtt{subst}\Rightarrow\forall A_\iota.\mathtt{in}A\,\mathtt{term}\Rightarrow\forall N_\iota.\mathtt{in}N\,\mathtt{subst}\Rightarrow$$
$$\mathtt{sub}\,(G\,M\,A)\,N = G\,(\mathtt{comp}\,M\,N)\,(\mathtt{sub}\,A\,N))\Rightarrow$$
$$\mathtt{hoaslam}\,\mathtt{id}\,(\lambda M_\iota A_\iota.F\,M\,A) = \mathtt{hoaslam}\,\mathtt{id}\,(\lambda M_\iota A_\iota.G\,M\,A)\Rightarrow$$
$$\forall M_\iota.\mathtt{in}M\,\mathtt{subst}\Rightarrow\forall A_\iota.\mathtt{in}A\,\mathtt{term}\Rightarrow F\,M\,A = G\,M\,A$$

hoaslaminj_gthm_1 is:

one_p⇒ap_p⇒lam_p⇒sub_p⇒id_p⇒sh_p⇒push_p⇒
comp_p⇒axapp⇒axvarcons⇒axvarid⇒axabs⇒axclos⇒
axidl⇒axshiftcons⇒axassoc⇒axmap⇒axidr⇒axvarshift⇒
axscons⇒ulamvar1⇒ulamvarsh⇒ulamvarind⇒apinj1⇒
apinj2⇒laminj⇒shinj⇒lamnotap⇒apnotvar⇒lamnotvar⇒
induction⇒pushprop⇒induction2lem⇒induction2⇒
substmonoid⇒termmset⇒hoasap_p⇒hoaslam_p⇒hoasapinj1⇒
hoasapinj2⇒hoaslaminj

hoaslaminj_lthm_1 is:

$$\texttt{axvarcons} \Rightarrow \texttt{axshiftcons} \Rightarrow \texttt{laminj} \Rightarrow \texttt{hoaslaminj}$$

We encoded this theorem in LEO with Representation II as follow:

$$\forall F_{\texttt{subst}\to\texttt{term}\to\texttt{term}}.$$
$$(\forall M_{\texttt{subst}} A_{\texttt{term}} N_{\texttt{subst}}.\texttt{sub}\,(F\,M\,A)\,N = F\,(\texttt{comp}\,M\,N)\,(\texttt{sub}\,A\,N)) \Rightarrow$$
$$\forall G_{\texttt{subst}\to\texttt{term}\to\texttt{term}}.$$
$$(\forall M_{\texttt{subst}} A_{\texttt{term}} N_{\texttt{subst}}.\texttt{sub}\,(G\,M\,A)\,N = G\,(\texttt{comp}\,M\,N)\,(\texttt{sub}\,A\,N)) \Rightarrow$$
$$\texttt{hoaslam id}\,(\lambda M_{\texttt{subst}} A_{\texttt{term}}.F\,M\,A) = \texttt{hoaslam id}\,(\lambda M_{\texttt{subst}} A_{\texttt{term}}.G\,M\,A) \Rightarrow$$
$$\forall M_{\texttt{subst}} A_{\texttt{term}}.F\,M\,A = G\,M\,A$$

hoaslaminj_gthm_2 is:

$$\texttt{axapp} \Rightarrow \texttt{axvarcons} \Rightarrow \texttt{axvarid} \Rightarrow \texttt{axabs} \Rightarrow \texttt{axclos} \Rightarrow \texttt{axidl} \Rightarrow$$
$$\texttt{axshiftcons} \Rightarrow \texttt{axassoc} \Rightarrow \texttt{axmap} \Rightarrow \texttt{axidr} \Rightarrow \texttt{axvarshift} \Rightarrow$$
$$\texttt{axscons} \Rightarrow \texttt{ulamvar1} \Rightarrow \texttt{ulamvarsh} \Rightarrow \texttt{ulamvarind} \Rightarrow \texttt{apinj1} \Rightarrow \texttt{apinj2} \Rightarrow$$
$$\texttt{laminj} \Rightarrow \texttt{shinj} \Rightarrow \texttt{lamnotap} \Rightarrow \texttt{apnotvar} \Rightarrow \texttt{lamnotvar} \Rightarrow \texttt{induction}$$
$$\Rightarrow \texttt{pushprop} \Rightarrow \texttt{induction2lem} \Rightarrow \texttt{induction2} \Rightarrow \texttt{substmonoid} \Rightarrow$$
$$\texttt{termmset} \Rightarrow \texttt{hoasapinj1} \Rightarrow \texttt{hoasapinj2} \Rightarrow \texttt{hoaslaminj}$$

hoaslaminj_lthm_2 is:

$$\texttt{axvarcons} \Rightarrow \texttt{axshiftcons} \Rightarrow \texttt{laminj} \Rightarrow \texttt{hoaslaminj}$$

LEO could prove hoaslaminj_gthm_2 and hoaslaminj_lthm_2 in Representation II, because using Representation II we have fewer clauses and unification steps.

4.3 Induction2

We want to prove this axiom in our model.

Theorem 3 (induction2) Let Φ be a property such that the following hold:

1. For all $x \in Var$, x satisfies Φ.
2. For all $a, b \in T$, if a and b satisfy Φ, then $(a\,b)^{\downarrow\sigma}$ satisfies Φ.
3. For all $a \in T$, if $(a[b.id])^{\downarrow\sigma}$ satisfies Φ whenever $b \in T$ satisfies Φ, then $(\lambda a)^{\downarrow\sigma}$ satisfies Φ.

Then for all $a \in T$, a satisfies Φ.

We encoded this theorem in LEO with Representation I as follows:

$$\forall P_{\iota\to o}.(\forall A_\iota.\mathtt{in}A\,\mathtt{term}\Rightarrow\mathtt{var}A\Rightarrow PA)\Rightarrow$$
$$(\forall A_\iota.\mathtt{in}A\,\mathtt{term}\Rightarrow\forall B_\iota.\mathtt{in}B\,\mathtt{term}\Rightarrow PA\Rightarrow PB\Rightarrow P(\mathtt{ap}A\,B))\Rightarrow$$
$$(\forall A_\iota.\mathtt{in}A\,\mathtt{term}\Rightarrow(\forall B_\iota.\mathtt{in}B\,\mathtt{term}\Rightarrow PB\Rightarrow P(\mathtt{sub}A\,(\mathtt{push}B\,\mathtt{id})))\Rightarrow P(\mathtt{lam}A))\Rightarrow$$
$$\forall A_\iota.\mathtt{in}A\,\mathtt{term}\Rightarrow PA$$

`induction2_gthm_1` is:

$$\mathtt{one\_p}\Rightarrow\mathtt{ap\_p}\Rightarrow\mathtt{lam\_p}\Rightarrow\mathtt{sub\_p}\Rightarrow\mathtt{id\_p}\Rightarrow\mathtt{sh\_p}\Rightarrow\mathtt{push\_p}\Rightarrow\mathtt{comp\_p}\Rightarrow$$
$$\mathtt{axapp}\Rightarrow\mathtt{axvarcons}\Rightarrow\mathtt{axvarid}\Rightarrow\mathtt{axabs}\Rightarrow\mathtt{axclos}\Rightarrow\mathtt{axidl}\Rightarrow$$
$$\mathtt{axshiftcons}\Rightarrow\mathtt{axassoc}\Rightarrow\mathtt{axmap}\Rightarrow\mathtt{axidr}\Rightarrow\mathtt{axvarshift}\Rightarrow$$
$$\mathtt{axscons}\Rightarrow\mathtt{ulamvar1}\Rightarrow\mathtt{ulamvarsh}\Rightarrow\mathtt{ulamvarind}\Rightarrow\mathtt{apinj1}\Rightarrow$$
$$\mathtt{apinj2}\Rightarrow\mathtt{laminj}\Rightarrow\mathtt{shinj}\Rightarrow\mathtt{lamnotap}\Rightarrow\mathtt{apnotvar}\Rightarrow\mathtt{lamnotvar}\Rightarrow$$
$$\mathtt{induction}\Rightarrow\mathtt{pushprop}\Rightarrow\mathtt{induction2lem}\Rightarrow\mathtt{induction2}$$

By the informal proof in chapter 2 we determined which axioms and previous results are needed to prove `induction2` and added these to `induction2_lthm_1`.

`induction2_lthm_1` is:

$$\mathtt{axvarid}\Rightarrow\mathtt{induction2lem}\Rightarrow\mathtt{induction2}$$

We encoded this theorem in LEO with Representation II as follows:

$$\forall P_{\mathtt{term}\to o}.(\forall A_{\mathtt{term}}.\mathtt{var}A\Rightarrow PA)\Rightarrow$$
$$(\forall A_{\mathtt{term}}B_{\mathtt{term}}.PA\Rightarrow PB\Rightarrow P(\mathtt{ap}A\,B))\Rightarrow$$
$$(\forall A_{\mathtt{term}}.(\forall B_{\mathtt{term}}.PB\Rightarrow P(\mathtt{sub}A\,(\mathtt{push}B\,\mathtt{id})))\Rightarrow P(\mathtt{lam}A))\Rightarrow$$
$$\forall A_{\mathtt{term}}.PA$$

induction2_gthm_2 is:

axapp⇒axvarcons⇒axvarid⇒axabs⇒axclos⇒axidl⇒
axshiftcons⇒axassoc⇒axmap⇒axidr⇒axvarshift⇒axscons⇒
ulamvar1⇒ulamvarsh⇒ulamvarind⇒apinj1⇒apinj2⇒laminj⇒
shinj⇒lamnotap⇒apnotvar⇒lamnotvar⇒induction
⇒pushprop⇒induction2lem⇒induction2

The induction2_lthm_2 is:

axvarid⇒induction2lem⇒induction2

LEO could neither prove the induction2_gthm_1 nor induction2_lthm_1 with Representation I, but could prove induction2_lthm_2 with Representation II.

Induction2_lthm_2 is the lthm version where we told LEO only to use axvarid and induction2lem. LEO could prove only induction2_lthm_2 with the Representation II, but not induction2_gthm_2. LEO-II is an extensional higher-order theorem prover based on resolution. The resolution rule is an inference rule that produces a new clause implied by two clauses containing complementary literals. After many steps of resolution the system generated thousands of children. Here by Induction2_lthm_2 we have much fewer clauses than Induction2_gthm_2. This is the reason why LEO could prove the Induction2_lthm_2 but not the Induction2_gthm_2.

4.4 Hoasinduction

Some of the theorems required intermediate lemmas to be formulated before the theorem could be proven automatically. These lemmas also needed to be automatically proven. We evaluated LEO based on how many intermediate lemmas were required in each case.

In the theory using higher order abstract syntax we have the following induction axiom
$\forall p((\forall x(\text{Var } x \Rightarrow (px)))$
$\wedge(\forall x \forall y(px \wedge py \Rightarrow p(\text{Ap } xy)))$
$\wedge(\forall f((\forall x(px \Rightarrow p(fx))) \Rightarrow p(\text{Lam } f))) \Rightarrow (\forall x(px))))$

We interpret this induction axiom in our model.

Theorem 14 (hoasinduction) Let $\Psi : M \times T \to \mathcal{P}(M)$ be a function such that

$$kn \in \Psi(m,a) \text{ iff } n \in \Psi(mk,ak)$$

for all $a \in T$ and $m,n,k \in M$. Suppose we have the following:

1. For all $x \in T$, if $id \in \mathtt{hoasvar}(id,x)$, then $id \in \Psi(id,x)$.

2. For all $a,b \in T$, if $id \in \Psi(id,a)$ and $id \in \Psi(id,b)$, then $id \in \Psi(id,\mathtt{hoasap}(id,a)(id,b))$

3. For all $f : M \times T \to T$ such that $f(m,a)n = f(mn,an)$ for all $a \in T$ and $m,n \in M$, if $id \in \Psi(id,a)$ implies $id \in \Psi(id,f(id,a))$ for all $a \in T$, then $id \in \Psi(id,\mathtt{hoaslam}(id,f))$.

Then for all $a \in T$, $id \in \Psi(id,a)$.

We encoded this theorem in LEO with Representation I as follows:

$$\forall P_{\iota\to\iota\to\iota\to o}.(\forall M_\iota.\mathtt{in}M\,\mathtt{subst}\Rightarrow\forall A_\iota.\mathtt{in}A\,\mathtt{term}\Rightarrow\forall N_\iota.\mathtt{in}N\,\mathtt{subst}\Rightarrow$$
$$\forall K_\iota.\mathtt{in}K\,\mathtt{subst}\Rightarrow PMA\,(\mathtt{comp}KN)\Rightarrow P\,(\mathtt{comp}MK)\,(\mathtt{sub}AK)N)\Rightarrow$$
$$(\forall M_\iota.\mathtt{in}M\,\mathtt{subst}\Rightarrow\forall A_\iota.\mathtt{in}A\,\mathtt{term}\Rightarrow\forall N_\iota.\mathtt{in}N\,\mathtt{subst}\Rightarrow$$
$$\forall K_\iota.\mathtt{in}K\,\mathtt{subst}\Rightarrow P\,(\mathtt{comp}MK)\,(\mathtt{sub}AK)N\Rightarrow PMA\,(\mathtt{comp}KN))\Rightarrow$$
$$(\forall A_\iota.\mathtt{in}A\,\mathtt{term}\Rightarrow\mathtt{hoasvar}\,\mathtt{id}A\,\mathtt{id}\Rightarrow P\,\mathtt{id}A\,\mathtt{id})\Rightarrow$$
$$(\forall A_\iota.\mathtt{in}A\,\mathtt{term}\Rightarrow\forall B_\iota.\mathtt{in}B\,\mathtt{term}\Rightarrow P\,\mathtt{id}A\,\mathtt{id}\Rightarrow P\,\mathtt{id}B\,\mathtt{id}\Rightarrow$$
$$P\,\mathtt{id}\,(\mathtt{hoasap}\,\mathtt{id}A\,\mathtt{id}B)\,\mathtt{id})\Rightarrow$$
$$(\forall F_{\iota\to\iota\to\iota}.(\forall M_\iota.\mathtt{in}M\,\mathtt{subst}\Rightarrow\forall A_\iota.\mathtt{in}A\,\mathtt{term}\Rightarrow\mathtt{in}\,(FMA)\,\mathtt{term})\Rightarrow$$
$$(\forall M_\iota.\mathtt{in}M\,\mathtt{subst}\Rightarrow\forall A_\iota.\mathtt{in}A\,\mathtt{term}\Rightarrow\forall N_\iota.\mathtt{in}N\,\mathtt{subst}\Rightarrow$$
$$\mathtt{sub}\,(FMA)N = F\,(\mathtt{comp}MN)\,(\mathtt{sub}AN))\Rightarrow$$
$$(\forall A_\iota.\mathtt{in}A\,\mathtt{term}\Rightarrow P\,\mathtt{id}A\,\mathtt{id}\Rightarrow P\,\mathtt{id}\,(F\,\mathtt{id}A)\,\mathtt{id})\Rightarrow$$
$$P\,\mathtt{id}\,(\mathtt{hoaslam}\,\mathtt{id}\lambda M_\iota A_\iota.FMA)\,\mathtt{id})\Rightarrow$$
$$\forall A_\iota.\mathtt{in}A\,\mathtt{term}\Rightarrow P\,\mathtt{id}A\,\mathtt{id}$$

We encoded this theorem in LEO with Representation II as follows:

$$\forall P_{\mathtt{subst}\to\mathtt{term}\to\mathtt{subst}\to o}.(\forall M_{\mathtt{subst}} A_{\mathtt{term}} N_{\mathtt{subst}} K_{\mathtt{subst}}.$$
$$PMA(\mathtt{comp}KN)\Rightarrow P(\mathtt{comp}MK)(\mathtt{sub}AK)N)\Rightarrow$$
$$(\forall M_{\mathtt{subst}} A_{\mathtt{term}} N_{\mathtt{subst}} K_{\mathtt{subst}}.$$
$$P(\mathtt{comp}MK)(\mathtt{sub}AK)N\Rightarrow PMA(\mathtt{comp}KN))\Rightarrow$$
$$(\forall A_{\mathtt{term}}.\mathtt{hoasvar}\,\mathtt{id}A\,\mathtt{id}\Rightarrow P\,\mathtt{id}A\,\mathtt{id})\Rightarrow$$
$$(\forall A_{\mathtt{term}} B_{\mathtt{term}}.P\,\mathtt{id}A\,\mathtt{id}\Rightarrow P\,\mathtt{id}B\,\mathtt{id}\Rightarrow P\,\mathtt{id}(\mathtt{hoasap}\,\mathtt{id}A\,\mathtt{id}B)\,\mathtt{id})$$
$$\Rightarrow(\forall F_{\mathtt{subst}\to\mathtt{term}\to\mathtt{term}}.(\forall M_{\mathtt{subst}} A_{\mathtt{term}} N_{\mathtt{subst}}.$$
$$\mathtt{sub}(FMA)N = F(\mathtt{comp}MN)(\mathtt{sub}AN))\Rightarrow$$
$$(\forall A_{\mathtt{term}}.P\,\mathtt{id}A\,\mathtt{id}\Rightarrow P\,\mathtt{id}(F\,\mathtt{id}A)\,\mathtt{id})\Rightarrow$$
$$P\,\mathtt{id}(\mathtt{hoaslam}\,\mathtt{id}\lambda M_{\mathtt{subst}} A_{\mathtt{term}}.FMA)\,\mathtt{id})\Rightarrow$$
$$\forall A_{\mathtt{term}}.P\,\mathtt{id}A\,\mathtt{id}$$

`hoasinduction_gthm` is:

$$\mathtt{axapp}\Rightarrow\mathtt{axvarcons}\Rightarrow\mathtt{axvarid}\Rightarrow\mathtt{axabs}\Rightarrow\mathtt{axclos}\Rightarrow\mathtt{axidl}$$
$$\Rightarrow\mathtt{axshiftcons}\Rightarrow\mathtt{axassoc}\Rightarrow\mathtt{axmap}\Rightarrow\mathtt{axidr}\Rightarrow\mathtt{axvarshift}\Rightarrow$$
$$\mathtt{axscons}\Rightarrow\mathtt{ulamvar1}\Rightarrow\mathtt{ulamvarsh}\Rightarrow\mathtt{ulamvarind}\Rightarrow\mathtt{apinj1}\Rightarrow$$
$$\mathtt{apinj2}\Rightarrow\mathtt{laminj}\Rightarrow\mathtt{shinj}\Rightarrow\mathtt{lamnotap}\Rightarrow\mathtt{apnotvar}\Rightarrow\mathtt{lamnotvar}$$
$$\Rightarrow\mathtt{induction}\Rightarrow\mathtt{pushprop}\Rightarrow\mathtt{induction2lem}\Rightarrow\mathtt{induction2}\Rightarrow$$
$$\mathtt{substmonoid}\Rightarrow\mathtt{termmset}\Rightarrow\mathtt{hoasapinj1}\Rightarrow\mathtt{hoasapinj2}\Rightarrow$$
$$\mathtt{hoaslaminj}\Rightarrow\mathtt{hoaslamnotap}\Rightarrow\mathtt{hoaslamnotvar}\Rightarrow\mathtt{hoasapnotvar}$$
$$\Rightarrow\mathtt{hoasinduction\_lem1}\Rightarrow\mathtt{hoasinduction\_lem2}\Rightarrow$$
$$\mathtt{hoasinduction\_lem3}\Rightarrow\mathtt{hoasinduction}$$

`hoasinduction_lthm_1` is:

$$\mathtt{induction2}\Rightarrow\mathtt{axvarid}\Rightarrow\mathtt{axclos}\Rightarrow\mathtt{axvarshift}\Rightarrow\mathtt{axmap}\Rightarrow\mathtt{axidl}$$
$$\Rightarrow\mathtt{hoasinduction}$$

We have already proven this theorem informally in chapter 2. LEO can not prove this theorem with both representations. Since Representation II is better than Representation I, we discuss only Representation II. Now we tried to split this

theorem into smaller pieces. We prove this theorem using `induction2` by hand (see chapter 2). So we can match three conditions of `hoasinduction` with three condition of `induction2`. We called these three lemmas:

`hoasinduction_lem1`, `hoasinduction_lem2`, `hoasinduction_lem3`

- `hoasinduction_lem1`

$$\forall P_{\texttt{subst}\to\texttt{term}\to\texttt{subst}\to o}.(\forall M_{\texttt{subst}} A_{\texttt{term}} N_{\texttt{subst}} K_{\texttt{subst}}.$$
$$PMA(\texttt{comp}KN)\Rightarrow P(\texttt{comp}MK)(\texttt{sub}AK)N)\Rightarrow$$
$$(\forall M_{\texttt{subst}} A_{\texttt{term}} N_{\texttt{subst}} K_{\texttt{subst}}.$$
$$P(\texttt{comp}MK)(\texttt{sub}AK)N\Rightarrow PMA(\texttt{comp}KN))\Rightarrow$$
$$(\forall A_{\texttt{term}}.\texttt{hoasvar}\,\texttt{id}A\,\texttt{id}\Rightarrow P\,\texttt{id}A\,\texttt{id})\Rightarrow$$
$$\forall A_{\texttt{term}}.\texttt{var}A\Rightarrow P\,\texttt{id}A\,\texttt{id}$$

- `hoasinduction_lem2`

$$\forall P_{\texttt{subst}\to\texttt{term}\to\texttt{subst}\to o}.(\forall M_{\texttt{subst}} A_{\texttt{term}} N_{\texttt{subst}} K_{\texttt{subst}}.$$
$$PMA(\texttt{comp}KN)\Rightarrow P(\texttt{comp}MK)(\texttt{sub}AK)N)\Rightarrow$$
$$(\forall M_{\texttt{subst}} A_{\texttt{term}} N_{\texttt{subst}} K_{\texttt{subst}}.$$
$$P(\texttt{comp}MK)(\texttt{sub}AK)N\Rightarrow PMA(\texttt{comp}KN))\Rightarrow$$
$$(\forall A_{\texttt{term}} B_{\texttt{term}}.P\,\texttt{id}A\,\texttt{id}\Rightarrow P\,\texttt{id}B\,\texttt{id}\Rightarrow$$
$$P\,\texttt{id}(\texttt{hoasap}\,\texttt{id}A\,\texttt{id}B)\,\texttt{id})\Rightarrow$$
$$\forall A_{\texttt{term}} B_{\texttt{term}}.P\,\texttt{id}A\,\texttt{id}\Rightarrow P\,\texttt{id}B\,\texttt{id}\Rightarrow P\,\texttt{id}(\texttt{ap}AB)\,\texttt{id}$$

- `hoasinduction_lem3`

$$\forall P_{\texttt{subst}\to\texttt{term}\to\texttt{subst}\to o}.(\forall M_{\texttt{subst}} A_{\texttt{term}} N_{\texttt{subst}} K_{\texttt{subst}}.$$
$$PMA(\texttt{comp}KN)\Rightarrow P(\texttt{comp}MK)(\texttt{sub}AK)N)\Rightarrow$$
$$(\forall M_{\texttt{subst}} A_{\texttt{term}} N_{\texttt{subst}} K_{\texttt{subst}}.$$
$$P(\texttt{comp}MK)(\texttt{sub}AK)N\Rightarrow PMA(\texttt{comp}KN))\Rightarrow$$
$$(\forall F_{\texttt{subst}\to\texttt{term}\to\texttt{term}}.(\forall M_{\texttt{subst}} A_{\texttt{term}} N_{\texttt{subst}}.$$
$$\texttt{sub}(FMA)N = F(\texttt{comp}MN)(\texttt{sub}AN))\Rightarrow$$
$$(\forall A_{\texttt{term}}.P\,\texttt{id}A\,\texttt{id}\Rightarrow P\,\texttt{id}(F\,\texttt{id}A)\,\texttt{id})\Rightarrow$$
$$P\,\texttt{id}(\texttt{hoaslam}\,\texttt{id}\lambda M_{\texttt{subst}} A_{\texttt{term}}.FMA)\,\texttt{id})\Rightarrow$$
$$\forall A_{\texttt{term}}.(\forall B_{\texttt{term}}.P\,\texttt{id}B\,\texttt{id}\Rightarrow P\,\texttt{id}(\texttt{sub}A(\texttt{push}B\,\texttt{id}))\,\texttt{id})\Rightarrow$$
$$P\,\texttt{id}(\texttt{lam}A)\,\texttt{id}$$

Then the hoasinduction_lthm_2 should be:

$$\texttt{induction2} \Rightarrow \texttt{hoasinduction\_lem1} \Rightarrow \texttt{hoasinduction\_lem2} \Rightarrow$$
$$\texttt{hoasinduction\_lem3} \Rightarrow \texttt{hoasinduction}$$

LEO could prove hoasinduction_lem1_gthm and hoasinduction_lem2_gthm, but hoasinduction_lem3_gthm (the λ case) was still too hard. We determined which axioms and previous results are needed to prove hoasinduction_lem3 and add these to hoasinduction_lem3_lthm.

- hoasinduction_lem3_lthm_1

$$\texttt{axvarid} \Rightarrow \texttt{axvarshift} \Rightarrow \texttt{axclos} \Rightarrow \texttt{axmap} \Rightarrow \texttt{hoasinduction\_lem3}$$

LEO could not prove this. After a careful examination of the informal proof, we notice the proof does not require the condition on Ψ:

$$kn \in \Psi(m,a) \text{ iff } n \in \Psi(mk,ak)$$

We modified the hoasinduction_lem3 to hoasinduction_lem3a by deleting the the Ψ- condition.

- hoasinduction_lem3a

$$\forall P_{\texttt{subst}\to\texttt{term}\to\texttt{subst}\to o}.(\forall F_{\texttt{subst}\to\texttt{term}\to\texttt{term}}.$$
$$(\forall M_{\texttt{subst}} A_{\texttt{term}} N_{\texttt{subst}}.$$
$$\texttt{sub}\,(F\,M\,A)\,N = F\,(\texttt{comp}\,M\,N)\,(\texttt{sub}\,A\,N)) \Rightarrow$$
$$(\forall A_{\texttt{term}}.P\,\texttt{id}\,A\,\texttt{id} \Rightarrow P\,\texttt{id}\,(F\,\texttt{id}\,A)\,\texttt{id}) \Rightarrow$$
$$P\,\texttt{id}\,(\texttt{hoaslam}\,\texttt{id}\,\lambda M_{\texttt{subst}} A_{\texttt{term}}.F\,M\,A)\,\texttt{id}) \Rightarrow$$
$$\forall A_{\texttt{term}}.(\forall B_{\texttt{term}}.P\,\texttt{id}\,B\,\texttt{id} \Rightarrow P\,\texttt{id}\,(\texttt{sub}\,A\,(\texttt{push}\,B\,\texttt{id}))\,\texttt{id}) \Rightarrow$$
$$P\,\texttt{id}\,(\texttt{lam}\,A)\,\texttt{id}$$

- hoasinduction_lem3a_lthm_1

$$\texttt{axvarid} \Rightarrow \texttt{axvarshift} \Rightarrow \texttt{axclos} \Rightarrow \texttt{axmap} \Rightarrow \texttt{hoasinduction\_lem3a}$$

LEO could not prove hoasinduction_lem3a_lthm_1. In the informal proof for the λ-case we choose f to be

$$f(m,a) = b[a.m]$$

so that

$$\lambda b = \lambda(f(\uparrow, 1)).$$

To test whether LEO can solve the unification problem, we created a new lemma `hoasinduction_lem3b`.

- `hoasinduction_lem3b`

$$\forall B_{\mathtt{term}}.(\exists F_{\mathtt{subst}\to\mathtt{term}\to\mathtt{term}}.\mathtt{sub}\,B\,(\mathtt{push}\,\mathtt{one}\,\mathtt{sh})) = F\,\mathtt{sh}\,\mathtt{one}$$

LEO could not prove this either. There is a flag `flag-max-uni-depth` in LEO, the standard setting for this flag is 5.
It is too low for LEO to prove `hoasinduction_lem3b`. When the flag is manually set to 10, LEO could find the proof, i.e. LEO could find the unifier.

We defined a new version of `hoasinduction_lem3a_lthm_1` that assumes `hoasinduction_lem3b` and see if LEO can prove it. It should look something like

- `hoasinduction_lem3a_lthm_2`

$$\mathtt{axvarid}\Rightarrow\mathtt{axvarshift}\Rightarrow\mathtt{axclos}\Rightarrow\mathtt{axmap}\Rightarrow\mathtt{hoasinduction\_lem3b}\Rightarrow\mathtt{hoasinduction\_lem3a}$$

LEO could not prove `hoasinduction_lem3a_lthm_2`.

LEO could not prove `hoasinduction_lem3a_lthm_1` and `hoasinduction_lem3a_lthm_2` We tried to modify `hoasinduction_lem3a` so that it no longer needs to use the sigma-equations.
Currently `hoasinduction_lem3a` needs `axvarid`, `axvarshift`, `axclos`, and `axmap`. We should be able to create `hoasinduction_lem3aa` which looks almost the same as `hoasinduction_lem3a`, but does not need any of `axvarid`, `axvarshift`, `axclos`, and `axmap`.

LEO can hopefully prove an `hoasinduction_lem3a_lthm_3` of the form:

$$\mathtt{axvarid}\Rightarrow\mathtt{axvarshift}\Rightarrow\mathtt{axclos}\Rightarrow\mathtt{axmap}\Rightarrow\mathtt{hoasinduction\_lem3aa}\Rightarrow\mathtt{hoasinduction\_lem3a}$$

- hoasinduction_lem3aa

$$\forall P_{\mathtt{subst}\to\mathtt{term}\to\mathtt{subst}\to o}.(\forall F_{\mathtt{subst}\to\mathtt{term}\to\mathtt{term}}.$$
$$(\forall M_{\mathtt{subst}} A_{\mathtt{term}} N_{\mathtt{subst}}.\mathtt{sub}\,(F\,M\,A)\,N = F\,(\mathtt{comp}\,M\,N)\,(\mathtt{sub}\,A\,N))\Rightarrow$$
$$(\forall A_{\mathtt{term}}.P\,\mathtt{id}\,A\,\mathtt{id}\Rightarrow P\,\mathtt{id}\,(F\,\mathtt{id}\,A)\,\mathtt{id})\Rightarrow$$
$$P\,\mathtt{id}\,(\mathtt{hoaslam}\,\mathtt{id}\,\lambda M_{\mathtt{subst}} A_{\mathtt{term}}.F\,M\,A)\,\mathtt{id})\Rightarrow$$
$$\forall A_{\mathtt{term}}.(\forall B_{\mathtt{term}}.P\,\mathtt{id}\,B\,\mathtt{id}\Rightarrow P\,\mathtt{id}\,(\mathtt{sub}\,A\,(\mathtt{push}\,B\,\mathtt{id}))\,\mathtt{id})\Rightarrow$$
$$P\,\mathtt{id}\,(\mathtt{lam}\,(\mathtt{sub}\,A\,(\mathtt{push}\,\mathtt{one}\,\mathtt{sh})))\,\mathtt{id}$$

With standard *flag-max-uni-depth* LEO could not prove hoasinduction-lem3aa_lthm. LEO even could not prove this when the *flag-max-uni-depth* is set as high as 10.

LEO could prove hoasinduction_lem3a_lthm_3, because from hoasinduction_lem3aa to hoasinduction_lem3a is easy. LEO could also prove hoasinduction_lem3_lthm_2 (with ψ-condition):

axvarid⇒axvarshift⇒hoasinduction_lem3aa⇒hoasinduction_lem3

We want LEO to prove hoasinduction_lthm_2 as following:

induction2⇒hoasinduction_lem1⇒hoasinduction_lem2⇒

hoasinduction_lem3⇒hoasinduction

But LEO could not prove this either.

We tried another way to break down the proof of hoasinduction.
We defined hoasinduction_lem0, which gives exactly the relationship between hoasinduction and induction2.

	Induction2	Hoasinduction
Property	$Q_{\mathtt{term}\to o}$	$P_{\mathtt{subst}\to\mathtt{term}\to\mathtt{subst}\to o}$

First we define the hoasinduction_p_and_p_prime.

- hoasinduction_p_and_p_prime

$$\lambda P Q.\forall X_{\mathtt{term}}.Q\,X \Leftrightarrow P\,\mathtt{id}\,X\,\mathtt{id}$$

- hoasinduction_lem0

$$\forall P.\exists Q.\mathtt{hoasinduction\_p\_and\_p\_prime}\,P\,Q$$

LEO can prove `hoasinduction_lem0`.

Now we match three condition of `hoasinduction` with three condition of `induction2` using the definition `hoasinduction_p_and_p_prime`.

- `hoasinduction_lem1v2`

$$\forall P_{\mathtt{subst}\to\mathtt{term}\to\mathtt{subst}\to o}\, Q_{\mathtt{term}\to o}.(\forall M_{\mathtt{subst}} A_{\mathtt{term}} N_{\mathtt{subst}} K_{\mathtt{subst}}.$$
$$P\,M\,A\,(\mathtt{comp}\,K\,N)\Rightarrow P\,(\mathtt{comp}\,M\,K)\,(\mathtt{sub}\,A\,K)\,N)\Rightarrow$$
$$(\forall M_{\mathtt{subst}} A_{\mathtt{term}} N_{\mathtt{subst}} K_{\mathtt{subst}}.$$
$$P\,(\mathtt{comp}\,M\,K)\,(\mathtt{sub}\,A\,K)\,N\Rightarrow P\,M\,A\,(\mathtt{comp}\,K\,N))\Rightarrow$$
$$(\forall A_{\mathtt{term}}.\mathtt{hoasvar}\,\mathtt{id}\,A\,\mathtt{id}\Rightarrow P\,\mathtt{id}\,A\,\mathtt{id})\Rightarrow$$
$$\mathtt{hoasinduction\_p\_and\_p\_prime}\,P\,Q\Rightarrow\forall A_{\mathtt{term}}.\mathtt{var}\,A\Rightarrow Q\,A$$

- `hoasinduction_lem2v2`

$$\forall P_{\mathtt{subst}\to\mathtt{term}\to\mathtt{subst}\to o}\, Q_{\mathtt{term}\to o}.(\forall M_{\mathtt{subst}} A_{\mathtt{term}} N_{\mathtt{subst}} K_{\mathtt{subst}}.$$
$$P\,M\,A\,(\mathtt{comp}\,K\,N)\Rightarrow P\,(\mathtt{comp}\,M\,K)\,(\mathtt{sub}\,A\,K)\,N)\Rightarrow$$
$$(\forall M_{\mathtt{subst}} A_{\mathtt{term}} N_{\mathtt{subst}} K_{\mathtt{subst}}.$$
$$P\,(\mathtt{comp}\,M\,K)\,(\mathtt{sub}\,A\,K)\,N\Rightarrow P\,M\,A\,(\mathtt{comp}\,K\,N))\Rightarrow$$
$$(\forall A_{\mathtt{term}} B_{\mathtt{term}}.P\,\mathtt{id}\,A\,\mathtt{id}\Rightarrow P\,\mathtt{id}\,B\,\mathtt{id}\Rightarrow$$
$$P\,\mathtt{id}\,(\mathtt{hoasap}\,\mathtt{id}\,A\,\mathtt{id}\,B)\,\mathtt{id})\Rightarrow$$
$$\mathtt{hoasinduction\_p\_and\_p\_prime}\,P\,Q\Rightarrow\forall A_{\mathtt{term}} B_{\mathtt{term}}.Q\,A\Rightarrow Q\,B\Rightarrow$$
$$Q\,(\mathtt{ap}\,A\,B)$$

- `hoasinduction_lem3v2`

$$\forall P_{\mathtt{subst}\to\mathtt{term}\to\mathtt{subst}\to o}\, Q_{\mathtt{term}\to o}.(\forall M_{\mathtt{subst}} A_{\mathtt{term}} N_{\mathtt{subst}} K_{\mathtt{subst}}.$$
$$P\,M\,A\,(\mathtt{comp}\,K\,N)\Rightarrow P\,(\mathtt{comp}\,M\,K)\,(\mathtt{sub}\,A\,K)\,N)\Rightarrow$$
$$(\forall M_{\mathtt{subst}} A_{\mathtt{term}} N_{\mathtt{subst}} K_{\mathtt{subst}}.P\,(\mathtt{comp}\,M\,K)\,(\mathtt{sub}\,A\,K)\,N\Rightarrow P\,M\,A\,(\mathtt{comp}\,K\,N))$$
$$\Rightarrow(\forall F_{\mathtt{subst}\to\mathtt{term}\to\mathtt{term}}.(\forall M_{\mathtt{subst}} A_{\mathtt{term}} N_{\mathtt{subst}}.$$
$$\mathtt{sub}\,(F\,M\,A)\,N = F\,(\mathtt{comp}\,M\,N)\,(\mathtt{sub}\,A\,N))\Rightarrow$$
$$(\forall A_{\mathtt{term}}.P\,\mathtt{id}\,A\,\mathtt{id}\Rightarrow P\,\mathtt{id}\,(F\,\mathtt{id}\,A)\,\mathtt{id})\Rightarrow$$
$$P\,\mathtt{id}\,(\mathtt{hoaslam}\,\mathtt{id}\,\lambda M_{\mathtt{subst}} A_{\mathtt{term}}.F\,M\,A)\,\mathtt{id})\Rightarrow$$
$$\mathtt{hoasinduction\_p\_and\_p\_prime}\,P\,Q\Rightarrow$$
$$\forall A_{\mathtt{term}}.(\forall B_{\mathtt{term}}.Q\,B\Rightarrow Q\,(\mathtt{sub}\,A\,(\mathtt{push}\,B\,\mathtt{id})))\Rightarrow Q\,(\mathtt{lam}\,A)$$

As before hoasinduction_lem1v2 and hoasinduction_lem2v2 were straightforward in LEO. We turned our attention to hoasinduction_lem3v2. LEO could not prove it. The hoasinduction_lem3v2_lthm should look like:

- hoasinduction_lem3v2_lthm

$$\texttt{axvarid}\Rightarrow\texttt{axvarshift}\Rightarrow$$
$$\texttt{axclos}\Rightarrow\texttt{axmap}\Rightarrow\texttt{hoasinduction\_lem3v2}$$

For the reasons discussed before we modified the hoasinduction_lem3v2 to hoasinduction_lem3v2a by deleting the ψ-condition.

- hoasinduction_lem3v2a

$$\forall P_{\texttt{subst}\to\texttt{term}\to\texttt{subst}\to o} Q_{\texttt{term}\to o}.(\forall F_{\texttt{subst}\to\texttt{term}\to\texttt{term}}.$$
$$(\forall M_{\texttt{subst}} A_{\texttt{term}} N_{\texttt{subst}}.\texttt{sub}\,(F\,M\,A)\,N = F\,(\texttt{comp}\,M\,N)\,(\texttt{sub}\,A\,N))\Rightarrow$$
$$(\forall A_{\texttt{term}}.P\,\texttt{id}\,A\,\texttt{id}\Rightarrow P\,\texttt{id}\,(F\,\texttt{id}\,A)\,\texttt{id})\Rightarrow$$
$$P\,\texttt{id}\,(\texttt{hoaslam}\,\texttt{id}\,\lambda M_{\texttt{subst}} A_{\texttt{term}}.F\,M\,A)\,\texttt{id})\Rightarrow$$
$$\texttt{hoasinduction\_p\_and\_p\_prime}\,P\,Q\Rightarrow$$
$$\forall A_{\texttt{term}}.(\forall B_{\texttt{term}}.Q\,B\Rightarrow Q\,(\texttt{sub}\,A\,(\texttt{push}\,B\,\texttt{id})))\Rightarrow Q\,(\texttt{lam}\,A)$$

The hoasinduction_lem3v2a_lthm looks like:

- hoasinduction_lem3v2a_lthm

$$\texttt{axvarid}\Rightarrow$$
$$\texttt{axvarshift}\Rightarrow\texttt{axclos}\Rightarrow\texttt{axmap}\Rightarrow\texttt{hoasinduction\_lem3v2a}$$

LEO could not prove this. But we hope, if we define the hoasinduction_lthm_3 as following, LEO can prove it.

$$\texttt{hoasinduction\_lem0}\Rightarrow\texttt{induction2}\Rightarrow\texttt{axvarid}\Rightarrow\texttt{hoasinduction\_lem3v2a}$$
$$\Rightarrow\texttt{hoasinduction}$$

Finally LEO could prove it, i.e. LEO could prove hoasinduction, but could not prove the sublemma hoasinduction_lem3v2a of it. Hoasinduction_lem3v2a is still too hard for LEO.

4.5 Pushprop

We want to prove this result in LEO.

Theorem 1 (pushprop) Let Φ be a property, $a \in T$ and $m \in M$. Assume for all $x \in Var$, $(x[m])^{\downarrow\sigma}$ satisfies Φ. Assume a satisfies Φ. Then $(x[a.m])^{\downarrow\sigma}$ satisfies Φ for all $x \in Var$.

We encoded this theorem in LEO with Representation II as follows:

$$\forall P_{\mathtt{term}\to o} A_{\mathtt{term}} M_{\mathtt{subst}}.(\forall B_{\mathtt{term}}.\mathtt{var}\,B \Rightarrow P\,(\mathtt{sub}\,B\,M)) \Rightarrow P\,A \Rightarrow \forall B_{\mathtt{term}}.\mathtt{var}\,B \Rightarrow P\,(\mathtt{sub}\,B\,(\mathtt{push}\,A\,M))$$

The pushprop_lthm is

$$\mathtt{ulamvar1} \Rightarrow \mathtt{axvarcons} \Rightarrow \mathtt{axclos} \Rightarrow \mathtt{axshiftcons} \Rightarrow \mathtt{ulamvarind} \Rightarrow \mathtt{pushprop}$$

We wanted to get LEO to prove pushprop. We now know that LEO can prove pushprop if it is given ulamvar1, ulamvarind, axvarcons, axclos, axshiftcons and the following Lemma:

Theorem 15 (pushprop_lem0) For every property ϕ, term a and substitution m,there is a property ϕ' such that for every term x, x satisfies ϕ' iff $(x[a.m])^{\downarrow\sigma}$ satisfies ϕ.

Proof: Just define ϕ' in this way.

Originally LEO was not able to prove pushprop_lem0. Define:

- pushprop_p_and_p_prime

$$\lambda A_{\mathtt{term}} M_{\mathtt{subst}} P_{\mathtt{term}\to o} Q_{\mathtt{term}\to o}.\forall X_{\mathtt{term}}.Q\,X \Leftrightarrow P\,(\mathtt{sub}\,X\,(\mathtt{push}\,A\,M))$$

- pushprop_lem0

$$\forall P_{\texttt{term}\to o} A_{\texttt{term}} M_{\texttt{subst}}.\exists Q_{\texttt{term}\to o}.\texttt{pushprop\_p\_and\_p\_prime}$$
$$A\,M\,P\,Q$$

We could write the term `pushprop_lem0` in ordinary notation as:

$$\forall P \forall a \forall m \exists Q \forall x. Qx \Leftrightarrow P\ (\texttt{sub}\ x(\texttt{push}\ a\ m))$$

Here the variables P and Q have type `term` $\to$ `o`, the variables a and x have type `term`, and the variable m has type `subst`. Also, the constant *sub* has type `term` $\to$ `subst` $\to$ `term`, and the constant *push* has type `term` $\to$ `subst` $\to$ `subst`.

When LEO creates the clauses from the negation of `pushprop_lem0`, there would be two clauses:

1. $Qsk^Q \vee P(\texttt{sub}\ sk^Q\ (\texttt{push}\ a\ m))$
2. $\neg Qsk^Q \vee \neg P(\texttt{sub}\ sk^Q(\texttt{push}\ a\ m))$

where P, a and m are skolem constants, $sk^Q$is a Skolem function of Q, and Q is a variable to be instantiated.

Solving this problem requires a *factoring rule.* For example, if we factor the first clause we should obtain a clause like

$$P\ (\texttt{sub}\ sk^Q(\texttt{push}\ a\ m))$$

with constraint

$$Q\ sk^Q\ =^?\ P(\texttt{sub}\ sk^Q(\texttt{push}\ a\ m))$$

where the unification constraint could be solved to give the desired value for Q:

$$\lambda x.P\ (\texttt{sub}\ x\ (\texttt{push}\ a\ m))$$

Since LEO did not have this factoring rule, it could not prove `pushprop_lem0`. After Chris Benzmüller added the factoring rule to LEO it could prove the `pushprop_lem0`. Also LEO could prove the following version of `pushprop_lthm`:

$$\texttt{pushprop\_lem0}\Rightarrow\texttt{ulamvar1}\Rightarrow\texttt{axvarcons}\Rightarrow\texttt{axclos}\Rightarrow\texttt{axshiftcons}$$
$$\Rightarrow\texttt{ulamvarind}\Rightarrow\texttt{pushprop}$$

Chapter 5

Conclusion and Future Work

5.1 Conclusion

LEO is sensitive to the representation. In this project we developed two representations. We defined only one base type ι in Representation I and two base types `term` and `subst` in Representation II. LEO could prove more theorems in Representation II. Using the Representation II LEO could prove the theorems much more quickly. For example LEO could prove `Substmonoid_gthm` in Representation I in 11.589s and 1.324s in Representation II.

LEO is sensitive to how many assumptions are given. There is one `lthm` and one `gthm` for each theorem. Sometimes LEO could only prove the `lthm` not the `gthm`, because `lthm` has much fewer assumptions than the `gthm`. For example `induction2`. There are 26 assumptions for `induction2_gthm`, but only 2 assumptions for `induction2_lthm`. LEO could prove `induction2_lthm` in 0.581s, but LEO could not prove `induction2_gthm`.

Instantiation of higher order variables is hard for LEO. Sometimes LEO can find the instantiation, for example theorem `hoaslaminj`. Sometimes LEO can not find the instantiation, for example theorem `hoasinduction`.

There are four of our 14 theorems, i.e. `pushprop`, `induction2lem`, `induction2` and `hoasinduction` which used some induction principle. At the beginning of our reseach LEO could not prove any of these, Finally LEO could prove `pushprop`, `induction2` and `hoasinduction`, but `induction2lem` is still hard. Induction is hard for LEO.

5.2 Future Work

One task left is to try to prove some intermediate lemmas for `hoasinduction`. By `hoasinduction` we created together 11 intermediate lemmas, 8 of these LEO was able to prove, `hoasinduction_lem3aa`, `hoasinduction_lem3v2` and `hoasinduction_lem3v2a` LEO could not prove.

We will also try to prove `induction2lem` by creating intermediate lemma. By informal proof we use `induction` to prove `induction2lem`, so we can match the first condition of `induction2lem` to first condition of `induction`, second condition of `induction2lem` to second condition of `induction`, and so on.

As a future project, we could try to build induction into LEO. In principle LEO can prove theorems which used some induction principle, but it is hard. For example, we want to prove `hoasinduction`, using `induction2`.

Theorem 3 (`induction2`) Let Φ be a property such that the following hold:

1. For all $x \in Var$, x satisfies Φ.
2. For all $a, b \in T$, if a and b satisfy Φ, then $(a\,b)^{\downarrow\sigma}$ satisfies Φ.
3. For all $a \in T$, if $(a[b.id])^{\downarrow\sigma}$ satisfies Φ whenever $b \in T$ satisfies Φ, then $(\lambda a)^{\downarrow\sigma}$ satisfies Φ.

Then for all $a \in T$, a satisfies Φ.

To use `induction2`, a theorem prover should:

1. Recognize `induction2` is an induction principle.
2. Choose an appropriate Φ.
3. Prove each of 1, 2 and 3 for this Φ.

LEO does not recognize `induction2`. Currently LEO-II employs a higher-order resolution calculus, where the search for an empty clause and higher-order pre-unification are interleaved, but LEO does not recognize the induction principle. It would be useful to build induction into LEO.

Bibliography

[1] Martín Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lèvy. Explicit substitutions. In *Conference Record of the Seventeenth Annual ACM Symposium on Principles of Programming Languages, San Francisco, California*, pages 31–46. ACM, 1990.

[2] Christoph Benzmüller, Florian Rabe, and Geoff Sutcliffe. The core tptp language for classical higher-order logic. In *Fourth International Joint Conference on Automated Reasoning (IJCAR'06)*, volume 5195 of *LNAI*. Springer, 2008.

[3] Christoph Benzmüller, Frank Theiss, Larry Paulson, and Arnaud Fietzke. LEO-II - a cooperative automatic theorem prover for higher-order logic. In *Fourth International Joint Conference on Automated Reasoning (IJCAR'06)*, volume 5195 of *LNAI*. Springer, 2008.

[4] Chad E. Brown. M-set models. In C. E. Benzmüller, C. E. Brown, J. Siekmann, and R. Statman, editors, *Reasoning in Simple Type Theory: Festschrift in Honor of Peter B. Andrews on His 70th Birthday*, Studies in Logic and the Foundations of Mathematics. IFCoLog, 2008. To appear.

[5] Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Higher-order unification via explicit substitutions. In D. Kozen, editor, *Proceedings of the Tenth Annual Symposium on Logic in Computer Science*, pages 366–374, San Diego, California, June 1995. IEEE Computer Society Press.

[6] Frank Pfenning and Conal Elliott. Higher-order abstract syntax. In *Proceedings of the ACM SIGPLAN '88 Symposium on Language Design and Implementation*, pages 199–208, Atlanta, Georgia, June 1988.

Appendix A

Representation II in THF Format

```
thf(one,constant,(one:term)).

thf(ap,constant,(ap:(term>(term>term)))).

thf(lam,constant,(lam:(term>term))).

thf(sub,constant,(sub:(term>(subst>term)))).

thf(id,constant,(id:subst)).

thf(sh,constant,(sh:subst)).

thf(push,constant,(push:(term>(subst>subst)))).

thf(comp,constant,(comp:(subst>(subst>subst)))).

thf(axapp,definition,(axapp := (![A:term]:(![B:term]:
(![M:subst]:((sub @ (ap @ A @ B) @ M)= (ap @ (sub @
A @ M) @ (sub @ B @ M))))))))).

thf(axvarcons,definition,(axvarcons := (![A:term]:
(![M:subst]:((sub @ one @ (push @ A @ M)) = A))))).

thf(axvarid,definition,(axvarid := (![A:term]:((sub
 @ A @ id) = A)))).
```

```
thf(axabs,definition,(axabs := (![A:term]:(![M:subst]
:((sub @ (lam @ A) @ M) = (lam @ (sub @ A @ (push @
one @ (comp @ M @ sh))))))))).

thf(axclos,definition,(axclos := (![A:term]:(![M:subst]
:(![N:subst]:((sub @ (sub @ A @ M) @ N) = (sub @ A @
 (comp @ M @ N))))))))).

thf(axidl,definition,(axidl := (![M:subst]:((comp @
id @ M) = M)))).

thf(axshiftcons,definition,(axshiftcons := (![A:term]
:(![M:subst]:((comp @ sh @ (push @ A @ M)) = M))))).

thf(axassoc,definition,(axassoc := (![M:subst]:(![N:subst]
:(![K:subst]:((comp @ (comp @ M @ N) @ K) = (comp @ M @
 (comp @ N @ K))))))))).

thf(axmap,definition,(axmap := (![A:term]:(![M:subst]:(![N:subst]:
((comp @ (push @ A @ M) @ N) = (push @ (sub @ A @ N) @ (comp @ M
@ N))))))))).

thf(axidr,definition,(axidr := (![M:subst]:((comp @ M @ id) = M)))).

thf(axvarshift,definition,(axvarshift := ((push @ one @ sh) = id))).

thf(axscons,definition,(axscons := (![M:subst]:((push @ (sub @ one
    @ M) @ (comp @ sh @ M)) = M)))).

thf(var,constant,(var:(term>$o))).

thf(ulamvar1,definition,(ulamvar1 := (var @ one))).

thf(ulamvarsh,definition,(ulamvarsh := (![A:term]:((var @ A) =>
   (var @ (sub @ A @ sh)))))).

thf(ulamvarind,definition,(ulamvarind := (![P:(term>$o)]:((P @ one)
  => ((![A:term]:((var @ A) => ((P @ A) => (P @ (sub @ A @ sh)))))
=>  (![A:term]:((var @ A) => (P @ A))))))))).

thf(apinj1,definition,(apinj1 := (![A:term]:(![B:term]:(![C:term]:
```

```
   (![D:term]:(((ap @ A @ C) = (ap @ B @ D)) => (A = B)))))))).

thf(apinj2,definition,(apinj2 := (![A:term]:(![B:term]:(![C:term]:
   (![D:term]:(((ap @ A @ C) = (ap @ B @ D)) => (C = D)))))))).

thf(laminj,definition,(laminj := (![A:term]:(![B:term]:(((lam @ A)
   = (lam @ B)) => (A = B)))))).

thf(shinj,definition,(shinj := (![A:term]:(![B:term]:(((sub @ A @
sh) = (sub @ B @ sh)) => (A = B)))))).

thf(lamnotap,definition,(lamnotap := (![A:term]:(![B:term]:(![C:term]:
    (~ ((lam @ A) = (ap @ B @ C)))))))).

thf(apnotvar,definition,(apnotvar := (![A:term]:(![B:term]:(~
   (var @ (ap @ A @ B)))))))).

thf(lamnotvar,definition,(lamnotvar := (![A:term]:(~ (var @ (lam @ A)
)))))).

thf(induction,definition,(induction := (![P:(term>$o)]:((![A:term]:
   ((var @ A) => (P @ A))) => ((![A:term]:(![B:term]:((P @ A) => ((
   P @ B)=> (P @ (ap @ A @ B)))))) => ((![A:term]:((P @ A) => (P @
    (lam @ A))))=> (![A:term]:(P @ A)))))))).

thf(pushprop_lem1,definition,(pushprop_lem1 := (![P:(term>$o)]:
   (!K:[term.$o]:(![A:term]:((![M:subst]:(![B:term]: (P @ A) =>
   (K @ (sub @ A @ (push @ B @ M))))))))))).

thf(pushprop_lem1_gthm,definition,(pushprop_lem1_gthm := (axapp
   => (axvarcons => (axvarid => (axabs => (axclos => (axidl =>
   (axshiftcons => (axassoc => (axmap => (axidr => (axvarshift =>
   (axscons => (ulamvar1 => (ulamvarsh => (ulamvarind => (apinj1
   => (apinj2 => (laminj => (shinj => (lamnotap => (apnotvar =>
   (lamnotvar => (induction => pushprop_lem1)))))))))))))))))))
   ))))))).

thf(pushprop_lem1_lthm,definition,(pushprop_lem1_lthm :=
(axvarcons => (axclos => (axshiftcons => (ulamvarind =>
pushprop_lem1)))))).
```

```
thf(pushprop_p_and_p_prime,definition,(pushprop_p_and_p
_prime := (^[A:term]:(^[M:subst]:(^[P:(term>$o)]:
(^[Q:(term>$o)]:![X:$term]:((var @ X) => ((Q @ X) <=>
 (P @ (sub @ X @ (push @ A @ M)))))))))).

thf(pushprop_lem1v2,definition,(pushprop_lem1v2 := (![P:
(term>$o)]:(![Q:(term>$o)]:(![A:term]:(![M:subst]:((P @ A) =>
 ((pushprop_p_and_p_prime @ A @ M @ P @ Q) => (Q @ one))))))))).

thf(pushprop_lem1v2_gthm,definition,(pushprop_lem1v2_gthm :=
(axapp => (axvarcons => (axvarid => (axabs => (axclos => (axidl
=>(axshiftcons =>(axassoc => (axmap => (axidr => (axvarshift =>
 (axscons => (ulamvar1 => (ulamvarsh => (ulamvarind => (apinj1
 => (apinj2 => (laminj => (shinj => (lamnotap => (apnotvar =>
 (lamnotvar => (induction => pushprop_lem1v2))))))))))))))))
 )))))))).

thf(pushprop_lem1v2_lthm,definition,(pushprop_lem1v2_lthm :=
(ulamvar1 => (axvarcons => pushprop_lem1v2)))).

thf(pushprop_p_and_p_prime,definition,(pushprop_p_
and_p_prime := (^[A:term]:(^[M:subst]:(^[P:(term>$o)]
:(^[Q:(term>$o)]:![X:$term]:((Q @ X) <=> (P @
(sub @ X @ (push @ A @ M)))))))))).

thf(pushprop_lem0,definition,(pushprop_lem0 := (![P:(term>
$o)]:(![A:term]:(![M:subst]:(?[Q:(term>$o)]:(pushprop_p_and
_p_prime @ A @ M @ P @ Q)))))).

thf(pushprop_lem0_gthm,definition,(pushprop_lem0_gthm :=
(axapp => (axvarcons => (axvarid => (axabs => (axclos =>
(axidl => (axshiftcons => (axassoc => (axmap => (axidr =>
(axvarshift => (axscons => (ulamvar1 =>(ulamvarsh =>
(ulamvarind => (apinj1 => (apinj2 => (laminj => (shinj =>
 (lamnotap => (apnotvar => (lamnotvar => (induction =>
 pushprop_lem0))))))))))))))))))))))))).

thf(pushprop_lem0_lthm,definition,(pushprop_lem0_lthm :=
pushprop_lem0)).

thf(pushprop_lem1v2,definition,(pushprop_lem1v2 := (![P:
```

```
(term>$o)]:(![Q:(term>$o)]:(![A:term]:(![M:subst]:((P @ A)
 => ((pushprop_p_and_p_prime @ A @ M @ P @ Q) =>
 (Q @ one))))))))).

thf(pushprop_lem1v2_gthm,definition,(pushprop_lem1v2_gthm
:= (axapp => (axvarcons => (axvarid => (axabs => (axclos
 => (axidl => (axshiftcons => (axassoc => (axmap => (axidr
  => (axvarshift => (axscons => (ulamvar1 => (ulamvarsh =>
   (ulamvarind => (apinj1 => (apinj2 => (laminj => (shinj =>
(lamnotap => (apnotvar => (lamnotvar => (induction =>
pushprop_lem1v2))))))))))))))))))))))).

thf(pushprop_lem1v2_lthm,definition,(pushprop_lem1v2_lthm :=
 (axvarcons => pushprop_lem1v2))).

thf(pushprop_lem2v2,definition,(pushprop_lem2v2 := (![P:
(term>$o)]:(![Q:(term>$o)]:(![A:term]:(![M:subst]:((pushprop
_p_and_p_prime @ A @ M @ P @ Q) => ((![B:term]:((var @ B)
=>(P @ (sub @ B @ M)))) => (![C:term]:((var @ C) => ((Q @ C)
 => (Q @ (sub @ C @ sh))))))))))))).

thf(pushprop_lem2v2_gthm,definition,(pushprop_lem2v2_gthm
:= (axapp => (axvarcons => (axvarid => (axabs => (axclos
 => (axidl => (axshiftcons => (axassoc => (axmap => (axidr
 => (axvarshift => (axscons => (ulamvar1 => (ulamvarsh =>
(ulamvarind => (apinj1 => (apinj2 => (laminj => (shinj =>
(lamnotap => (apnotvar => (lamnotvar => (induction =>
pushprop_lem2v2))))))))))))))))))))))).

thf(pushprop_lem2v2_lthm,definition,(pushprop_lem2v2_lthm
:= (axclos =>
 (axshiftcons => pushprop_lem2v2)))).

thf(pushprop_lem3v2,definition,(pushprop_lem3v2 := (![P:
(term>$o)]:(![Q:(term>$o)]:(![A:term]:(![M:subst]:((pushprop
_p_and_p_prime @ A @ M @ P @ Q) => ((![B:term]:((var @ B)
=> (Q @ B))) => (![B:term]:((var @ B) => (P @ (sub @ B @
(push @ A @ M))))))))))))).

thf(pushprop_lem3v2_lthm,definition,(pushprop_lem3v2_lthm
:= pushprop_lem3v2)).
```

```
thf(pushprop,definition,(pushprop := (![P:(term>$o)]:(![A:
term]:(![M:subst]:((![B:term]:((var @ B) => (P @ (sub @ B
@ M)))) => ((P @ A) => (![B:term]:((var @ B) => (P @ (sub
@ B @ (push @ A @ M)))))))))))).

thf(pushprop_gthm,definition,(pushprop_gthm := (axapp =>
(axvarcons => (axvarid => (axabs => (axclos => (axidl =>
(axshiftcons => (axassoc => (axmap => (axidr => (axvarshift
 => (axscons => (ulamvar1 => (ulamvarsh => (ulamvarind =>
 (apinj1 => (apinj2 => (laminj => (shinj => (lamnotap =>
(apnotvar => (lamnotvar => (induction => pushprop)))))))))
))))))))))))))).


thf(pushprop_lthm,definition,(pushprop_lthm := (pushprop_
lem0 => (pushprop_lem1v2 => (pushprop_lem2v2 => (pushprop
_lem3v2 => (ulamvarind => pushprop)))))))).

thf(pushprop_lthm,definition,(pushprop_lthm := (pushprop
_lem0 => (ulamvar1 => (axvarcons => (axclos => (axshiftcons
 => (ulamvarind => pushprop)))))))).

thf(induction2lem,definition,(induction2lem := (![P:(term
>$o)]:((![A:term]:(![B:term]:((P @ A) => ((P @ B) => (P @
(ap @ A @ B))))))=> ((![A:term]:((![B:term]:((P @ B) =>
(P @ (sub @ A @ (push @ B @ id))))) => (P @ (lam @ A))))
 => (![A:term]:(![M:subst]:((![B:term]:((var @ B) => (P @
 (sub @ B @ M)))) => (P @ (sub @ A @ M)))))))))).

thf(induction2lem_gthm,definition,(induction2lem_gthm :=
(axapp => (axvarcons => (axvarid => (axabs => (axclos =>
 (axidl => (axshiftcons => (axassoc => (axmap => (axidr
 => (axvarshift => (axscons => (ulamvar1 => (ulamvarsh =>
  (ulamvarind => (apinj1 => (apinj2 => (laminj => (shinj
 => (lamnotap => (apnotvar => (lamnotvar => (induction =>
 (pushprop =>induction2lem))))))))))))))))))))))))))).

thf(induction2lem_lthm,definition,(induction2lem_lthm :=
(axapp => (axvarcons => (axabs => (axclos => (axshiftcons
=> (axassoc => (axmap=> (axidr => (induction => (pushprop
```

```
=> induction2lem))))))))))).

thf(induction2,definition,(induction2 := (![P:(term>$o)]:
((![A:term]:((var @ A) => (P @ A))) => ((![A:term]:(![B:
term]:((P @ A) => ((P @ B)=> (P @ (ap @ A @ B)))))) => (
(![A:term]:((![B:term]:((P @ B) => (P @ (sub @ A @ (push
 @ B @ id))))) => (P @ (lam @ A)))) => (![A:term]:(P @ A)
 )))))).

thf(induction2_gthm,definition,(induction2_gthm := (axapp
=> (axvarcons => (axvarid => (axabs => (axclos => (axidl =>
 (axshiftcons => (axassoc => (axmap => (axidr => (axvarshift
  => (axscons => (ulamvar1 => (ulamvarsh => (ulamvarind =>
  (apinj1 => (apinj2 => (laminj => (shinj=> (lamnotap =>
  (apnotvar => (lamnotvar => (induction => (pushprop =>
 (induction2lem => induction2))))))))))))))))))))))))).

thf(induction2_lthm,definition,(induction2_lthm := (axvarid
=> (induction2lem => induction2)))).

thf(substmonoid,definition,(substmonoid := (((![M:subst]:
(![N:subst]:(![K:subst]:((comp @ (comp @ M @ N) @ K) =
(comp @ M @ (comp @ N @ K)))))) & (![M:subst]:((comp @
id @ M) = M))) & (![M:subst]:((comp @ M @ id) = M))))).

thf(substmonoid_gthm,definition,(substmonoid_gthm :=
(axapp =>(axvarcons => (axvarid => (axabs => (axclos =>
 (axidl => (axshiftcons => (axassoc => (axmap => (axidr
  => (axvarshift => (axscons => (ulamvar1 => (ulamvarsh
   => (ulamvarind => (apinj1 => (apinj2 => (laminj =>
    (shinj => (lamnotap => (apnotvar => (lamnotvar =>
 (induction => (pushprop => (induction2lem => (induction2
 => substmonoid))))))))))))))))))))))))).

thf(substmonoid_lthm,definition,(substmonoid_lthm :=
(axidl => (axassoc => (axidr => substmonoid))))).

thf(termmset,definition,(termmset := ((![A:term]:(
![M:subst]:(![N:subst]:((sub @ (sub @ A @ M) @ N) = (sub
 @ A @ (comp @ M @ N)))))) & (![A:term]:((sub @ A @ id)
  = A))))).
```

```
thf(termmset_gthm,definition,(termmset_gthm := (axapp =>
(axvarcons => (axvarid => (axabs => (axclos => (axidl =>
(axshiftcons => (axassoc => (axmap => (axidr => (axvarshift
 => (axscons => (ulamvar1 => (ulamvarsh => (ulamvarind =>
 (apinj1 => (apinj2 => (laminj => (shinj => (lamnotap =>
 (apnotvar => (lamnotvar => (induction => (pushprop =>
  (induction2lem => (induction2 => (substmonoid =>
  termmset))))))))))))))))))))))))))).

thf(termmset_lthm,definition,(termmset_lthm := (axvarid =>
(axclos => termmset)))).

thf(hoasap,definition,(hoasap := (^[M:subst]:(^[A:term]:
(^[N:subst]:(^[B:term]:(ap @ (sub @ A @ N) @ B)))))).

thf(hoaslam,definition,(hoaslam := (^[M:subst]:(^[F:
(subst>(term>term))]:(lam @ (F @ sh @ one)))))).

thf(hoasvar,definition,(hoasvar := (^[M:subst]:
(^[A:term]:(^[N:subst]:(var @ (sub @ A @ N)))))).

thf(hoasapinj1,definition,(hoasapinj1 := (![A:term]
:(![B:term]:(![C:term]:(![D:term]:(((hoasap @ id @ A @
id @ C) = (hoasap @ id @ B @ id @ D)) => (A = B))))))).

thf(hoasapinj1_gthm,definition,(hoasapinj1_gthm := (axapp
=>(axvarcons => (axvarid => (axabs => (axclos => (axidl =>
(axshiftcons => (axassoc => (axmap => (axidr => (axvarshift
=>(axscons => (ulamvar1 => (ulamvarsh => (ulamvarind =>
(apinj1 => (apinj2 => (laminj => (shinj => (lamnotap =>
(apnotvar => (lamnotvar => (induction => (pushprop =>
(induction2lem => (induction2 => (substmonoid => (termmset
=> hoasapinj1))))))))))))))))))))))))))).

thf(hoasapinj1_lthm,definition,(hoasapinj1_lthm := (axvarid
=>(apinj1 => hoasapinj1)))).

thf(hoasapinj2,definition,(hoasapinj2 := (![A:term]:(
![B:term]:(![C:term]:(![D:term]:(((hoasap @ id @ A @ id @
C) = (hoasap @ id @ B @ id @ D)) => (C = D))))))).
```

```
thf(hoasapinj2_gthm,definition,(hoasapinj2_gthm := (axapp =>
(axvarcons => (axvarid => (axabs => (axclos => (axidl =>
(axshiftcons => (axassoc => (axmap => (axidr => (axvarshift
=> (axscons => (ulamvar1 => (ulamvarsh => (ulamvarind =>
(apinj1=> (apinj2 => (laminj => (shinj => (lamnotap =>
(apnotvar => (lamnotvar => (induction => (pushprop =>
(induction2lem => (induction2 => (substmonoid => (termmset
 => (hoasapinj1 => hoasapinj2))))))))))))))))))))))))).

thf(hoasapinj2_lthm,definition,(hoasapinj2_lthm := (apinj2 =>
hoasapinj2))).

thf(hoaslaminj,definition,(hoaslaminj := (![F:(subst>(term>
term))]:((![M:subst]:(![A:term]:(![N:subst]:((sub @ (F @ M
@ A) @ N) = (F @(comp @ M @ N) @ (sub @ A @ N)))))) => (
![G:(subst>(term>term))]:((![M:subst]:(![A:term]:(![N:subst]
:((sub @ (G @ M @ A) @ N) = (G @ (comp @ M @ N) @ (sub @ A
@ N)))))) => (((hoaslam @ id @ (^[M:subst]:(^[A:term]:(F @
 M @ A)))) = (hoaslam @ id @ (^[M:subst]:(^[A:term]:(G @ M
 @ A))))) => (![M:subst]:(![A:term]:((F @ M @ A)= (G @ M @
 A)))))))))).

thf(hoaslaminj_gthm,definition,(hoaslaminj_gthm := (axapp =>
(axvarcons => (axvarid => (axabs => (axclos => (axidl =>
(axshiftcons => (axassoc => (axmap => (axidr => (axvarshift =>
 (axscons => (ulamvar1 => (ulamvarsh => (ulamvarind => (apinj1
 => (apinj2 => (laminj => (shinj => (lamnotap => (apnotvar =>
 (lamnotvar => (induction => (pushprop => (induction2lem =>
 (induction2 => (substmonoid => (termmset => (hoasapinj1 =>
 (hoasapinj2 => hoaslaminj))))))))))))))))))))))))))).

thf(hoaslaminj_lthm,definition,(hoaslaminj_lthm := (axvarcons
=> (axshiftcons => (laminj => hoaslaminj))))).

thf(hoaslamnotap,definition,(hoaslamnotap := (![F:(subst>
(term>term))]:((![M:subst]:(![A:term]:(![N:subst]:((sub @
(F @ M @ A) @ N) = (F @ (comp @ M @ N) @ (sub @ A @ N))))))
 => (![A:term]:(![B:term]:(~ ((hoaslam @ id @ (^[M:subst]:
(^[C:term]:(F @ M @ C)))) = (hoasap @ id @ A @ id @ B)))))))).
```

```
thf(hoaslamnotap_gthm,definition,(hoaslamnotap_gthm :=
(axapp=> (axvarcons => (axvarid => (axabs => (axclos =>
(axidl =>(axshiftcons => (axassoc => (axmap => (axidr =>
(axvarshift => (axscons => (ulamvar1 => (ulamvarsh =>
(ulamvarind => (apinj1 => (apinj2 => (laminj => (shinj
=> (lamnotap => (apnotvar => (lamnotvar => (induction =>
(pushprop => (induction2lem => (induction2 => (substmonoid
 => (termmset => (hoasapinj1 => (hoasapinj2 => (hoaslaminj
 => hoaslamnotap))))))))))))))))))))))))))).

thf(hoaslamnotap_lthm,definition,(hoaslamnotap_lthm :=
(lamnotap => hoaslamnotap))).

thf(hoaslamnotvar,definition,(hoaslamnotvar := (![F:
(subst>(term>term))]:((![M:subst]:(![A:term]:(![N:subst]:
((sub @ (F @ M @ A) @ N) = (F @ (comp @ M @ N) @ (sub @
 A @ N)))))) => (~ (hoasvar @ id @ (hoaslam @ id @
 (^[M:subst]:(^[A:term]:(F @ M @ A)))) @ id)))))).

thf(hoaslamnotvar_gthm,definition,(hoaslamnotvar_gthm :=
(axapp => (axvarcons => (axvarid => (axabs => (axclos =>
(axidl => (axshiftcons => (axassoc => (axmap => (axidr =>
(axvarshift => (axscons => (ulamvar1 => (ulamvarsh =>
(ulamvarind => (apinj1 => (apinj2 => (laminj => (shinj =>
(lamnotap => (apnotvar => (lamnotvar => (induction =>
(pushprop => (induction2lem => (induction2 => (substmonoid
 => (termmset => (hoasapinj1 => (hoasapinj2 => (hoaslaminj
  => (hoaslamnotap => hoaslamnotvar))))))))))))))))))))))
  ))))))))))))))).

thf(hoaslamnotvar_lthm,definition,(hoaslamnotvar_lthm :=
(axvarid => (lamnotvar => hoaslamnotvar)))).

thf(hoasapnotvar,definition,(hoasapnotvar := (![A:term]:
(![B:term]:(~ (hoasvar @ id @ (hoasap @ id @ A @ id @ B)
@ id)))))).

thf(hoasapnotvar_gthm,definition,(hoasapnotvar_gthm :=
(axapp => (axvarcons => (axvarid => (axabs => (axclos =>
 (axidl => (axshiftcons => (axassoc => (axmap => (axidr
  => (axvarshift => (axscons => (ulamvar1 => (ulamvarsh
```

```
  => (ulamvarind => (apinj1 => (apinj2 => (laminj =>
  (shinj => (lamnotap => (apnotvar => (lamnotvar =>
  (induction => (pushprop => (induction2lem =>
  (induction2 => (substmonoid => (termmset => (hoasapinj1
   => (hoasapinj2 => (hoaslaminj => (hoaslamnotap =>
   (hoaslamnotvar => hoasapnotvar))))))))))))))))))
   ))))))))))))).

thf(hoasapnotvar_lthm,definition,(hoasapnotvar_lthm :=
(axvarid => (apnotvar => hoasapnotvar))).

thf(hoasinduction_p_and_p_prime,definition,
(hoasinduction_p_and_p_prime := (^[P:(subst>(term>(subst>
$o))]:(^[Q:(term>$o)]:![X:term]:((Q @ X) <=> (P @ id @ X
@ id))))))).

thf(hoasinduction_lem0,definition,(hoasinduction_lem0 :=

(![P:(subst>(term>(subst>$o)))]:(?[Q:(term>$o)]:
(hoasinduction_p_and_p_prime @ P @ Q))))).

thf(hoasinduction_lem0_lthm,definition,(hoasinduction_lem0
_lthm := hoasinduction_lem0)).

thf(hoasinduction_lem1v2,definition,(hoasinduction_lem1v2 :=
(![P:(subst>(term>(subst>$o)))]:(![Q:(term>$o)]:((![M:subst]:
(![A:term]:(![N:subst]:(![K:subst]:((P @ M @ A @ (comp @ K @
 N)) => (P @ (comp @ M @ K) @ (sub @ A @ K) @ N)))))) =>
 ((![M:subst]:(![A:term]:(![N:subst]:(![K:subst]:((P @ (comp
  @ M @ K) @ (sub @ A @ K) @ N) => (P @ M @ A @ (comp @ K @
   N))))))) => ((![A:term]:((hoasvar @ id @ A @ id) => (P @
   id @ A @ id))) => ((hoasinduction_p_and_p_prime @ P @ Q)
    => ![A:term]:((var @ A) => (Q @ A)))))))))).

thf(hoasinduction_lem1v2_gthm,definition,(hoasinduction_
lem1v2_gthm := (axapp => (axvarcons => (axvarid => (axabs
 => (axclos => (axidl => (axshiftcons => (axassoc =>
 (axmap => (axidr =>(axvarshift => (axscons =>(ulamvar1 =>
(ulamvarsh => (ulamvarind => (apinj1 => (apinj2 =>
(laminj => (shinj =>(lamnotap => (apnotvar => (lamnotvar
=>(induction => (pushprop => (induction2lem => (induction2
```

```
 => (substmonoid  => (termmset => (hoasapinj1 => (hoasapinj2
  => (hoaslaminj=> (hoaslamnotap => (hoaslamnotvar =>
  (hoasapnotvar => hoasinduction_lem1v2))))))))))))))))
  ))))))))))))))))).

thf(hoasinduction_lem2v2,definition,(hoasinduction_lem2v2 :=
 (![P:(subst>(term>(subst>$o)))]:(![Q:(term>$o)]:((![M:subst]:
(![A:term]:(![N:subst]:(![K:subst]:((P @ M @ A @ (comp @ K @
 N)) => (P @ (comp @ M @ K) @ (sub @ A @ K) @ N)))))) =>
 ((![M:subst]:(![A:term]:(![N:subst]:(![K:subst]:((P @ (comp
  @ M @ K) @ (sub @ A @ K) @ N) => (P @ M @ A @ (comp @ K @
  N)))))))) =>(![A:term]:(![B:term]:((P @ id @ A @ id) =>
  ((P @ id @ B @ id) => (P @ id @ (hoasap @ id @ A @ id @ B)
   @ id))))) =>((hoasinduction_p_and_p_prime @ P @ Q) =>
   (![A:term]:(![B:term]:((Q @ A) => ((Q @ B) => (Q @ (ap
    @ A @ B)))))))))))).

thf(hoasinduction_lem2v2_gthm,definition,(hoasinduction_
lem2v2_gthm := (axapp => (axvarcons => (axvarid => (axabs
=> (axclos => (axidl => (axshiftcons => (axassoc => (axmap
 => (axidr => (axvarshift => (axscons =>(ulamvar1 =>
(ulamvarsh => (ulamvarind => (apinj1 => (apinj2 => (laminj
=> (shinj => (lamnotap => (apnotvar => (lamnotvar =>
(induction => (pushprop => (induction2lem => (induction2
 => (substmonoid => (termmset => (hoasapinj1 => (hoasapinj2
 => (hoaslaminj => (hoaslamnotap => (hoaslamnotvar =>
 (hoasapnotvar =>hoasinduction_lem2v2))))))))))))))))))
 ))))))))))))))).

thf(hoasinduction_lem3v2_f,definition,(hoasinduction_lem3v2_f
:= (![B:term]:(?[F:(subst>(term>term))]:(![A:term]:(![M:subst]
:((F @ M @ A) = (sub @ B @ (push @ A @ M))))))))).

thf(hoasinduction_lem3v2_f_lthm,definition,(hoasinduction_
lem3v2_f_lthm := hoasinduction_lem3v2_f)).

thf(hoasinduction_lem3v2,definition,(hoasinduction_lem3v2 :=
(![P:(subst>(term>(subst>$o)))]:(![Q:(term>$o)]:((![M:subst]:
(![A:term]:(![N:subst]:(![K:subst]:((P @ M @ A @ (comp @ K @
 N))=> (P @ (comp @ M @ K) @ (sub @ A @ K) @ N)))))) => ((
 ![M:subst]:(![A:term]:(![N:subst]:(![K:subst]:((P @ (comp
```

```
  @ M @ K) @ (sub @ A @ K) @ N) => (P @ M @ A @ (comp @ K @
  N))))))) => ((![F:(subst>(term>term))]:((![M:subst]:(
  ![A:term]:(![N:subst]:((sub @ (F @ M @ A) @ N) = (F @
  (comp @ M @ N) @ (sub @ A @ N)))))) => ((![A:term]:((P @
  id @ A @ id) => (P @ id @ (F @ id @ A) @ id))) => (P
  @ id @ (hoaslam @ id @ (^[M:subst]:(^[A:term]:(F @ M @ A)
  ))) @ id)))) => ((hoasinduction_p_and_p_prime @ P @ Q ) =>
  (![A:term]:((![B:term]:((Q @ B) => (Q @ (sub @ A @ (push @
   B @ id)))) => (Q @ (lam @ A)))))))))))).

thf(hoasinduction_lem3v2_gthm,definition,(hoasinduction_
lem3v2_gthm := (axapp => (axvarcons => (axvarid => (axabs
=> (axclos => (axidl => (axshiftcons => (axassoc => (axmap
 => (axidr => (axvarshift => (axscons =>(ulamvar1 =>
 (ulamvarsh => (ulamvarind => (apinj1 => (apinj2 => (laminj
  => (shinj => (lamnotap => (apnotvar => (lamnotvar =>
 (induction => (pushprop => (induction2lem => (induction2
 => (substmonoid => (termmset =>hoasapinj1 => (hoasapinj2
 => (hoaslaminj => (hoaslamnotap=> (hoaslamnotvar =>
 (hoasapnotvar => hoasinduction_lem3v2))))))))))))))))
 )))))))))))))))).

thf(hoasinduction_lem3v2_lthm,definition,(hoasinduction_
lem3v2_lthm := (axvarid => (axvarshift => (hoasinduction_
lem3aa =>hoasinduction_lem3v2))))).

thf(hoasinduction_lem3v2_lthm,definition,(hoasinduction_
lem3v2_lthm := (axvarid => (axvarshift => (axclos => (axmap
 => hoasinduction_lem3v2))))).

thf(hoasinduction_lem3v2a,definition,(hoasinduction_lem3v2a
:= (![P:(subst>(term>(subst>$o)))]:(![Q:(term>$o)]:((![F:
(subst>(term>term))]:((![M:subst]:(![A:term]:(![N:subst]:
((sub @ (F @ M @ A) @ N) = (F @ (comp @ M @ N) @ (sub @ A
 @ N)))))) => ((![A:term]:((P @ id @ A @ id) => (P @ id @
  (F @ id @ A) @ id))) => (P @ id @ (hoaslam @ id @ (^[M:
subst]:(^[A:term]:(F @ M @ A)))) @ id)))) => ((hoasinduction
_p_and_p_prime @P @ Q ) => (![A:term]:((![B:term]:((Q @ B)
=> (Q @ (sub @ A @ (push @ B @ id)))) => (Q @ (lam @ A)))))
)))))).
```

```
thf(hoasinduction_lem3v2a_lthm,definition,(hoasinduction_
lem3v2a_lthm:= (hoasinduction_lem3v2_f => (axvarid =>
(axvarshift => (axclos => (axmap => hoasinduction_lem3v2a))
))))).

thf(hoasinduction_lem1,definition,(hoasinduction_lem1 :=
(![P:(subst>(term>(subst>$o)))]:((![M:subst]:(![A:term]:
(![N:subst]:(![K:subst]:((P @ M @ A @ (comp @ K @ N)) =>
(P @ (comp @ M @ K) @ (sub @ A @ K) @ N)))))) => ((
![M:subst]:(![A:term]:(![N:subst]:(![K:subst]:((P @
(comp @ M @ K) @ (sub @ A @ K) @ N) => (P @ M @ A @
(comp @ K @ N)))))) => ((![A:term]:((hoasvar @ id @
A @ id) => (P @ id @ A @ id))) => ![A:term]:((var @ A)
 => (P @ id @ A @ id)))))))).

thf(hoasinduction_lem1_gthm,definition,(hoasinduction
_lem1_gthm := (axapp => (axvarcons => (axvarid => (axabs
=> (axclos => (axidl => (axshiftcons => (axassoc => (axmap
 => (axidr => (axvarshift => (axscons => (ulamvar1 =>
 (ulamvarsh => (ulamvarind => (apinj1 => (apinj2 => (laminj
  => (shinj => (lamnotap => (apnotvar => (lamnotvar =>
  (induction => (pushprop => (induction2lem => (induction2
   => (substmonoid => (termmset => (hoasapinj1 =>
   (hoasapinj2 => (hoaslaminj => (hoaslamnotap =>
   (hoaslamnotvar => (hoasapnotvar => hoasinduction_lem1)))
   ))))))))))))))))))))))))))).

thf(hoasinduction_lem1_lthm,definition,(hoasinduction_lem1
_lthm:= (axapp => (axvarcons => (axvarid => (axabs => (axclos
 => (axidl => (axshiftcons => (axassoc => (axmap => (axidr =>
 (axvarshift => (axscons =>(ulamvar1 => (ulamvarsh =>
 (ulamvarind => (apinj1 => (apinj2 => (laminj => (shinj =>
 (lamnotap => (apnotvar => (lamnotvar => (induction =>
 (pushprop => (induction2lem => (induction2 => (substmonoid
 => (termmset => (hoasapinj1 => (hoasapinj2 => (hoaslaminj =>
  (hoaslamnotap => (hoaslamnotvar => (hoasapnotvar =>
  hoasinduction_lem1))))))))))))))))))))))))))))).

thf(hoasinduction_lem2,definition,(hoasinduction_lem2 :=
(![P:(subst>(term>(subst>$o)))]:((![M:subst]:(![A:term]:
(![N:subst]:(![K:subst]:((P @ M @ A @ (comp @ K @ N)) =>
```

```
(P @ (comp @ M @ K) @ (sub @ A @ K) @ N)))))) => ((![M:subst]
:(![A:term]:(![N:subst]:(![K:subst]:((P @ (comp @ M @ K) @
(sub @ A @ K) @ N) => (P @ M @ A @ (comp @ K @ N)))))) =>
  (![A:term]:(![B:term]:((P @ id @ A @ id) => ((P @ id @ B
   @ id) => (P @ id @ (hoasap @ id @ A @ id @ B) @ id)))))
 =>
  (![A:term]:(![B:term]:((P @ id @ A @ id) => ((P @ id @ B @
  id) => (P @ id @ (ap @ A @ B) @ id))))))))))).

thf(hoasinduction_lem2_gthm,definition,(hoasinduction_lem2_gthm
:= (axapp => (axvarcons => (axvarid => (axabs => (axclos =>
 (axidl => (axshiftcons => (axassoc => (axmap => (axidr =>
  (axvarshift => (axscons => (ulamvar1 => (ulamvarsh =>
  (ulamvarind => (apinj1 => (apinj2 => (laminj => (shinj =>
  (lamnotap => (apnotvar => (lamnotvar => (induction =>
  (pushprop => (induction2lem => (induction2 => (substmonoid
=> (termmset => (hoasapinj1 => (hoasapinj2 => (hoaslaminj =>
(hoaslamnotap => (hoaslamnotvar => (hoasapnotvar =>
     hoasinduction_lem2))))))))))))))))))))))))))))).

thf(hoasinduction_lem2_lthm,definition,(hoasinduction_lem2_lthm
:= (axapp => (axvarcons => (axvarid => (axabs => (axclos =>
(axidl => (axshiftcons => (axassoc => (axmap => (axidr =>
(axvarshift => (axscons =>(ulamvar1 => (ulamvarsh => (ulamvarind
 => (apinj1 => (apinj2 => (laminj => (shinj => (lamnotap =>
  (apnotvar => (lamnotvar => (induction => (pushprop =>
  (induction2lem => (induction2 => (substmonoid => (termmset
  => (hoasapinj1 => (hoasapinj2 => (hoaslaminj =>
  (hoaslamnotap => (hoaslamnotvar => (hoasapnotvar =>
  hoasinduction_lem2))))))))))))))))))))))))))).

thf(hoasinduction_lem3aa,definition,(hoasinduction_lem3aa
:= (![P:(subst>(term>(subst>$o))]:((![F:(subst>(term>
term))]:((![M:subst]:(![A:term]:(![N:subst]:((sub @ (F @
M @ A) @ N) = (F @ (comp @ M @ N) @ ( sub@ A @ N)))))) =>
 ((![A:term]:((P @ id @ A @ id) => (P @ id @ (F @ id @ A)
  @ id))) => (P @ id @ (hoaslam @ id @ (^[M:subst]:(^
  [A:term]:(F @ M @ A)))) @ id))))  =>  (![A:term]:((
  ![B:term]:((P @ id @ B @ id) =>(P @ id @ (sub @ A @
  (push @ B @ id)) @ id))) =>(P @ id @ (lam @ (sub @ A
  @ (push @ one @ sh)))@ id)))))))).
```

```
thf(hoasinduction_lem3aa_lthm,definition,(hoasinduction_lem3aa
_lthm:= (axclos => (axmap => hoasinduction_lem3aa)))).

thf(hoasinduction_lem3aaa,definition,(hoasinduction_lem3aaa
:= (![P: (subst>(term>(subst>$o)))]:((![F:(subst>(term>term))]
:(?[C:term]:(![M:subst]:(![A:term]:(![N:subst]:(((sub @ (F @
M @ A) @ N) = (sub @ (sub @C @ (push @ A @ M)) @ N)) & ((sub
@ C @ (push @ (sub @ A @ N) @ (comp @ M @ N))) = (F @ (comp @
 M @ N) @ (sub@ A @ N)))))))) => ((![A:term]:((P @ id @ A @ id)
  => (P @ id @ (F @ id @ A) @ id))) => (P @ id @ (hoaslam @ id
  @ (^[M:subst]:(^[A:term]:(F @ M @ A)))) @ id)))) =>
  (![A:term]:((![B:term]:((P @ id @ B @ id) =>(P @ id @ (sub
  @ A @  (push @ B @ id)) @ id))) =>(P @ id @ (lam @ (sub @
  A @ (push @ one @ sh)))@ id))))))).

thf(hoasinduction_lem3,definition,(hoasinduction_lem3 :=
(![P:(subst>(term>(subst>$o)))]:((![M:subst]:(![A:term]:
(![N:subst]:(![K:subst]:((P @ M @ A @ (comp @ K @ N)) =>
(P @ (comp @ M @ K) @ (sub @ A @ K) @ N)))))) => ((![M:subst]
:(![A:term]:(![N:subst]:(![K:subst]:((P @ (comp @ M @ K) @
(sub @ A @ K) @ N) => (P @ M @ A @ (comp @ K @ N))))))) =>
((![F:(subst>(term>term))]:((![M:subst]:(![A:term]:
(![N:subst]:((sub @ (F @ M @ A) @ N) = (F @ (comp @ M @ N) @
 (sub @ A @ N)))))) => ((![A:term]:((P @ id @ A @ id) =>
 (P @ id @ (F @ id @ A) @ id))) => (P @ id @ (hoaslam @ id @
  (^[M:subst]:(^[A:term]:(F @ M @ A)))) @ id))))  =>
  (![A:term]:((![B:term]:((P @ id @ B @ id) => (P @ id @
  (sub @ A @ (push @ B @ id)) @ id))) =>(P @ id @ (lam @ A)
   @ id))))))))).

thf(hoasinduction_lem3_gthm,definition,(hoasinduction_lem3_
gthm := (axapp => (axvarcons => (axvarid => (axabs => (axclos
=> (axidl => (axshiftcons => (axassoc => (axmap => (axidr =>
(axvarshift => (axscons => (ulamvar1 => (ulamvarsh =>
(ulamvarind => (apinj1 => (apinj2 => (laminj => (shinj =>
 (lamnotap => (apnotvar => (lamnotvar => (induction =>
 (pushprop => (induction2lem => (induction2 => (substmonoid
  => (termmset => (hoasapinj1 => (hoasapinj2 => (hoaslaminj
  => (hoaslamnotap => (hoaslamnotvar => (hoasapnotvar =>
  (hoasinduction_lem1 => (hoasinduction_lem2 =>
```

```
     hoasinduction_lem3)))))))))))))))))))))))))))))))))))))).

thf(hoasinduction_lem3_lthm,definition,(hoasinduction_lem3_lthm
:= (axvarid => (axvarshift => (hoasinduction_lem3aa =>
hoasinduction_lem3))))).

thf(hoasinduction_lem3a,definition,(hoasinduction_lem3a :=
(![P:(subst>(term>(subst>$o))):((![F:(subst>(term>term))]:
((![M:subst]:(![A:term]:(![N:subst]:((sub @ (F @ M @ A) @ N)
 = (F @ (comp @ M @ N) @ (sub @ A @ N)))))) => ((![A:term]:
 ((P @ id @ A @ id) => (P @ id @ (F @ id @ A) @ id))) =>
 (P @ id @ (hoaslam @ id @ (^[M:subst]:(^[A:term]:(F @ M @
 A)))) @ id))))  =>  (![A:term]:((![B:term]:((P @ id @ B @
 id) => (P @ id @ (sub @ A @ (push @ B @ id)) @ id))) =>
 (P @ id @ (lam @ A) @ id)))))))).

thf(hoasinduction_lem3a_gthm,definition,(hoasinduction_lem3a
_gthm := (axapp => (axvarcons => (axvarid => (axabs => (axclos
=> (axidl => (axshiftcons => (axassoc => (axmap => (axidr =>
(axvarshift => (axscons => (ulamvar1 => (ulamvarsh =>
(ulamvarind => (apinj1 => (apinj2 => (laminj => (shinj =>
 (lamnotap => (apnotvar => (lamnotvar => (induction =>
 (pushprop => (induction2lem => (induction2 => (substmonoid
 => (termmset => (hoasapinj1 => (hoasapinj2 => (hoaslaminj
 => (hoaslamnotap => (hoaslamnotvar => (hoasapnotvar =>
 (hoasinduction_lem1 => (hoasinduction_lem2 =>
     hoasinduction_lem3a)))))))))))))))))))))))))))))))))))))).

thf(hoasinduction_lem3a_lthm,definition,(hoasinduction_lem3a
_lthm := (axvarid => (axvarshift => (axclos => (axmap =>
hoasinduction_lem3a))))))).

thf(hoasinduction_lem3a_lthm,definition,(hoasinduction_lem3a
_lthm := (axvarid => (axvarshift => (hoasinduction_lem3aa =>
hoasinduction_lem3a))))).

thf(hoasinduction_lem3b,definition,(hoasinduction_lem3b :=
(![B:term]:(?[F:(subst>(term>term))]:(sub @ B @ (push @ one
 @ sh))= (F @ sh @ one))))).

thf(hoasinduction_lem3b_gthm,definition,(hoasinduction_lem3b
```

```
_gthm := (axapp => (axvarcons => (axvarid => (axabs => (axclos
 => (axidl => (axshiftcons => (axassoc => (axmap => (axidr =>
(axvarshift => (axscons => (ulamvar1 => (ulamvarsh =>
(ulamvarind => (apinj1 => (apinj2 => (laminj => (shinj =>
 (lamnotap => (apnotvar => (lamnotvar => (induction =>
 (pushprop => (induction2lem => (induction2 => (substmonoid
 => (termmset => (hoasapinj1 => (hoasapinj2 => (hoaslaminj
 => (hoaslamnotap => (hoaslamnotvar => (hoasapnotvar =>
(hoasinduction_lem1 => (hoasinduction_lem2 => hoasinduction_
lem3b))))))))))))))))))))))))))))))))))))))).

thf(hoasinduction_lem3b_lthm,definition,(hoasinduction_lem3b_
lthm := hoasinduction_lem3b)).

thf(hoasinduction_lem3b,definition,(hoasinduction_lem3b :=
((![F:(subst>(term>term))]:((![M:subst]:(![A:term]:(![N:subst]
:((sub @ (F @ M @ A) @ N) = (F @ (comp @ M @ N) @ (sub @ A @
 N)))))))) => (![B:term]:(?[F:(subst>(term>term))]:(sub @ B
 @ (push @ one @ sh)) = (F @  sh @ one))))).

thf(hoasinduction_lem3b,definition,(hoasinduction_lem3b :=
(![P:(subst>(term>(subst>$o)))]:((![F:(subst>(term>term))]:
((![M:subst]:(![A:term]:(![N:subst]:((sub @ (F @ M @ A) @ N)
 = (F @ (comp @ M @ N) @ (sub @ A @ N)))))) => (![B:term]:
 (?[F:(subst>(term>term))]:(sub @ B @ (push @ one @ sh)) =
 (F @ sh @ one)))))))).

thf(hoasinduction_lem3bb,definition,(hoasinduction_lem3bb
:=(![P:(subst>(term>(subst>$o)))]:((![F:(subst>(term>term))]
:((![M:subst]:(![A:term]:(![N:subst]:((sub @ (F @ M @ A) @ N)
 = (F @ (comp @ M @ N) @ (sub@ A @ N)))))))) =>  (![B:term]
 :(?[F:(subst>(term>term))]:(sub @ B @ (push @ one @ sh)) =
 (F @ sh @ one)))))).

thf(hoasinduction_lem3bb_lthm,definition,(hoasinduction_lem3
bb_lthm :=(axvarid => (axvarshift => (axclos => (axmap =>
 hoasinduction_lem3bb))))))).

thf(hoasinduction,definition,(hoasinduction :=
(![P:(subst>(term>(subst>$o)))]:((![M:subst]:(![A:term]:
(![N:subst]:(![K:subst]:((P @ M @ A @ (comp @ K @ N)) =>
```

```
 (P @ (comp @ M @ K) @ (sub @ A @ K) @ N)))))) =>
 ((![M:subst]:(![A:term]:(![N:subst]:(![K:subst]:((P @
 (comp @ M @ K) @ (sub @ A @ K) @ N) => (P @ M @ A @
 (comp @ K @ N))))))) => ((![A:term]:((hoasvar @ id @
  A @ id) => (P @ id @ A @ id))) => ((![A:term]:(![B:term]
  :((P @ id @ A @ id) => ((P @ id @ B @ id) => (P @ id @
   (hoasap @ id @ A @ id @ B) @ id))))) => ((![F:(subst>
   (term>term))]:((![M:subst]:(![A:term]:(![N:subst]:
   ((sub @ (F @ M @ A) @ N) = (F @ (comp @ M @ N) @ (sub
   @ A @ N)))))) => ((![A:term]:((P @ id @ A @ id) => (P
    @ id @ (F @ id @ A) @ id))) => (P @ id @ (hoaslam @
    id @ (^[M:subst]:(^[A:term]:(F @ M @ A)))) @ id))))
    => (![A:term]:(P @ id @ A @ id))))))))))).

thf(hoasinduction_gthm,definition,(hoasinduction_gthm :=
 (axapp => (axvarcons => (axvarid => (axabs => (axclos =>
  (axidl => (axshiftcons => (axassoc => (axmap => (axidr
   => (axvarshift => (axscons => (ulamvar1 => (ulamvarsh
    => (ulamvarind => (apinj1 => (apinj2 => (laminj =>
    (shinj => (lamnotap => (apnotvar => (lamnotvar =>
    (induction => (pushprop => (induction2lem =>
    (induction2 => (substmonoid => (termmset => (hoasapinj1
     => (hoasapinj2 => (hoaslaminj => (hoaslamnotap =>
     (hoaslamnotvar => (hoasapnotvar => (hoasinduction_lem0
     => (hoasinduction_lem1 => (hoasinduction_lem2 =>
     (hoasinduction_lem3 => hoasinduction))))))))))))))
     ))))))))))))))))))))))).

thf(hoasinduction_lthm,definition,(hoasinduction_lthm :=(
induction2 => (hoasinduction_lem1 => (hoasinduction_lem2
=> (hoasinduction_lem3 => hoasinduction))))))).

thf(hoasinduction_lthm,definition,(hoasinduction_lthm :=
 (induction2 => (hoasinduction_lem1 => (hoasinduction_lem2
  => (hoasinduction_lem3a => hoasinduction))))))).

thf(hoasinduction_lthm,definition,(hoasinduction_lthm :=
(induction2 => (hoasinduction_lem1v2 => (hoasinduction_lem2v2
 => (hoasinduction_lem3v2 => hoasinduction))))))).

thf(hoasinduction_lthm,definition,(hoasinduction_lthm :=
```

```
 (hoasinduction_lem0 => (induction2 => (hoasinduction_lem1v2
  => (hoasinduction_lem2v2 => (hoasinduction_lem3v2 =>
  hoasinduction))))))).

thf(hoasinduction_lthm,definition,(hoasinduction_lthm :=
 (hoasinduction_lem0 => (induction2 => (axvarid =>
 (hoasinduction_lem3v2 => hoasinduction))))))).

thf(hoasinduction_lthm,definition,(hoasinduction_lthm :=
(hoasinduction_lem0 => (induction2 => (axvarid =>
(hoasinduction_lem3v2a => hoasinduction))))))).

thf(hoasinduction_no_psi_cond,definition,(hoasinduction_no_
psi_cond := (![P:(subst>(term>(subst>$o))))]:((![A:term]:(
![B:term]:((P @ id @ A @ id) => ((P @ id @ B @ id) => (P @
 id @ (hoasap@ id @ A @ id @ B) @ id))))) => ((![F:(subst>
 (term>term))]:((![M:subst]:(![A:term]:(![N:subst]:((sub @
  (F @ M @ A) @ N)= (F @ (comp @ M @ N) @ (sub @ A @ N))))))
  => ((![A:term]:((P @ id @ A @ id) => (P @ id @ (F @ id @
  A) @ id))) => (P@ id @ (hoaslam @ id @ (^[M:subst]:(^
  [A:term]:(F @ M @ A)))) @ id)))) => (![A:term]:(P @ id @
  A @ id))))))).

thf(hoasinduction_no_psi_cond_lthm,definition,(hoasinduction_
no_psi_cond_lthm:= (hoasinduction_lem0 => (induction2 =>
(axvarid => (hoasinduction_lem3v2a => hoasinduction_no_psi_
cond))))))).
```