A Higher-Order Logical Framework for Reasoning about Programming Languages

Chelsea Battell, Professor Amy Felty (University of Ottawa)

Objective

Mechanize reasoning about programming languages and logics.

Example: May want to prove type soundness of the simply-typed lambda calculus (subject reduction).

i.e. if $\vdash e \Downarrow v$ and $\vdash e : t$, then $\vdash v : t$.

Solution (a):

Encode the object logic (OL) in an existing proof assistant.

Problem:

Many tedious computations for each encoding with binding structures.

For example, performing capture-avoiding substitutions and keeping track of free and bound variables.

Solution (b):

Use higher-order abstract syntax (HOAS) for encoding OL expressions. Implement inference rules in Coq to use built-in induction utilities.

HOAS:
- OL bound variables encoded as reasoning layer bound variables, so no need to implement α-renaming or substitution.
- Inference rules encoded using parametric and hypothetical judgments, so no need for explicit representation of contexts for OL.

Problem:

Some judgments cannot be encoded as inductive types in Coq. Too restrictive.

To ensure termination, the calculus of constructions requires strict positivity for inductive definitions. For example, typing of abstractions, written

$$\frac{\Gamma \vdash E : expr \to expr}{\Gamma \vdash \lambda x.E\ x : expr}\ tp\_abs$$

fails this condition since the type of the argument is an arrow type. This rule cannot be the type of a constructor of an inductive type.

Solution (c):

Add intermediate layer called specification logic (SL).

The SL is a deductive system defined in Coq as an inductive type encapsulating OL judgments, resulting in a two-level system.

Hybrid

- two-level logical framework for reasoning about programming language and logic metatheory
- uses HOAS for encoding OL expressions
- implemented as a library in Coq

Contribution

- implement a new SL to increase the class of OL judgments that Hybrid can reason about

Object Logic (OL)

The layer where the desired syntax and judgments are encoded and properties proven.

For example, we can encode a logic to show a bijection between HOAS and De Bruijn representations of lambda-terms with types `tm` and `dtm` for HOAS terms and De Bruijn terms, respectively, defined by the following signature:

$\texttt{hApp} : \texttt{tm} \to \texttt{tm} \to \texttt{tm}$	HOAS application
$\texttt{hAbs} : (\texttt{tm} \to \texttt{tm}) \to \texttt{tm}$	HOAS abstraction
$\texttt{dApp} : \texttt{dtm} \to \texttt{dtm} \to \texttt{dtm}$	De Bruijn application
$\texttt{dAbs} : \texttt{dtm} \to \texttt{dtm}$	De Bruijn abstraction
$\texttt{dVar} : \mathbb{N} \to \texttt{dtm}$	De Bruijn variable

Our only judgment is a relation stating HOAS term H is in bijection with De Bruijn term D under n abstractions, written $H \equiv_n D$, and is defined by:

$$\frac{\Gamma \vdash H_1 \equiv_n D_1 \quad \Gamma \vdash H_2 \equiv_n D_2}{\Gamma \vdash \texttt{hApp}\ H_1\ H_2 \equiv_n \texttt{dApp}\ D_1\ D_2}\ \texttt{hodb\_app} \qquad \frac{\Gamma, (\forall k, x \equiv_{n+k} \texttt{dVar}\ k) \vdash H \equiv_{n+1} D}{\Gamma \vdash \texttt{hAbs}\ (\lambda x.H) \equiv_n \texttt{dAbs}\ D}\ \texttt{hodb\_abs}$$

Specification Logic (SL)

A new SL based on higher-order hereditary-Harrop formulas is defined as a mutually inductive type corresponding to the rules in figures 1 and 2.

Goal-reduction sequents reduce the goal to atomic. If the atom is an OL judgment, backchain over static assumptions (via `s_bc`), else focus a formula in the context and backchain over dynamic assumptions (by `s_init`).

Structural Properties Proven:

Thin Exchange: $\dfrac{\Gamma_1 \subseteq \Gamma_2 \quad \Gamma_1 \rhd G}{\Gamma_2 \rhd G}$ and $\dfrac{\Gamma_1 \subseteq \Gamma_2 \quad \Gamma_1, [F] \rhd G}{\Gamma_2, [F] \rhd G}$

Contraction, weakening, and exchange are all corollaries of above. Proof is by mutual structural induction over sequent premises.

Cut Elimination: $\dfrac{\Gamma, G_1 \rhd G_2 \quad \Gamma \rhd G_1}{\Gamma \rhd G_2}$ and $\dfrac{\Gamma, G_1, [F] \rhd G_2 \quad \Gamma \rhd G_1}{\Gamma, [F] \rhd G_2}$

Proof of Cut Elimination:

By a nested induction with structural induction on G_1 then mutual induction over the structure of $\Gamma, G_1 \rhd G_2$ and $\Gamma, G_1, [F] \rhd G_2$.

Case: $G_1 = \langle\, a_1 \,\rangle$ for atom a_1.

Subcase: $\Gamma, G_1 \rhd G_2$ derived by `s_init`. Then $G_2 = \langle\, a_2 \,\rangle$ for atom a_2.

1	$\Gamma \rhd \langle\, a_1 \,\rangle$	(case premise)
2	$F \in \Gamma, \langle\, a_1 \,\rangle$	(subcase premise)
3	$\Gamma, \langle\, a_1 \,\rangle, [F] \rhd a_2$	(subcase premise)
4	$\forall \Gamma, \Gamma \rhd \langle\, a_1 \,\rangle \Rightarrow \Gamma, [F] \rhd a_2$	(induction hypothesis)
5	$F = \langle\, a_1 \,\rangle$ *or* $F \in \Gamma$	(inversion 2)
6	$F = \langle\, a_1 \,\rangle$	(assumption)
7	$\Gamma, \langle\, a_1 \,\rangle, [\langle\, a_1 \,\rangle] \rhd a_2$	(rewrite 6 in 3)
8	$a_1 = a_2$	(inversion 7)
9	$\Gamma \rhd \langle\, a_2 \,\rangle$	(rewrite 8 in 1)
10	$F \in \Gamma$	(assumption)
11	$\Gamma, [F] \rhd a_2$	(modus ponens 1 4)
12	$\Gamma \rhd \langle\, a_2 \,\rangle$	(`s_init` 10 11)
13	$\Gamma \rhd \langle\, a_2 \,\rangle$	(or elim 5, 6-9, 10-12)

The remaining 97 subcases follow from assumptions and induction hypotheses, using contraction, weakening, and exchange and some context lemmas.

$$\frac{A :- G \quad \Sigma; \Gamma \rhd G}{\Sigma; \Gamma \rhd \langle\, A \,\rangle}\ \texttt{s\_bc}$$

$$\frac{F \in \Gamma \quad \Sigma; \Gamma, [F] \rhd A}{\Sigma; \Gamma \rhd \langle\, A \,\rangle}\ \texttt{s\_init}$$

$$\frac{}{\Sigma; \Gamma \rhd \top}\ \texttt{s\_tt}$$

$$\frac{\Sigma; \Gamma \rhd G_1 \quad \Sigma; \Gamma \rhd G_2}{\Sigma; \Gamma \rhd G_1 \wedge G_2}\ \texttt{s\_and}$$

$$\frac{\Sigma; \Gamma, G_1 \rhd G_2}{\Sigma; \Gamma \rhd G_1 \to G_2}\ \texttt{s\_imp}$$

$$\frac{x \notin \Sigma \quad \Sigma, (x : \tau); \Gamma \rhd G\, x}{\Sigma; \Gamma \rhd \forall^{\tau} G}\ \texttt{s\_alls}$$

$$\frac{x \notin \Sigma \quad \Sigma, (x : expr); \Gamma \rhd G\, x}{\Sigma; \Gamma \rhd \forall^{expr} G}\ \texttt{s\_all}$$

$$\frac{\Sigma \vdash t : expr \quad \Sigma; \Gamma \rhd G\, t}{\Sigma; \Gamma \rhd \exists^{expr} G}\ \texttt{s\_exists}$$

Figure 1: Goal-Reduction Rules

$$\frac{}{\Sigma; \Gamma, [\langle\, A \,\rangle] \rhd A}\ \texttt{b\_match}$$

$$\frac{\Sigma; \Gamma, [F_1] \rhd A}{\Sigma; \Gamma, [F_1 \wedge F_2] \rhd A}\ \texttt{b\_and1}$$

$$\frac{\Sigma; \Gamma, [F_2] \rhd A}{\Sigma; \Gamma, [F_1 \wedge F_2] \rhd A}\ \texttt{b\_and2}$$

$$\frac{\Sigma; \Gamma \rhd G \quad \Sigma; \Gamma, [F] \rhd A}{\Sigma; \Gamma, [G \to F] \rhd A}\ \texttt{b\_imp}$$

$$\frac{\Sigma \vdash t : expr \quad \Sigma; \Gamma, [F\, t] \rhd A}{\Sigma; \Gamma, [\forall^{expr} F] \rhd A}\ \texttt{b\_alls}$$

$$\frac{\Sigma \vdash t : \tau \quad \Sigma; \Gamma, [F\, t] \rhd A}{\Sigma; \Gamma, [\forall^{\tau} F] \rhd A}\ \texttt{b\_all}$$

$$\frac{x \notin \Sigma \quad \Sigma; \Gamma, (x : expr), [G\, x] \rhd A}{\Sigma; \Gamma, [\exists^{expr} G] \rhd A}\ \texttt{b\_exists}$$

Figure 2: Backchaining Rules

HOAS

The layer managing the higher-order abstract syntax. All computations involving binding operators are handled here. Expressions are defined as De Bruijn representations of lambda-terms. This is described in detail in [1].

Reasoning Logic

The layer on which the whole system is built. Coq is an interactive theorem proving system based on the calculus of inductive constructions (CIC). In CIC, a theorem is a proposition P proven by giving an object of type P.

CIC is strongly normalizing, so computations guaranteed to terminate. Incomplete but consistent, so appropriate for program specification and proving.

Case Study

Correspondence between HOAS and De Bruijn representations of lambda-terms.

This follows the Abella implementation presented in [2] introduced here by the OL example to the left.

Define the function symbols using the Hybrid library [1]:

```
Definition hApp : tm -> tm -> tm :=
 fun (t1 : tm) => fun (t2 : tm) =>
   APP (APP (CON chAPP) t1) t2.

Definition hAbs : (tm -> tm) -> tm :=
 fun (f : tm -> tm) => APP (CON chABS) (lambda f).

Definition dApp : dtm -> dtm -> dtm :=
 fun (d1 : dtm) => fun (d2 : dtm) =>
   APP (APP (CON cdAPP) t1) t2.

Definition dAbs : dtm -> dtm :=
 fun (d : dtm) => APP (CON cdABS) d.

Definition dVar : var -> dtm := VAR Econ.
```

Define the atoms and program clauses:

```
Inductive atm : Set :=
| hodb : tm -> nat -> dtm -> atm.

Inductive prog : atm -> oo -> Prop :=
| hodb_app : forall (t u : tm) (n : nat) (d e : dtm),
   prog (hodb (hApp t u) n (dApp d e))
        (<hodb t n d> & <hodb u n e>)
| hodb_abs : forall (f : tm -> tm) (n : nat) (d : dtm),
   abstr f ->
   prog (hodb (hAbs f) n (dFun d))
    (All (fun x =>
     (Alls (fun m =>
      <hodb x (n+m) (dVar m)>)) ---> <hodb (f x) (n + 1) d>)).
```

Prove theorems about this encoding:

```
Theorem ident_bijection :
 seq0 <hodb (hAbs (fun x => x)) 0 (dAbs (dVar 1))>.
Proof.
eapply s_bc.
 eapply hodb_abs. apply fun_abstr.
 apply s_all; intros. apply s_imp.
 eapply s_init.
  apply elem_self.
  eapply b_alls. eapply b_match.
Qed.
```

Proof of $\rhd \lambda x.x \equiv_0 \lambda.1$

Future and Related Work

Prove that encoding in case study is correct:

$$\frac{\rhd\texttt{hodb}\ H_1\ n\ D \quad \rhd\texttt{hodb}\ H_2\ n\ D}{H_1 = H_2} \quad \text{and} \quad \frac{\rhd\texttt{hodb}\ H\ n\ D_1 \quad \rhd\texttt{hodb}\ H\ n\ D_2}{D_1 = D_2}$$

Construct more general induction principles to be used with dependent sequent predicates or encoded expressions.

Related systems implementing HOAS for programming language metatheory:

- Abella implements the G logic and uses two-level reasoning
- Beluga is based on contextual modal type theory

References

[1] Amy Felty and Alberto Momigliano.
Hybrid: A definitional two-level approach to reasoning with higher-order abstract syntax.
Journal of Automated Reasoning, 48:43–105, 2010.

[2] Yuting Wang, Kaustuv Chaudhuri, Andrew Gacek, and Gopalan Nadathur.
Reasoning about higher-order relational specifications.
In 15th International Symposium on Principles and Practice of Declarative Programming (PPDP), pages 157–168. ACM Press, 2013.

Acknowledgements

Thank you to NSERC for supporting this research and to Amy Felty for her support and guidance.

Contact

Chelsea Battell – cbattell@uottawa.ca
University of Ottawa, Department of Mathematics and Statistics

STUDC CMS SMC

CMS SMC

uOttawa