Multi-Level Meta-Reasoning
with
Higher-Order Abstract Syntax

Alberto Momigliano
Department of Mathematics and Computer Science
University of Leicester

Joint work with Simon Ambler

FOSSACS 08/04/2003

Overview of the Talk

- The big issue: a logical framework which combines higher-order abstract syntax (HOAS), interactive theorem proving and (co)induction.

- Introducing and using Hybrid: encoding Mini-ML.

- Multi-level reasoning with Hybrid.

- Case study: compilation.

Motivation

- EPSRC grant: formal validation of the effect-based optimizations performed by the MLj compiler for SML into Java bytecode [Benton & Kennedy '98,'99], as (contextual) equivalence preserving transformations.

- Compilation into a typed intermediate language with computational effects.

- Need to devise techniques which are truly **scalable** to such languages. This is a function of how *abstract* the encoding can be.

- We propose a new logical framework (Hybrid), which aims to marry HOAS and interactive theorem proving with a traditional HOL-like set-up (Coq too).

- Case study: **compilation** of Mini-ML to the CAM. Prove the compiler sound and complete, with an eye to the extension with computational effect.

- The prototypical model of a (strict) applicative language [Kahn et al'86]. In this talk we are going to use this fragment:

$$\begin{array}{llll} \textit{Types} & \tau & ::= & \mathbf{i} \mid \tau_1 \text{ arrow } \tau_2 \\ \textit{Terms} & e & ::= & x \mid \mathbf{fun}\, x\,.\, e \mid e \mathbin{@} e' \mid \mathbf{fix}\, x\,.\, e \end{array}$$

- $\mathbf{fix}\, x\,.\, E$ (and $\mathbf{fun}\, x\,.$) is a **binder**. It induces an equivalence class under α-renaming, and requires capture avoiding substitutions: encoded how?

- This is the crucial choice, as it will dominate the formal development, in particular the representation of judgments.

- Why reinvent the wheel? Use a (weak) λ-calculus as a meta-language [Church'40]. It has a single binder λ, from which any other one is definable: built-in α-equivalence, well-behaved substitutions and more.

- Big step (strict) evaluation $e \Downarrow v$:

$$\frac{e_1 \Downarrow \textbf{fun}\, x\,.\, e_1' \qquad e_2 \Downarrow v_2 \qquad [v_2/x]e_1' \Downarrow v}{e_1 \mathbin{@} e_2 \Downarrow v}\ \textbf{ev\_app}$$

$$\frac{}{\textbf{fun}\, x\,.\, e \Downarrow \textbf{fun}\, x\,.\, e}\ \textbf{ev\_fun} \qquad \frac{[\textbf{fix}\, x\,.\, e/x]e \Downarrow v}{\textbf{fix}\, x\,.\, e \Downarrow v}\ \textbf{ev\_fix}$$

- Key notion is **substitution** of expressions, cf. $\textbf{ev\_fix}, \textbf{ev\_app}$.

- The latter is a major hassle and so easy to get it wrong.

- Why not use λ-calculus substitution?

- Typing judgments in sequent style, with explicit contexts (lists, partial functions):

$$\frac{\Gamma, x{:}\tau \vdash e : \tau'}{\Gamma \vdash \mathbf{fun}\ x\,.\ e : \tau \ \textsf{arrow}\ \tau'}\ \mathbf{tp\_fun} \qquad \frac{\Gamma, x{:}\tau \vdash e : \tau}{\Gamma \vdash \mathbf{fix}\ x\,.\ e : \tau}\ \mathbf{tp\_fix}$$

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau}\ \mathbf{tp\_var} \qquad \frac{\Gamma \vdash e_1 : \tau' \to \tau \qquad \Gamma \vdash e_2 : \tau'}{\Gamma \vdash e_1 \mathbin{@} e_2 : \tau}\ \mathbf{tp\_app}$$

- However, by using Schroeder-Heister's *higher-order* natural deduction, contexts $(\vdash)$ can be **implicit**,

$$\frac{\dfrac{}{x : \tau}\,u \\ \vdots \\ e : \tau}{\mathbf{fun}\ x\,.\ e : \tau}\ \mathbf{tp\_fun}^{x,u}$$

no rule for **tp_var**

Mini-ML in ELF

```
exp  : type.

fun   : (exp -> exp) -> exp.
@     : exp -> exp -> exp.

tp : type.

arrow : tp -> tp -> tp.

eval : exp -> exp -> type.

ev_fun  : eval (fun E) (fun E).
ev_app  : eval (E1 @ E2) V
            <- eval E1 (fun E1')
            <- eval E2 V2
            <- eval (E1' V2) V.

of : exp -> tp -> type.

tp_fun : of (fun E) (arrow T1 T2)
          <- ({x:exp} of x T1 -> of (E x) T2).
tp_app : of (E1 @ E2) T1
          <- of E1 (arrow T2 T1)
          <- of E2 T2.
```

HOAS: the Good, the Bad and the Ugly

- Object logic functions are mapped to the function space of a meta-language to which α-equivalence is delegated.

- The body of a fix point as a function $E :: exp \Rightarrow exp$. Thus OL substitution = Meta-L β-conversion. One more notion that can be delegated to the meta-L.

- **Implicit** contexts: $x_1 : \tau_1 \ldots x_n : \tau_n \vdash e : \tau$ represented by inferring `(of E T)` from assumptions `of !x1 T1...of !xn Tn`, where `!xi` is an eigenvariable.

- *Weakening* is implicit by the assumption/discharge rules of natural deduction — for **all** judgments in **context**. No distinction between closed and open terms (see applicative bisimulation).

- For execution, you just need a logic programming interpreter (λProlog, Elf).

- But what about **meta-reasoning** over those specifications? Is the sequent `eval (lam E) F ==> F = (lam E)` provable? Is `eval` deterministic?

- In textbooks, e.g. [Winskel], this relation, as well as the data-structures involved, are **inductive**, which can be formalized either set or type-theoretically.

- The catch: HOAS and induction are not best friends. The following is **not** a data-type definition: `datatype??` $expr \ = \ \mathbf{fix}\ (expr \Rightarrow expr) \mid \ldots$
The rule **tp_fun** is not an inductive definition (type).

- This because of **negative** occurrence. Recall: a *rule set* $\mathcal{R}$ is set of pairs $\langle G, a\rangle$ on an universe $\mathcal{U}$, s.t. a is an atom and G a set of atoms. For $\Phi_{\mathcal{R}} : 2^{\mathcal{U}} \to 2^{\mathcal{U}}$, $\Phi_{\mathcal{R}}$ **monotone** define $\Phi_{\mathcal{R}}(A)$ as $\{a \in \mathcal{U} \mid \langle a, G\rangle \in \mathcal{R}, G \subseteq A\}$. $Id(\mathcal{R})$ is the least fix point of this operator.

Isabelle/HOL and Induction

- A simply-typed set theory, where sets are predicates.
- A **definitional** approach: new types are introduced by definition as subsets of existing ones.
- (Co)inductive definitions and data-types are built via the Knaster-Tarski construction, i.e. formalizing the required fragment of (set-theoretic) fixed points theory [Paulson'96].
- Thus, inductive clauses satisfy the **positivity** condition, so that the resulting operator is monotone [Wenzel'98].
- So how do we squeeze HOAS, hypothetical judgments and (co)induction in?

Road-map

- The big issue: a logical framework which combines HOAS, interactive theorem proving and (co)induction.

- $\boxed{\textbf{Introducing and using Hybrid: encoding Mini-ML}}$.

- Multi-level reasoning with Hybrid:

- Case study: compilation.

Hybrid in short

- A package on top of Isabelle/HOL.

- Generalizing [A. Gordon'94], a deep encoding in Isabelle/HOL of a λ-calculus which provides a form of *logical framework* where the syntax of an object level logic can be adequately represented by HOAS.

- Supports *tactical theorem proving*, *induction and co-induction*, but is *definitional*, so consistent *within a classical type theory*.

- Bottom line: 'Hybrid syntax' provides the user with a form of HOAS, but this is syntactic sugar (more properly a **definition**) for an underlying de Bruijn representation.

- This gives access to all the features of Isabelle HOL.

- Start with a de Bruijn datatype for the the untyped λ-calculus with constants:

$$\begin{array}{rcl} CON & :: & con \Rightarrow expr \\ \$\$ & :: & expr \Rightarrow expr \Rightarrow expr \end{array} \qquad \begin{array}{rcl} VAR & :: & nat \Rightarrow expr \\ ABS & :: & expr \Rightarrow expr \end{array}$$

- From that **define** a function $LAM\,.\ :: (expr \Rightarrow expr) \Rightarrow expr$/

- A predicate $proper :: expr \Rightarrow bool$ models the terms one-one with the untyped λ-calculus modulo α-conversion and a predicate $abstr :: (expr \Rightarrow expr) \Rightarrow bool$ to rule out exotic functional term.

- Example: a terms such as $\lambda V_1.\lambda V_2.V_1\ V_2$ is rendered in Hybrid as $LAM\, v_1.(LAM\, v_2.(v_1\ \$\$\ v_2))$ and reduced by rewriting to the de Bruijn term $ABS\ (ABS\ (BND\ 1\ \$\$\ BND\ 0))$.

- Datatype properties of Hybrid expressions **proved** from the properties of the underlying de Bruijn representation:

$$LAM\, v. F\ v \neq E_1 \ \$\$\ E_2$$

$$\llbracket\, abstr\ E;\ abstr\ F\, \rrbracket \Longrightarrow (LAM\, v. E\ v = LAM\, v. F\ v) = (E = F)$$

$$\llbracket\, abstr\ E;\ abstr\ F;\ \forall i. E\ (VAR\ i) = F\ (VAR\ i)\, \rrbracket \Longrightarrow E = F$$

- Several induction principles over proper terms also proved:

$$\llbracket\, proper\ u;$$

$$\bigwedge i.\ P(VAR\ i);$$

$$\bigwedge s\ t.\ \llbracket\, proper\ s;\ P\ s;\ proper\ t;\ P\ t\, \rrbracket \Longrightarrow P(s\ \$\$\ t);$$

$$\bigwedge e.\ \llbracket\, abstr\ e;\ \forall t.\ proper\ t \longrightarrow Pt \longrightarrow P(e\ t)\, \rrbracket \Longrightarrow P(LAM\, x. e\ x)\, \rrbracket$$

$$\Longrightarrow Pu$$

Back to Mini-ML (in Hybrid)

- Encode the syntax on Mini-ML as an object logic in the Hybrid:

$$e_1 @ e_2 \quad == \quad (CON\ cAPP)\ \$\$\ e_1\ \$\$\ e_2 \qquad fun\, x.\, E\ x \quad == \quad (CON\ cABS)\ \$\$\ LAM\, x.\, e\ E$$

- Evaluation will be an *inductive* relation and add *abstraction* annotations as Isabelle HOL premises to allow inversion $\_ \Downarrow \_ :: [exp, exp] \Rightarrow bool$:

$$[\![\ E_1 \Downarrow fun\, x.\, E'\ x;\ abstr\ E'; E_2 \Downarrow V_2;\ (E'\ V_2) \Downarrow V\ ]\!] \implies (E_1 @ E_2) \Downarrow V$$

$$[\![\ abstr\ E;\ E\ (fix\ x.\ E\ x) \Downarrow V\ ]\!] \implies fix\ x.\ E\ x \Downarrow V \ldots$$

- Yet typing is not inductive: this clause fails the positivity condition:

$$[\![\ abstr\ E; \forall x.(of\ x\ T_1)\ \rightarrow\ (of\ (E\ x)\ T_2)\ ]\!] \implies (of\ fun\, x.\, E\ x)\ T_1\ \text{arrow}\ T_2$$

- **Multi-level reasoning** to combine HOAS, induction and hypothetical judgments so that OL contexts are implicit but consistent with Isabelle HOL.

Road-map

- The big issue: a logical framework which combines HOAS, interactive theorem proving and (co)induction.

- Introducing and using Hybrid: encoding Mini-ML.

- $\boxed{\textbf{Multi-level reasoning with Hybrid}}$:

- Case study: compilation.

Multi-Levels Reasoning in Hybrid

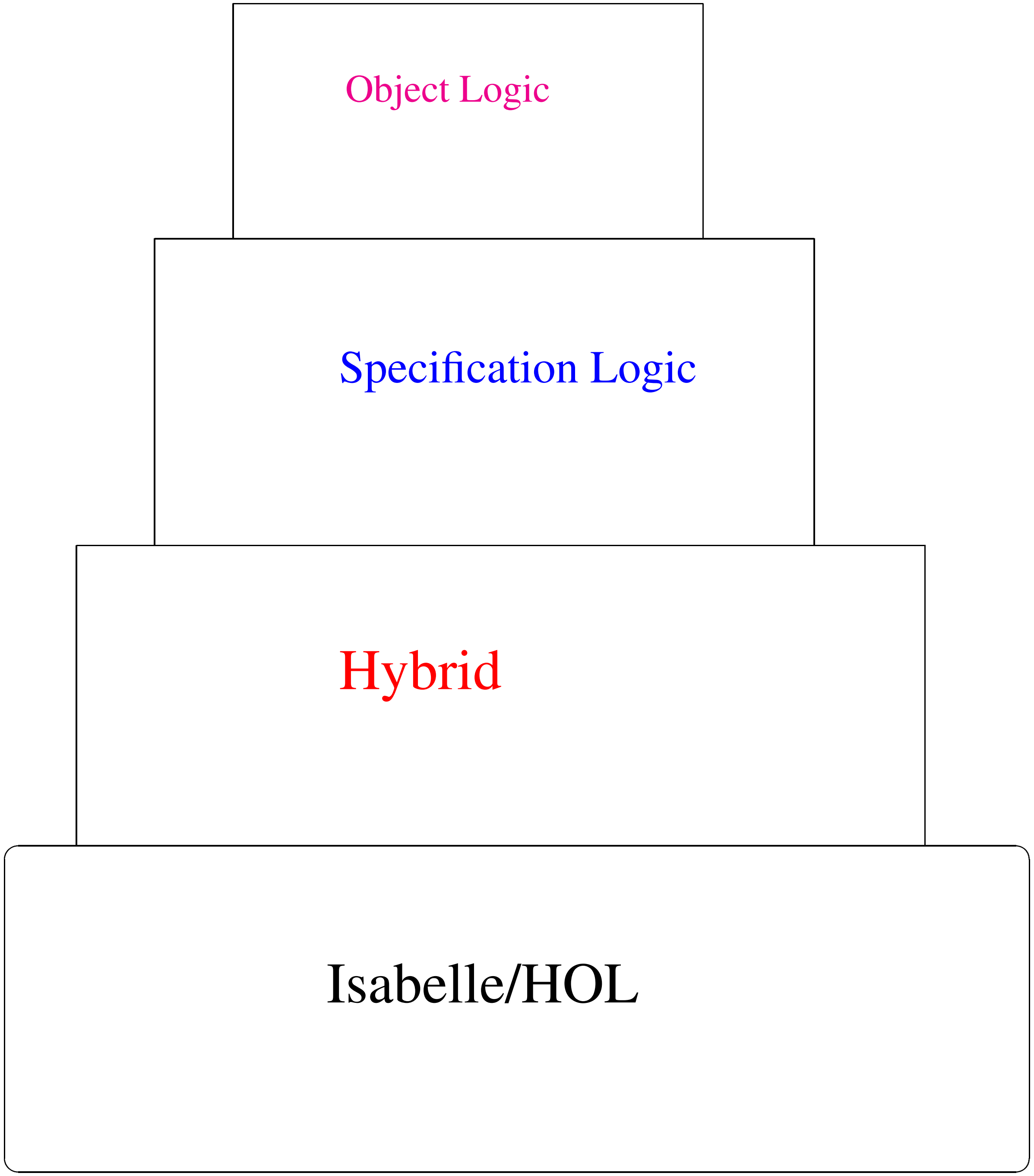
- Moral: HOAS without hypothetical judgments falls short.

- Adopt $FO\lambda^{\Delta\mathbb{N}}$ 2-level approach [Miller'01]: separate the specification (via hypothetical judgments) from the (inductive) meta-reasoning.

- Analogous to the architecture in [Felty'02], but *definitionally* rather than axiomatically.

- $FO\lambda^{\Delta\mathbb{N}}$ definitional reflection [Halnass'90] simulated by inversion on inductive definitions aided by Hybrid frenees of constructors and extensionality properties at higher types

- $FO\lambda^{\Delta\mathbb{N}}$ number induction replaced by Isabelle HOL's structural induction on stratifiable judgments (viz. evaluation).



Syntax: E1 @ E2 N, fun x . E x
Semantics: typing
via prog clauses: $A \leftarrow G$

Sequent Calculus: Gamma $\vdash$ B ::: i
complete induction for
hypothetical judgments

”Datatype” for the l-calculus
primitives: $$, LAM, VAR n

Tactics/Simplifier
induction principles for
stratified jugments

Multi-Levels Reasoning in Hybrid 2/2

- Inherit all the automation (simplifier, classical reasoner) of Isabelle. Full access to all libraries and assorted goodies. For example, reasoning by *logical relations*.

- Fruitful interaction among Hybrid constructs and hypothetical judgment with Isabelle HOL (co)-induction principles, data-types, classical reasoning. More automation than in Coq.

- The mixing of meta-level (evaluation) and OL (typing) judgments makes proofs more mechanizable, compared to all being encapsulated as `prog`.

- This allows us to use co-induction still unaccounted for in a logic such as $FO\lambda^{\Delta I\!N}$. Better: enrich the framework with a proof-theoretic notion of co-induction, possibly *circular proofs* (*guarded* induction [Gimenez95]) — work in progress with D. Miller and A. Tiu.

Sort of a logic programming meta-interpreter; *uniform proofs system* here for a fragment (i.e. hereditary Harrop, without predicate quantification) of second order minimal logic, whose syntax is realized as an Isabelle HOL datatype:

$$
\begin{array}{rrcl}
\textit{Clauses} & D & ::= & A \mid D_1 \; \mathit{and} \; D_2 \mid G \; \mathit{imp} \; A \mid \mathit{tt} \mid \mathit{all}\, x.\, D \\
\textit{Goals} & G & ::= & A \mid G_1 \; \mathit{and} \; G_2 \mid A \; \mathit{imp} \; G \mid \mathit{tt} \mid \mathit{all}\, x.\, G \\
\textit{datatype} & \mathit{oo} & ::= & \langle \mathit{atm} \rangle \mid \mathit{oo} \;\; \mathit{and} \;\; \mathit{oo} \mid \mathit{atm} \;\; \mathit{imp} \;\; \mathit{oo} \mid \mathit{tt} \mid \mathit{all} \; (\mathrm{prpr} \Rightarrow \mathit{oo})
\end{array}
$$

$$
\begin{array}{rcl}
\mathit{Inductive} \; \_ \rhd_{\_} \; \_ & :: & [\mathit{nat}, \mathit{atm}\; \mathit{list}, \mathit{oo}] \Rightarrow \mathit{bool} \\
 & \Longrightarrow & \Phi \rhd_n \mathit{tt} \\
[\![\; \Phi \rhd_n G_1; \Phi \rhd_n G_2 \;]\!] & \Longrightarrow & \Phi \rhd_{n+1} (G_1 \; \mathit{and} \; G_2) \\
[\![\; \forall x.\, \Phi \rhd_n (G\, x) \;]\!] & \Longrightarrow & \Phi \rhd_{n+1} (\mathit{all}\, x.\, G\, x) \\
[\![\; \Phi, A \rhd_n G \;]\!] & \Longrightarrow & \Phi \rhd_{n+1} (A \; \mathit{imp} \; G) \\
[\![\; A \in \Phi \;]\!] & \Longrightarrow & \Phi \rhd_n \langle A \rangle \\
[\![\; A \longleftarrow G; \Phi \rhd_n G \;]\!] & \Longrightarrow & \Phi \rhd_{n+1} \langle A \rangle
\end{array}
$$

- Atoms provable by assumption or via *backchaining* over Prolog-like rules $A \longleftarrow G$ corresponding to an inductive definition of a set `prog` of type $[atm, oo] \Rightarrow bool$ which encode the properties of the object-logic in question.

- Sequents decorated with numerals representing the *height* of a proof so that complete induction in the SL captures induction on derivation of judgments.

- Develop the required meta-theory (but once and for all for a given SL); define $\Phi \rhd G$ iff $\exists n$ such that $\Phi \rhd_n G$ and $(\rhd G)$ if $[] \rhd G$.

  1. Weakening for numerical bounds: for all n, m $[\![\ \Phi \rhd_m G;\ m < n\ ]\!] \Longrightarrow \Phi \rhd_n G$
  2. Context weakening and exchange: $[\![\ \Phi \rhd G;\ \Phi \subseteq \Phi'\ ]\!] \Longrightarrow \Phi' \rhd G$
  3. (Atomic) cut: $[\![\ \Phi, A \rhd G;\ \Phi \rhd \langle A \rangle\ ]\!] \Longrightarrow \Phi \rhd G$

The Object Language Level

- Non-inductive judgments encapsulated as Prolog-like clauses and executed in the SL:

$$\textsf{datatype} \quad atm \quad ::= \quad of\ exp\ tp \mid (val\ env) \vdash exp' \leftrightarrow exp \mid val \Leftrightarrow exp$$

$$\begin{array}{rcl}
Inductive\ _ \longleftarrow _ & :: & [atm, oo] \Rightarrow bool \\
& \Longrightarrow & of\ (E_1 @ E_2)\ T \longleftarrow \langle of\ E_1\ (T'\ \textsf{arrow}\ T)\rangle\ and\ \langle of\ E_2\ T'\rangle \\
[\![abstr\ E]\!] & \Longrightarrow & of\ (fun\, x.\, E\ x)\ (T_1\ \textsf{arrow}\ T_2) \longleftarrow all\, x.\, (of\ x\ T_1)\ imp\ \langle of\ (E\ x)\ T_2\rangle \\
[\![abstr\ E]\!] & \Longrightarrow & of\ (fix\, x.\, E\ x)\ T \longleftarrow all\, x.\, (of\ x\ T)\ imp\ \langle of\ (E\ x)\ T\rangle
\end{array}$$

- Meta-theory: subject reduction is proven by HOL structural induction on evaluation, using only inversion principles on provability and OL typing:

$$E \Downarrow V \Longrightarrow \forall T.\, (\rhd\ of\ E\ T) \to (\rhd\ of\ V\ T)$$

Noteworthy

- Evaluation encoded at the Hybrid level, typing at the OL level.
- **Implicit** contexts: the mathematical statement $x_1 : \tau_1 \ldots x_n : \tau_n \vdash e : \tau$ is represented at the SL level as the **SL sequent** $[of\ !x_1\ \tau_1 \ldots of\ !x_n\ \tau_n] \rhd \langle of\ e\ \tau \rangle$, where the $of\ !x_i \tau_i$'s are SL atoms and $!x_i$'s are Isabelle's eigen-variables.
- Structural rules and cut are proven **once and for all** at the SL level and do not need repeating for every judgment over open terms.
- The call to the substitution lemma is **avoided** altogether by the combination of hypothetical judgments and (atomic) cut.
- This is extremely elegant and invaluable in formal development.

Road-map

- The big issue: a logical framework which combines HOAS, interactive theorem proving and (co)induction.

- Introducing and using Hybrid: encoding Mini-ML.

- Multi-level reasoning with Hybrid.

- **Case study: compilation**

Compiler Verification

- Based on operational (rather than denotational) semantics, so that meaning of both high and low level languages are expressed uniformly.

- Both the operational semantics and the program translations are represented using deductive systems [Despeyroux'86], implemented and (Elf-style) verified in [Pfenning&Hannah'92].

- Stepwise approach: form big-step to closure evaluation to stack machine.

- We add a co-inductive twist due to [Milner & Tofte'91] in the treatment of fix points which evaluate to a "circular" closure, where free variables point to the closure itself.

- Mini-ML compiled into variable-free terms with shift:

$$\begin{array}{rrcl} \textit{dB Terms} & F & ::= & 1 \mid F\uparrow \mid \mathbf{fun}'\ F \mid F\ @'\ F' \mid \mathbf{fix}'\ F \\ \textit{Environments} & \eta & ::= & \cdot \mid \eta;W \mid \eta + F \\ \textit{Values} & W & ::= & \{\eta, F\} \end{array}$$

- dB terms encoded as Hybrid expressions, environments and closure as mutually recursive Isabelle HOL datatypes.

- Standard operational semantics $\eta \vdash F \hookrightarrow W$: environments to associate variables to values and closures for terms with free variables, implemented at the meta-level as a first-order inductive definition.

- Translation ($\eta \vdash F \leftrightarrow e$) to the dB language uses parametric and hypothetical judgments for the binding constructs.

$$\dfrac{\dfrac{}{w \Leftrightarrow x}\,u \\ \vdots \\ \eta; w \vdash F \leftrightarrow e}{\eta \vdash \mathbf{fun}'\ F \leftrightarrow \mathbf{fun}\ x\,.\,e}\ \mathbf{tr\_lam}^{w,x,u} \qquad \dfrac{\dfrac{}{\eta \vdash f \leftrightarrow x}\,u \\ \vdots \\ \eta + f \vdash F \leftrightarrow e}{\eta \vdash \mathbf{fix}'\ F \leftrightarrow \mathbf{fix}\ x\,.\,e}\ \mathbf{tr\_fix}^{f,x,u}$$

$$\dfrac{\eta \vdash F_1 \leftrightarrow e_1 \qquad \eta \vdash F_2 \leftrightarrow e_2}{\eta \vdash F_1\ @'\ F_2 \leftrightarrow e_1 @\ e_2}\ \mathbf{tr\_app}$$

$$\dfrac{W \Leftrightarrow e}{\eta; W \vdash 1 \leftrightarrow e}\ \mathbf{tr\_1} \qquad \dfrac{\eta \vdash F \leftrightarrow e}{\eta; W \vdash F\uparrow \leftrightarrow e}\ \mathbf{tr\_}\uparrow$$

$$\dfrac{\eta \vdash \mathbf{fun}'\ F \leftrightarrow \mathbf{fun}\ x\,.\,e}{\{\eta, \mathbf{fun}'\ F\} \Leftrightarrow \mathbf{fun}\ x\,.\,e}\ \mathbf{vtr\_lam}$$

- The translation is again implemented at the OL level:

$$
\begin{array}{rcl}
 & \Longrightarrow & \eta \vdash (F_1 \; @' \; F_2) \leftrightarrow (E_1 @ E_2) \longleftarrow \langle \eta \vdash F_1 \leftrightarrow E_1 \rangle \; \mathit{and} \; \langle \eta \vdash F_2 \leftrightarrow E_2 \rangle \\
[\![\, \mathit{abstr} \, E \,]\!] & \Longrightarrow & \eta \vdash \mathbf{fun}' \, F \leftrightarrow (\mathit{fun} \, x. E \; x) \longleftarrow \\
 & & \qquad \mathit{all} \, w. \mathit{all} \, x. w \Leftrightarrow x \; \mathit{imp} \; \langle (\eta ; w) \vdash F \leftrightarrow (E \; x) \rangle \\
[\![\, \mathit{abstr} \, E \,]\!] & \Longrightarrow & \eta \vdash \mathbf{fix}' \, F \leftrightarrow (\mathit{fix} \, x. \; E \; x) \longleftarrow \\
 & & \qquad \mathit{all} \, f. \mathit{all} \, x. \eta \vdash f \leftrightarrow x \; \mathit{imp} \; \langle (\eta + f) \vdash F \leftrightarrow (E \; x) \rangle \\
 & \Longrightarrow & (\eta ; W) \vdash 1 \leftrightarrow E \longleftarrow \langle W \Leftrightarrow E \rangle \\
 & \Longrightarrow & (\eta ; W) \vdash F \uparrow \; \leftrightarrow E \longleftarrow \langle \eta \vdash F \leftrightarrow E \rangle \\
 & & \\
[\![\, \mathit{abstr} \, E \,]\!] & \Longrightarrow & \{\eta, \mathbf{fun}' \, F\} \Leftrightarrow \mathit{fun} \, x. E \; x \longleftarrow \langle \eta \vdash \mathbf{fun}' \, F \leftrightarrow \mathit{fun} \, x. E \; x \rangle
\end{array}
$$

$$
\begin{tikzcd}
E \arrow[r, "e \Downarrow V"] \arrow[d, "\eta \vdash F \leftrightarrow E"'] \arrow[dr, phantom, "\equiv"] & V \\
F \arrow[r, "\eta \vdash F \hookrightarrow W"'] & W \arrow[u, "W \Leftrightarrow V"']
\end{tikzcd}
$$

- Soundness of compilation:
  $E \Downarrow V \Longrightarrow \forall n\, \eta\, F. ([] \triangleright_n \eta \vdash F \leftrightarrow E) \rightarrow \exists W. \eta \vdash F \hookrightarrow W \wedge (\triangleright W \Leftrightarrow V)$, by nested induction, outermost on the structure of $E \Downarrow V$ innermost by complete induction on n, i.e. the height of the SL's proof of $\eta \vdash F \leftrightarrow E$.

- Completeness of compilation:
  $\eta \vdash F \hookrightarrow W \Longrightarrow \forall E. (\triangleright \eta \vdash F \leftrightarrow E) \rightarrow \exists V. E \Downarrow V \wedge \triangleright W \Leftrightarrow V$

- Proofs are from first principles, without any technical lemma.

- Unrestricted fix-points requires environments to contain unevaluated expressions so variable evaluation is not just a look-up.

- With the SML restriction that body of a fix point is always a lambda, fix point can evaluate to non-well founded closures — drop the '$+$' constructor altogether in every judgment:

$$[\![\, cl = \{(\eta; cl), \mathbf{fun}'\, F\}^{\infty} \,]\!] \implies \eta \vdash \mathbf{fix}'\, (\mathbf{fun}'\, F) \hookrightarrow cl$$

- Needs co-induction reasoning, to prove, for example subject reduction for dB language (typing of closures and type consistency of value and type environments are *mutually co-inductive* definitions [Milner &Tofte'91]).

- The latter are encoded directly in Isabelle HOL, exploiting the co-induction principle provided by the data-type package.

Conclusions

- A logical framework giving a definitional account of HOAS, compatible with tactic based theorem proving and (co)induction.

- Object languages are represented via a full HOAS λ-calculus and crucial properties such as injectivity are proved via the de Bruijn representation, rather than postulated as axioms.

- **Judgments** encoded as HOL's inductive definitions, if stratifiable.

- Two-level architecture allows the use of hypothetical judgments similarly to Twelf, but in the interactive and well-known setting of higher-order logic.

- Results about the SL (such as cut elimination) are proven once and for all. Different logics can be selected, without changing infrastructure.

Current and Future Work

- At the infrastructure level, generalization to n-ary abstraction and definitions by primitive recursion and case analysis.

- Case studies in other areas: authenticity-by-typing [Gordon & Jeffrey'03]

- Different specification logics (the "next 700 syndrome"):

  1. "Term-in-infinite-contexts" encoding for induction over open terms and Miller's ∇ quantifier to reason about **names**.

  2. Linear logic, both in Lolli [Miller'92] and Forum presentation [Miller'95], with application to security [Miller'03]. Also, Olli *non-commutative* intuitionistic linear logic [Polakow'01], to model imperative features (storage, continuations and exceptions).