A Hybrid Logical Framework

Alberto Momigliano
Department of Mathematics and Computer Science
University of Leicester

Joint work with Simon Ambler

INRIA Rocquencourt 31/1/2003

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: case study Mini-ML.
- Multi-level reasoning with Hybrid.
- Case study: compilation.
- (Mini-ML with references in a linear logic framework).

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).

- Main case study: **compilation** of an (effect-free) fragment of ML (Mini-ML). Prove the compiler sound and complete. Extend it to the imperative fragment.

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

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

- $\mathbf{fix}\, x\,.\, E$ (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 $e \Downarrow v$:

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

$$\frac{}{\mathbf{fun}\, x\,.\, e \Downarrow \mathbf{fun}\, x\,.\, e}\ \mathbf{ev_fun} \qquad \frac{[\mathbf{fix}\, x\,.\, e/x]e \Downarrow v}{\mathbf{fix}\, x\,.\, e \Downarrow v}\ \mathbf{ev_fix}$$

- Key notion is **substitution** of expressions, cf. **ev_fix**, **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):

$$\frac{\Gamma, x{:}\tau \vdash e : \tau'}{\Gamma \vdash \mathbf{fun}\, x\,.\, e : \tau \text{ arrow } \tau'}\,\mathbf{tp_fun} \qquad \frac{\Gamma, x{:}\tau \vdash e : \tau}{\Gamma \vdash \mathbf{fix}\, x\,.\, e : \tau}\,\mathbf{tp_fix}$$

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau}\,\mathbf{tp_var} \qquad \frac{\Gamma \vdash e_1 : \tau' \to \tau \qquad \Gamma \vdash e_2 : \tau'}{\Gamma \vdash e_1 \mathbin{@} e_2 : \tau}\,\mathbf{tp_app}$$

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

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

no rule for **tp_var**

```
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, (example, 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 datatype 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.

- Name-carrying syntax: i.e. fix :: $var * exp \Rightarrow exp$; then quotient these trees by α-conversion. Incredibly, people do go thru this [Ford & Mason'02], while the essence is *FM sets* [Gabbay and Pitts'99].

- De Bruijn syntax: α-equivalence is identity, but the handling of there pointers is better left to a compiler (need well formed-ness, substitution is complex).

- Toward higher-order syntax: fix is a function from **var** to expressions: [Despeyroux et al'95]. Then fix :: $(var \Rightarrow exp) \Rightarrow exp$. Example: $\ulcorner \mathbf{fix}\, x\,.\, x \urcorner = \mathrm{fix}(\lambda x. VAR\ x)$.

- This is inductive, but the function space is too rich. Hence **Exotic terms**: $\mathrm{fix}(\lambda x. if\ x = 0\ then\ 0\ else\ 1)$ does not represent a Mini-ML terms.

- *Elf*: specify a meta-theorem as a relation between judgments, induced by operational semantics of a logic programming-like interpretation. External checks (termination, moded-ness, totality) confirms that it is a realizer for that theorem.

- *Twelf*: an explicit meta-meta-logic for reasoning (inductively) about logical frameworks, in a fully automated way. $\mathcal{M}_\omega$ is a constructive first-order logic, whose quantifiers range over LF object over a signature.

- $FO\lambda^{\Delta\mathbb{N}}$ [Miller & McDowell '02]: higher-order intuitionistic logic with definitional reflection and induction on natural numbers (only a proof-checker is implemented)

- What about a tactic-driven system such as Isabelle HOL?

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 datatypes 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.

- **Introducing and using Hybrid: case study Mini-ML**.

- Multi-level reasoning with Hybrid:

- Case study: compilation.

- (Mini-ML with references in a linear logic framework).

- 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:

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

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

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

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

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

$$[\![\ \text{abstr}\ E;\ \text{abstr}\ F\ ]\!] \Longrightarrow (\text{LAM}\, v.E\ v = \text{LAM}\, v.F\ v) = (E = F)$$

$$[\![\ \text{abstr}\ E;\ \text{abstr}\ F;\ \forall i.E\ (\text{VAR}\ i) = F\ (\text{VAR}\ i)\ ]\!] \Longrightarrow E = F$$

- Several induction principles over proper terms also proved:

$$[\![\ \text{proper}\ u;$$

$$\bigwedge i.\ P(\text{VAR}\ i);$$

$$\bigwedge s\ t.\ [\![\ \text{proper}\ s;\ P\ s;\ \text{proper}\ t;\ P\ t\ ]\!] \Longrightarrow P(s\ \$\$\ t);$$

$$\bigwedge e.\ [\![\ \text{abstr}\ e;\ \forall t.\ \text{proper}\ t \longrightarrow Pt \longrightarrow P(e\ t)\ ]\!] \Longrightarrow P(\text{LAM}\, x.e\ x)\ ]\!]$$

$$\Longrightarrow Pu$$

- Let's encode a fragment of Quantified Boolean formulae: introduce a type synonym oo for Hybrid's proper expressions.

$$\begin{array}{rcl} Q & ::= & V_i \mid Q \wedge Q' \mid \forall V_i . Q \mid \ldots \\ \mathsf{All}. & :: & [oo \Rightarrow oo] \Rightarrow oo \qquad\qquad \mathsf{And} :: [oo, oo] \Rightarrow oo \\ Q_1\ \mathsf{And}\ Q_2 & == & \mathsf{CON}\ cAnd\ \$\$\ Q_1\ \$\$\ Q_2 \qquad \mathsf{All}\, v . Q\ v == \mathsf{CON}\ cForall\ \$\$\ \mathsf{LAM}\, v . Q\, v \end{array}$$

- Underlying form of $\mathsf{All}\, v_1 . \mathsf{All}\, v_2 .\ v_1\ \mathsf{And}\ v_2$:

$$\mathsf{CON}\ cForall\ \$\$\ (\mathsf{LAM}\, v_1 . \mathsf{CON}\ cForall\ \$\$\ \mathsf{LAM}\, v_2 . (\mathsf{CON}\ cAnd\ \$\$\ v_1\ \$\$\ v_2)$$

- The constructors have distinct images. For example: $\mathsf{All}\, x . P\ x \neq\ Q\ \mathsf{And}\ Q'$

- Every *binding* constructor is injective on abstractions.

$$[\![\ \mathsf{abstr}\ P; \mathsf{abstr}\ Q; \mathsf{All}\, v . e\ v = \mathsf{All}\, v . f\ v\ ]\!] \Longrightarrow P = Q$$

Back to Mini-ML (in Hybrid)

- Encode the syntax as before and evaluation as an *inductive* relation:

- *abstraction* annotations as Isabelle HOL premises to derive inversion principles:

$$\mathit{Inductive}\ _ \Downarrow _ \quad :: \quad [\mathit{exp}, \mathit{exp}] \Rightarrow \mathit{bool}$$

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

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

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

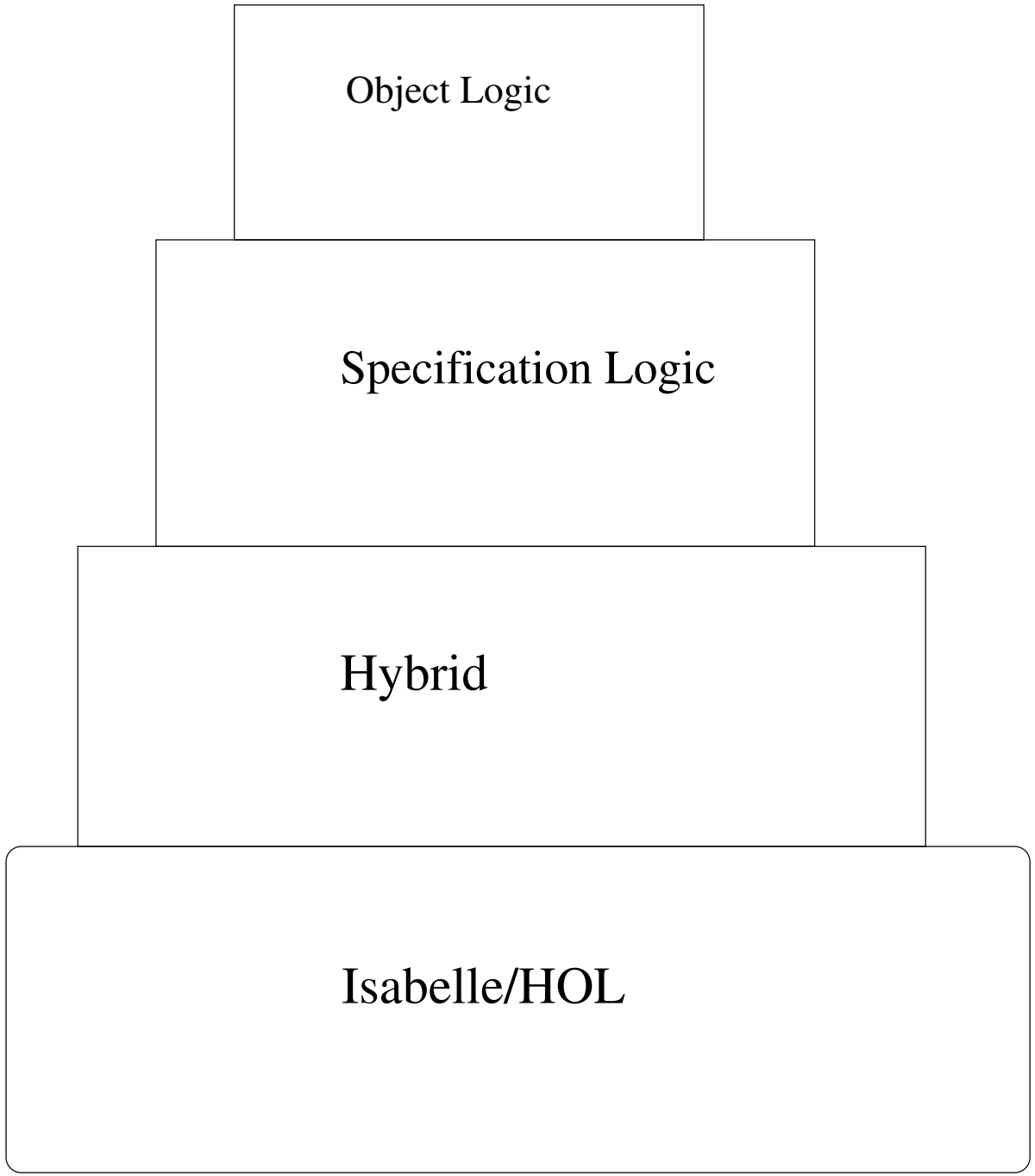
$$[\![\ \mathsf{abstr}\ E; \forall x.(x : T)\ \rightarrow\ (E\ x : T)\ ]\!] \implies (\mathbf{fix}\ x\ .\ E\ x) : T$$

- **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: case study Mini-ML.
- **Multi-level reasoning with Hybrid**:
- Case study: compilation.
- (Mini-ML with references in a linear logic framework).

- Moral: HOAS without hypothetical judgments falls short.
- Adopt $FO\lambda^{\Delta\mathbb{N}}$ 2-level approach, where we 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 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: M @ N, fun x . E x
Semantics:typing
via prog clauses: A <== G

Sequent Calculus: Gs |– B : i
complete induction for
hypothetical judgments

"Datatype" for the l–calculus
primitives: $$, LAM, VAR n

Tactics/Simplifier
induction principles for
stratified jugments

- 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, datatypes, 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 $FO\lambda^{\Delta\mathbb{N}}$. Better: enrich the framework with a proof-theoretic notion of coinduction, 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:

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

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

- 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 $\cdot \rhd G$.

  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:

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

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

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

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

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: case study Mini-ML.
- Multi-level reasoning with Hybrid.
- **Case study: compilation**
- (Mini-ML with references in a linear logic framework).

- 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.

$$\frac{\dfrac{}{w \Leftrightarrow x}\,u \quad \vdots \quad \eta; w \vdash F \leftrightarrow e}{\eta \vdash \mathbf{fun}'\ F \leftrightarrow \mathbf{fun}\ x\,.\,e}\,\mathbf{tr_lam}^{w,x,u} \qquad \frac{\dfrac{}{\eta \vdash f \leftrightarrow x}\,u \quad \vdots \quad \eta + f \vdash F \leftrightarrow e}{\eta \vdash \mathbf{fix}'\ F \leftrightarrow \mathbf{fix}\ x\,.\,e}\,\mathbf{tr_fix}^{f,x,u}$$

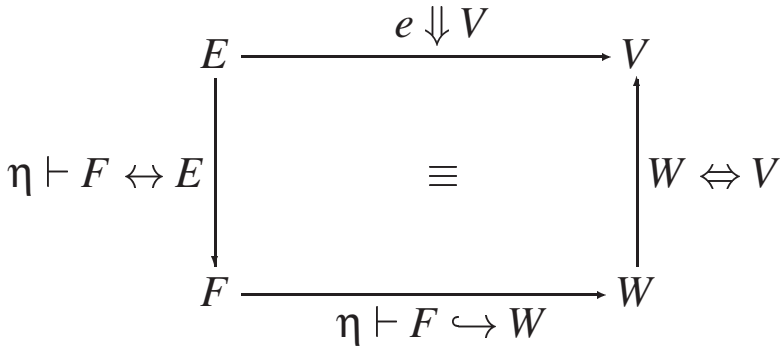
$$\frac{\eta \vdash F_1 \leftrightarrow e_1 \qquad \eta \vdash F_2 \leftrightarrow e_2}{\eta \vdash F_1\ @'\ F_2 \leftrightarrow e_1 @\ e_2}\,\mathbf{tr_app}$$

$$\frac{W \Leftrightarrow e}{\eta; W \vdash 1 \leftrightarrow e}\,\mathbf{tr_1} \qquad \frac{\eta \vdash F \leftrightarrow e}{\eta; W \vdash F \uparrow \leftrightarrow e}\,\mathbf{tr_}\uparrow$$

$$\frac{\eta \vdash \mathbf{fun}'\ F \leftrightarrow \mathbf{fun}\ x\,.\,e}{\{\eta, \mathbf{fun}'\ F\} \Leftrightarrow \mathbf{fun}\ x\,.\,e}\,\mathbf{vtr_lam}$$

- This is again implemented at the OL level:

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



- Soundness of compilation:
  $E \Downarrow V \Longrightarrow \forall n\ \eta\ F.\,(\cdot \rhd_n \eta \vdash F \leftrightarrow E) \to \exists W.\,\eta \vdash F \hookrightarrow W \wedge (\rhd W \Leftrightarrow V)$, 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:
  $\eta \vdash F \hookrightarrow W \Longrightarrow \forall n\ E.\,(\cdot \rhd_n \eta \vdash F \leftrightarrow E) \to \exists V.\,E \Downarrow V \wedge \rhd W \Leftrightarrow V$

- 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, luckily, as we don't have a proof-theoretic notion of coinductive proof yet.

Road-map

- The big issue: a logical framework which combines HOAS, interactive theorem proving and (co)induction.
- Introducing and using Hybrid: case study Mini-ML.
- Multi-level reasoning with Hybrid.
- Case study: compilation.
- **Mini-ML with references in a linear logic framework**

- Evaluation in Mini-ML with references involves imperative **state**-based computations, i.e. judgments such as $St, e \Downarrow St', v$.

$$\begin{array}{lcll} \textit{Types} & \tau & ::= & \ldots \tau \ \mathbf{refty} \\ \textit{Terms} & e & ::= & \ldots \mathbf{ref}\ e \mid e{:=}\ e' \mid !\ e \mid c : loc \end{array}$$

- Representing the state as a list or a function is too much of a low-level description; aim for an **implicit** store, via hypothetical judgments.

- Updating a state is a destructive operation, compatible with the monotone notion of intuitionistic context only via awkward encodings.

- *Linear logic* views context formulae as resources: a state (store) $\{c_1 \mapsto v_1 \ldots c_n \mapsto v_n\}$ will be represented implicitly by a linear context of atoms $contains\ C_1\ V_1 \ldots contains\ C_n\ V_n$, for C_i eigenvariable [Cervesato96].

- Updating the store $C := V$ means consuming the linear assumption $contains\ C\ W$ and asserting the new fact $contains\ C\ V$.

- **Freshness** of cells is again enforced by a parametrical judgments.

- Specification logic is a meta-interpreter for Lolli [Hodas & Miller94], with additive conjunction, top (erase), linear and minimal implication.

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

- Clauses have the form $A \longleftarrow LL; IL$, where two lists form the body.

- *Backchaining* is more complex since it involves splitting "$*$" the linear context:

$$\begin{array}{rcl} [\![\ (A \longleftarrow LL; IL) \wedge \Gamma; \Delta \models_n LL \wedge \Gamma; \cdot \models_n IL\ ]\!] & \Longrightarrow & \Gamma; \Delta \rhd_{n+1} \langle A \rangle \\ & \Longrightarrow & \Gamma; \cdot \models_n \cdot \\ [\![\ \Delta = \Delta_1 * \Delta_2, \Gamma; \Delta_1 \rhd_n B \wedge \Gamma; \Delta_2 \models_n \Phi\ ]\!] & \Longrightarrow & \Gamma; \Delta \models_{n+1} B, \Phi \end{array}$$

- Context splitting is difficult to implement in a linear lp, but here it is less problematic, as guided by tactics and we are interested in the meta-theory.

- Show structural rules for intuitionistic context and (two) cut rules admissible.

- Crucially, this SL is a **conservative** extension of the previous one, so that when you do not need linearity, your proofs stay the same.

- towards subject reduction: the implementation of typing needs no linear assumption, so Δ will be empty and we may as well use linear body:

```
tp_abs "[| abstr E |] ==> (Fun E) ::: ( T1 ->  T2) <--
            [all x . ( x ::: T1 ->  <  (E x) ::: T2 >)] ; []
```

- Evaluation is *continuation-based*, so that with this sequentalization we can express what to do next (i.e. , the next expression on the stack).

- We need an auxiliary level of *instructions*, as well as *answers* yielding the judgment $K \models (S, i) \Downarrow a$.

- The implicit state is reified and returned (by consuming it) in the final answer a, where cells are bound by a “new” quantifier.

- The description of the implementation of evaluation with a judgment of type $cntn \to instr \to answr \to bool$ is a topic in its own. A sample is below:

```
ev_cell  "K |= ieval (cell C) >>> Ans <-- [<K|= ireturn (cell C) >>> Ans>
ev_ref   "K |= ieval (ref E) >>> Ans <--
          [< (klam (%x. iref x) K) |= ieval E >>> Ans>] ; [] "
ev_iref  "[| abstr F |] ==> K|= iref V >>> (New F) <--
                [all c. (contains c V -* <K|= ireturn (cell c) >>> (F c)>)
ev_idref "K|= ideref (cell C) >>> Ans <--
                [<contains C V>,  contains C V -* <K|= ireturn V >>> F>] ;
ev_ass2 " K|=  iassgn2 (cell C) V  >>> Ans <--
                [<contains C W>,  contains C V -* <K|= ireturn V >>> F>] ;
Subject reduction:
        "[]; De |- (K |=  I >>> Ans) -->
           !Ga T1 T2. Ga; [] |- ofK K (T1 -> T2) & Ga; [] |- ofI I T1
                    --> Ga ; [] |- ofA Ans T2"
```

Conclusions

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

- Crucial properties of expressions are proved via the de Bruijn representation, rather than postulated as axioms.

- One-level architecture sufficient when hypothetical judgments are rare.

- Multi-level architecture allows to simulate $FO\lambda^{\Delta\mathbb{N}}$ and Twelf 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.

- Different specification logics:

  1. Miller & McDowell's explicit eigenvariable encoding for induction over generic judgments (i.e. proof of type uniqueness).

  2. Logic with a "New" quantifier to reason about **names**.

  3. More experiments with linear logic, also in the Forum presentation [Miller95] with application to security [Miller03]. Also, *non-commutative* ones [Polakow01], to model continuations and exceptions.

- Co-induction via a coinduction rule at the meta-level and/or a notion of circular proofs (case study: polymorphic references 'a la Tofte [91]).

More Info

- `http://www.mcs.le.ac.uk/mechsem/Hybrid`
  - TPHOLs '02 paper with the theory of Hybrid, scripts for the current infrastructure and example suite: quantified propositional logic, higher-order π-calculus, functional programming.
- `http://www.mcs.le.ac.uk/mechsem/Hybrid/Howe`
  - LFM'02 paper and code for Howe's style proof of applicative bisimulation being a congruence.
- `http://www.mcs.le.ac.uk/mechsem/Hybrid/2level/Compilation`
  - FOSSACS'03 paper with the multi-level architecture and the compilation example, both inductive and co-inductive.