Relational semantics for effect-based program transformations: higher-order store

Martin Hofmann

Ludwig-Maximilians-Universität München

IFIP Working Group 2.8, June 2009

Effect-dependent program equivalences

$$x = e; y = e; e'(x, y) \quad \text{is equivalent to} \quad x = e; e'(x, x)$$

provided that x, y are fresh and

- e's reads and writes are disjoint and
- e does not allocate, or
- none of the above, but somehow e' doesn't care.

Effect-dependent program equivalences

$$x = e; y = e; e'(x, y) \quad \text{is equivalent to} \quad x = e; e'(x, x)$$

provided that x, y are fresh and

- e's reads and writes are disjoint and
- e does not allocate, or
- none of the above, but somehow e' doesn't care.

Ongoing research programme:

- Justify such conditional equivalences by interpreting effectful types as relations ("logical relation")
- Global integer references (APLAS06)
- Dynamically allocated integer references with regions (PPDP07)
- Ultimate goal: Dynamically allocated references of arbitrary type.

Acknowledgements: Nick Benton, Lennart Beringer, Andrew Kennedy (collaborators)
MOBIUS (IST-FET-15905).

This talk

- Global references of arbitrary (including functional) type
- Relational semantics requires solving mixed-variance equations.
- Existing solution theory found insufficient.
- Extension to solution theory
- Definition of logical relation that proves soundness of effect-dependent program equivalences
- Fly in the ointment: in latent effects of stored functions we cannot distinguish reading and writing.

Syntax

$$
\begin{aligned}
e \quad ::= \quad & x \mid n \mid \texttt{true} \mid \texttt{false} \mid x_1 \ op\ x_2 \mid () \mid (x_1, x_2) \mid x.1 \mid \\
& x.2 \mid x_1\ x_2 \mid \texttt{let}\ x \Leftarrow e_1\ \texttt{in}\ e_2 \mid !\ell \mid \ell := x \mid \\
& \texttt{if}\ x\ \texttt{then}\ e_2\ \texttt{else}\ e_3 \mid \texttt{rec}\ f\ x.e \mid \lambda x.e
\end{aligned}
$$

In examples we use ML notation such as this

```
val f = fn g => fn n =>
        if n=0 then 1 else n * g (n-1);
val r = ref (fn x => 0);
val fac = fn n => (r := (fn x => f (!r) x); !r n);
```

Denotational semantics

$$\begin{array}{rcl} \mathbf{V} & \cong & \{\mathit{wrong}\} + \mathit{unit}(\mathbf{1}) + \mathit{int}(\mathbb{Z}) + \mathit{bool}(\mathbb{B}) \; + \\ & & \mathit{pair}(\mathbf{V} \times \mathbf{V}) + \mathit{fun}(\mathbf{V} \to \mathbf{C}) \\ \mathbf{C} & = & \mathbf{S} \to (\mathbf{S} \times \mathbf{V})_\bot \\ \mathbf{S} & = & \mathbb{L} \to \mathbf{V} \end{array}$$

V is the least predomain solving this. Predomain: CPO not nec. with $\bot$.
NB **C** happens to have least element $\lambda x.\bot$.
We have retracts $p_i : \spadesuit \to \spadesuit$ where $\spadesuit \in \{\mathbf{V}, \mathbf{S}, \mathbf{C}\}$.

Properties of the retracts

$$
\begin{array}{rcl}
p_i(\mathit{wrong}) & = & \mathit{wrong} \\
p_i(\mathit{int}(n)) & = & \mathit{int}(n) \\
p_i(\mathit{unit}()) & = & \mathit{unit}() \\
p_i(\mathit{bool}(x)) & = & \mathit{bool}(x) \\
p_i(\mathit{pair}(v_1, v_2)) & = & \mathit{pair}(p_i(v_1), p_i(v_2)) \\
p_i(\mathit{fun}(f)) & = & \mathit{fun}(p_i; f; p_i) \\
p_0(f)(s) & = & \bot \\
p_{i+1}(f)(s) & = & \bot \text{ if } f(p_i(s)) = \bot \\
p_{i+1}(f)(s) & = & (p_i(s_1), p_i(v)) \text{ if } f(p_i(s)) = (s_1, v) \\
p_i(s)(\ell) & = & p_i(s(\ell))
\end{array}
$$

Moreover, $p_i \sqsubseteq p_{i+1}$ and $p_i; p_j = p_{\min(i,j)}$ and $\bigsqcup_i p_i(x) = x$ for all $x \in \mathbf{V} \cup \mathbf{S} \cup \mathbf{C}$.

Useful for proving properties/defining functions over $\mathbf{V}$.

Semantics of untyped language

$[\![e]\!]\theta \in \mathbf{C}$ when $\theta : FV(e) \to \mathbf{V}$

$$
\begin{array}{lcl}
[\![x]\!]\theta\ s & = & (s, \theta(x)) \\
[\![x\ y]\!]\theta\ s & = & f(\theta(y))s \text{ where } \theta(x) = fun(f) \\
[\![\texttt{let}\ x \Leftarrow e_1\ \texttt{in}\ e_2]\!]\theta\ s & = & [\![e_2]\!]\theta[x \mapsto v]\ s_1 \text{when } [\![e_1]\!]\theta\ s = (s_1, v) \\
[\![\texttt{if}\ x\ \texttt{then}\ e_2\ \texttt{else}\ e_3]\!]\theta & = & [\![e_2]\!]\theta, \text{ when } \theta(x) = bool(\texttt{true}) \\
[\![!\ell]\!]\theta\ s & = & (s, s.\ell) \\
[\![\ell := y]\!]\theta\ s & = & (s[\ell \mapsto \theta(y)], unit()) \\
[\![\texttt{rec}\ f\ x.e]\!]\theta\ s & = & (s, fun(g)) \text{ where } g = \bigsqcup_i g_i \text{ and} \\
 & & g_0 = \lambda x.\lambda s.\bot \text{ and} \\
 & & g_{i+1} = \lambda v.[\![e]\!]\theta[x \mapsto v, f \mapsto fun(g_i)] \\
[\![\lambda x.e]\!]\theta\ s & = & (s, fun(f)) \text{ where } f\ \ v = [\![e]\!]\theta[x \mapsto v] \\
[\![e]\!]\theta\ s & = & wrong, \text{ if no clause applies}
\end{array}
$$

Types

Effects (ε): Finite subsets of $\{rd_\ell, wr_\ell \mid \ell \in \mathbb{L}\}$.

Types:

$$A, B, C \quad ::= \quad \texttt{int} \mid \texttt{unit} \mid \texttt{bool} \mid A \times B \mid A \xrightarrow{\varepsilon} B$$

Store type (Σ): $\ell_1{:}A_1, \ldots, \ell_n{:}A_n$.

Typing context (Θ): $x_1{:}A_1, \ldots, x_m{:}A_m$.

Typing judgement: $\Pi; \Sigma; \Theta \vdash e : A, \varepsilon$. Here $\Pi \subseteq \mathbb{L}$, all ℓ appearing in jugement are listed in Π.

Typing rules

$$\frac{}{\Pi; \Sigma; \Theta \vdash n : \mathtt{int}}(\text{T-INT})$$

$$\frac{x \in \mathrm{dom}(\Theta) \qquad \Pi \vdash \Theta\ ok}{\Pi; \Sigma; \Theta \vdash x : \Theta(x)}(\text{T-VAR})$$

$$\frac{\Pi; \Sigma; \Theta}{\Pi; \Sigma; \Theta \vdash !\ell : \Sigma(\ell), \{rd_\ell\}}(\text{T-READ})$$

$$\frac{\Pi; \Sigma; \Theta \vdash y : \Sigma(\ell)}{\Pi; \Sigma; \Theta \vdash \ell := y : \mathtt{unit}, \{wr_\ell\}}(\text{T-WRITE})$$

$$\frac{\Pi; \Sigma; \Theta \vdash e : A, \varepsilon_1 \qquad A <: B \qquad \varepsilon_1 \subseteq \varepsilon_2}{\Pi; \Sigma; \Theta \vdash e : B, \varepsilon_2}(\text{T-SUB})$$

$$\frac{\Pi; \Sigma; \Theta \vdash x : A \xrightarrow{\varepsilon} B \qquad \Pi; \Sigma; \Theta \vdash y : A}{\Pi; \Sigma; \Theta \vdash x\ y : B, \varepsilon}(\text{T-APP})$$

Typing rules, cont'd

$$\frac{\Pi;\Sigma;\Theta, x{:}A \vdash e : B, \varepsilon}{\Pi;\Sigma;\Theta \vdash \lambda x.e : A \xrightarrow{\varepsilon} B}\,(\text{T-LAM})$$

$$\frac{\begin{array}{c}\Pi;\Sigma;\Theta \vdash x : \texttt{bool} \\ \Pi;\Sigma;\Theta \vdash e_1 : A, \varepsilon \qquad \Pi;\Sigma;\Theta \vdash e_2 : A, \varepsilon\end{array}}{\Pi;\Sigma;\Theta \vdash \texttt{if}\ x\ \texttt{then}\ e_1\ \texttt{else}\ e_2 : A, \varepsilon}\,(\text{T-IF})$$

$$\frac{\Pi;\Sigma;\Theta \vdash e_1 : A_1, \varepsilon_1 \qquad \Pi;\Sigma;\Theta, x{:}A_1 \vdash e_2 : A_2, \varepsilon_2}{\Pi;\Sigma;\Theta \vdash \texttt{let}\ x \Leftarrow e_1\ \texttt{in}\ e_2 : A_2, \varepsilon_1 \cup \varepsilon_2}\,(\text{T-LET})$$

$$\frac{\Pi;\Sigma;\Theta \vdash x : A \qquad \Pi;\Sigma;\Theta \vdash y : B}{\Pi;\Sigma;\Theta \vdash (x, y) : A \times B}\,(\text{T-PAIR})$$

$$\frac{\Pi;\Sigma;\Theta, f{:}A \xrightarrow{\varepsilon} B, x{:}A \vdash e : B, \varepsilon}{\Pi;\Sigma;\Theta \vdash \texttt{rec}\ f\ x.e : A \xrightarrow{\varepsilon} B}\,(\text{T-REC})$$

Subtyping

$$\frac{}{A <: A}(\textsc{s-refl})$$

$$\frac{A_1 <: A_2 \qquad B_1 <: B_2}{A_1 \times B_1 <: A_2 \times B_2}(\textsc{s-prod})$$

$$\frac{A_2 <: A_1 \qquad B_1 <: B_2 \qquad \varepsilon_1 \subseteq \varepsilon_2}{A_1 \xrightarrow{\varepsilon_1} B_1 <: A_2 \xrightarrow{\varepsilon_2} B_2}(\textsc{s-arr})$$

Example again

```
val f = fn g => fn n =>
        if n=0 then 1 else n * g (n-1);
val r = ref (fn x => 0);
val fac = fn n => (r := (fn x => f (!r) x); !r n);
```

$$\mathtt{r}; \mathtt{r} : \mathtt{int} \xrightarrow{rd_{\mathtt{r}}} \mathtt{int}; \emptyset \vdash \mathtt{f} : (\mathtt{int} \xrightarrow{rd_{\mathtt{r}}} \mathtt{int}) \rightarrow \mathtt{int} \xrightarrow{rd_{\mathtt{r}}} \mathtt{int}$$

$$\mathtt{r}; \mathtt{r} : \mathtt{int} \xrightarrow{rd_{\mathtt{r}}} \mathtt{int}; \emptyset \vdash \mathtt{fac} : \mathtt{int} \xrightarrow{rd_{\mathtt{r}}, wr_{\mathtt{r}}} \mathtt{int}.$$

More examples: Vector multiplication, event handling.

Equational theory

$$\frac{\forall\theta.[\![e_1]\!]\theta = [\![e_2]\!]\theta \qquad \Pi;\Sigma;\Theta \vdash e_i : A, \varepsilon}{\Pi;\Sigma;\Theta \vdash e_1 = e_2 : A, \varepsilon}(\text{E-BASIC})$$

Sym, Trans, Cong.

$$\frac{\Pi;\Sigma;\Theta \vdash e : A, \varepsilon \qquad \mathrm{rds}(\varepsilon) \cap \mathrm{wrs}(\varepsilon) = \emptyset \qquad x \notin \mathrm{dom}(\Theta)}{\begin{array}{c}\Pi;\Sigma;\Theta \vdash \mathtt{let}\ x \Leftarrow e\ \mathtt{in}\ \mathit{pair}(x, x) = \\ \mathtt{let}\ x \Leftarrow e\ \mathtt{in}\ \mathtt{let}\ y \Leftarrow e\ \mathtt{in}\ \mathit{pair}(x, y) : A \times A, \varepsilon\end{array}}(\text{E-DUP})$$

Typing rules cont'd

$$\frac{\Pi;\Sigma;\Theta \vdash e_i : A_i, \varepsilon_i \qquad \forall i = 1,2. \mathrm{rds}(\varepsilon_i) \cap \mathrm{wrs}(\varepsilon_{3-i}) = \emptyset \quad \mathrm{wrs}(\varepsilon_i) \cap \mathrm{wrs}(\varepsilon_{3-i}) = \emptyset \quad x_i \cap (\mathrm{dom}(\Theta) \cup \{x_{3-i}\}) = \emptyset}{\Pi;\Sigma;\Theta \vdash \mathtt{let}\ x_1 \Leftarrow e_1\ \mathtt{in}\ \mathtt{let}\ x_2 \Leftarrow e_2\ \mathtt{in}\ \mathit{pair}(x_1,x_2) = \mathtt{let}\ x_2 \Leftarrow e_2\ \mathtt{in}\ \mathtt{let}\ x_1 \Leftarrow e_1\ \mathtt{in}\ \mathit{pair}(x_1,x_2) : A_1 \times A_2, \varepsilon_1 \cup \varepsilon_2}\ (\text{E-SWAP})$$

$$\frac{\Pi;\Sigma;\Theta \vdash e_1 : A, \emptyset \qquad \Pi;\Sigma;\Theta, x{:}A, y{:}B \vdash e_2 : C, \varepsilon \qquad x \neq y}{\Pi;\Sigma;\Theta \vdash \mathtt{let}\ \_ \Leftarrow e_1\ \mathtt{in}\ \lambda y{:}B.\mathtt{let}\ x \Leftarrow e_1\ \mathtt{in}\ e_2 = \mathtt{let}\ x \Leftarrow e_1\ \mathtt{in}\ \lambda y{:}B.e_2 : B \xrightarrow{\varepsilon} C, \emptyset}\ (\text{E-HOIST})$$

Goal: Semantic interpretation of eq.thy as logical relation.

- Justifies soundness eq.thy for obs.eq.
- Allows for semantic reasoning (justify obs.eq using the log.rel rather than rules)

The logical relation

Define

$[\![\Pi; \Sigma \vdash A]\!] \subseteq \mathbf{V} \times \mathbf{V}$

$[\![\Pi; \Sigma \vdash A, \varepsilon]\!] \subseteq \mathbf{C} \times \mathbf{C}$

$[\![\Pi; \Sigma \vdash \varepsilon]\!] \subseteq$ sets of relations on $\mathbf{S}$

$$[\![\Pi; \Sigma \vdash A, \varepsilon]\!] = \mathrm{per}(\mathtt{T}_E^O(A))$$

$$\begin{aligned}
&(f, f') \in \mathtt{T}_E^O(A) \iff \forall s\ s'\ s_1\ s_1'\ v\ v'.\forall R \in E.(sRs' \Rightarrow \\
&\quad (f\ s = \bot \Leftrightarrow f'\ s' = \bot) \wedge \\
&\quad ((f\ s) = (s_1, v) \wedge (f'\ s') = (s_1', v') \Rightarrow s_1 R s_1' \wedge (v, v') \in [\![\Pi; \Sigma \vdash A]\!])
\end{aligned}$$

Logical relation cont'd

$$
\begin{aligned}
[\![\Pi;\Sigma \vdash \mathtt{unit}]\!] &= \mathtt{Unit}\\
[\![\Pi;\Sigma \vdash \mathtt{int}]\!] &= \mathtt{Int}\\
[\![\Pi;\Sigma \vdash \mathtt{bool}]\!] &= \mathtt{Bool}\\
[\![\Pi;\Sigma \vdash A \times B]\!] &= \mathtt{Prod}[\![\Pi;\Sigma \vdash A]\!], [\![\Pi;\Sigma \vdash B]\!]\\
[\![\Pi;\Sigma \vdash A \xrightarrow{\varepsilon} B]\!] &= \mathtt{Arr}[\![\Pi;\Sigma \vdash A]\!], [\![\Pi;\Sigma \vdash B, \varepsilon]\!])
\end{aligned}
$$

Problem: It is not clear whether $[\![\ldots]\!]$ satisfying these exists!

Logical relation cont'd

$$
\begin{array}{rcl}
[\![\Pi;\Sigma \vdash \texttt{unit}]\!] & = & \texttt{Unit} \\
[\![\Pi;\Sigma \vdash \texttt{int}]\!] & = & \texttt{Int} \\
[\![\Pi;\Sigma \vdash \texttt{bool}]\!] & = & \texttt{Bool} \\
[\![\Pi;\Sigma \vdash A \times B]\!] & = & \texttt{Prod}[\![\Pi;\Sigma \vdash A]\!],[\![\Pi;\Sigma \vdash B]\!] \\
[\![\Pi;\Sigma \vdash A \stackrel{\varepsilon}{\to} B]\!] & = & \texttt{Arr}[\![\Pi;\Sigma \vdash A]\!],[\![\Pi;\Sigma \vdash B,\varepsilon]\!])
\end{array}
$$

Problem: It is not clear whether $[\![\ldots]\!]$ satisfying these exists!

We can show existence for a special case: latent effects of stored functions "storable", i.e. both rd_ℓ, wr_ℓ or ℓ not mentioned at all.

Logical relation cont'd

$$
\begin{array}{rcl}
[\![\Pi;\Sigma\vdash \texttt{unit}]\!] & = & \texttt{Unit} \\
[\![\Pi;\Sigma\vdash \texttt{int}]\!] & = & \texttt{Int} \\
[\![\Pi;\Sigma\vdash \texttt{bool}]\!] & = & \texttt{Bool} \\
[\![\Pi;\Sigma\vdash A\times B]\!] & = & \texttt{Prod}[\![\Pi;\Sigma\vdash A]\!],[\![\Pi;\Sigma\vdash B]\!] \\
[\![\Pi;\Sigma\vdash A\xrightarrow{\varepsilon} B]\!] & = & \texttt{Arr}[\![\Pi;\Sigma\vdash A]\!],[\![\Pi;\Sigma\vdash B,\varepsilon]\!])
\end{array}
$$

Problem: It is not clear whether $[\![\ldots]\!]$ satisfying these exists!

We can show existence for a special case: latent effects of stored functions "storable", i.e. both rd_ℓ, wr_ℓ or ℓ not mentioned at all.

We can "define" log.rel. even for dynamic allocation

Hereditarily pure

Consider

$$\mathbf{V} \cong \mathbf{V} \times \mathbf{V} \to (\mathbf{V} \times \mathbf{V})_\bot$$

models untyped functional programs with one global reference.
Retracts:

$$\begin{array}{rcl} p_0(f)(s,x) & = & \bot \\ p_{i+1}(f)(s,x) & = & \bot,\text{ if } f(p_i(s),p_i(x)) = \bot \\ p_{i+1}(f)(s,x) & = & (p_i(s_1),p_i(y)),\text{ if} \\ & & \qquad f(p_i(s),p_i(x)) = (s_1,y) \end{array}$$

We seek $P \subseteq \mathbf{V}$ such that:

$$\begin{array}{l} f \in P \iff \forall x \in P.\ (\forall s \in \mathbf{V}.f(s,x) = \bot)\vee \\ \qquad (\exists u \in P.\forall s \in \mathbf{V}.f(s,x) = (s,u)) \end{array}$$

Does such P exist?

Problem with existing solution theory

A. Pitts (1996) ("minimal invariants"): Essentially define $P_i := P \cap \mathrm{Im}(p_i)$ by induction on i. Then define $P = \{x \mid \forall i. p_i(x) \in P_i\}$.

Problem: the predicate P so obtained is closed under the p_i.
However, $\mathit{fun}(\mathit{id})$ should be in P, yet $\mathit{fun}(p_i) = p_i(\mathit{fun}()\mathit{id})$ should not.
Projecting down the store isn't "pure".

Our solution

Replace the p_i with q_i given by:

$$\begin{array}{rcl} q_0(f)(s,x) & = & \bot \\ q_{i+1}(f)(s,x) & = & \bot, \text{ if } f(s, q_i(x)) = \bot \\ q_{i+1}(f)(s,x) & = & (s_1, q_i(y)), \text{ if } f(s, q_i(x)) = (s_1, y) \end{array}$$

We can thus establish the existence of P.
This also allows us to establish the existence of the desired logical relation.

Challenge: Hereditarily read only commands

Consider $\mathbf{V} \cong \mathbf{V} \to \mathbf{V}_\perp$.
Think of $f : \mathbf{V} \to \mathbf{V}_\perp$ as stateful function of type `unit->unit` ("command") manipulating single untyped reference.
We want to single out hereditarily read only, i.e., define P such that

$$f \in P \iff \forall x \in P. f\ x \in \{x, \perp\}$$

Note that $\nabla = \lambda x.xx$ would be in P if P exists.

Not all predicates exist!

Same predomain **V** as before. Want to define "hereditarily total":

$$f \in T \iff \forall x{\in}T. f(x) \neq \bot \wedge f(x) \in T$$

Not all predicates exist!

Same predomain **V** as before. Want to define "hereditarily total":

$$f \in T \iff \forall x {\in} T. f(x) \neq \bot \wedge f(x) \in T$$

If T existed then $\nabla \in T$, yet $\nabla\nabla = \bot$. A contradiction.

Conclusion

- Slogan "Boldly define mixed-variance predicates and appeal to "minimal invariants" is dangerous.
- Open problem: Existence of hereditarily read-only.
- If we succeed in showing existence: we obtain powerful equational theory to reason about effectful programs.
- Partial solution: global references with restriction on effects of stored functions.