Refining Effects with Relations

Refined Monads and Effect Systems

- Gifford and Lucassen (86,88). Expressions get both a type and an *effect* – a safe, static overapproximation of possible side effects

- CBV translation

$$A, B ::= \mathrm{int} \mid A \to B \mid T_\varepsilon A$$

$$(\Gamma \vdash M : \sigma)^* = \Gamma^* \vdash M^* : T_\varepsilon \sigma^*$$

$$(\sigma \to \tau)^* = \sigma \to T_\varepsilon \tau^*$$

$$\left(\frac{\Gamma, x : \tau \vdash M : \sigma}{\Gamma \vdash (\lambda x : \tau . M) : \tau \to \sigma} \right)^* = \cfrac{\cfrac{\Gamma^*, x : \tau^* \vdash M^* : T_\varepsilon \sigma^*}{\Gamma^* \vdash (\lambda x : \tau^* . M^*) : \tau^* \to T_\varepsilon \sigma^*}}{\Gamma^* \vdash val(\lambda x : \tau^* . M^*) : T_\emptyset(\tau^* \to T_\varepsilon \sigma^*)}$$

- Effect system

$$\sigma, \tau ::= \mathrm{int} \mid \sigma \xrightarrow{\varepsilon} \tau$$

$$\Gamma \vdash M : \sigma, \varepsilon$$

$$\frac{\Gamma, x : \tau \vdash M : \sigma, \varepsilon}{\Gamma \vdash (\lambda x{:}\tau . M) : \tau \xrightarrow{\varepsilon} \sigma, \emptyset}$$

Monads vs. Effect Systems

- Monads:

$$\frac{\Gamma^* \vdash M^* : T_{\varepsilon}(\tau^* \to T_{\varepsilon'}\sigma^*) \qquad \dfrac{\Gamma^* \vdash N^* : T_{\varepsilon''}\tau^* \qquad \Gamma^*, f : \tau^* \to T_{\varepsilon'}\sigma^*, x : \tau^* \vdash f\ x : T_{\varepsilon'}\sigma^*)}{\Gamma^*, f : \tau^* \to T_{\varepsilon'}\sigma^* \vdash let\ x \Leftarrow N\ in\ f\ x : T_{\varepsilon'' \cup \varepsilon'}\sigma^*}}{\Gamma^* \vdash let\ f \Leftarrow M^*\ in\ (let\ x \Leftarrow N^*\ in\ f\ x) : T_{\varepsilon \cup \varepsilon' \cup \varepsilon''}\sigma^*}$$

- Effect system:

$$\frac{\Gamma \vdash M : \tau \xrightarrow{\varepsilon'} \sigma, \varepsilon \qquad \Gamma \vdash N : \tau, \varepsilon''}{\Gamma \vdash (M\ N) : \sigma, \varepsilon \cup \varepsilon' \cup \varepsilon''}$$

Effect-Refined Monadic Intermediate Languages

- Tolmach 98
- Wadler 98
- B, Kennedy, Russell 98
  - Implemented in MLJ Standard ML to Java bytecode compiler
  - Tracks reading, writing, allocating, exceptions, divergence

- First go at (extensional) correctness in HOOTS'99
  - Heavy operational techniques (Howe's method)
  - Based on sets of tests expressed in the language
  - Worked, but pretty icky

Tracking exceptions

Framework

Base type system

Base semantics
(sets and functions)

$$\begin{tikzcd} \Gamma \vdash \mathrm{M} : \mathrm{A} \arrow[r] & [\![\Gamma \vdash \mathrm{M} : \mathrm{A}]\!] \arrow[d, "\text{“in diagonal of”}"] \\ \Theta \vdash \mathrm{M} : \mathrm{X} \arrow[u, "\text{“erases to”}"] \arrow[r] & [\![\Theta \vdash \mathrm{M} : \mathrm{X}]\!] \end{tikzcd}$$

Refined type system
(effect annotation,
subtyping)

Refined semantics
(partial equivalence relations
over base semantics)

Store Effects

- What does it actually *mean* to "read" or "write"? Let *f* be the denotation of a command i.e. $f \in \mathbb{Z}\times\mathbb{Z}\to\mathbb{Z}\times\mathbb{Z}$.
- Suppose C does not *write* to the first location.
  Extensionally: there is some $g : \mathbb{Z}\times\mathbb{Z}\to\mathbb{Z}$ such that $f(x,y) = (x,g(x,y))$
- Suppose C does not *read or write* the first location.
  Extensionally: there is some $g : \mathbb{Z}\to\mathbb{Z}$ such that $f(x,y) = (x,g(y))$
- Suppose C does not *read* from the first location.
  Extensionally: there is some $h : \mathbb{Z}\to\mathbb{B}$, $g_1,g_2 : \mathbb{Z}\to\mathbb{Z}$ such that

$$f(x,y) = (h(y)\ ?\ x : g_1(y), g_2(y))$$

Observation

- Move to a *relational* interpretation, and things look much slicker, if slightly mysterious:
  Δ = diagonal relation , $\times$ and $\rightarrow$ usual constructions on relations
  $f : R$ shorthand for $(f,f) \in R$
- C does not write to the first location:
  $$\forall\, R \subseteq \Delta.\ f : R \times \Delta \rightarrow R \times \Delta.$$
- C does not read or write the first location:
  $$\forall\, R.\ f : R \times \Delta \rightarrow R \times \Delta.$$
- C does not read from the first location:
  $$\forall\, R \supseteq \Delta.\ f : R \times \Delta \rightarrow R \times \Delta.$$

Framework

Base type system

Base semantics
(sets and functions)

$$\begin{tikzcd} \Gamma \vdash M : A \arrow[r] & [\![\, \Gamma \vdash M : A \,]\!] \arrow[d, "\text{“in diagonal of”}"] \\ \Theta \vdash M : X \arrow[u, "\text{“erases to”}"] \arrow[r] & [\![\, \Theta \vdash M : X \,]\!] \end{tikzcd}$$

Refined type system
(effect annotation,
subtyping)

Refined semantics
(partial equivalence relations
over base semantics)

Base language

- Types

$$
\begin{array}{rcl}
A, B & := & \mathtt{unit} \mid \mathtt{int} \mid \mathtt{bool} \mid A \times B \mid A \to TB \\
\Gamma & := & x_1 : A_1, \ldots, x_n : A_n
\end{array}
$$

- Terms

$$
\begin{array}{rcl}
V, W & := & () \mid n \mid b \mid (V, W) \mid \lambda X : A.M \mid V + W \mid \pi_i V \mid \ldots \\
M, N & := & \mathtt{val}\, V \mid \mathtt{let}\, x \Leftarrow M \;\mathtt{in}\; N \mid V\, W \\
 & & \mid \mathtt{if}\ V\ \mathtt{then}\ M\ \mathtt{else}\ N \mid \mathtt{read}\,\ell \mid \mathtt{write}(\ell, V)
\end{array}
$$

Standard typing rules

$$\frac{\Gamma \vdash V_1 : A \quad \Gamma \vdash V_2 : B}{\Gamma \vdash (V_1, V_2) : A \times B} \qquad \frac{\Gamma \vdash V : A_1 \times A_2}{\Gamma \vdash \pi_i\, V : A_i}$$

$$\frac{\Gamma, x : A \vdash M : TB}{\Gamma \vdash \lambda x : A.M : A \to TB} \qquad \frac{\Gamma \vdash V_1 : A \to TB \quad \Gamma \vdash V_2 : A}{\Gamma \vdash V_1\, V_2 : TB}$$

$$\frac{\Gamma \vdash V : A}{\Gamma \vdash \mathtt{val}\, V : TA} \qquad \frac{\Gamma \vdash M : TA \quad \Gamma, x : A \vdash N : TB}{\Gamma \vdash \mathtt{let}\; x \Leftarrow M\; \mathtt{in}\; N : TB}$$

$$\frac{\Gamma \vdash V : \mathtt{bool} \quad \Gamma \vdash M : TA \quad \Gamma \vdash N : TA}{\Gamma \vdash \mathtt{if}\; V\; \mathtt{then}\; M\; \mathtt{else}\; N : TA}$$

$$\frac{}{\Gamma \vdash \mathtt{read}(\ell) : T\mathtt{int}} \qquad \frac{\Gamma \vdash V : \mathtt{int}}{\Gamma \vdash \mathtt{write}(\ell, V) : T\mathtt{unit}}$$

Base semantics in Set

$$
\begin{array}{rcl}
S & = & \mathrm{Locs} \to \mathbb{Z} \\
[\![\mathtt{unit}]\!] & = & 1 \\
[\![\mathtt{int}]\!] & = & \mathbb{Z} \\
[\![\mathtt{bool}]\!] & = & \mathbb{B} \\
[\![A \times B]\!] & = & [\![A]\!] \times [\![B]\!] \\
[\![A \to TB]\!] & = & [\![A]\!] \to [\![TB]\!] \\
[\![TA]\!] & = & S \to S \times [\![A]\!]
\end{array}
$$

Refined types and subtyping

- Types

$$
\begin{array}{rcl}
X, Y & := & \texttt{unit} \mid \texttt{int} \mid \texttt{bool} \mid X \times Y \mid X \to T_\varepsilon Y \\
\Theta & := & x_1 : X_1, \ldots, x_n : X_n \\
\varepsilon & \subseteq & \bigcup_{\ell \in \mathcal{L}} \{\texttt{r}_\ell, \texttt{w}_\ell\}
\end{array}
$$

- Subtyping

$$
\frac{}{X \leq X} \qquad \frac{X \leq Y \quad Y \leq Z}{X \leq Z} \qquad \frac{X \leq X' \quad Y \leq Y'}{X \times Y \leq X' \times Y'}
$$

$$
\frac{X' \leq X \quad T_\varepsilon Y \leq T_{\varepsilon'} Y'}{(X \to T_\varepsilon Y) \leq (X' \to T_{\varepsilon'} Y')} \qquad \frac{\varepsilon \subseteq \varepsilon' \quad X \leq X'}{T_\varepsilon X \leq T_{\varepsilon'} X'}
$$

Selected typing rules for refined types

$$\frac{\Theta, x : X \vdash M : T_\varepsilon Y}{\Theta \vdash \lambda x : U(X).M : X \to T_\varepsilon Y} \qquad \frac{\Theta \vdash V_1 : X \to T_\varepsilon Y \quad \Theta \vdash V_2 : X}{\Theta \vdash V_1\, V_2 : T_\varepsilon Y}$$

$$\frac{\Theta \vdash V : X}{\Theta \vdash \texttt{val}\, V : T_\emptyset X} \qquad \frac{\Theta \vdash M : T_\varepsilon X \quad \Theta, x : X \vdash N : T_{\varepsilon'} Y}{\Theta \vdash \texttt{let}\, x \Leftarrow M\, \texttt{in}\, N : T_{\varepsilon \cup \varepsilon'} Y}$$

$$\frac{\Theta \vdash V : \texttt{bool} \quad \Theta \vdash M : T_\varepsilon X \quad \Theta \vdash N : T_\varepsilon X}{\Theta \vdash \texttt{if}\ V\ \texttt{then}\ M\ \texttt{else}\ N : T_\varepsilon X}$$

$$\frac{}{\Theta \vdash \texttt{read}(\ell) : T_{\{\texttt{r}_\ell\}}(\texttt{int})} \qquad \frac{\Theta \vdash V : \texttt{int}}{\Theta \vdash \texttt{write}(\ell, V) : T_{\{\texttt{w}_\ell\}}(\texttt{unit})}$$

$$\frac{\Theta \vdash V : X \quad X \leq X'}{\Theta \vdash V : X'} \qquad \frac{\Theta \vdash M : T_\varepsilon X \quad T_\varepsilon X \leq T_{\varepsilon'} X'}{\Theta \vdash M : T_{\varepsilon'} X'}$$

Semantics of refined types

$$
\begin{array}{rcl}
[\![X]\!] & \subseteq & [\![U(X)]\!] \times [\![U(X)]\!] \\
[\![\mathtt{int}]\!] & = & \Delta_{\mathbb{Z}} \\
[\![\mathtt{bool}]\!] & = & \Delta_{\mathbb{B}} \\
[\![\mathtt{unit}]\!] & = & \Delta_{1} \\
[\![X \times Y]\!] & = & [\![X]\!] \times [\![Y]\!] \\
[\![X \to T_\varepsilon Y]\!] & = & [\![X]\!] \to [\![T_\varepsilon Y]\!] \\
[\![T_\varepsilon X]\!] & = & \bigcap_{R \in \mathcal{R}_\varepsilon} R \to R \times [\![X]\!]
\end{array}
$$

Values of base type are related just to themselves (diagonal relation)

Functions are related in the usual "logical" fashion: related arguments $\rightarrow$ related results

Computations are related if they preserve all state relations that respect the effect

$$
\begin{array}{rcl}
\mathcal{R}_\varepsilon, \mathcal{R}_e & \subseteq & \mathbb{P}(S \times S) \\
\mathcal{R}_\varepsilon & = & \bigcap_{e \in \varepsilon} \mathcal{R}_e \\
\mathcal{R}_{\mathtt{r}_\ell} & = & \{R \mid \forall (s, s') \in R,\ s\,\ell = s'\,\ell\} \\
\mathcal{R}_{\mathtt{w}_\ell} & = & \{R \mid \forall (s, s') \in R,\ \forall n \in \mathbb{Z}.\, (s[\ell \mapsto n], s'[\ell \mapsto n]) \in R\}
\end{array}
$$

Results

- Soundness of subtyping: If $X \leq Y$ then $[\![X]\!] \subseteq [\![Y]\!]$.
- Fundamental theorem:

$$\begin{array}{l}\text{If } \Theta \vdash V : X, (\rho, \rho') \in [\![\Theta]\!] \\ \text{then } ([\![U(\Theta) \vdash V : U(X)]\!]\, \rho,\ [\![U(\Theta) \vdash V : U(X)]\!]\, \rho') \in [\![X]\!].\end{array}$$

- Meaning of top effect: $[\![G(A)]\!] = \Delta_{[\![A]\!]}$.
- Equivalences
  - Effect-independent: congruence rules, β, η rules, commuting conversions
  - Effect-dependent: dead computation, duplicated computation, commuting computations, pure lambda hoist
  - Reasoning is quite intricate, involving construction of specific effect-respecting relations.

Effect-dependent equivalences (1)

Dead Computation:

$$\frac{\Theta \vdash M : T_\varepsilon X \quad \Theta \vdash N : T_{\varepsilon'} Y}{\Theta \vdash \mathtt{let}\, x \Leftarrow M \,\mathtt{in}\, N = N : T_{\varepsilon'} Y} \, x \notin \Theta, \mathrm{wrs}(\varepsilon) = \emptyset$$

Duplicated Computation:

$$\frac{\Theta \vdash M : T_\varepsilon X \quad \Theta, x : X, y : X \vdash N : T_{\varepsilon'} Y}{\Theta \vdash \begin{array}{cl} & \mathtt{let}\, x \Leftarrow M \,\mathtt{in}\, \mathtt{let}\, y \Leftarrow M \,\mathtt{in}\, N \\ = & \mathtt{let}\, x \Leftarrow M \,\mathtt{in}\, N[x/y] \end{array} : T_{\varepsilon \cup \varepsilon'} Y} \, \mathrm{rds}(\varepsilon) \cap \mathrm{wrs}(\varepsilon) = \emptyset$$

Effect-dependent equivalences (2)

Commuting Computations:

$$\frac{\Theta \vdash M_1 : T_{\varepsilon_1} X_1 \quad \Theta \vdash M_2 : T_{\varepsilon_2} X_2 \quad \Theta, x_1 : X_1, x_2 : X_2 \vdash N : T_{\varepsilon'} Y}{\Theta \vdash \begin{array}{ll} & \texttt{let}\, x_1 \Leftarrow M_1 \,\texttt{in}\, \texttt{let}\, x_2 \Leftarrow M_2 \,\texttt{in}\, N \\ = & \texttt{let}\, x_2 \Leftarrow M_2 \,\texttt{in}\, \texttt{let}\, x_1 \Leftarrow M_1 \,\texttt{in}\, N \end{array} : T_{\varepsilon_1 \cup \varepsilon_2 \cup \varepsilon'} Y} \quad \begin{array}{l} \mathrm{rds}(\varepsilon_1) \cap \mathrm{wrs}(\varepsilon_2) = \emptyset \\ \mathrm{wrs}(\varepsilon_1) \cap \mathrm{rds}(\varepsilon_2) = \emptyset \\ \mathrm{wrs}(\varepsilon_1) \cap \mathrm{wrs}(\varepsilon_2) = \emptyset \end{array}$$

Pure Lambda Hoist:

$$\frac{\Theta \vdash M : T_{\{\}} Z \quad \Theta, x : X, y : Z \vdash N : T_{\varepsilon} Y}{\Theta \vdash \begin{array}{ll} & \texttt{val}\, (\lambda x : U(X).\texttt{let}\, y \Leftarrow M \,\texttt{in}\, N) \\ = & \texttt{let}\, y \Leftarrow M \,\texttt{in}\, \texttt{val}\, (\lambda x : U(X).N) \end{array} : T_{\{\}}(X \to T_{\varepsilon} Y)}$$

A language with dynamic allocation

- Axiomatic (abstract) treatment of state equipped with dom, lookup, update and new
- Set of *regions* Regs, refined types include ref_r
- $\epsilon \subseteq \{rd_r, wr_r, al_r \mid r \in \mathrm{Regs}\}$

$$\frac{\Gamma(x) = \mathtt{int}}{\Gamma \vdash \mathtt{ref}(x) : \mathtt{ref}_{\mathsf{r}} : \{al_{\mathsf{r}}\}} \quad \frac{\Gamma(x) = \mathtt{ref}_{\mathsf{r}} \quad \Gamma(y) = \mathtt{int}}{\Gamma \vdash x{:=}y : \mathtt{unit}, \{wr_{\mathsf{r}}\}}$$

$$\frac{\Gamma \vdash e : A, \varepsilon \quad \mathsf{r} \text{ does not occur in } \Gamma \text{ or } A}{\Gamma \vdash e : A, \varepsilon \setminus \{wr_{\mathsf{r}}, rd_{\mathsf{r}}, al_{\mathsf{r}}\}}$$

Parametric Logical Relation

- A parameter φ assigns every $r \in$ Regs $\cup \{\tau\}$ a finite partial bijection on $\mathbb{L}$ (all disjoint)
- State relation on $L, L' \subseteq \mathbb{L}$ is $R \subseteq S \times S$ st. sRs', $s \sim_L s_1$, $s' \sim_{\mathcal{L}'} s_1' \Rightarrow s_1 R s_1'$
- If R state relation on dom(φ), dom'(φ) then
  - R respects $\{rd_r\}$ at φ if $sRs' \Rightarrow s.l = s'.l'$ $\forall (l,l') \in \varphi(r)$
  - R respects $\{wr_r\}$ at φ if sRs', $(l,l') \in \varphi(r)$, $v \in \mathbb{Z} \Rightarrow s[l \mapsto v]Rs[l' \mapsto v]$
  - R respects $\{al_r\}$ always
- $\mathcal{R}_\epsilon(\varphi) = \{R \in \mathrm{StRel}(\mathrm{dom}(\varphi), \mathrm{dom}'(\varphi))$ st. $\forall e \in \epsilon$, R resp. e at $\varphi\}$
- $[\![A]\!]_\varphi$ will be a QPER on $[\![|A|]\!]$
  - Relation R such that $R;R^{-1};R = R$

$$
\begin{aligned}
[\![A]\!]_\varphi &\equiv \{(v,v) \mid v \in [\![|A|]\!]\} \text{ when } A \in \{\texttt{int}, \texttt{bool}, \texttt{unit}\} \\
[\![\texttt{ref}_{\mathsf{r}}]\!]_\varphi &\equiv \varphi(\mathsf{r}) \\
[\![A \times B]\!]_\varphi &\equiv [\![A]\!]_\varphi \times [\![B]\!]_\varphi \\
[\![A \xrightarrow{\varepsilon} B]\!]_\varphi &\equiv \{(f, f') \mid \forall\ \varphi' \geq \varphi. \forall (x, x') \in [\![A]\!]_{\varphi'}. \\
&\qquad (f(x), f'(x')) \in (T_\varepsilon [\![B]\!])_{\varphi'}\} \\
(T_\varepsilon Q)_\varphi &\equiv QPER(\{(f, f') \mid s, s' \models \varphi \Rightarrow \\
&\quad \forall\ R \in \mathcal{R}_\varepsilon(\varphi).s\ R\ s' \Rightarrow s_1\ R\ s'_1\ \ \wedge \\
&\quad \exists \psi.(\psi(\mathsf{r}) \neq \emptyset \Rightarrow \mathsf{r} \in \mathrm{als}(\varepsilon))\ \ \wedge s_1, s'_1 \models \varphi \otimes \psi \wedge \\
&\quad s_1 \sim_\psi s'_1 \wedge (v, v') \in Q_{\varphi \otimes \psi} \\
&\quad \text{where } (s_1, v) {=} f\ s \text{ and } (s'_1, v') {=} f'\ s'\})
\end{aligned}
$$

- Monotonicity: $\varphi' \geq \varphi \Rightarrow [\![A]\!]_{\varphi'} \supseteq [\![A]\!]_{\varphi}$
- Masking: If r not in A, $[\![A]\!]_{\varphi}$ = $[\![A]\!]_{\varphi - \mathbf{r}}$ where $\varphi -$ r moves φ(r) into the silent region $\boldsymbol{\tau}$
- Fundamental theorem
- Semantic equality, quantifying over parameters, is PER and yields same equations except duplicated computations requires no allocations as well as disjoint reads and writes

Conclusion

- Relational parametricity can give elegant, useful extensional semantics to effect systems
- Related
  - Fancier system for exceptions
  - Global higher order
  - Abstract regions
- Can be (should be) extended to regular effects or other transition systems
- Note emphasis on operations. In retrospect this is what we were doing all along.

Extensional Semantics for Program Analyses and Optimising Transformations

- Program analysis and optimising transformations *ought* to be a killer app for semantics
  - *"An assignment [x := a]^l* may reach *a certain program point if there is an execution of the program where x was last assigned a value at l when the program point is reached."*
  - *"...by the standard technique of proving preservation and progress"*
  - *"denotational and operational methods seem ill-suited to validating transformations that involve a program's computational future or computational past"*
- This seems wrong
  - Confusion of semantics of analysis results (true precondition for transformation) with syntactic, approximate technique used to obtain them
  - Original and transformed programs equal in standard, extensional semantics; surely the *reason* why is expressible in those terms too
  - Instrumented semantics both a cheat and a poor basis for equational reasoning
  - Doesn't go through abstraction levels (machine code programs don't go wrong)

Relational Semantics for Traditional Dataflow and Transformations

Type system mapping variables to the total relation, the diagonal or singleton {(n,n)}, interpreted as pre- and post-relations on stores, captures constant propagation, dead code elimination, slicing and Smith/Volpano style secure information flow.

$$(\texttt{if } X = 3 \texttt{ then } X\!:=\!7 \texttt{ else skip}; Z\!:=\!X+1) \sim (Z\!:=\!8) \quad : \Phi, X : \{3\}, Z : \mathbb{T}_{\texttt{int}} \Rightarrow \Phi, X : \mathbb{T}_{\texttt{int}}, Z : \{8\}$$

Generalizes to "Relational Hoare Logic", which can deal with code motion, available expressions and other flow-sensitive analyses

```
while I<N do                X := Y+1;
  X := Y+1;      ==>        while I<N do
  I := I+X;                   I := I+X;
```

at type $\Phi \Rightarrow \Phi$ where Φ is $I\langle 1\rangle = I\langle 2\rangle \wedge N\langle 1\rangle = N\langle 2\rangle \wedge Y\langle 1\rangle = Y\langle 2\rangle$

Separation logic version of this idea used in establishing semantic type safety of a compiler. Probabilistic version (CertiCrypt) by Barthe, Zanella et al used to establish impressive results on correctness of crypto protocols