Monads in Action

Andrzej Filinski

Department of Computer Science (DIKU)
University of Copenhagen

Principles of Programming Languages
January 20–22, 2010, Madrid

Background and overview

- Two approaches to uniformly specifying computational effects:
  1. Monad of computations + pure definitions of operators
     - Translate client program using monad components, plug in operator definitions, evaluate by core semantics
     - Typically used in Haskell-like settings
  2. Stylized evaluation contexts + context-rewriting operators
     - Formalize context shapes, extend core semantics with new rules for effectful operators
     - Typically used in ML/Scheme-like settings
- Can we generically derive (2) from (1)
  - ... without giving up the monadic equational theory?
- In paper: details for full multimonadic metalanguage
  - Here: for single effect only; expressed in Haskell subset (slightly oversimplified)
  - There is a complementary story for ML-like settings

Monadic Haskell with *Int*-exceptions

$$
\begin{array}{lcl}
t & ::= & \overbrace{\mathit{Int} \mid t_1 \to t_2 \mid \mathit{Either}\ t_1\ t_2}^{\mathit{Core\ lang.}} \mid M_\varepsilon\ t \\
\varepsilon & ::= & \mathit{id} \mid \mathit{ex} \mid \cdots \\
e & ::= & \left.\begin{array}{l} n \mid x \mid \lambda x \to e \mid e_1\ e_2 \mid \mathit{Left}\ e \mid \mathit{Right}\ e \\ \mid \mathbf{case}\ e_0\ \mathbf{of}\ \{\mathit{Left}\ x \to e_1; \mathit{Right}\ y \to e_2\} \end{array}\right\} \mathit{Core\ lang.} \\
 & & \mid \mathbf{return}^\varepsilon\ e \mid \mathbf{do}^\varepsilon\ x \leftarrow e_1; e_2 \mid \mathbf{raise}\ e \mid \mathbf{try}^\varepsilon\ e_1\ \mathbf{with}\ x \to e_2
\end{array}
$$

Typing judgment $\boxed{\Gamma \vdash e : t}$. Usual rules for core constructs +

$$\frac{\Gamma \vdash e : t}{\Gamma \vdash \mathbf{return}^\varepsilon\ e : M_\varepsilon\ t} \qquad \frac{\Gamma \vdash e_1 : M_\varepsilon\ t_1 \quad \Gamma, x : t_1 \vdash e_2 : M_\varepsilon\ t_2}{\Gamma \vdash \mathbf{do}^\varepsilon\ x \leftarrow e_1; e_2 : M_\varepsilon\ t_2}$$

$$\frac{\Gamma \vdash e : \mathit{Int}}{\Gamma \vdash \mathbf{raise}\ e : M_{ex}\ t} \qquad \frac{\Gamma \vdash e_1 : M_{ex}\ t \qquad \Gamma, x : \mathit{Int} \vdash e_2 : M_\varepsilon\ t}{\Gamma \vdash \mathbf{try}^\varepsilon\ e_1\ \mathbf{with}\ x \to e_2 : M_\varepsilon\ t}$$

Superscripts on **return**, **do**, **try**: from overloading resolution.

Specifying exceptions with a monad

Transparent/*concrete* definition of exception monad:

$$\textbf{type}\ T_{ex}\ a = Either\ a\ Int$$

$$unit_{ex} :: a \to T_{ex}\ a$$
$$unit_{ex}\ a = Left\ a$$

$$bind_{ex} :: T_{ex}\ a \to (a \to T_{ex}\ b) \to T_{ex}\ b$$
$$bind_{ex}\ t\ f = \textbf{case}\ t\ \textbf{of}\ \{Left\ a \to f\ a; Right\ n \to Right\ n\}$$

Used to implement *abstract* effect of exceptions:

$$\textbf{newtype}\ M_\varepsilon\ a = Reflect_\varepsilon\ (T_\varepsilon\ a) \qquad Reflect_\varepsilon :: T_\varepsilon\ a \to M_\varepsilon\ a$$
$$reify_\varepsilon\ (Reflect_\varepsilon\ t) = t \qquad reify_\varepsilon :: M_\varepsilon\ a \to T_\varepsilon\ a$$

$$\textbf{return}^\varepsilon\ e = Reflect_\varepsilon\ (unit_\varepsilon\ e)$$
$$\textbf{do}^\varepsilon\ x \leftarrow e_1; e_2 = Reflect_\varepsilon\ (bind_\varepsilon\ (reify_\varepsilon\ e_1)\ (\lambda x \to reify_\varepsilon\ e_2))$$
$$\textbf{raise}\ e \equiv Reflect_{ex}\ (Right\ e) \qquad \text{-- definitional abbrevs.}$$
$$\textbf{try}^\varepsilon\ e_1\ \textbf{with}\ x \to e_2 \equiv$$
$$\quad \textbf{case}\ reify_{ex}\ e_1\ \textbf{of}\ \{Left\ a \to \textbf{return}^\varepsilon\ a; Right\ x \to e_2\}$$

Standard (JIT-translational) operational semantics

$$e ::= (core) \mid \mathbf{return}^{\varepsilon}\ e \mid \mathbf{do}^{\varepsilon}\ x \leftarrow e_1; e_2 \mid \mathit{reify}_{\varepsilon}\ e \mid \mathit{Reflect}_{\varepsilon}\ e$$

Reduction judgment $\boxed{e \longrightarrow e'}$ for *closed* terms:

$$\frac{e_1 \longrightarrow e_1'}{e_1\ e_2 \longrightarrow e_1'\ e_2} \qquad \frac{}{(\lambda x \to e_1)\ e_2 \longrightarrow e_1[e_2/x]}$$

$$\frac{e_0 \longrightarrow e_0'}{\mathbf{case}\ e_0\ \mathbf{of}\ \{\cdots\} \longrightarrow \mathbf{case}\ e_0'\ \mathbf{of}\ \{\cdots\}}$$

$$\frac{}{\mathbf{case}\ \mathit{Left}\ e\ \mathbf{of}\ \{\mathit{Left}\ x_1 \to e_1; \mathit{Right}\ x_2 \to e_2\} \longrightarrow e_1[e/x]}\ (+\mathit{symm})$$

$$\frac{}{\mathbf{return}^{\varepsilon}\ e \longrightarrow \mathit{Reflect}_{\varepsilon}\ (\mathit{unit}_{\varepsilon}\ e)}$$

$$\frac{}{\mathbf{do}^{\varepsilon}\ x \leftarrow e_1; e_2 \longrightarrow \mathit{Reflect}_{\varepsilon}\ (\mathit{bind}_{\varepsilon}\ (\mathit{reify}_{\varepsilon}\ e_1)\ (\lambda x \to \mathit{reify}_{\varepsilon}\ e_2))}$$

$$\frac{e \longrightarrow e'}{\mathit{reify}_{\varepsilon}\ e \longrightarrow \mathit{reify}_{\varepsilon}\ e'} \qquad \frac{}{\mathit{reify}_{\varepsilon}\ (\mathit{Reflect}_{\varepsilon}\ e) \longrightarrow e}$$

Note: tags on $\mathbf{return}^{\varepsilon}$, $\mathbf{do}^{\varepsilon}$ play essential role in behavior.

Note: code for $\mathit{unit}_{\varepsilon}$, $\mathit{bind}_{\varepsilon}$ traversed on every $\mathbf{return}^{\varepsilon}$, $\mathbf{do}^{\varepsilon}$.

Equational theory

If $(T_\varepsilon, \mathit{unit}_\varepsilon, \mathit{bind}_\varepsilon)$ satisfy monad laws, get additional valid reasoning principles for observational equivalence:

$$
\begin{array}{rcl}
(\lambda x \to e_1)\, e_2 & = & e_1[e_2/x] \\
\lambda x \to e\, x & = & e \quad (x \notin FV(e)) \\
 & \vdots & \\
\mathbf{do}^\varepsilon\ x \leftarrow \mathbf{return}^\varepsilon\ e_1; e_2 & = & e_2[e_1/x] \\
\mathbf{do}^\varepsilon\ x \leftarrow e; \mathbf{return}^\varepsilon\ x & = & e \\
\mathbf{do}^\varepsilon\ y \leftarrow (\mathbf{do}^\varepsilon\ x \leftarrow e_1; e_2); e_3 & = & \mathbf{do}^\varepsilon\ x \leftarrow e_1; \mathbf{do}^\varepsilon\ y \leftarrow e_2; e_3 \quad (x \notin FV(e_3)) \\
\mathit{reify}_\varepsilon\, (\mathbf{return}^\varepsilon\ e) & = & \mathit{unit}_\varepsilon\, e \\
\mathit{reify}_\varepsilon\, (\mathbf{do}^\varepsilon\ x \leftarrow e_1; e_2) & = & \mathit{bind}_\varepsilon\, (\mathit{reify}_\varepsilon\, e_1)\, (\lambda x \to \mathit{reify}_\varepsilon\, e_2) \\
\mathit{reify}_\varepsilon\, (\mathit{Reflect}_\varepsilon\, e) & = & e \\
\mathit{Reflect}_\varepsilon\, (\mathit{reify}_\varepsilon\, e) & = & e
\end{array}
$$

Can we operationalize these equations in a different way?

New (effectful) operational semantics

$$e ::= (\mathit{core}) \mid \mathbf{return}\ e \mid \mathbf{do}\ x \leftarrow e_1; e_2 \mid \mathit{reify}_\varepsilon\ e \mid \mathbf{do}\ x \leftarrow \mathit{Reflect}_\varepsilon\ e_1; e_2$$

$$\mathit{reflect}_\varepsilon\ e \equiv \mathbf{do}\ x \leftarrow \mathit{Reflect}_\varepsilon\ e; \mathbf{return}\ x$$

$\boxed{e \longrightarrow e'}$

(Unmodified rules for core constructs)

$$\frac{e_1 \longrightarrow e_1'}{\mathbf{do}\ x \leftarrow e_1; e_2 \longrightarrow \mathbf{do}\ x \leftarrow e_1'; e_2}$$

$$\frac{}{\mathbf{do}\ x \leftarrow \mathbf{return}\ e_1; e_2 \longrightarrow e_2[e_1/x]}$$

$$\frac{}{\mathbf{do}\ y \leftarrow (\mathbf{do}\ x \leftarrow \mathit{Reflect}_\varepsilon\ e_1; e_2); e_3 \longrightarrow \mathbf{do}\ x \leftarrow \mathit{Reflect}_\varepsilon\ e_1; \mathbf{do}\ y \leftarrow e_2; e_3}\ (*)$$

$$\frac{e \longrightarrow e'}{\mathit{reify}_\varepsilon\ e \longrightarrow \mathit{reify}_\varepsilon\ e'} \qquad \frac{}{\mathit{reify}_\varepsilon\ (\mathbf{return}\ e) \longrightarrow \mathit{unit}_\varepsilon\ e}$$

$$\frac{}{\mathit{reify}_\varepsilon\ (\mathbf{do}\ x \leftarrow \mathit{Reflect}_\varepsilon\ e_1; e_2) \longrightarrow \mathit{bind}_\varepsilon\ e_1\ (\lambda x \to \mathit{reify}_\varepsilon\ e_2)}\ (*)$$

Marked rules: match up $\mathit{Reflect}_\varepsilon$ with nearest enclosing $\mathit{reify}_\varepsilon$.

Properties of reduction semantics

- **Sound:** if $e \longrightarrow e'$, then $e = e'$ in equational theory.
- **Deterministic:** if $e \longrightarrow e'$ and $e \longrightarrow e''$, then $e' \equiv e''$.
- **Type-preserving:** if $\cdot \vdash e : t$ and $e \longrightarrow e'$, then $\cdot \vdash e' : t$.
- **Progressing:** if $\cdot \vdash e : t$, then either e canonical, or $e \longrightarrow e'$. Canonical forms are:

$$\overbrace{n}^{Int} \mid \overbrace{\lambda x \longrightarrow e}^{t_1 \longrightarrow t_2} \mid \overbrace{Left\ e \mid Right\ e}^{Either\ t_1\ t_2} \mid \overbrace{\mathbf{return}\ e \mid \mathbf{do}\ x \leftarrow Reflect_\varepsilon\ e_1; e_2}^{M_\varepsilon\ t}$$

  Note: an M-computation is either finished, or an effect invocation.

- In particular, a closed term of type *Int* (but using monadic effects internally) must reduce to an n, or diverge.

Evaluation-context formulation of new semantics

General and restricted evaluation contexts:

$$E ::= [] \mid E\ e \mid \textbf{case}\ E\ \textbf{of}\ \{\cdots\} \mid \textbf{do}\ x \leftarrow E; e \mid \mathit{reify}_\varepsilon\ E$$
$$F ::= [] \mid \textbf{do}\ x \leftarrow F; e \qquad \text{(in particular, no } \mathit{reify}_\varepsilon\ F\text{)}$$

Bigger-step judgment $\boxed{e \longrightarrow\!\!\!\!\!\rightarrow e'}$:

$$\frac{e \longrightarrow\!\!\!\!\!\rightarrow e'}{E[e] \longrightarrow\!\!\!\!\!\rightarrow E[e']} \qquad \frac{}{(\lambda x \rightarrow e_1)\ e_2 \longrightarrow\!\!\!\!\!\rightarrow e_1[e_2/x]}$$

$$\frac{}{\textbf{case}\ \mathit{Left}\ e\ \textbf{of}\ \{\mathit{Left}\ x_1 \rightarrow e_1; \mathit{Right}\ x_2 \rightarrow e_2\} \longrightarrow\!\!\!\!\!\rightarrow e_1[e/x]}\ (+\mathit{symm})$$

$$\frac{}{\textbf{do}\ x \leftarrow \textbf{return}\ e_1; e_2 \longrightarrow\!\!\!\!\!\rightarrow e_2[e_1/x]} \qquad \frac{}{\mathit{reify}_\varepsilon\ (\textbf{return}\ e) \longrightarrow\!\!\!\!\!\rightarrow \mathit{unit}_\varepsilon\ e}$$

$$\frac{}{\mathit{reify}_\varepsilon\ (F[\mathit{reflect}_\varepsilon\ e]) \longrightarrow\!\!\!\!\!\rightarrow \mathit{bind}_\varepsilon\ e\ (\lambda x \rightarrow \mathit{reify}_\varepsilon\ (F[\textbf{return}\ x]))}\ (*)$$

Sound: if $e \longrightarrow\!\!\!\!\!\rightarrow e'$ then $e \longrightarrow^+ e'$.

Example: exceptions

$$
\begin{aligned}
T_{ex}\ a &= \mathit{Either}\ a\ \mathit{Int};\ \mathit{unit}_{ex}\ a = \mathit{Left}\ a;\ \mathit{bind}_{ex} = \cdots \\
\mathbf{raise}\ e &\equiv \mathit{reflect}_{ex}\ (\mathit{Right}\ e) \\
\mathbf{try}\ e_1\ \mathbf{with}\ x \to e_2 &\equiv \\
&\quad \mathbf{case}\ \mathit{reify}_{ex}\ e_1\ \mathbf{of}\ \{\mathit{Left}\ a \to \mathbf{return}\ a;\ \mathit{Right}\ x \to e_2\}
\end{aligned}
$$

Derivable typing and reduction rules:

$$\frac{\Gamma \vdash e : \mathit{Int}}{\Gamma \vdash \mathbf{raise}\ e : M_{ex}\ t} \qquad \frac{\Gamma \vdash e_1 : M_{ex}\ t \qquad \Gamma, x : \mathit{Int} \vdash e_2 : M_{\varepsilon}\ t}{\Gamma \vdash \mathbf{try}\ e_1\ \mathbf{with}\ x \to e_2 : M_{\varepsilon}\ t}$$

$$\frac{e_1 \longrightarrow e_1'}{\mathbf{try}\ e_1\ \mathbf{with}\ x \to e_2 \longrightarrow \mathbf{try}\ e_1'\ \mathbf{with}\ x \to e_2}$$

$$\frac{}{\mathbf{try}\ \mathbf{return}\ e_0\ \mathbf{with}\ x \to e_2 \longrightarrow^{+} \mathbf{return}\ e_0}$$

$$\frac{}{\mathbf{try}\ F[\mathbf{raise}\ e_0]\ \mathbf{with}\ x \to e_2 \longrightarrow^{+} e_2[e_0/x]}$$

Example: state

$$T_{st}\ a \quad = \quad Int \to (a, Int);\ unit_{st}\ a = \lambda s \to (a, s);\ bind_{st} = \cdots$$

$$\textbf{withst}\ e_1\ \textbf{do}\ e_2 \quad \equiv \quad \textbf{let}\ (a, s') = (reify_{st}\ e_2)\ e_1\ \textbf{in}\ a$$

$$\textbf{getst} \quad \equiv \quad reflect_{st}\ (\lambda s \to (s, s))$$

$$\textbf{setst}\ e \quad \equiv \quad reflect_{st}\ (\lambda s \to ((), e))$$

Derived typing and reduction rules:

$$\frac{\Gamma \vdash e_1 : Int \quad \Gamma \vdash e_2 : M_{st}\ t}{\Gamma \vdash \textbf{withst}\ e_1\ \textbf{do}\ e_2 : t} \qquad \frac{}{\Gamma \vdash \textbf{getst} : M_{st}\ Int} \qquad \frac{\Gamma \vdash e : Int}{\Gamma \vdash \textbf{setst}\ e : M_{st}\ ()}$$

$$\frac{e_2 \longrightarrow e_2'}{\textbf{withst}\ e_1\ \textbf{do}\ e_2 \longrightarrow \textbf{withst}\ e_1\ \textbf{do}\ e_2'} \qquad \frac{}{\textbf{withst}\ e_1\ \textbf{do return}\ e_2 \longrightarrow^{+} e_2}$$

$$\frac{}{\textbf{withst}\ e\ \textbf{do}\ F[\textbf{getst}] \longrightarrow^{+} \textbf{withst}\ e\ \textbf{do}\ F[\textbf{return}\ e]}$$

$$\frac{}{\textbf{withst}\ e\ \textbf{do}\ F[\textbf{setst}\ e'] \longrightarrow^{+} \textbf{withst}\ e'\ \textbf{do}\ F[\textbf{return}\ ()]}$$

Conclusions

- Monadic definitions of effects can be given *direct* operational interpretation; Curry-style type system.
- Independent reconstruction of evaluation-context semantics.
  - Related construction: taking implementation type M to be a delimited-continuations monad $\Rightarrow$ embedding arbitrary monadic effects in Scheme.
- In paper:
  - Full core language with product, sum, function, recursive, and generalized-effect types; effect-subtyping.
  - Explicit syntax for effect definitions with layering.
  - Precise formulation of semantics (explicit and context-based), type system, type soundness (all formalized in Twelf).
- Current work: correspondence between (domain-theoretic) denotational and operational semantics for monadic effects.