Liberating effects with rows and handlers

Sam Lindley

Laboratory for Foundations of Computer Science
The University of Edinburgh

Sam.Lindley@ed.ac.uk

March 31st, 2016

(joint work with Daniel Hillerström)

Motivation

A framework for modular programming with effects

- starting point: abstract computations over signatures of effectful operations $\{op_i : A_i \to B_i\}_i$
- modularity
  - interpreters for abstract computations defined in terms of interpretations of the underlying effectful operations
  - multiple interpreters
  - composable interpreters
- an *effect handler* is an interpreter for abstract computations
  - a *closed* effect handler interprets a fixed set of operations
  - an *open* effect handler interprets a fixed set of operations, and generically forwards all others (crucial for composition)

Abstract computations as trees

An abstract computation of type Comp E A over *effect signature* $E = \{op_i : A_i \to B_i\}_i$ with *return type* A is a tree where

- nodes are labelled with operations and operation parameters
- edges are labelled with operation result values
- leaves are labelled with final return values of type A

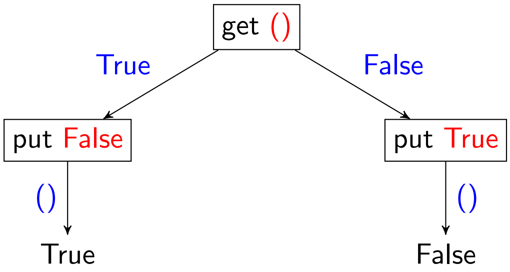
Algebraic effects [Plotkin and Power, 2001]: same story, but trees are quotiented by equations

Example: bit toggling

$$E = \{\text{get} : 1 \rightarrow \text{Bool}, \quad \text{put} : \text{Bool} \rightarrow 1\}$$

$$A = \text{Bool}$$

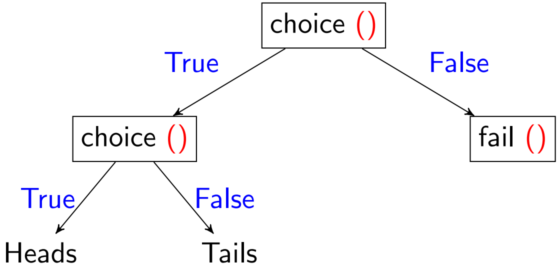
$$\text{toggle} = \textbf{let } x \leftarrow \text{get}() \textbf{ in } \text{put}(\neg x); x$$



Example: drunk coin toss

$$E = \{\text{choice} : 1 \to \text{Bool},\ \text{fail} : 1 \to 0\}$$
$$A = \text{Heads} + \text{Tails}$$

drunkToss =
  if choice() **then**
    if choice() **then** Heads
    else Tails
  else fail()



Effect handlers (Pretnar and Plotkin)

An effect handler is an interpreter for abstract computations of type

$$\mathsf{Comp}\ E\ A \to B$$

for some target type B (which may itself be an abstract computation over a different signature).

An effect handler is defined as a fold over a computation tree, specifying how return values and operations are interpreted.

$$\begin{array}{lcl} \mathbf{return}\, x & \mapsto & M \\ op_1\ p\ k & \mapsto & N_1 \\ \quad \ldots & & \\ op_n\ p\ k & \mapsto & N_n \end{array}$$

In each N_i, the variable k is bound to a function for invoking H on the subtrees of the current node.

Example: closed state handlers

$$\text{State}\, S = \{\text{get} : 1 \to S, \text{put} : S \to 1\}$$

$$\begin{aligned}
&\text{evalState} : \text{Comp}\ (\text{State}\, S)\ A \to (S \to A) \\
&\text{evalState} = \textbf{return}\, x \mapsto \lambda s.x \\
&\qquad\qquad\quad \text{get}\ ()\ k \quad \mapsto \lambda s.k\ s\ s \qquad\qquad (k{:}S{\to}(S{\to}A)) \\
&\qquad\qquad\quad \text{put}\ t\ k \quad \mapsto \lambda s.k\ ()\ t \qquad\qquad (k{:}1{\to}(S{\to}A))
\end{aligned}$$

$$\begin{aligned}
&\text{logState} : \text{Comp}\ (\text{State}\, S)\ A \to (S \to A \times \text{List}\, S) \\
&\text{logState} = \textbf{return}\, x \mapsto \lambda s.(x, [s]) \\
&\qquad\qquad\quad \text{get}\ ()\ k \quad \mapsto \lambda s.k\ s\ s \\
&\qquad\qquad\quad \text{put}\ t\ k \quad \mapsto \lambda s.\textbf{let}\ (x, ss) \leftarrow k\ ()\ t\ \textbf{in} \\
&\qquad\qquad\qquad\qquad\qquad\qquad (x, s :: ss)
\end{aligned}$$

$$(\textbf{handle}\ \text{toggle}\ \textbf{with}\ \text{evalState})\ \text{true} = \text{true}$$

$$(\textbf{handle}\ \text{toggle}\ \textbf{with}\ \text{logState})\ \text{true} = (\text{true}, [\text{true}, \text{false}])$$

Composing handlers

- *closed handlers* only handle the operations explicitly listed in the operation clauses
- *open handlers*, in addition to handling the explicitly specified operations, also *forward* all other operations
- open handlers support composition

Links + effect handlers (Hillerström, Lindley)

Links

- statically-typed functional programming language for the web [Cooper, Lindley, Wadler, Yallop, 2006]
- relevant features: call-by-value, type inference, first-class continuations, row types (for records, variants, session types, and **effects**)

Effect handler implementation

- adapts existing infrastructure for row-based effects
- adapts existing infrastructure for first-class continuations

(Links demo)

Kinding rules

TyVar
$$\frac{}{\Delta, \alpha : K \vdash \alpha : K}$$

Comp
$$\frac{\Delta \vdash A : \mathsf{Type} \qquad \Delta \vdash E : \mathsf{Effect}}{\Delta \vdash A!E : \mathsf{Comp}}$$

Fun
$$\frac{\Delta \vdash A : \mathsf{Type} \qquad \Delta \vdash C : \mathsf{Comp}}{\Delta \vdash A \to C : \mathsf{Type}}$$

Forall
$$\frac{\Delta, \alpha : K \vdash C : \mathsf{Comp}}{\Delta \vdash \forall \alpha^K . C : \mathsf{Type}}$$

Record
$$\frac{\Delta \vdash R : \mathsf{Row}_\emptyset}{\Delta \vdash \langle R \rangle : \mathsf{Type}}$$

Variant
$$\frac{\Delta \vdash R : \mathsf{Row}_\emptyset}{\Delta \vdash [R] : \mathsf{Type}}$$

Effect
$$\frac{\Delta \vdash R : \mathsf{Row}_\emptyset}{\Delta \vdash \{R\} : \mathsf{Effect}}$$

EmptyRow
$$\frac{}{\Delta \vdash \cdot : \mathsf{Row}_\mathcal{L}}$$

ExtendRow
$$\frac{\Delta, P : \mathsf{Presence} \qquad \Delta, R : \mathsf{Row}_{\mathcal{L} \uplus \{\ell\}}}{\Delta \vdash \ell : P; R : \mathsf{Row}_\mathcal{L}}$$

Present
$$\frac{\Delta \vdash A : \mathsf{Type}}{\Delta \vdash \mathsf{Pre}(A) : \mathsf{Presence}}$$

Absent
$$\frac{}{\Delta \vdash \mathsf{Abs} : \mathsf{Presence}}$$

Value typing rules

$$\textsc{T-Var}\quad \frac{x : A \in \Gamma}{\Delta; \Gamma \vdash x : A}$$

$$\textsc{T-Lam}\quad \frac{\Delta; \Gamma, x : A \vdash M : C}{\Delta; \Gamma \vdash \lambda x^A . M : A \to C}$$

$$\textsc{T-PolyLam}\quad \frac{\Delta, \alpha : K; \Gamma \vdash M : C \qquad \alpha \notin FTV(\Gamma)}{\Delta; \Gamma \vdash \Lambda \alpha^K . M : \forall \alpha^K . C}$$

$$\textsc{T-Unit}\quad \frac{}{\Delta; \Gamma \vdash \langle\rangle : \langle\rangle}$$

$$\textsc{T-Extend}\quad \frac{\Delta; \Gamma \vdash V : A \qquad \Delta; \Gamma \vdash W : \langle \ell : \mathsf{Abs}; R \rangle}{\Delta; \Gamma \vdash \langle \ell = V; W \rangle : \langle \ell : \mathsf{Pre}(A); R \rangle}$$

$$\textsc{T-Inject}\quad \frac{\Delta; \Gamma \vdash V : A}{\Delta; \Gamma \vdash (\ell\, V)^R : [\ell : \mathsf{Pre}(A); R]}$$

Computation typing rules

T-App
$$\frac{\Delta;\Gamma \vdash V : A \to C \qquad \Delta;\Gamma \vdash W : B}{\Delta;\Gamma \vdash V\,W : C}$$

T-PolyApp
$$\frac{\Delta;\Gamma \vdash V : \forall \alpha^K.C \qquad \Delta \vdash A : K}{\Delta;\Gamma \vdash V\,A : C[A/\alpha]}$$

T-Split
$$\frac{\Delta;\Gamma \vdash V : \langle \ell : \mathsf{Pre}(A); R\rangle \qquad \Delta;\Gamma, x : A, y : \langle \ell : \mathsf{Abs}; R\rangle \vdash N : C}{\Delta;\Gamma \vdash \mathbf{let}\ \langle \ell = x; y\rangle \leftarrow V\ \mathbf{in}\ N : C}$$

T-Case
$$\frac{\Delta;\Gamma \vdash V : [\ell : \mathsf{Pre}(A); R] \qquad \Delta;\Gamma, x : A \vdash M : C \qquad \Delta;\Gamma, y : [\ell : \mathsf{Abs}; R] \vdash N : C}{\Delta;\Gamma \vdash \mathbf{case}\ V\{\ell\ x \mapsto M; y \mapsto N\} : C}$$

T-Absurd
$$\frac{\Delta;\Gamma \vdash V : []}{\Delta;\Gamma \vdash \mathbf{absurd}^A\ V : C}$$

T-Return
$$\frac{\Delta;\Gamma \vdash V : A}{\Delta;\Gamma \vdash \mathbf{return}\ V : A!E}$$

T-Let
$$\frac{\Delta;\Gamma \vdash M : A!E \qquad \Delta;\Gamma, x : A \vdash N : B!E}{\Delta;\Gamma \vdash \mathbf{let}\ x \leftarrow M\ \mathbf{in}\ N : B!E}$$

Effect and handler typing rules

$$\frac{\Delta;\Gamma \vdash V : A \qquad E = \{\ell : A \to B; R\}}{\Delta;\Gamma \vdash (\mathbf{do}\ \ell\ V)^E : B!E}\ \textsc{T-Do}$$

$$\frac{\Delta;\Gamma \vdash M : C \qquad \Delta;\Gamma \vdash H : C \Rightarrow D}{\Delta;\Gamma \vdash \mathbf{handle}\ M\ \mathbf{with}\ H : D}\ \textsc{T-Handle}$$

$$\frac{\begin{array}{c} C = A!\{(\ell_i : A_i \to B_i)_i; R\} \qquad D = B!\{(\ell_i : P_i)_i; R\} \\ H = \{\mathbf{return}\ x \mapsto M\} \uplus \{\ell_i\ y\ k \mapsto N_i\}_i \\ [\Delta;\Gamma, y : A_i, k : B_i \to D \vdash N_i : D]_i \qquad \Delta;\Gamma, x : A \vdash M : D \end{array}}{\Delta;\Gamma \vdash H : C \Rightarrow D}\ \textsc{T-Handler}$$

Operational semantics for effect handlers

S-Handle-Ret

$$\textbf{handle}\ (\textbf{return}\ V)\ \textbf{with}\ H \leadsto M[V/x],$$
$$\text{where } \{\textbf{return}\ x \mapsto M\} \in H$$

S-Handle-Op

$$\textbf{handle}\ \mathcal{E}[\textbf{do}\ \ell\ V]\ \textbf{with}\ H \leadsto$$
$$M[V/x, \lambda y.\,\textbf{handle}\ \mathcal{E}[\textbf{return}\ y]\ \textbf{with}\ H/k],$$
$$\text{where } \ell \notin BL(\mathcal{E}) \text{ and } \{\ell\ x\ k \mapsto M\} \in H$$

Evaluation contexts

$$\mathcal{E} ::= [\,] \mid \textbf{let}\ x \leftarrow \mathcal{E}\ \textbf{in}\ N \mid \textbf{handle}\ \mathcal{E}\ \textbf{with}\ H$$

Abstract machine syntax

Configurations	$\mathcal{C} ::= \langle M \mid \gamma \mid \kappa \rangle$
	$\mid \langle M \mid \gamma \mid \kappa \mid \kappa' \rangle_{\mathsf{op}}$
Value environments	$\gamma ::= \emptyset \mid \gamma[x \mapsto v]$
Values	$v, w ::= \lambda^{\gamma} x^{A}.M \mid \Lambda^{\gamma} \alpha^{K}.M$
	$\mid \langle\rangle \mid \langle \ell = v; w \rangle \mid (\ell\, v)^{R} \mid \kappa^{A}$
Continuations	$\kappa ::= [\,] \mid \delta :: \kappa$
Continuation frames	$\delta ::= (\sigma, \chi)$
Pure continuations	$\sigma ::= [\,] \mid \phi :: \sigma$
Pure continuation frames	$\phi ::= (\gamma, x, N)$
Handlers	$\chi ::= (\gamma, H)$

Abstract machine semantics

$$
\begin{aligned}
\langle V\ W \mid \gamma \mid \kappa\rangle &\longrightarrow \langle M \mid \gamma'[x \mapsto [\![W]\!]\gamma] \mid \kappa\rangle, && \text{if } [\![V]\!]\gamma = \lambda^{\gamma'} x.\, M \\
\langle V\ W \mid \gamma \mid \kappa\rangle &\longrightarrow \langle \mathbf{return}\ W \mid \gamma \mid \kappa' \mathbin{+\!\!+} \kappa\rangle, && \text{if } [\![V]\!]\gamma = \kappa' \\
\langle V\ A \mid \gamma \mid \kappa\rangle &\longrightarrow \langle M[A/\alpha] \mid \gamma' \mid \kappa\rangle, && \text{if } [\![V]\!]\gamma = \Lambda^{\gamma'} \alpha.\, M \\
\langle \mathbf{let}\ \langle \ell = x; y\rangle \leftarrow V\ \mathbf{in}\ N \mid \gamma \mid \kappa\rangle &\longrightarrow \langle N \mid \gamma[x \mapsto v, y \mapsto w] \mid \kappa\rangle, && \text{if } [\![V]\!]\gamma = \langle \ell = v; w\rangle \\
\langle \mathbf{case}\ V\ \{\ell\, x \mapsto M; y \mapsto N\} \mid \gamma \mid \kappa\rangle &\longrightarrow \begin{cases}\langle M \mid \gamma[x \mapsto v] \mid \kappa\rangle, \\ \langle N \mid \gamma[y \mapsto \ell'\ v] \mid \kappa\rangle,\end{cases} && \begin{matrix}\text{if } [\![V]\!]\gamma = \ell\ v \\ \text{if } [\![V]\!]\gamma = \ell'\ v \text{ and } \ell \neq \ell'\end{matrix}
\end{aligned}
$$

$$
\begin{aligned}
\langle \mathbf{let}\ x \leftarrow M\ \mathbf{in}\ N \mid \gamma \mid (\sigma, \chi) :: \kappa\rangle &\longrightarrow \langle M \mid \gamma \mid ((\gamma, x, N) :: \sigma, \chi) :: \kappa\rangle \\
\langle \mathbf{handle}\ M\ \mathbf{with}\ H \mid \gamma \mid \kappa\rangle &\longrightarrow \langle M \mid \gamma \mid ([\,], (\gamma, H)) :: \kappa\rangle
\end{aligned}
$$

$$
\begin{aligned}
\langle \mathbf{return}\ V \mid \gamma \mid ((\gamma', x, N) :: \sigma, \chi) :: \kappa\rangle &\longrightarrow \langle N \mid \gamma'[x \mapsto [\![V]\!]\gamma] \mid (\sigma, \chi) :: \kappa\rangle \\
\langle \mathbf{return}\ V \mid \gamma \mid ([\,], (\gamma', H)) :: \kappa\rangle &\longrightarrow \langle M \mid \gamma'[x \mapsto [\![V]\!]\gamma] \mid \kappa\rangle, \\
&\qquad \text{if } H(\mathbf{return}) = \{\mathbf{return}\ x \mapsto M\} \\
\langle \mathbf{return}\ V \mid \gamma \mid [\,]\rangle &\longrightarrow [\![V]\!]\gamma
\end{aligned}
$$

$$
\begin{aligned}
\langle (\mathbf{do}\ \ell\ V)^E \mid \gamma \mid \kappa\rangle &\longrightarrow \langle (\mathbf{do}\ \ell\ V)^E \mid \gamma \mid \kappa \mid [\,]\rangle_{\mathrm{op}} \\
\langle (\mathbf{do}\ \ell\ V)^E \mid \gamma \mid \delta :: \kappa \mid \kappa'\rangle_{\mathrm{op}} &\longrightarrow \langle M \mid \gamma'[x \mapsto [\![V]\!]\gamma, k \mapsto (\kappa' \mathbin{+\!\!+} [\delta])^B] \mid \kappa\rangle, \\
&\qquad \text{if } \ell : A \to B \in E \text{ and } \delta(\ell) = \{\ell\ x\ k \mapsto M\} \\
\langle (\mathbf{do}\ \ell\ V)^E \mid \gamma \mid \delta :: \kappa \mid \kappa'\rangle_{\mathrm{op}} &\longrightarrow \langle (\mathbf{do}\ \ell\ V)^E \mid \gamma \mid \kappa \mid \kappa' \mathbin{+\!\!+} [\delta]\rangle_{\mathrm{op}}, \quad \text{if } \delta(\ell) = \emptyset
\end{aligned}
$$

Value interpretation

$$
\begin{aligned}
[\![x]\!]\gamma &= \gamma(x) \\
[\![\lambda x^A.M]\!]\gamma &= \lambda^\gamma x^A.M \\
[\![\Lambda\alpha^K.M]\!]\gamma &= \Lambda^\gamma\alpha^K.M \\
[\![\langle\rangle]\!]\gamma &= \langle\rangle \\
[\![\langle\ell = V; W\rangle]\!]\gamma &= \langle\ell = [\![V]\!]\gamma; [\![W]\!]\gamma\rangle \\
[\![(\ell\, V)^R]\!]\gamma &= (\ell\, [\![V]\!]\gamma)^R
\end{aligned}
$$

Mapping configurations to terms

Configurations

$$\begin{aligned} (\!|\langle M \mid \gamma \mid \kappa\rangle|\!) &= (\!|\kappa|\!)(M, \gamma) \\ (\!|\langle M \mid \gamma \mid \kappa \mid \kappa'\rangle_{\mathsf{op}}|\!) &= (\!|\kappa' +\!\!+ \kappa|\!)(M, \gamma) = (\!|\kappa'|\!)((\!|\kappa|\!)(M, \gamma), \emptyset) \end{aligned}$$

Continuations

$$\begin{aligned} (\!|[]|\!)(M, \gamma) &= (\!|M|\!)\gamma \\ (\!|((\gamma', x, N) :: \sigma, \chi) :: \kappa|\!)(M, \gamma) &= (\!|(\sigma, \chi) :: \kappa|\!)(\mathbf{let}\ x \leftarrow M\ \mathbf{in}\ (\!|N|\!)(\gamma' \backslash \{x\}), \gamma) \\ (\!|([], (\gamma', H)) :: \kappa|\!)(M, \gamma) &= (\!|\kappa|\!)(\mathbf{handle}\ M\ \mathbf{with}\ (\!|H|\!)\gamma', \gamma) \end{aligned}$$

Computation terms

$$\begin{aligned} (\!|V\ W|\!)\gamma &= (\!|V|\!)\gamma\ (\!|W|\!)\gamma \\ (\!|V\ A|\!)\gamma &= (\!|V|\!)\gamma\ A \\ (\!|\mathbf{let}\ \langle \ell = x; y\rangle \leftarrow V\ \mathbf{in}\ N|\!)\gamma &= \mathbf{let}\ \langle \ell = x; y\rangle \leftarrow (\!|V|\!)\gamma\ \mathbf{in}\ (\!|N|\!)(\gamma \backslash \{x, y\}) \\ (\!|\mathbf{case}\ V\ \{\ell\ x \mapsto M; y \mapsto N\}|\!)\gamma &= \mathbf{case}\ (\!|V|\!)\gamma\ \{\ell\ x \mapsto (\!|M|\!)(\gamma \backslash \{x\}); y \mapsto (\!|N|\!)(\gamma \backslash \{y\})\} \\ (\!|\mathbf{return}\ V|\!)\gamma &= \mathbf{return}\ (\!|V|\!)\gamma \\ (\!|\mathbf{let}\ x \leftarrow M\ \mathbf{in}\ N|\!)\gamma &= \mathbf{let}\ x \leftarrow (\!|M|\!)\gamma\ \mathbf{in}\ (\!|N|\!)(\gamma \backslash \{x\}) \\ (\!|\mathbf{do}\ \ell\ V|\!)\gamma &= \mathbf{do}\ \ell\ (\!|V|\!)\gamma \\ (\!|\mathbf{handle}\ M\ \mathbf{with}\ H|\!)\gamma &= \mathbf{handle}\ (\!|M|\!)\gamma\ \mathbf{with}\ (\!|H|\!)\gamma \end{aligned}$$

Handler definitions

$$\begin{aligned} (\!|\{\mathbf{return}\ x \mapsto M\}|\!)\gamma &= \{\mathbf{return}\ x \mapsto (\!|M|\!)(\gamma \backslash \{x\})\} \\ (\!|\{\ell\ x\ k \mapsto M\} \uplus H|\!)\gamma &= \{\ell\ x\ k \mapsto (\!|M|\!)(\gamma \backslash \{x, k\}\} \uplus (\!|H|\!)\gamma \end{aligned}$$

Mapping configurations to terms (continued)

Value terms and values

$$
\begin{aligned}
(\!|x|\!)\gamma &= (\!|v|\!), \quad \text{if } \gamma(x) = v \\
(\!|x|\!)\gamma &= x, \quad \text{if } x \notin dom(\gamma) \\
(\!|\lambda x^A.M|\!)\gamma &= \lambda x^A.(\!|M|\!)(\gamma \backslash \{x\}) \\
(\!|\Lambda \alpha^K.M|\!)\gamma &= \Lambda \alpha^K.(\!|M|\!)\gamma \\
(\!|\langle\rangle|\!)\gamma &= \langle\rangle \\
(\!|\langle \ell = V; W\rangle|\!)\gamma &= \langle \ell = (\!|V|\!)\gamma; (\!|W|\!)\gamma\rangle \\
(\!|(\ell\ V)^R|\!)\gamma &= (\ell\ (\!|V|\!)\gamma)^R
\end{aligned}
$$

$$
\begin{aligned}
(\!|\lambda^\gamma x^A.M|\!) &= \lambda x^A.(\!|M|\!)(\gamma \backslash \{x\}) \\
(\!|\Lambda^\gamma \alpha^K.M|\!) &= \Lambda \alpha^K.(\!|M|\!)\gamma \\
(\!|\langle\rangle|\!) &= \langle\rangle \\
(\!|\langle \ell = v; w\rangle|\!) &= \langle \ell = (\!|v|\!); (\!|w|\!)\rangle \\
(\!|(\ell\ v)^R|\!) &= (\ell\ (\!|v|\!))^R \\
(\!|\kappa^A|\!) &= \lambda x^A.(\!|\kappa|\!)(\textbf{return}\ x, \emptyset)
\end{aligned}
$$

Theorems

Definition

Computation term N is normal with respect to effect E, if N is either of the form **return** V, or $\mathcal{E}[\mathbf{do}\ \ell\ W]$, where $\ell \in E$ and $\ell \notin BL(\mathcal{E})$.

Theorem (Type Soundness)

If $\vdash M : A!E$, *then there exists* $\vdash N : A!E$, *such that* $M \leadsto^{+} N \not\leadsto$, *and* N *is normal with respect to effect* E.

Definition

$$\Longrightarrow = \longrightarrow_a^{*} \longrightarrow_{\beta}$$

Theorem (Simulation)

If $M \leadsto N$, *then for any* $\mathcal{C}$, *such that* $(\!|\mathcal{C}|\!) = M$, *there exists* $\mathcal{C}'$, *such that* $\mathcal{C} \Longrightarrow \mathcal{C}'$ *and* $(\!|\mathcal{C}'|\!) = N$.

Frank (McBride, McLaughlin, Lindley)

- bidirectional type system
- shadowing instead of presence information
- shallow handlers
- everything is a handler (or rather *multihandler*)
  - handlers (multihandlers) generalise functions
  - "call-by-handling" generalises call-by-value
- effect polymorphism with a single invisible effect variable

Related abstractions

- monad transformers
- free monads
  - deep handler: fold over free monad
  - shallow handler: case-split over free monad
- containers
- monadic reflection
- exceptional syntax
- delimited continuations
  - Bauer:

    $$\text{“effects + handlers”} : \text{“delimited continuations”} = \text{“while”} : \text{“goto”}$$

  - deep handlers: shift0/reset0
  - shallow handlers: control0/prompt0
- modules, type classes, dynamic binding

Some related work

Algebraic effects [Plotkin and Power, 2001]
Effect handlers [Pretnar and Plotkin, 2009]
Other effect handler languages and libraries:

- Eff [Bauer and Pretnar, 2012]
- Haskell, SML, OCaml, Racket, Scheme effect libraries [Kammar, Lindley, Oury, 2013]
- Haskell extensible effects [Kiselyov, Sabry, Swords, 2013]
- Idris effects library [Brady, 2013]
- Experimental OCaml development branch [Dolan, Silvaramakrishnan, White, Yallop, Madhavapeddy, 2015]
- Prolog library [Schrijvers, Wu, Desouter, Demoen, 2014]
- Shonky [McBride, 2016]

Other stuff:

- Extensible denotational language specifications [Cartwright and Felleisen, 1994]
- Data types a la carte [Swierstra, 2008]
- Kleisli arrows of outrageous fortune [McBride, 2011]
- Scoped effect handlers [Wu, Schrijvers, Hinze, 2014]
- Fusion for free [Wu and Schrijvers, 2015]