Expressiveness of effect handlers

Sam Lindley

The University of Edinburgh

OPLSS 2022

On the expressive power of programming languages

Many different notions of expressive power, for instance:

- Computability
- Algorithmic complexity
- **Macro expressiveness**

Matthias Felleisen

[Felleisen, 1990]

Example: nondeterminism

choose	binary nondeterministic choice (tt / ff)
fail	nullary nondeterministic choice
run	run a nondeterministic computation

$$
\begin{aligned}
&\mathit{drunkToss}\ () = \textbf{if choose then}\\
&\qquad\qquad\qquad\qquad\ \textbf{if choose then}\ \mathrm{Heads}\ \textbf{else}\ \mathrm{Tails}\\
&\qquad\qquad\qquad\quad\ \textbf{else}\\
&\qquad\qquad\qquad\qquad\ \textbf{fail}
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{drunkTosses}\ n = \textbf{if}\ n = 0\ \textbf{then}\ []\\
&\qquad\qquad\qquad\qquad\ \textbf{else}\ \mathit{drunkToss}\ () :: \mathit{drunkTosses}\ (n-1)
\end{aligned}
$$

$$
\begin{aligned}
&\textbf{run}\ (\mathit{drunkTosses}\ 2) =\\
&\quad [[\mathrm{Heads}, \mathrm{Heads}], [\mathrm{Heads}, \mathrm{Tails}], [\mathrm{Tails}, \mathrm{Heads}], [\mathrm{Tails}, \mathrm{Tails}]]
\end{aligned}
$$

Example: nondeterminism (plain λ-calculus)

We can implement nondeterminism with plain λ-calculus using a **global** transformation.

$$
\begin{aligned}
[\![x]\!] &= [x] \\
[\![\lambda x.M]\!] &= [\lambda x.[\![M]\!]] \\
[\![M\ N]\!] &= \textbf{concat}\ [f\ x \mid f \leftarrow [\![M]\!], x \leftarrow [\![N]\!]] \\
[\![\textbf{choose}]\!] &= [\mathrm{tt}, \mathrm{ff}] \\
[\![\textbf{fail}]\!] &= [] \\
[\![\textbf{run}\ M]\!] &= [\![M]\!]
\end{aligned}
$$

(assuming standard encodings of booleans and lists)

Example: nondeterminism (plain λ-calculus)

We can implement nondeterminism with plain λ-calculus using a **global** transformation.

$$
\begin{aligned}
[\![x]\!] &= [x] \\
[\![\lambda x.M]\!] &= [\lambda x.[\![M]\!]] \\
[\![M\ N]\!] &= \textbf{concat}\ [f\ x \mid f \leftarrow [\![M]\!], x \leftarrow [\![N]\!]] \\
[\![\textbf{choose}]\!] &= [\mathsf{tt}, \mathsf{ff}] \\
[\![\textbf{fail}]\!] &= [] \\
[\![\textbf{run}\ M]\!] &= [\![M]\!]
\end{aligned}
$$

(assuming standard encodings of booleans and lists)

What about a **local** transformation?

Example: nondeterminism (structure of a local transformation)

$$
\begin{aligned}
[\![x]\!] &= x \\
[\![\lambda x.M]\!] &= \lambda x.[\![M]\!] \\
[\![M\ N]\!] &= [\![M]\!]\ [\![N]\!] \\
[\![\textbf{choose}]\!] &= \mathit{Choose} \\
[\![\textbf{fail}]\!] &= \mathit{Fail} \\
[\![\textbf{run}\ M]\!] &= \mathit{Run}([\![M]\!])
\end{aligned}
$$

A local transformation of nondeterminism is determined solely by the parameters *Choose*, *Fail*, and *Run*.

Example: nondeterminism (effect handlers)

We can implement nondeterminsm with **effect handlers** using a **local** transformation.

$$
\begin{aligned}
[\![\textbf{choose}]\!] &= \mathsf{choose}\ () \\
[\![\textbf{fail}]\!] &= \mathsf{fail}\ () \\
[\![\textbf{run}\ M]\!] &= \textbf{handle}\ [\![M]\!]\ \textbf{with} \\
&\qquad \textbf{return}\ x \quad \mapsto [x] \\
&\qquad \mathsf{choose}\ ()\ r \mapsto r\ \mathsf{tt} +\!\!+ r\ \mathsf{ff} \\
&\qquad \mathsf{fail}\ ()\ r \quad \mapsto []
\end{aligned}
$$

Operational semantics for effect handlers

$$\textbf{handle}\ V\ \textbf{with}\ H \rightsquigarrow N[V/x]$$
$$\textbf{handle}\ \mathcal{E}[\mathsf{op}_i\ V]\ \textbf{with}\ H \rightsquigarrow N_i[V/p, \lambda x.\textbf{handle}\ \mathcal{E}[x]\ \textbf{with}\ H/r]$$

where

$$\begin{aligned} H = \textbf{return}\ x &\mapsto N \\ \mathsf{op}_1\ p\ r &\mapsto N_1 \\ &\cdots \\ \mathsf{op}_n\ p\ r &\mapsto N_n \end{aligned}$$

Effect handler typing rules

Effects

$$E ::= \emptyset \mid E \uplus \{\mathsf{op} : A \to B\}$$

Computations

$$C, D ::= A!E$$

Operations

$$\frac{\Gamma \vdash V : A}{\Gamma \vdash \mathsf{op}\, V : B!(E \uplus \{\mathsf{op} : A \to B\})}$$

Handlers

$$\frac{\Gamma \vdash M : C \qquad \Gamma \vdash H : C \Rightarrow D}{\Gamma \vdash \textbf{handle}\ M\ \textbf{with}\ H : D}$$

$$\frac{\Gamma, x : A \vdash M : C \qquad [\Gamma, p : A_i, r : B_i \to C \vdash N_i : C]_i}{\Gamma \vdash \begin{array}{l}\textbf{return}\ x \mapsto M \\ (\mathsf{op}_i\ p\ r \mapsto N_i)_i\end{array} : A!\{\mathsf{op}_i : A_i \to B_i\}_i \Rightarrow C}$$

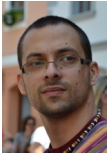
Algebraic effects

Gordon Plotkin

John Power

Effect handlers

Gordon Plotkin



Matija Pretnar

[Plotkin and Power, 2001–2003; Plotkin and Pretnar, 2009]

Example: nondeterminism (monadic reflection)

We can implement nondeterminsm with **monadic reflection** using a **local** transformation.

$$
\begin{aligned}
[\![\textbf{choose}]\!] &= \mu(\lambda().[\mathrm{tt}, \mathrm{ff}]) \\
[\![\textbf{fail}]\!] &= \mu(\lambda().[]) \\
[\![\textbf{run}\ M]\!] &= \llbracket [\![M]\!] \rrbracket^{\mathit{List}}
\end{aligned}
$$

$$
\mathit{List} = \left\{ \begin{array}{l} \textbf{return}\ x \mapsto [x] \\ m \gg\!= k \mapsto \textbf{concat}\ [k\ x \mid x \leftarrow m\ ()] \end{array} \right\}
$$

Operational semantics for monadic reflection

$$[V]^L \rightsquigarrow N_{\text{return}}[V/x]$$
$$[\mathcal{E}[\mu(V)]]^L \rightsquigarrow N_{\text{bind}}[V/m, (\lambda x.\,[\mathcal{E}[x]]^L)/k]$$

where

$$L = \left\{ \begin{array}{l} \textbf{return}\ x \mapsto N_{\text{return}} \\ m \gg\!= k \mapsto N_{\text{bind}} \end{array} \right\}$$

Monadic reflection typing rules

Effects

$$E ::= \emptyset \mid E, (T, L)$$

Computations

$$C, D ::= A!E$$

Reflect

$$\frac{\Gamma \vdash V : 1 \to T\ A!E}{\Gamma \vdash \mu(V) : A!E, (T, L)}$$

Reify

$$\frac{\vdash E, (T, L) \qquad \Gamma \vdash M : A!E, (T, L)}{\Gamma \vdash [M]^L : T\ A!E}$$

Monads

$$\frac{\begin{array}{l} x : \alpha \vdash M : T\ \alpha!E \\ m : 1 \to T\ \alpha!E, k : \alpha \to T\ \beta!E \vdash N : T\ \beta!E \end{array} \qquad L = \left\{ \begin{array}{l} \textbf{return}\ x \mapsto M \\ m \gg\!= k \mapsto N \end{array} \right\}}{\vdash E, (T, L)}$$

Monadic reflection

Andrzej Filinski

[Filinski 1994; Filinski 2010]

Example: nondeterminism (delimited continuations)

We can implement nondeterminsm with **delimited continuations** using a **local** transformation.

$$\begin{aligned} [\![\textbf{choose}]\!] &= \mathcal{S}k.k\ \mathsf{tt} +\!\!+ k\ \mathsf{ff} \\ [\![\textbf{fail}]\!] &= \mathcal{S}k.[] \\ [\![\textbf{run}\ M]\!] &= \langle [\![M]\!] \mid \lambda x.[x] \rangle \end{aligned}$$

Operational semantics for delimited continuations

Static delimited continuations (shift and reset)

$$\begin{aligned} \langle \mathcal{E}[\mathcal{S}k.M] \rangle &\rightsquigarrow \langle M[(\lambda x.\langle \mathcal{E}[x] \rangle)/k] \rangle \\ \langle V \rangle &\rightsquigarrow V \end{aligned}$$

Dynamic delimited continuations (shift0 and reset0)

$$\begin{aligned} \langle \mathcal{E}[\mathcal{S}k.M] \rangle &\rightsquigarrow M[(\lambda x.\langle \mathcal{E}[x] \rangle)/k] \\ \langle V \rangle &\rightsquigarrow V \end{aligned}$$

Slight variation (shift0 and reset0 with a success continuation)

$$\begin{aligned} \langle \mathcal{E}[\mathcal{S}k.M] \mid W \rangle &\rightsquigarrow M[(\lambda x.\langle \mathcal{E}[x] \mid W \rangle)/k] \\ \langle V \mid \lambda x.N \rangle &\rightsquigarrow N[V/x] \end{aligned}$$

CPS translations for delimited continuations

Static delimited continuations (shift and reset)

$$\begin{aligned}[\![\mathcal{S}k.M]\!] &= \lambda k'.(\lambda k.[\![M]\!])\,(\lambda x\,k''.k''\,(k'\,x))\,(\lambda x.x)\\ [\![\langle M\rangle]\!] &= \lambda k.k\,([\![M]\!]\,(\lambda x.x))\end{aligned}$$

Dynamic delimited continuations (shift0 and reset0)

$$\begin{aligned}[\![\mathcal{S}k.M]\!] &= \lambda k.[\![M]\!]\\ [\![\langle M\rangle]\!] &= [\![M]\!]\,(\lambda k.k)\end{aligned}$$

Slight variation (shift0 and reset0 with a success continuation)

$$\begin{aligned}[\![\mathcal{S}k.M]\!] &= \lambda k.[\![M]\!]\\ [\![\langle M \mid W\rangle]\!] &= [\![M]\!]\,[\![W]\!]\end{aligned}$$

Delimited continuation typing rules

Effects

$$E ::= \emptyset \mid E, A$$

Computations

$$C, D ::= A!E$$

Shift

$$\frac{\Gamma, k : A \to B!E \vdash M : B!E}{\Gamma \vdash \mathcal{S}k.M : A!E, B}$$

Reset

$$\frac{\Gamma \vdash M : A!E, B \qquad \Gamma \vdash W : A \to B!E}{\Gamma \vdash \langle M \mid W \rangle : B!E}$$

Delimited continuations

Control and prompt

Matthias Felleisen

Shift and reset

Olivier Danvy

Andrzej Filinski

[Felleisen, 1988; Danvy and Filinski, 1990]

Delimited continuations versus monads

Delimited continuations can “express” any “definable” monad

Andrzej Filinski

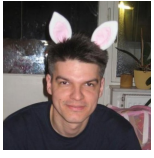
[Filinski, 1994]

Effect handlers versus delimited continuations

"effects + handlers" : "delimited continuations"

=

"while" : "goto"

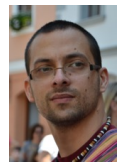


Andrej Bauer

On the expressive power of user-defined effects

effect handlers, monadic reflection, delimited continuations



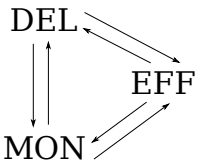
Yannick Forster Ohad Kammar Sam Lindley Matija Pretnar

[Forster, Kammar, Lindley, and Pretnar, 2017]

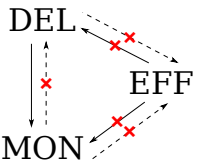
In the paper

- Effect handlers, monadic reflection, delimited continuations each formulated as extensions of CBPV
- Operational and denotational semantics
- Untyped macro translations (formalised in Abella)
- Nonexistence result for simply-typed macro translations (using denotational semantics)

No types



Simple types



Delimited continuations as monadic reflection

$$[\![\mathcal{S}k.M]\!] = \mu(\lambda()\ k.[\![M]\!])$$
$$[\![\langle M \mid W \rangle]\!] = \llbracket [\![M]\!] \rrbracket^{Cont}\ [\![W]\!]$$

where

$$Cont = \left\{ \begin{array}{l} \textbf{return}\ x = \lambda k.k\,x \\ m \gg\!= f = \lambda k.m\,(\lambda x.f\ x\ k) \end{array} \right\}$$

Theorem

If $M \rightsquigarrow N$ *then* $[\![M]\!] \rightsquigarrow^{+}_{\mathrm{cong}} [\![N]\!]$.

Monadic reflection as delimited continuations

$$[\![\mu(V)]\!] = \mathcal{S}k.\mathcal{S}b.b\ ([\![V]\!], \lambda x.\langle k\ x \mid b\rangle)$$
$$[\![[M]^L]\!] = \langle\langle [\![M]\!] \mid \lambda x.\mathcal{S}b.[\![N_{\mathrm{return}}]\!]\rangle \mid \lambda(m,k).[\![N_{\mathrm{bind}}]\!]\rangle$$

where

$$L = \begin{Bmatrix} \textbf{return}\ x \mapsto N_{\mathrm{return}} \\ m \gg\!= k \mapsto N_{\mathrm{bind}} \end{Bmatrix}$$

Theorem

If $M \rightsquigarrow N$ *then* $[\![M]\!] \rightsquigarrow^{+}_{\mathrm{cong}} [\![N]\!]$.

Monadic reflection as effect handlers

$$
\begin{aligned}
[\![\mu(V)]\!] &= \text{reflect } [\![V]\!] \\
[\![[M]^L]\!] &= \textbf{handle } [\![M]\!] \textbf{ with} \\
&\qquad \textbf{return } x \quad \mapsto [\![N_{\text{return}}]\!] \\
&\qquad \text{reflect } m\ k \mapsto [\![N_{\text{bind}}]\!]
\end{aligned}
$$

where

$$
L = \begin{Bmatrix} \textbf{return } x \mapsto N_{\text{return}} \\ m \gg\!= k \mapsto N_{\text{bind}} \end{Bmatrix}
$$

Theorem

If $M \rightsquigarrow N$ *then* $[\![M]\!] \rightsquigarrow^{+} [\![N]\!]$.

Effect handlers as monadic reflection (cont)

$$\llbracket \mathsf{op}\ V \rrbracket = \mu(\lambda()\ k\ h.h\ (\mathbf{inj}\ \mathsf{op}\ (\llbracket V \rrbracket, \lambda x.k\ x\ h)))$$
$$\llbracket \mathbf{handle}\ M\ \mathbf{with}\ H \rrbracket = \lceil \llbracket M \rrbracket \rceil^{Cont}\ \llbracket H^{\mathrm{ret}} \rrbracket\ \llbracket H^{\mathrm{ops}} \rrbracket$$

where

$$\llbracket \mathbf{return}\ x \mapsto N \rrbracket = \lambda x\, h.\llbracket N \rrbracket$$
$$\llbracket (\mathsf{op}_i\ p\ r \mapsto N_i)_i \rrbracket = \lambda z.\mathbf{case}\ z\ \mathbf{of}\ \{(\mathbf{inj}\ \mathsf{op}_i\ (p, r) \mapsto \llbracket N_i \rrbracket)_i\}$$

$$Cont = \left\{ \begin{array}{l} \mathbf{return}\ x = \lambda k.k\, x \\ m \gg\!= f = \lambda k.m\ (\lambda x.f\ x\ k) \end{array} \right\}$$

Theorem

If $M \rightsquigarrow N$ *then* $\llbracket M \rrbracket \rightsquigarrow^{+}_{\mathrm{cong}} \llbracket N \rrbracket$.

Effect handlers as monadic reflection (free)

$$[\![\mathsf{op}\ V]\!] = \mu(\lambda().\mathbf{inj}\ \mathsf{op}\ ([\![V]\!], \lambda x.x))$$
$$[\![\mathbf{handle}\ M\ \mathbf{with}\ H]\!] = H^\star\ [[\![M]\!]]^{\mathit{Free}}$$

where

$$H = \begin{array}{ll} \mathbf{return}\ x & \mapsto [\![N]\!] \\ (\mathsf{op}_i\ p\ r & \mapsto [\![N_i]\!])_i \end{array}$$

$$\mathit{Free} = \left\{ \begin{array}{ll} \mathbf{return}\ x = \mathbf{inj}\ \mathsf{ret}\ x & \\ \mathbf{inj}\ z \gg\!= k = \mathbf{case}\ z\ \mathbf{of}\ \{\ \mathbf{inj}\ \mathsf{ret}\ x & \mapsto k\ x \\ \qquad\qquad (\mathbf{inj}\ \mathsf{op}_i\ (p, r') & \mapsto \mathbf{inj}\ \mathsf{op}_i\ (p, \lambda x.r'\ x \gg\!= k))_i\} \end{array} \right\}$$

$$H^\star\ M = \mathbf{letrec}\ h\ z = \mathbf{case}\ z\ \mathbf{of}\ \{\ \mathbf{inj}\ \mathsf{ret}\ x \mapsto [\![N]\!]$$
$$(\mathbf{inj}\ \mathsf{op}_i\ (p, r') \mapsto \mathbf{let}\ r = \lambda x.h\,(r'\ x)\ \mathbf{in}\ [\![N_i]\!])_i\}$$
$$\mathbf{in}\ h\ M$$

Theorem

If $M \rightsquigarrow N$ *then* $[\![M]\!] \rightsquigarrow^{+}_{\mathrm{cong}} [\![N]\!]$.

Delimited continuations as effect handlers

$$
\begin{aligned}
[\![\mathcal{S}k.M]\!] &= \text{shift } (\lambda k.[\![M]\!]) \\
[\![\langle M \mid W \rangle]\!] &= \textbf{handle } [\![M]\!] \textbf{ with} \\
&\qquad \textbf{return } x \mapsto [\![W]\!]\ x \\
&\qquad \text{shift } p\ k \mapsto p\ k
\end{aligned}
$$

Theorem

If $M \rightsquigarrow N$ *then* $[\![M]\!] \rightsquigarrow^{+} [\![N]\!]$.

Effect handlers as delimited continuations

$$\llbracket \mathsf{op}\ V \rrbracket = \mathcal{S}k.\mathcal{S}h.h\ (\mathbf{inj}\ \mathsf{op}\ (\llbracket V \rrbracket, \lambda x.\langle k\ x \mid h \rangle))$$
$$\llbracket \mathbf{handle}\ M\ \mathbf{with}\ H \rrbracket = \langle \langle \llbracket M \rrbracket \mid \llbracket H^{\mathrm{ret}} \rrbracket \rangle \mid \llbracket H^{\mathrm{ops}} \rrbracket \rangle$$

where

$$\llbracket \mathbf{return}\ x \mapsto N \rrbracket = \lambda x.\mathcal{S}h.\llbracket N \rrbracket$$
$$\llbracket (\mathsf{op}_i\ p\ r \mapsto N_i)_i \rrbracket = \lambda z.\mathbf{case}\ z\ \mathbf{of}\ \{(\mathbf{inj}\ \mathsf{op}_i\ (p, r) \mapsto \llbracket N_i \rrbracket)_i\}$$

Theorem

If $M \rightsquigarrow N$ *then* $\llbracket M \rrbracket \rightsquigarrow^{+}_{\mathrm{cong}} \llbracket N \rrbracket$.

CPS for effect handlers

$$[\![\text{op } V]\!] = \lambda k\, h.h\ (\textbf{inj}\ \text{op}\ ([\![V]\!], \lambda x.k\ x\ h))$$
$$[\![\textbf{handle } M \textbf{ with } H]\!] = [\![M]\!]\ [\![H^{\text{ret}}]\!]\ [\![H^{\text{ops}}]\!]$$

where

$$[\![\textbf{return } x \mapsto N]\!] = \lambda x\, h.[\![N]\!]$$
$$[\![(\text{op}_i\ p\ r \mapsto N_i)_i]\!] = \lambda z.\textbf{case } z \textbf{ of } \{(\textbf{inj}\ \text{op}_i\ (p, r) \mapsto [\![N_i]\!])_i\}$$

[Hillerström, Lindley, Atkey, and Sivaramakrishnan, 2017]

Daniel Hillerström

Sam Lindley

Robert Atkey

KC Sivaramakrishnan

Typing: delimited continuations as monadic reflection

$$[\![\mathcal{S}k.M]\!] = \mu(\lambda()\ k.[\![M]\!])$$
$$[\![\langle M \mid W \rangle]\!] = [[\![M]\!]]^{Cont}\ [\![W]\!]$$

This translation type checks!

Because the continuation monad is sufficiently parameteric to simulate shift + reset

$$Cont_C : \alpha.(\alpha \to C) \to C$$

Typing: monadic reflection as delimited continuations

$$[\![\mu(V)]\!] = \mathcal{S}k.\mathcal{S}b.b\ ([\![V]\!], \lambda x.\langle k\ x \mid b\rangle)$$
$$[\![[M]^L]\!] = \langle\langle [\![M]\!] \mid \lambda x.\mathcal{S}b.[\![N_{\mathrm{return}}]\!]\rangle \mid \lambda(m,k).[\![N_{\mathrm{bind}}]\!]\rangle$$

where

$$L = \left\{ \begin{array}{l} \textbf{return}\ x \mapsto N_{\mathrm{return}} \\ m \gg\!= k \mapsto N_{\mathrm{bind}} \end{array} \right\}$$

- Return types of k and b determined by the context
- Bind may be invoked at different input types

Typing: monadic reflection as delimited continuations

$$[\![\mu(V)]\!] = \mathcal{S}k.\mathcal{S}b.b\ ([\![V]\!], \lambda x.\langle k\ x \mid b\rangle)$$
$$[\![\,[M]^L]\!] = \langle\langle [\![M]\!] \mid \lambda x.\mathcal{S}b.[\![N_{\text{return}}]\!]\rangle \mid \lambda(m,k).[\![N_{\text{bind}}]\!]\rangle$$

where

$$L = \left\{ \begin{array}{l} \textbf{return}\ x \mapsto N_{\text{return}} \\ m \gg\!= k \mapsto N_{\text{bind}} \end{array} \right\}$$

- Return types of k and b determined by the context
- Bind may be invoked at different input types

Some kind of **polymorphic** delimited continuations?

Typing: monadic reflection as effect handlers

$$\frac{\Gamma \vdash V : 1 \to T\ A!E}{\Gamma \vdash \mu(V) : A!E, (T, L)}$$

$$\mathsf{reflect}_{A,T,E} : (1 \to T\ A!E) \to A$$

$$\begin{aligned}
[\![\mu(V)]\!] &= \mathsf{reflect}\ [\![V]\!] \\
[\![[M]^L]\!] &= \textbf{handle}\ [\![M]\!]\ \textbf{with} \\
&\qquad \textbf{return}\ x \quad \mapsto [\![N_{\mathrm{return}}]\!] \\
&\qquad \mathsf{reflect}\ m\ k \mapsto [\![N_{\mathrm{bind}}]\!]
\end{aligned}$$

where

$$L = \left\{ \begin{array}{l} \textbf{return}\ x \mapsto N_{\mathrm{return}} \\ m \gg\!= k \mapsto N_{\mathrm{bind}} \end{array} \right\}$$

Within a given reify

- T and E are fixed
- A can vary

Typing: monadic reflection as effect handlers

$$\frac{\Gamma \vdash V : 1 \to T\, A!E}{\Gamma \vdash \mu(V) : A!E, (T, L)}$$

$\mathrm{reflect}_{A,T,E} : (1 \to T\, A!E) \to A$

$$\begin{aligned}
[\![\mu(V)]\!] &= \mathrm{reflect}\ [\![V]\!] \\
[\![[M]^L]\!] &= \textbf{handle}\ [\![M]\!]\ \textbf{with} \\
&\qquad \textbf{return}\ x \quad \mapsto [\![N_{\mathrm{return}}]\!] \\
&\qquad \mathrm{reflect}\ m\ k \mapsto [\![N_{\mathrm{bind}}]\!]
\end{aligned}$$

where

$$L = \left\{ \begin{array}{l} \textbf{return}\ x \mapsto N_{\mathrm{return}} \\ m \gg\!= k \mapsto N_{\mathrm{bind}} \end{array} \right\}$$

Within a given reify

- T and E are fixed
- A can vary

We need **parametric** operations

$\mathrm{reflect}_{T,E} : \alpha.(1 \to T\, \alpha!E) \to \alpha$

Typing: effect handlers as monadic reflection

$$[\![\mathsf{op}\ V]\!] = \mu(\lambda()\ k\ h.h\ (\mathbf{inj}\ \mathsf{op}\ ([\![V]\!], \lambda x.k\ x\ h)))$$
$$[\![\mathbf{handle}\ M\ \mathbf{with}\ H]\!] = \lceil [\![M]\!] \rceil^{Cont}\ [\![H^{\mathrm{ret}}]\!]\ [\![H^{\mathrm{ops}}]\!]$$

where

$$[\![\mathbf{return}\ x \mapsto N]\!] = \lambda x\, h.[\![N]\!]$$
$$[\![(\mathsf{op}_i\ p\ r \mapsto N_i)_i]\!] = \lambda z.\mathbf{case}\ z\ \mathbf{of}\ \{(\mathbf{inj}\ \mathsf{op}_i\ (p, r) \mapsto [\![N_i]\!])_i\}$$

Return types of k and h determined by the context

Typing: effect handlers as monadic reflection

$$[\![\mathsf{op}\ V]\!] = \mu(\lambda()\ k\ h.h\ (\mathbf{inj}\ \mathsf{op}\ ([\![V]\!], \lambda x.k\ x\ h)))$$
$$[\![\mathbf{handle}\ M\ \mathbf{with}\ H]\!] = \lceil [\![M]\!] \rceil^{Cont}\ [\![H^{\mathrm{ret}}]\!]\ [\![H^{\mathrm{ops}}]\!]$$

where

$$[\![\mathbf{return}\ x \mapsto N]\!] = \lambda x\, h.[\![N]\!]$$
$$[\![(\mathsf{op}_i\ p\ r \mapsto N_i)_i]\!] = \lambda z.\mathbf{case}\ z\ \mathbf{of}\ \{(\mathbf{inj}\ \mathsf{op}_i\ (p, r) \mapsto [\![N_i]\!])_i\}$$

Return types of k and h determined by the context

Standard **polymorphism**?

Replace continuation monad by a codensity monad

$$Cont_C : \alpha.(\alpha \to C) \to C$$
$$Cod_{\beta.C} : \alpha.\forall\beta.(\alpha \to C) \to C$$

Typing: delimited continuations as effect handlers

$$\frac{\Gamma, k : A \to B!E \vdash M : B!E}{\Gamma \vdash \mathcal{S}k.M : A!(E, B)} \qquad \mathsf{shift}_{A,B!E} : ((A \to B!E) \to B!E) \to A$$

$$\begin{aligned} [\![\mathcal{S}k.M]\!] &= \mathsf{shift}\ (\lambda k.[\![M]\!]) \\ [\![\langle M \mid W \rangle]\!] &= \textbf{handle}\ [\![M]\!]\ \textbf{with} \\ &\qquad \textbf{return}\ x \mapsto [\![W]\!]\ x \\ &\qquad \mathsf{shift}\ p\ k \mapsto p\ k \end{aligned}$$

Within a given reset:

- B and E are fixed
- A can be instantiated at different types

Typing: delimited continuations as effect handlers

$$\frac{\Gamma, k : A \to B!E \vdash M : B!E}{\Gamma \vdash \mathcal{S}k.M : A!(E, B)} \qquad \mathsf{shift}_{A,B!E} : ((A \to B!E) \to B!E) \to A$$

$$\begin{aligned} [\![\mathcal{S}k.M]\!] &= \mathsf{shift}\ (\lambda k.[\![M]\!]) \\ [\![\langle M \mid W \rangle]\!] &= \textbf{handle}\ [\![M]\!]\ \textbf{with} \\ &\qquad \textbf{return}\ x \mapsto [\![W]\!]\ x \\ &\qquad \mathsf{shift}\ p\ k \mapsto p\ k \end{aligned}$$

Within a given reset:

- B and E are fixed
- A can be instantiated at different types

We need **parametric** operations

$$\mathsf{shift}_C : \alpha.((\alpha \to C) \to C) \to \alpha$$

Typing: effect handlers as delimited continuations

$$\llbracket \mathrm{op}\ V \rrbracket = \mathcal{S}k.\mathcal{S}h.h\ (\mathbf{inj}\ \mathrm{op}\ (\llbracket V \rrbracket, \lambda x.\langle k\ x \mid h \rangle))$$
$$\llbracket \mathbf{handle}\ M\ \mathbf{with}\ H \rrbracket = \langle \langle \llbracket M \rrbracket \mid \llbracket H^{\mathrm{ret}} \rrbracket \rangle \mid \llbracket H^{\mathrm{ops}} \rrbracket \rangle$$

where

$$\llbracket \mathbf{return}\ x \mapsto N \rrbracket = \lambda x.\mathcal{S}h.\llbracket N \rrbracket$$
$$\llbracket (\mathrm{op}_i\ p\ r \mapsto N_i)_i \rrbracket = \lambda z.\mathbf{case}\ z\ \mathbf{of}\ \{(\mathbf{inj}\ \mathrm{op}_i\ (p, r) \mapsto \llbracket N_i \rrbracket)_i\}$$

Return types of k and h determined by the context

Typing: effect handlers as delimited continuations

$$[\![\mathsf{op}\ V]\!] = \mathcal{S}k.\mathcal{S}h.h\ (\mathbf{inj}\ \mathsf{op}\ ([\![V]\!], \lambda x.\langle k\ x \mid h\rangle))$$
$$[\![\mathbf{handle}\ M\ \mathbf{with}\ H]\!] = \langle\langle [\![M]\!] \mid [\![H^{\mathrm{ret}}]\!]\rangle \mid [\![H^{\mathrm{ops}}]\!]\rangle$$

where

$$[\![\mathbf{return}\ x \mapsto N]\!] = \lambda x.\mathcal{S}h.[\![N]\!]$$
$$[\![(\mathsf{op}_i\ p\ r \mapsto N_i)_i]\!] = \lambda z.\mathbf{case}\ z\ \mathbf{of}\ \{(\mathbf{inj}\ \mathsf{op}_i\ (p, r) \mapsto [\![N_i]\!])_i\}$$

Return types of k and h determined by the context

Some kind of **polymorphic** delimited continuations?

Typing: effect handlers as delimited continuations

$$[\![\text{op } V]\!] = \mathcal{S}k.\mathcal{S}h.h\ (\textbf{inj}\ \text{op}\ ([\![V]\!], \lambda x.\langle k\ x \mid h\rangle))$$
$$[\![\textbf{handle } M \textbf{ with } H]\!] = \langle\langle [\![M]\!] \mid [\![H^{\text{ret}}]\!]\rangle \mid [\![H^{\text{ops}}]\!]\rangle$$

where

$$[\![\textbf{return } x \mapsto N]\!] = \lambda x.\mathcal{S}h.[\![N]\!]$$
$$[\![(\text{op}_i\ p\ r \mapsto N_i)_i]\!] = \lambda z.\textbf{case } z \textbf{ of } \{(\textbf{inj}\ \text{op}_i\ (p, r) \mapsto [\![N_i]\!])_i\}$$

Return types of k and h determined by the context

Some kind of **polymorphic** delimited continuations?

Yes! [Maciej Piróg, Piotr Polesiuk, Filip Sieczkowski, FSCD 2019]

Typing: CPS for effect handlers

Computation types

$$\mathcal{C}_R^{A!E} = (\llbracket A \rrbracket \to (\llbracket E \rrbracket R \to R) \to R) \to (\llbracket E \rrbracket R \to R) \to X$$
$$\llbracket A!E \rrbracket = \forall X.\mathcal{C}_X^{A!E}$$

Effect types

$$\llbracket \{\mathsf{op}_i : \llbracket A_i \rrbracket \to \llbracket B_i \rrbracket\}_i \rrbracket C = [\mathsf{op}_i : \llbracket A_i \rrbracket \times (\llbracket B_i \rrbracket \to C)]_i$$

Typing: CPS for effect handlers

Computation types

$$\mathcal{C}_R^{A!E} = (\llbracket A \rrbracket \to (\llbracket E \rrbracket R \to R) \to R) \to (\llbracket E \rrbracket R \to R) \to X$$
$$\llbracket A!E \rrbracket = \forall X.\mathcal{C}_X^{A!E}$$

Effect types

$$\llbracket \{\mathsf{op}_i : \llbracket A_i \rrbracket \to \llbracket B_i \rrbracket\}_i \rrbracket C = [\mathsf{op}_i : \llbracket A_i \rrbracket \times (\llbracket B_i \rrbracket \to C)]_i$$

Operations and handlers

$$\llbracket (\mathsf{op}\ V)^E \rrbracket = \Lambda X.\lambda k^{\llbracket B \rrbracket \to (\llbracket E \rrbracket X \to X) \to X}\ h^{\llbracket E \rrbracket X \to X}.h\ (\mathbf{inj}\ \mathsf{op}\ (\llbracket V \rrbracket, \lambda x^{\llbracket B \rrbracket}.k\ x\ h))^{\llbracket E \rrbracket X}$$
$$\llbracket \mathbf{handle}\ M\ \mathbf{with}\ H^{C \Rightarrow D} \rrbracket = \Lambda X.\llbracket M \rrbracket\ \mathcal{C}_X^D\ \llbracket H^{\mathrm{ret}} \rrbracket^{C \Rightarrow D}\ \llbracket H^{\mathrm{ops}} \rrbracket^{C \Rightarrow D}$$

where

$$\mathsf{op}\ :\ A \twoheadrightarrow B \in E$$
$$\llbracket \mathbf{return}\ x \mapsto N \rrbracket^{A!E \Rightarrow D} = \lambda x^{\llbracket A \rrbracket}\ h^{\llbracket E \rrbracket \mathcal{C}_X^D \to \mathcal{C}_X^D}.\llbracket N \rrbracket X$$
$$\llbracket (\mathsf{op}_i\ p\ k \mapsto N_i)_i \rrbracket^{C \Rightarrow D} = \lambda z^{\llbracket E \rrbracket \mathcal{C}_X^D}.\mathbf{case}\ z\ \{(\mathbf{inj}\ \mathsf{op}_i\ (p,k) \mapsto \llbracket N_i \rrbracket X)_i\}$$

Typing: CPS for effect handlers

Computation types

$$\mathcal{C}_R^{A!E} = (\llbracket A \rrbracket \to (\llbracket E \rrbracket R \to R) \to R) \to (\llbracket E \rrbracket R \to R) \to X$$
$$\llbracket A!E \rrbracket = \forall X.\mathcal{C}_X^{A!E}$$

Effect types

$$\llbracket \{\mathsf{op}_i : \llbracket A_i \rrbracket \to \llbracket B_i \rrbracket\}_i \rrbracket C = [\mathsf{op}_i : \llbracket A_i \rrbracket \times (\llbracket B_i \rrbracket \to C)]_i$$

Operations and handlers

$$\llbracket\ \mathsf{op}\ V\ \rrbracket = \quad \lambda k \qquad\qquad h \qquad .h\ (\mathbf{inj}\ \mathsf{op}\ (\llbracket V \rrbracket, \lambda x \quad .k\ x\ h))$$
$$\llbracket \mathbf{handle}\ M\ \mathbf{with}\ H \quad \rrbracket = \quad \llbracket M \rrbracket \quad \llbracket H^{\mathrm{ret}} \rrbracket \quad \llbracket H^{\mathrm{ops}} \rrbracket$$

where

$$\llbracket \mathbf{return}\ x \mapsto N \rrbracket \quad = \lambda x \quad h \quad .\llbracket N \rrbracket$$
$$\llbracket (\mathsf{op}_i\ p\ k \mapsto N_i)_i \rrbracket \quad = \lambda z \quad .\mathbf{case}\ z\ \{(\mathbf{inj}\ \mathsf{op}_i\ (p,k) \mapsto \llbracket N_i \rrbracket\ )_i\}$$

Typing: CPS for effect handlers

Computation types

$$\mathcal{C}_R^{A!E} = (\llbracket A \rrbracket \to (\llbracket E \rrbracket R \to R) \to R) \to (\llbracket E \rrbracket R \to R) \to X$$
$$\llbracket A!E \rrbracket = \forall X.\mathcal{C}_X^{A!E}$$

Effect types

$$\llbracket \{\mathsf{op}_i : \llbracket A_i \rrbracket \to \llbracket B_i \rrbracket\}_i \rrbracket C = [\mathsf{op}_i : \llbracket A_i \rrbracket \times (\llbracket B_i \rrbracket \to C)]_i$$

Operations and handlers

$$\llbracket \mathsf{op}\ V \rrbracket = \lambda k\ h.h\ (\mathbf{inj}\ \mathsf{op}\ (\llbracket V \rrbracket, \lambda x.k\ x\ h))$$
$$\llbracket \mathbf{handle}\ M\ \mathbf{with}\ H \rrbracket = \llbracket M \rrbracket\ \ \llbracket H^{\mathrm{ret}} \rrbracket\ \llbracket H^{\mathrm{ops}} \rrbracket$$

where

$$\llbracket \mathbf{return}\ x \mapsto N \rrbracket = \lambda x\ h.\llbracket N \rrbracket$$
$$\llbracket (\mathsf{op}_i\ p\ r \mapsto N_i)_i \rrbracket = \lambda z.\mathbf{case}\ z\ \{(\mathbf{inj}\ \mathsf{op}_i\ (p, r) \mapsto \llbracket N_i \rrbracket)_i\}$$

Typing: CPS for effect handlers

Computation types

$$\mathcal{C}_R^{A!E} = (\llbracket A \rrbracket \to (\llbracket E \rrbracket R \to R) \to R) \to (\llbracket E \rrbracket R \to R) \to X$$
$$\llbracket A!E \rrbracket = \forall X.\mathcal{C}_X^{A!E}$$

Effect types

$$\llbracket \{\mathsf{op}_i : \llbracket A_i \rrbracket \to \llbracket B_i \rrbracket\}_i \rrbracket C = [\mathsf{op}_i : \llbracket A_i \rrbracket \times (\llbracket B_i \rrbracket \to C)]_i$$

Operations and handlers

$$\llbracket\ \mathsf{op}\ V\ \rrbracket = \quad \lambda k \qquad\qquad h \qquad .h\ (\mathbf{inj}\ \mathsf{op}\ (\llbracket V \rrbracket, \lambda x \quad .k\ x\ h))$$
$$\llbracket \mathbf{handle}\ M\ \mathbf{with}\ H \quad \rrbracket = \quad \llbracket M \rrbracket \quad \llbracket H^{\mathrm{ret}} \rrbracket \quad \llbracket H^{\mathrm{ops}} \rrbracket$$

where

$$\llbracket \mathbf{return}\ x \mapsto N \rrbracket \quad = \lambda x \quad h \quad .\llbracket N \rrbracket$$
$$\llbracket (\mathsf{op}_i\ p\ r \mapsto N_i)_i \rrbracket \quad = \lambda z \quad .\mathbf{case}\ z\ \{(\mathbf{inj}\ \mathsf{op}_i\ (p, r) \mapsto \llbracket N_i \rrbracket\ )_i\}$$

Typing: CPS for effect handlers

Computation types

$$\mathcal{C}_R^{A!E} = (\llbracket A \rrbracket \to (\llbracket E \rrbracket R \to R) \to R) \to (\llbracket E \rrbracket R \to R) \to X$$
$$\llbracket A!E \rrbracket = \forall X.\mathcal{C}_X^{A!E}$$

Effect types

$$\llbracket \{\mathsf{op}_i : \llbracket A_i \rrbracket \to \llbracket B_i \rrbracket\}_i \rrbracket C = [\mathsf{op}_i : \llbracket A_i \rrbracket \times (\llbracket B_i \rrbracket \to C)]_i$$

Operations and handlers

$$\llbracket (\mathsf{op}\ V)^E \rrbracket = \Lambda X.\lambda k^{\llbracket B \rrbracket \to (\llbracket E \rrbracket X \to X) \to X}\ h^{\llbracket E \rrbracket X \to X}.h\ (\mathbf{inj}\ \mathsf{op}\ (\llbracket V \rrbracket, \lambda x^{\llbracket B \rrbracket}.k\ x\ h))^{\llbracket E \rrbracket X}$$
$$\llbracket \mathbf{handle}\ M\ \mathbf{with}\ H^{C \Rightarrow D} \rrbracket = \Lambda X.\llbracket M \rrbracket\ \mathcal{C}_X^D\ \llbracket H^{\mathrm{ret}} \rrbracket^{C \Rightarrow D}\ \llbracket H^{\mathrm{ops}} \rrbracket^{C \Rightarrow D}$$

where

$$\mathsf{op}\ :\ A \twoheadrightarrow B \in E$$
$$\llbracket \mathbf{return}\ x \mapsto N \rrbracket^{A!E \Rightarrow D} = \lambda x^{\llbracket A \rrbracket}\ h^{\llbracket E \rrbracket \mathcal{C}_X^D \to \mathcal{C}_X^D}.\llbracket N \rrbracket X$$
$$\llbracket (\mathsf{op}_i\ p\ r \mapsto N_i)_i \rrbracket^{C \Rightarrow D} = \lambda z^{\llbracket E \rrbracket \mathcal{C}_X^D}.\mathbf{case}\ z\ \{(\mathbf{inj}\ \mathsf{op}_i\ (p, r) \mapsto \llbracket N_i \rrbracket X)_i\}$$

Typing summary

- (DEL $\mapsto$ MON) preserves **simple** typing
- (MON $\mapsto$ EFF) and (DEL $\mapsto$ EFF) require **parametric** operations
- (MON $\mapsto$ DEL) and (EFF $\mapsto$ DEL) require **polymorphic** delimited continuations
- (EFF $\mapsto$ MON) requires value **polymorphism** or algebraic data types and recursion