Delimited continuations, macro expressiveness, and effect handlers

Sam Lindley

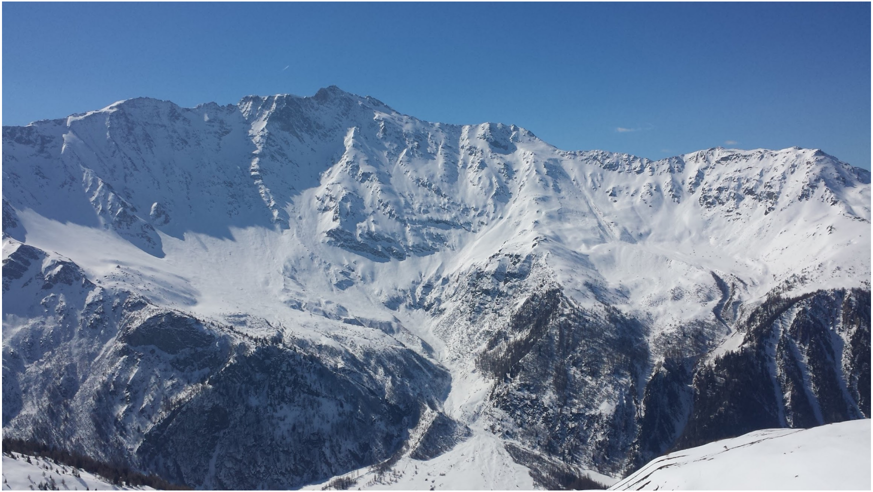
Laboratory for Foundations of Computer Science
The University of Edinburgh

Sam.Lindley@ed.ac.uk

May 19th, 2017

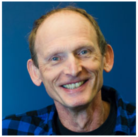
(based on joint work with Yannick Forster, Ohad Kammar, and Matija Pretnar)

Delimited continuations



Delimited continuations

Control and prompt



Matthias Felleisen

Shift and reset

Olivier Danvy

Andrzej Filinski

Delimited continuations can “express” any “definable” monad.

[Felleisen, 1988]
[Danvy and Filinski, 1990]
[Filinski, 1994]

Static delimited continuations

$$\text{“Alice”} +\!\!+ \langle \text{“ has ”} +\!\!+ (\mathcal{S}k.(k\ \text{“a dog”}) +\!\!+ \text{“ and the dog”} +\!\!+ (k\ \text{“a bone.”}))\rangle$$

- $\langle - \rangle$ is called *reset*: it delimits a continuation
- $\mathcal{S}$ is called *shift*: it captures a delimited continuation

(example from [Materzok and Biernacki, 2011])

Static delimited continuations

$$\text{“Alice”} +\!\!+ \langle \text{“ has ”} +\!\!+ (\mathcal{S}k.(k\ \text{“a dog”}) +\!\!+ \text{“ and the dog”} +\!\!+ (k\ \text{“a bone.”}))\rangle$$

evaluates to:

“Alice has a dog and the dog has a bone.”

- $\langle - \rangle$ is called *reset*: it delimits a continuation
- $\mathcal{S}$ is called *shift*: it captures a delimited continuation

(example from [Materzok and Biernacki, 2011])

Static delimited continuations

$$\text{“Alice”} \mathbin{+\!\!+} \langle \text{“ has ”} \mathbin{+\!\!+} (\mathcal{S}k.(k\ \text{“a dog”}) \mathbin{+\!\!+} \text{“ and the dog”} \mathbin{+\!\!+} (k\ \text{“a bone.”}))\rangle$$

evaluates to:

$$\text{“Alice has a dog and the dog has a bone.”}$$

The *delimited continuation* k is bound to

$$\langle \text{“ has ”} \mathbin{+\!\!+} [\,]\rangle$$

where the hole $[\,]$ is filled with “a dog” and “a bone.”.

- $\langle - \rangle$ is called *reset*: it delimits a continuation
- $\mathcal{S}$ is called *shift*: it captures a delimited continuation

(example from [Materzok and Biernacki, 2011])

Dynamic delimited continuations

$$\langle \text{“Alice”} +\!\!+ \langle \text{“ has ”} +\!\!+ (\mathcal{S}k_1.\mathcal{S}k_2.\text{“A cat”} +\!\!+ (k_1\ (k_2\ \text{“.”})))\rangle\rangle$$

(example from [Materzok and Biernacki, 2011])

Dynamic delimited continuations

$$\langle \text{“Alice”} \mathbin{++} \langle \text{“ has ”} \mathbin{++} (\mathcal{S}k_1.\mathcal{S}k_2.\text{“A cat”} \mathbin{++} (k_1\ (k_2\ \text{“.”}))) \rangle \rangle$$

evaluates to:

$$\text{“A cat has Alice.”}$$

(example from [Materzok and Biernacki, 2011])

Dynamic delimited continuations

$$\langle \text{“Alice”} +\!\!+ \langle \text{“ has ”} +\!\!+ (\mathcal{S}k_1.\mathcal{S}k_2.\text{“A cat”} +\!\!+ (k_1\ (k_2\ \text{“.”})))\rangle\rangle$$

evaluates to:

$$\text{“A cat has Alice.”}$$

k_1 is bound to $\langle \text{“ has ”} +\!\!+ [\,]\rangle$
k_2 is bound to $\langle \text{“Alice”} +\!\!+ [\,]\rangle$

(example from [Materzok and Biernacki, 2011])

Dynamic delimited continuations

$$\langle\text{“Alice”} +\!\!+ \langle\text{“ has ”} +\!\!+ (\mathcal{S}k_1.\mathcal{S}k_2.\text{“A cat”} +\!\!+ (k_1\ (k_2\ \text{“.”})))\rangle\rangle$$

evaluates to:

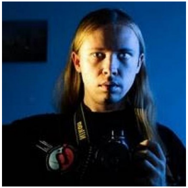
$$\text{“A cat has Alice.”}$$

$$k_1 \text{ is bound to } \langle\text{“ has ”} +\!\!+ [\,]\rangle$$
$$k_2 \text{ is bound to } \langle\text{“Alice”} +\!\!+ [\,]\rangle$$

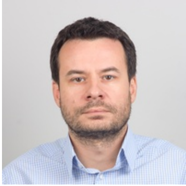
$$\begin{array}{ll}
 & \langle\text{“Alice”} +\!\!+ \langle\text{“ has ”} +\!\!+ (\mathcal{S}k_1.\mathcal{S}k_2.\text{“A cat”} +\!\!+ (k_1\ (k_2\ \text{“.”})))\rangle\rangle \\
\rightsquigarrow & \langle\text{“Alice”} +\!\!+ (\mathcal{S}k_2.\text{“A cat”} +\!\!+ (\langle\text{“ has ”} +\!\!+ [\,]\rangle\ (k_2\ \text{“.”})))\rangle \\
\rightsquigarrow & \text{“A cat”} +\!\!+ \langle\text{“ has ”} +\!\!+ [\,]\rangle\ (\langle\text{“Alice”} +\!\!+ [\,]\rangle\ \text{“.”}) \\
\rightsquigarrow^* & \text{“A cat has Alice.”}
\end{array}$$

(example from [Materzok and Biernacki, 2011])

Subtyping delimited continuations



Marek Materzok



Dariusz Biernacki

[Materzok and Biernacki, 2011]

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)

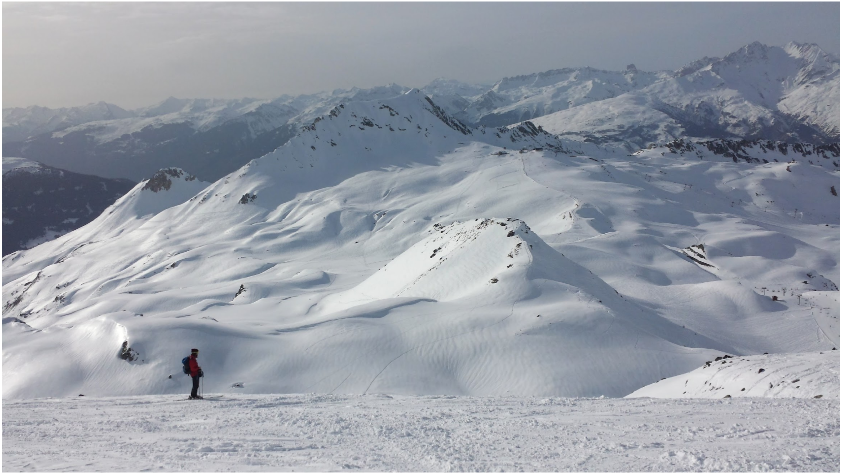
$$\begin{aligned}\langle \mathcal{E}[\mathcal{S}k.M] \rangle &\rightsquigarrow M[(\lambda x.\langle \mathcal{E}[x] \rangle)/k] \\ \langle V \rangle &\rightsquigarrow V\end{aligned}$$

Slight variation ("dollar" in place of reset0)

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

$$\langle M \rangle \equiv \langle M \mid x.x \rangle$$

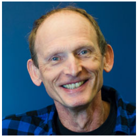
Macro expressiveness



On the expressive power of programming languages

There are many different notions of expressive power. Examples:

- What functions can be expressed (not very interesting for Turing-complete languages)
- Algorithmic complexity
- *Macro expressiveness*



Matthias Felleisen

[Felleisen, 1990]

Macro expressiveness

Language L macro expresses language L' if there exists a *local* transformation of L' into L.

Analogy with logic:

$$\begin{array}{r}\text{local transformation} \simeq \text{derivable judgement}\\ \text{global transformation} \simeq \text{admissible judgement}\end{array}$$

Example: nondeterminism

choose	binary nondeterministic choice (true / false)
fail	nullary nondeterministic choice
run	run a nondeterministic computation

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

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

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

Example: nondeterminism (plain λ-calculus)

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

$$
\begin{aligned}
[\![x]\!] &= [x] \\
[\![\lambda x.M]\!] &= [\lambda x.[\![M]\!]] \\
[\![M\,N]\!] &= \mathbf{concat}\;[f\;x \mid f \leftarrow [\![M]\!], x \leftarrow [\![N]\!]] \\
[\![\mathbf{choose}]\!] &= [\mathsf{true}, \mathsf{false}] \\
[\![\mathbf{fail}]\!] &= [] \\
[\![\mathbf{run}\;M]\!] &= [\![M]\!]
\end{aligned}
$$

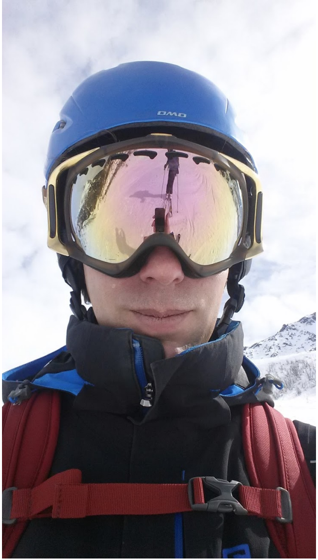
(We assume standard encodings of booleans and lists.)

Example: nondeterminism (delimited continuations)

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

$$
\begin{aligned}
[\![\mathbf{choose}]\!] &= \mathcal{S}k.k\ \mathsf{true} \mathbin{+\!\!+} k\ \mathsf{false} \\
[\![\mathbf{fail}]\!] &= \mathcal{S}k.[] \\
[\![\mathbf{run}\ M]\!] &= \langle M \mid x.[x] \rangle
\end{aligned}
$$

Effect handlers



Effect handlers structure delimited continuations

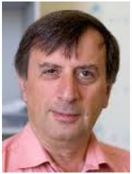
"effects + handlers" : "delimited continuations"

=

"while" : "goto"

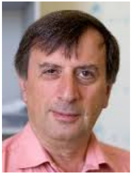
Andrej Bauer

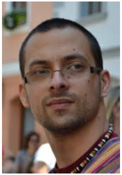
Algebraic effects


Gordon Plotkin


John Power

Effect handlers


Gordon Plotkin


Matija Pretnar

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

Example: nondeterminism (effect handlers)

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

$$
\begin{aligned}
[\![\mathbf{choose}]\!] &= \mathsf{choose}\ () \\
[\![\mathbf{fail}]\!] &= \mathsf{fail}\ () \\
[\![\mathbf{run}\ M]\!] &= \mathbf{handle}\ M\ \mathbf{with} \\
&\qquad \begin{array}{ll}
\mathbf{return}\ x & \mapsto [x] \\
\mathsf{choose}\ ()\ k & \mapsto k\ \mathsf{true} \mathbin{+\!\!+} k\ \mathsf{false} \\
\mathsf{fail}\ ()\ k & \mapsto []
\end{array}
\end{aligned}
$$

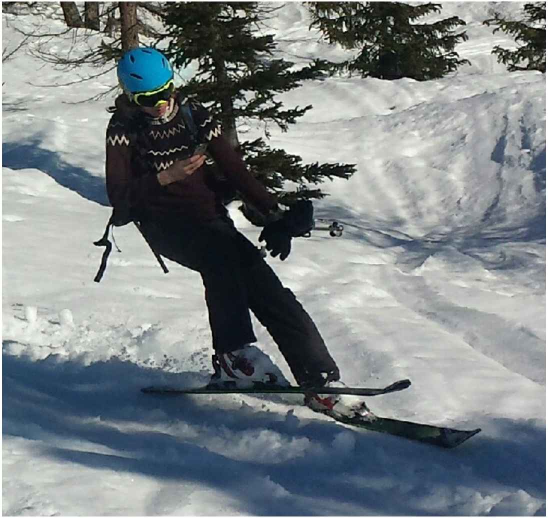
Operational semantics for effect handlers

$$
\begin{aligned}
\textbf{handle } V \textbf{ with } H &\leadsto N[V/x] \\
\textbf{handle } \mathcal{E}[\mathsf{op_i}\ V] \textbf{ with } H &\leadsto N_i[V/p, \lambda x.\textbf{handle } \mathcal{E}[x] \textbf{ with } H/k]
\end{aligned}
$$

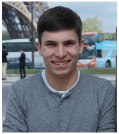
where

$$
\begin{aligned}
H = \textbf{return } x &\mapsto N \\
\mathsf{op_1}\ p\ k \quad &\mapsto N_1 \\
\dots \\
\mathsf{op_n}\ p\ k \quad &\mapsto N_n
\end{aligned}
$$

Delimited continuations versus effect handlers

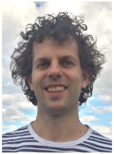


On the expressive power of user-defined effects: effect handlers, monadic reflection, delimited continuations

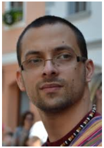


Yannick Forster

Ohad Kammar



Sam Lindley



Matija Pretnar

[Forster et al., 2017]

Delimited continuations as effect handlers

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

Theorem

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

Effect handlers as delimited continuations

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

where

$$
\begin{aligned}
H = \mathbf{return}\ x &\mapsto N \qquad & H^{\mathrm{ret}} &= x.\mathcal{S}h.[\![N]\!] \\
\mathsf{op}_1\ p\ k &\mapsto N_1 & H^{\mathrm{ops}} &= y.\mathbf{case}\ y\ \mathbf{of}\ \mathsf{op}_1\ (p, k) \to [\![N_1]\!] \\
\ldots & & & \qquad\qquad\qquad \ldots \\
\mathsf{op}_n\ p\ k &\mapsto N_n & & \qquad\qquad\qquad \mathsf{op}_n\ (p, k) \to [\![N_n]\!]
\end{aligned}
$$

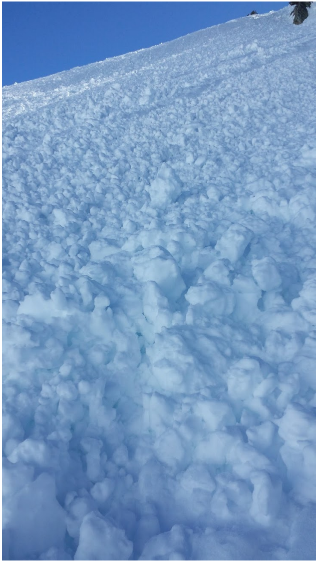
Theorem

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

Types

- Simple types for delimited continuations and effect handlers
- Neither local transformation preserves typeability of terms
- No typeability-preserving local transformations exist
- Polymorphic operations (Del $\mapsto$ Eff)
- Polymorphism (Eff $\mapsto$ Del)

Conclusions



- Expressiveness is subtle
- Untyped delimited continuations and effect handlers can macro express one another
- Simply typed delimited continuations and effect handlers cannot macro express one another
- Polymorphism may allow type-preserving macro translations between delimited continuations and effect handlers
- Macro expressiveness has limitations

References

Olivier Danvy and Andrzej Filinski. Abstracting Control. *LISP and functional programming 1990*.

Yannick Forster, Ohad Kammar, Sam Lindley, and Matija Pretnar. On the expressive power of user defined effects: effect handlers, monadic reflection, delimited continuations. *ICFP 2017*.

Matthias Felleisen. The theory and practice of first-class prompts. *POPL 1988*.

Matthias Felleisen. On the expressive power of programming languages. *ESOP 1990*.

Andrzej Filinski. Representing monads. *POPL 1994*.

Marek Materzok and Dariusz Biernacki. Subtyping delimited continuations. *ICFP 2011*.

Gordon Plotkin and John Power. Adequacy for algebraic effects. *FoSSaCS 2001*.

Gordon Plotkin and Matija Pretnar. Handlers of algebraic effects. *ESOP 2009*.

Static delimited continuations example

Delimited continuations

$$\text{“Alice”} +\!\!+ \langle \text{“ has ”} +\!\!+ (\mathcal{S}k.(k\ \text{“a dog”}) +\!\!+ \text{“ and the dog”} +\!\!+ (k\ \text{“a bone.”}))\rangle$$

Effect handlers

$$\begin{array}{l}
\text{“Alice”} +\!\!+ \\
\quad \textbf{handle}\ \text{“ has ”} +\!\!+ \mathsf{trans}\ (\text{“dog”}, \text{“bone”})\ \textbf{with} \\
\qquad \begin{array}{ll}
\textbf{return}\ x & \mapsto x \\
\mathsf{trans}\ (p, q)\ k & \mapsto (k\ (\text{“a ”} +\!\!+ p)) +\!\!+ \text{“ and the ”} +\!\!+ p +\!\!+ \\
& \quad (k\ (\text{“a ”} +\!\!+ q))
\end{array}
\end{array}$$

Dynamic delimited continuations example

Delimited continuations

$$\langle \text{“Alice”} \mathbin{+\!\!+} \langle \text{“ has ”} \mathbin{+\!\!+} (\mathcal{S}k_1.\mathcal{S}k_2.\text{“A cat”} \mathbin{+\!\!+} (k_1\ (k_2\ \text{“.”}))) \rangle \rangle$$

Effect handlers

$$
\begin{array}{l}
\textbf{handle}\ \text{“Alice”} \mathbin{+\!\!+} \\
\quad \textbf{handle}\ \text{“ has ”} \mathbin{+\!\!+} \mathsf{subject}\ (\text{“A cat”})\ \textbf{with} \\
\quad\quad \textbf{return}\ x \quad \mapsto x \\
\quad\quad \mathsf{subject}\ s\ k \mapsto s \mathbin{+\!\!+} k\ (\mathsf{object}\ \text{“.”}) \\
\textbf{with} \\
\quad \textbf{return}\ x \ \ \mapsto x \\
\quad \mathsf{object}\ p\ k \mapsto k\ p
\end{array}
$$