Delimited Control with Multiple Prompts in Theory and Practice

Paul Downen   Zena M. Ariola

University of Oregon

HOPE'14 — August 31, 2014

Crash course on control

Separating a redex from its evaluation context

$$1 + 2 + (3 \times 4)$$

Separating a redex from its evaluation context

$$1 + 2 + (3 \times 4)$$

$$1 + 2 + \square \qquad\qquad 3 \times 4$$

Separating a redex from its evaluation context

$$1 + 2 + (3 \times 4)$$

$$\boxed{1 + 2 + \square} \qquad \boxed{3 \times 4}$$

$$\Downarrow$$

$$\boxed{1 + 2 + \square} \qquad \boxed{12}$$

$$1 + 2 + 12$$

Classical control: Abortive continuations

$$1 + 2 + \mathsf{call/cc}(\lambda k.3 \times (k\ 4))$$

Classical control: Abortive continuations

$$1 + 2 + \text{call/cc}(\lambda k.3 \times (k\ 4))$$

$$1 + 2 + \square \qquad\qquad \text{call/cc}(\lambda k.3 \times (k\ 4))$$

Classical control: Abortive continuations

$$1 + 2 + \text{call/cc}(\lambda k.3 \times (k\ 4))$$

$$\boxed{1 + 2 + \boxed{\square}} \qquad \boxed{\text{call/cc}(\lambda k.3 \times (k\ 4))}$$

$$\Downarrow$$

$$k\ :\ \boxed{1 + 2 + \boxed{\square}} \qquad \boxed{3 \times (k\ 4)}$$

$$1 + 2 + 3 \times\ (k\ 4)$$

Classical control: Abortive continuations

$$1 + 2 + \text{call/cc}(\lambda k.3 \times (k\ 4))$$

$$1 + 2 + \square \qquad\qquad \text{call/cc}(\lambda k.3 \times (k\ 4))$$

$$\Downarrow$$

$$k\ :\ 1 + 2 + \square \qquad\qquad 3 \times (k\ 4)$$

$$1 + 2 + 3 \times (k\ 4)$$

$$1 + 2 + 3 \times \square \qquad\qquad k\ 4$$

Classical control: Abortive continuations

$$1 + 2 + \text{call/cc}(\lambda k.3 \times (k\ 4))$$

$$1 + 2 + \Box \qquad\qquad \text{call/cc}(\lambda k.3 \times (k\ 4))$$

$$\Downarrow$$

$$\boxed{k} : 1 + 2 + \Box \qquad\qquad 3 \times (k\ 4)$$

$$1 + 2 + 3 \times (k\ 4)$$

$$\boxed{1 + 2 + 3 \times \boxed{\Box}} \qquad\qquad \boxed{k\ 4}$$

$$\Downarrow$$

$$\boxed{1 + 2 + \boxed{\Box}} \qquad\qquad \boxed{4}$$

$$1 + 2 + 4$$

Delimited control: Composable continuations

$$1 + \#(\, 2 + \mathcal{F}(\lambda k.3 \times (k\ 4))\ )$$

Delimited control: Composable continuations

$$1 + \#(2 + \mathcal{F}(\lambda k.3 \times (k\ 4)))$$

$1 + \# \square$ $\quad$ $2 + \square$ $\quad$ $\mathcal{F}(\lambda k.3 \times (k\ 4))$

Delimited control: Composable continuations

$$1 + \#(\,2 + \,\mathcal{F}(\lambda k.3 \times (k\ 4))\,\,)$$

$1 + \#\square$ $\quad k\ :\ 2 + \square$ $\quad \mathcal{F}(\lambda k.3 \times (k\ 4))$

$$\Downarrow$$

$1 + \#\square$ $\quad \square$ $\quad 3 \times (k\ 4)$

$$1 + \#(\,3 \times\ (k\ 4)\,\,)$$

Delimited control: Composable continuations

$$1 + \#(\, 2 + \mathcal{F}(\lambda k.3 \times (k\ 4))\ )$$

$$1 + \# \Box \qquad k\ :\ 2 + \Box \qquad \mathcal{F}(\lambda k.3 \times (k\ 4))$$

$$\Downarrow$$

$$1 + \# \Box \qquad \Box \qquad 3 \times (k\ 4)$$

$$1 + \#(\, 3 \times (k\ 4)\ )$$

$$1 + \# \Box \qquad 3 \times \Box \qquad k\ 4$$

Delimited control: Composable continuations

$$1 + \#(\, 2 + \ \mathcal{F}(\lambda k.3 \times (k\ 4))\ )$$

$$1 + \# \square \qquad k : \ 2 + \square \qquad \mathcal{F}(\lambda k.3 \times (k\ 4))$$

$$\Downarrow$$

$$1 + \# \square \qquad \square \qquad 3 \times (k\ 4)$$

$$1 + \#(\, 3 \times \ (k\ 4)\ )$$

$$1 + \# \square \qquad 3 \times \square \qquad k\ 4$$

$$\Downarrow$$

$$1 + \# \square \qquad 3 \times \square \qquad 2 + 4$$

$$1 + \#(\, 3 \times \ (2 + 4)\ )$$

A zoo of delimited control operators

Design decisions

$$E[\#(E'[\mathcal{F}\ V])]$$

- Does $\mathcal{F}$ remove $\#$ surrounding E'?
- Does continuation guard its call-site with a $\#$?

A family of operators: $*\mathcal{F}*$

$$E[\#(\,E'[\,+\mathcal{F}+\;\;V\,]\,)] \mapsto E[\#(\;V\;\;k\;)]$$

where $k\;\;x = \#(\,E'[\,x\,]\,)$

$$E[\#(\,E'[\,+\mathcal{F}-\;\;V\,]\,)] \mapsto E[\#(\;V\;\;k\;)]$$

where $k\;\;x = E'[\,x\,]$

$$E[\#(\,E'[\,-\mathcal{F}+\;\;V\,]\,)] \mapsto E[\,V\;\;k\,]$$

where $k = \#(\,E'[\,x\,]\,)$

$$E[\#(\,E'[\,-\mathcal{F}-\;\;V\,]\,)] \mapsto E[\,V\;\;k\,]$$

where $k\;\;x = E'[\,x\,]$

A family of operators: $+\mathcal{F}*$ vs $-\mathcal{F}*$

$$E[\#(E'[+\mathcal{F}+\ V])] \mapsto E[\#(V\ k)]$$

where $k\ x = \#(E'[x])$

$$E[\#(E'[+\mathcal{F}-\ V])] \mapsto E[\#(V\ k)]$$

where $k\ x = E'[x]$

$$E[\#(E'[-\mathcal{F}+\ V])] \mapsto E[V\ k]$$

where $k = \#(E'[x])$

$$E[\#(E'[-\mathcal{F}-\ V])] \mapsto E[V\ k]$$

where $k\ x = E'[x]$

A family of operators: $*\mathcal{F}+$ vs $*\mathcal{F}-$

$$E[\#(\,E'[+\mathcal{F}+\ V]\,)] \mapsto E[\#(\,V\ k\,)]$$

$$\textbf{where}\ k\ x = \#(\,E'[x]\,)$$

$$E[\#(\,E'[+\mathcal{F}-\ V]\,)] \mapsto E[\#(\,V\ k\,)]$$

$$\textbf{where}\ k\ x = E'[x]$$

$$E[\#(\,E'[-\mathcal{F}+\ V]\,)] \mapsto E[\,V\ k\,]$$

$$\textbf{where}\ k = \#(\,E'[x]\,)$$

$$E[\#(\,E'[-\mathcal{F}-\ V]\,)] \mapsto E[\,V\ k\,]$$

$$\textbf{where}\ k\ x = E'[x]$$

A family of operators

- $+\mathcal{F}+$: shift ($\mathcal{S}$) and reset ($\langle\_\rangle$) of Danvy and Filinski
- $+\mathcal{F}-$: control ($\mathcal{F}$) and prompt ($\#$) of Felleisen
- $-\mathcal{F}+$: shift$_0$ ($\mathcal{S}_0$) and reset$_0$ ($\langle\_\rangle_0$)
- $-\mathcal{F}-$: control$_0$ ($\mathcal{F}_0$) and prompt$_0$ ($\#_0$)

shift = control?

List traversal two ways (Biernacki et al., 2005)

$$
\begin{aligned}
&\mathit{Straverse}\ \mathit{xs} = \langle \mathit{visit}\ \mathit{xs} \rangle \\
&\quad \textbf{where}\ \mathit{visit}\ [\,] \qquad\qquad = [\,] \\
&\qquad\qquad\quad \mathit{visit}\ (x :: \mathit{xs}) \quad = \mathit{visit}\ (\mathcal{S}(\lambda k.\, x :: (\ k\ \mathit{xs}\ )\ )) \\
&\mathit{Ftraverse}\ \mathit{xs} = \#(\mathit{visit}\ \mathit{xs}) \\
&\quad \textbf{where}\ \mathit{visit}\ [\,] \qquad\qquad = [\,] \\
&\qquad\qquad\quad \mathit{visit}\ (x :: \mathit{xs}) \quad = \mathit{visit}\ (\mathcal{F}(\lambda k.\, x :: (\ k\ \mathit{xs}\ )\ ))
\end{aligned}
$$

What's the difference?

shift $\neq$ control

List traversal two ways (Biernacki et al., 2005)

$$
\begin{aligned}
&\mathit{Straverse}\ xs = \langle \mathit{visit}\ xs \rangle \\
&\quad \textbf{where}\ \mathit{visit}\ [] \qquad\qquad = [] \\
&\qquad\qquad \mathit{visit}\ (x :: xs) \quad = \mathit{visit}\ (\mathcal{S}(\lambda k.\, x :: (k\ xs)\,)) \\
&\mathit{Ftraverse}\ xs = \#(\mathit{visit}\ xs) \\
&\quad \textbf{where}\ \mathit{visit}\ [] \qquad\qquad = [] \\
&\qquad\qquad \mathit{visit}\ (x :: xs) \quad = \mathit{visit}\ (\mathcal{F}(\lambda k.\, x :: (k\ xs)\,))
\end{aligned}
$$

What's the difference?

$\mathit{Straverse}\ [1,2,3] \mapsto^* [1,2,3]$	list copy
$\mathit{Ftraverse}\ [1,2,3] \mapsto^* [3,2,1]$	list reverse

shift $=$ shift$_0$?

Continuation swap two ways

$$\begin{aligned} \mathit{swap}\ x &= \mathcal{S}(\lambda k_1.\mathcal{S}(\lambda k_2.k_1\ (k_2\ x))) \\ \mathit{swap}_0\ x &= \mathcal{S}_0(\lambda k_1.\mathcal{S}_0(\lambda k_2.k_1\ (k_2\ x))) \end{aligned}$$

What's the difference?

shift $\neq$ shift$_0$

Continuation swap two ways

$$\begin{aligned} swap\ x &= \mathcal{S}(\lambda k_1.\mathcal{S}(\lambda k_2.k_1\ (k_2\ x))) \\ swap_0\ x &= \mathcal{S}_0(\lambda k_1.\mathcal{S}_0(\lambda k_2.k_1\ (k_2\ x))) \end{aligned}$$

What's the difference?

$$\langle 10 + \langle 2 \times (swap\ 1)\rangle\rangle \mapsto^* \langle 10 + \langle k_1\ (k_2\ 1)\rangle\rangle \mapsto^* 12$$

$$\textbf{where } k_1\ x = \langle 2 \times x\rangle \qquad k_2\ x = \langle x\rangle$$

identity function

$$\langle 10 + \langle 2 \times (swap_0\ 1)\rangle_0\rangle_0 \mapsto^* k_1\ (k_2\ 1) \mapsto^* 22$$

$$\textbf{where } k_1\ x = \langle 2 \times x\rangle_0 \qquad k_2\ x = \langle 10 + x\rangle_0$$

Context switch

Theory vs. Practice

Theory

- Focus on $*\mathcal{F}+$ operators
- shift and reset are heavily studied
- shift_0 and reset_0 recently gaining interest
- Both have theories with desirable properties
  - Simple continuation-passing style semantics
  - Sound and complete axiomatizations
  - Error-free type and effect systems
  - “Observational purity”

Theory vs. Practice

Practice

- Focus on $*\mathcal{F}-$ operators
- Major implementations of delimited control
  - Racket: control and prompt
  - Haskell library CC-delcont: control_0 and prompt_0
  - OCaml library delcontcc: control_0 and prompt_0
- Practical extensions of delimited control
  - Integrated into languages with other effects
  - Multiple prompts

Theory vs. Practice

Practice

- Focus on $*\mathcal{F}-$ operators
- Major implementations of delimited control
  - Racket: control and prompt
  - Haskell library CC-delcont: control_0 and prompt_0
  - OCaml library delcontcc: control_0 and prompt_0
- Practical extensions of delimited control
  - Integrated into languages with other effects
  - Multiple prompts

Giving prompts a name

Multiple prompts

- Multiple prompts, referred to by name
- Similar to exception handling
  - $\mathcal{F}^{\widehat{\alpha}}$: go to nearest prompt (handler) for $\widehat{\alpha}$
  - $\#^{\widehat{\alpha}}$: delimit (handle) control effects for $\widehat{\alpha}$

Multiple prompts: Dynamic continuations

$$\#^{\widehat{\gamma}}(1 + \#^{\widehat{\beta}}(\,2 + \#^{\widehat{\alpha}}(3 + \,\mathcal{F}^{\widehat{\beta}}\;(\lambda k.4 \times (k\;5))\,)\,))$$

Multiple prompts: Dynamic continuations

$$\#^{\widehat{\gamma}}(1 + \#^{\widehat{\beta}}(2 + \#^{\widehat{\alpha}}(3 + \mathcal{F}^{\widehat{\beta}}\ (\lambda k.4 \times (k\ 5)))))$$

$$\#^{\widehat{\gamma}}(1 + \#^{\widehat{\beta}}\square) \qquad 2 + \#^{\widehat{\alpha}}(3 + \square) \qquad \mathcal{F}^{\widehat{\beta}}\ (\lambda k.4 \times (k\ 5))$$

Multiple prompts: Dynamic continuations

$$\#^{\widehat{\gamma}}(1 + \#^{\widehat{\beta}}(\, 2 + \#^{\widehat{\alpha}}(3 + \; \mathcal{F}^{\widehat{\beta}} \; (\lambda k.4 \times (k \; 5)) \,) \,))$$

$$\#^{\widehat{\gamma}}(1 + \#^{\widehat{\beta}} \square \,) \qquad k \; : \; 2 + \#^{\widehat{\alpha}}(3 + \square \,) \qquad \mathcal{F}^{\widehat{\beta}} \; (\lambda k.4 \times (k \; 5))$$

$$\Downarrow$$

$$\#^{\widehat{\gamma}}(1 + \#^{\widehat{\beta}} \square \,) \qquad \square \qquad 4 \times (k \; 5)$$

$$\#^{\widehat{\gamma}}(1 + \#\widehat{\beta}(\, 4 \times \; (k \; 5) \;))$$

Multiple prompts: Dynamic continuations

$$\#^{\widehat{\gamma}}(1 + \#^{\widehat{\beta}}(\,2 + \#^{\widehat{\alpha}}(3 + \ \mathcal{F}^{\widehat{\beta}}\ (\lambda k.4 \times (k\ \ 5))\,)\,))$$

$$\#^{\widehat{\gamma}}(1 + \#^{\widehat{\beta}}\,\square\,) \qquad k\ :\ 2 + \#^{\widehat{\alpha}}(3 + \square\,) \qquad \mathcal{F}^{\widehat{\beta}}\ (\lambda k.4 \times (k\ \ 5))$$

$$\Downarrow$$

$$\#^{\widehat{\gamma}}(1 + \#^{\widehat{\beta}}\,\square\,) \qquad \square \qquad 4 \times (k\ \ 5)$$

$$\#^{\widehat{\gamma}}(1 + \#^{\widehat{\beta}}(\,4 \times (k\ \ 5)\,))$$

$$\#^{\widehat{\gamma}}(1 + \#^{\widehat{\beta}}\,\square\,) \qquad 4 \times \square \qquad k\ \ 5$$

Multiple prompts: Dynamic continuations

$$\#^{\widehat{\gamma}}(1+\#^{\widehat{\beta}}(2+\#^{\widehat{\alpha}}(3+\mathcal{F}^{\widehat{\beta}}\ (\lambda k.4\times(k\ 5)))))$$

$$\#^{\widehat{\gamma}}(1+\#^{\widehat{\beta}}\square) \qquad k : 2+\#^{\widehat{\alpha}}(3+\square) \qquad \mathcal{F}^{\widehat{\beta}}\ (\lambda k.4\times(k\ 5))$$

$$\Downarrow$$

$$\#^{\widehat{\gamma}}(1+\#^{\widehat{\beta}}\square) \qquad \square \qquad 4\times(k\ 5)$$

$$\#^{\widehat{\gamma}}(1+\#\widehat{\beta}(4\times\ (k\ 5)\ ))$$

$$\#^{\widehat{\gamma}}(1+\#^{\widehat{\beta}}\square) \qquad 4\times\square \qquad k\ 5$$

$$\Downarrow$$

$$\#^{\widehat{\gamma}}(1+\#^{\widehat{\beta}}\square) \qquad 4\times\square \qquad 2+\#^{\widehat{\alpha}}(3+5)$$

$$\#^{\widehat{\gamma}}(1+\#^{\widehat{\beta}}(4\times(2+\#^{\widehat{\alpha}}(3+5))))$$

Putting practice to theory

Multiple prompts via marked stacks

- Monadic Framework for Delimited Continuations (Dybvig et al., 2007)
- control_0 and prompt_0 style control with multiple prompts
- Use hybrid abstract/concrete continuation monad
  - Stack of ordinary continuations
  - Special markers representing prompts

Multiple prompts via dynamic binding

- Systematic Approach to Delimited Control with Multiple Prompts (Downen and Ariola, 2012)
- shift_0 and reset_0 style control with multiple prompts
- $\lambda\widehat{\mu}_0$: Conservative extension of CBV Parigot's $\lambda\mu$ (i.e., λ-calculus with call/cc)
  - Dynamic continuation variables
  - Splitting/joining dynamic environment of continuations

Comparing the two frameworks

- Biggest mismatch comes down to representation of meta-contexts
- Monadic framework: marked stack

$$MetaCont = [Ident + Cont]$$
$$[k_3, \widehat{\alpha}_3, \widehat{\alpha}_2, k_2, k_1, \widehat{\alpha}_1]$$

- $\lambda\widehat{\mu}_0$: dynamic environment

$$MetaCont = [Ident * Cont]$$
$$[\widehat{\alpha}_3 \mapsto k_3, \widehat{\alpha}_2 \mapsto k_2, \widehat{\alpha}_1 \mapsto k_1]$$

Dynamic environment to marked stack

- Embed $\lambda\widehat{\mu}_0$ into Monadic Framework
- Flatten $[\mathit{Ident} * \mathit{Cont}]$ to $[\mathit{Ident} + \mathit{Cont}]$?
- Easy!

$$
\begin{aligned}
&[\widehat{\alpha}_3 \mapsto k_3, \widehat{\alpha}_2 \mapsto k_2, \widehat{\alpha}_1 \mapsto k_1] \\
&= [k_3, \widehat{\alpha}_3, k_2, \widehat{\alpha}_2, k_1, \widehat{\alpha}_1]
\end{aligned}
$$

Marked stack to dynamic environment

- Embed Monadic Framework into $\lambda\widehat{\mu}$
- Restore $[\mathit{Iden} + \mathit{Cont}]$ to $[\mathit{Ident} * \mathit{Cont}]$
- Not so easy...

Distinguished return

- Reserve one dynamic continuation variable ($\widehat{\mathsf{tp}}$)
  - "Return" value by sending it to $\widehat{\mathsf{tp}}$
  - "Empty" continuation $k_{\widehat{\mathsf{tp}}}$ "returns" input to $\widehat{\mathsf{tp}}$
  - Continuations associated with $\widehat{\mathsf{tp}}$ (next return point)
  - Prompts associated with empty continuation $k_{\widehat{\mathsf{tp}}}$ (skip other prompts, go to next return point)
- Embedding back into environment!

$$\begin{aligned}
&[k_3, \widehat{\alpha}_3, \widehat{\alpha}_2, k_2, k_1, \widehat{\alpha}_1] \\
&= [\widehat{\mathsf{tp}} \mapsto k_3, \widehat{\alpha}_3 \mapsto k_{\widehat{\mathsf{tp}}}, \widehat{\alpha}_2 \mapsto k_{\widehat{\mathsf{tp}}}, \\
&\qquad \widehat{\mathsf{tp}} \mapsto k_2, \widehat{\mathsf{tp}} \mapsto k_1, \widehat{\alpha}_1 \mapsto k_{\widehat{\mathsf{tp}}}]
\end{aligned}$$

From practice to theory

- Embedding from Monadic Framework to $\lambda\widehat{\mu}_0$
- With multiple prompts, shift_0 and reset_0 style control implements $\mathsf{control}_0$ and prompt_0
- Corollary: shift_0 and reset_0 style control with 2 prompts implements $\mathsf{control}_0$ and prompt_0

Simulation of $*\mathcal{F}-$ style operators

$$\# \equiv \text{Prompt handler for } \mathcal{F}_0, \mathcal{F}$$

$$\langle \_ \rangle \equiv \text{Traces of ``naked'' context composition}$$

$$E ::= \square \mid E\ M \mid V\ E \mid \langle M \rangle$$

$$E[\#(E'[\mathcal{F}_0 V])] \mapsto E[\langle V\ k \rangle] \quad \textbf{where } k = \lambda x.\langle E[x] \rangle$$

$$E[\#(E'[\mathcal{F} V])] \mapsto E[\langle \#(V\ k) \rangle] \quad \textbf{where } k = \lambda x.\langle E[x] \rangle$$

Dynamic traversal

$$\#(\,E[\,\mathit{visit}(x :: xs)\,]\,) \mapsto^* \langle \#(x :: \langle E[\mathit{visit}\ xs]\rangle)\rangle$$

$$\mathit{Ftraverse}\ [1,2,3] \mapsto\ \#(\ \mathit{visit}\ [1,2,3]\ )$$

$$\mapsto^*\ \langle \#(\,1 :: \langle\, \mathit{visit}\ [2,3]\,\rangle\,)\rangle$$

$$\mapsto^*\ \langle\langle \#(\,2 :: \langle 1 :: \langle\, \mathit{visit}\ [3]\,\rangle\rangle\,)\rangle\rangle$$

$$\mapsto^*\ \langle\langle\langle \#(\,3 :: \langle 2 :: \langle 1 :: \langle\, \mathit{visit}\ []\,\rangle\rangle\rangle\,)\rangle\rangle\rangle$$

$$\mapsto \langle\langle\langle \#(3 :: \langle 2 :: \langle 1 :: \langle [] \rangle\rangle\rangle)\rangle\rangle\rangle$$

$$\mapsto^* [3,2,1]$$

Dynamic traversal

$$\#(\, E[\, \mathit{visit}(x :: xs) \,]\,) \mapsto^* \langle \#(x :: \langle E[\mathit{visit}\ xs]\rangle)\rangle$$

$$\mathit{Ftraverse}\ [1,2,3] \mapsto \#(\, \mathit{visit}\ [1,2,3]\,)$$

$$\mapsto^* \langle \#(\, 1 :: \langle\, \mathit{visit}\ [2,3]\,\rangle\,)\rangle$$

$$\mapsto^* \langle\langle \#(\, 2 :: \langle 1 :: \langle\, \mathit{visit}\ [3]\,\rangle\rangle\,)\rangle\rangle$$

$$\mapsto^* \langle\langle\langle \#(\, 3 :: \langle 2 :: \langle 1 :: \langle\, \mathit{visit}\ []\,\rangle\rangle\rangle\,)\rangle\rangle\rangle$$

$$\mapsto \langle\langle\langle \#(3 :: \langle 2 :: \langle 1 :: \langle [] \rangle\rangle\rangle)\rangle\rangle\rangle$$

$$\mapsto^* [3,2,1]$$

Dynamic traversal

$$\#(\, E[\, visit(x :: xs)\,]\,) \mapsto^* \langle \#(x :: \langle E[visit\ xs]\rangle)\rangle$$

$$Ftraverse\ [1,2,3] \mapsto\ \#(\ visit\ [1,2,3]\ )$$

$$\mapsto^*\ \langle \#(\, 1 :: \langle\, visit\ [2,3]\, \rangle\,)\rangle$$

$$\mapsto^*\ \langle\langle \#(\, 2 :: \langle 1 :: \langle\ visit\ [3]\, \rangle\rangle\,)\rangle\rangle$$

$$\mapsto^*\ \langle\langle\langle \#(\, 3 :: \langle 2 :: \langle 1 :: \langle\ visit\ []\, \rangle\rangle\rangle\,)\rangle\rangle\rangle$$

$$\mapsto \langle\langle\langle \#(3 :: \langle 2 :: \langle 1 :: \langle []\rangle\rangle\rangle\rangle)\rangle\rangle\rangle$$

$$\mapsto^* [3,2,1]$$

Dynamic traversal

$$\#(\, E[\, visit(x :: xs)\,]\,) \mapsto^* \langle \#(x :: \langle E[visit\ xs] \rangle) \rangle$$

$$Ftraverse\ [1, 2, 3] \mapsto\ \#(\ visit\ [1, 2, 3]\ )$$

$$\mapsto^*\ \langle \#(\, 1 :: \langle\, visit\ [2, 3]\, \rangle\,) \rangle$$

$$\mapsto^*\ \langle \langle \#(\, 2 :: \langle 1 :: \langle\, visit\ [3] \rangle \rangle\,) \rangle \rangle$$

$$\mapsto^*\ \langle \langle \langle \#(\, 3 :: \langle 2 :: \langle 1 :: \langle\, visit\ []\, \rangle \rangle \rangle\,) \rangle \rangle \rangle$$

$$\mapsto \langle \langle \langle \#(3 :: \langle 2 :: \langle 1 :: \langle [] \rangle \rangle \rangle) \rangle \rangle \rangle$$

$$\mapsto^* [3, 2, 1]$$

Questions?

References

D. Biernacki, O. Danvy, and K. Millikin. A dynamic continuation-passing style for dynamic delimited continuations. BRICS, Department of Computer Science, Univ., 2005.

P. Downen and Z. M. Ariola. A systematic approach to delimited control with multiple prompts. In Procedings of the 21st European Symposium on Programming, pages 234–253. Springer, 2012.

R. K. Dybvig, S. P. Jones, and A. Sabry. A monadic framework for delimited continuations. Journal of Functional Programming, 17(06):687–730, 2007.