An approach to call-by-name delimited continuations

Hugo Herbelin, INRIA Futurs, France

Silvia Ghilezan, University of Novi Sad, Serbia

POPL 2008

Context of the talk

Languages with control operators

$\hookrightarrow$ have operators that capture and modify the flow of control

$$\#1 + 5 * \texttt{Abort}\ 2 \quad \rightarrow \quad 2$$

Languages with delimited control

$$\#1 + \#5 * \texttt{Abort}\ 2 \quad \rightarrow \quad 3$$

Fundamental property (Filinski 1994): delimited control is *complete* for implementing monads in direct style (e.g. exceptions, references, ...)

Outline of the talk

I- Introduction and background

- Drawbacks of previous calculi of control
- A better foundation for control: $\lambda\mu$tp-calculus
- A foundation for call-by-value delimited control: call-by-value $\lambda\mu\widehat{tp}$-calculus

II- Our main results

- A remarkable connection between two a priori unrelated calculi:

  A call-by-name calculus of control known to satisfy observational (Böhm) completeness (namely de Groote and Saurin's $\Lambda\mu$-calculus) is the exact canonical call-by-name variant of $\lambda\mu\widehat{tp}$-calculus.

- A uniform presentation of four calculi of delimited control

Introduction and background

Frameworks to reason about call-by-value (non delimited) control

$\lambda_{\mathcal{C}}$ [Felleisen-Friedman-Kohlbecker-Duba 1986]

- pioneering control calculus (*aiming* at modelling e.g. `call-with-current-continuation`)
- continuations (i.e. the rest of the computation) are *regular functions*

$\lambda\mu$ [Parigot 1992]

- continuations are treated primitively (*structural substitution*)
- more fine-grained (has a primitive notion of terms and of *machine states*)

$\overline{\lambda}\mu\tilde{\mu}$ [Curien-Herbelin 2000]

- even more fine-grained (has a primitive notion of terms, *stacks* and machine states)
- but less *natural*... let's not focus on it in this talk

The operational semantics of `call-with-current-continuation`

definition of evaluation contexts

$$E ::= \Box \mid V\ E \mid E\ t$$

expected semantics

$$E[\texttt{callcc}(\lambda k.t)]$$

$$\downarrow$$

$$E[t[\lambda x.\mathcal{A}(E[x])/k]]$$

$\downarrow$ not reified

$\hookrightarrow$ reified occurrence

The operational semantics of `call-with-current-continuation`

(how to simulate it in $\lambda_{\mathcal{C}}$?)

abbreviations in $\lambda_{\mathcal{C}}$

$$
\begin{aligned}
\texttt{callcc}(\lambda k.t) &\triangleq \mathcal{C}(\lambda k.k\ t) \\
\mathcal{A}(t) &\triangleq \mathcal{C}(\lambda\_.t)
\end{aligned}
$$

simulation

$$E[\mathcal{C}(\lambda k.k\ t)]$$

$$\downarrow$$

$$(\lambda x.\mathcal{A}(E[x]))\ t[\lambda x.\mathcal{A}(E[x])/k]$$

$$\downarrow$$

should not be reified as a function!

Consequences of the *reification* of continuations in $\lambda\mathcal{C}$

- cannot faithfully express the operational semantics of `call-with-current-continuation`
- its *reduction* system cannot faithfully express its own *operational* semantics
- may introduce space leaks in computation

... more in Ariola and Herbelin (JFP, to appear)

The operational semantics of `call-with-current-continuation`

(how to simulate it in $\lambda\mu$?)

abbreviations in $\lambda\mu$

$$\begin{array}{lcl} \texttt{callcc}(\lambda k.t) & \triangleq & \mu\alpha.[\alpha](t[\lambda x.\mu\_.[\alpha]x/k]) \\ \mathcal{A}(t) & \triangleq & \mu\_.[\;?\;].t \end{array}$$

$\mathcal{A}$ discards the current evaluation context and jumps to the toplevel:
we need a toplevel continuation constant to express it

The operational semantics of `call-with-current-continuation`

(how to simulate it in $\lambda\mu$ extended with tp?)

abbreviations in $\lambda\mu$ extended with tp

$$\begin{array}{lcl} \texttt{callcc}(\lambda k.t) & \triangleq & \mu\alpha.[\alpha](t[\lambda x.\mu\_.[\alpha]x/k]) \\ \mathcal{A}(t) & \triangleq & \mu\_.[\mathsf{tp}].t \end{array}$$

simulation

$$\begin{array}{c} E[\texttt{callcc}(\lambda k.t)] \\ = \\ E[\mu\alpha.[\alpha](t[\lambda x.\mu\_.[\alpha]x/k])] \\ \downarrow \\ E[t[\lambda x.\mu\_.[\mathsf{tp}](E[x])/k]] \\ = \\ E[t[\lambda x.\mathcal{A}(E[x])/k]] \end{array}$$

A foundation for control: $\lambda\mu$tp-calculus

$$\begin{array}{lcll} V & ::= & x \mid \lambda x.t & \text{(values)} \\ t, u & ::= & V \mid t\,u \mid \mu\alpha.c & \text{(terms)} \\ c & ::= & [\beta]t \mid [\text{tp}]t & \text{(commands or states)} \end{array}$$

$\lambda\mu$tp-calculus satisfies:

- Faithful simulation of call-with-current-continuation, $\mathcal{A}$, $\mathcal{C}$, ...

- Observationally equivalent to $\lambda\mathcal{C}$ (but not operationally equivalent)

- Confluence, termination in simply-typed case, standardisation, ...

- Internal notion of *state*: evaluations are of the unique form tp V

How to make $\lambda\mu$tp suitable for *delimited* control?

A foundation for delimited control: $\lambda\mu\widehat{tp}$-calculus

Let's turn tp into a dynamically bound variable $\widehat{tp}$:

$$\begin{array}{lcll} V & ::= & x \mid \lambda x.t & \text{(values)} \\ t, u & ::= & V \mid t\,u \mid \mu\alpha.c \mid \mu\widehat{tp}.c & \text{(terms)} \\ c & ::= & [\beta]t \mid [\widehat{tp}]t & \text{(commands or states)} \end{array}$$

The new operator $\mu\widehat{tp}.c$ *delimits* a local toplevel.

The binding is *dynamic* in exactly the same way an exception is *dynamically* caught by the closest surrounding handler.

... more in Ariola, Herbelin and Sabry (HOSC, to appear)

Expressiveness of $\lambda\mu\widehat{tp}$-calculus

Danvy and Filinski's delimited control	**reset** t	$\triangleq$	$\mu\widehat{tp}.[\widehat{tp}]\, t$
	shift	$\triangleq$	$\lambda y.\mu\alpha.[\widehat{tp}]\,(y\ \lambda x.\mu\widehat{tp}.[\alpha]x)$
exception handling	**raise** t	$\triangleq$	$\mu\_.[\widehat{tp}]\, t$
	t **handle** *patterns*	$\triangleq$	$\texttt{case}\ \mu\widehat{tp}.[\widehat{tp}]\,(\texttt{Val}\ t)\ \texttt{of}$ $\mid\ \texttt{Val}\ x \Rightarrow x$ $\mid\ \mathit{patterns}$ $\mid\ x \Rightarrow \mu\_.[\widehat{tp}]\, x$
monads in direct style	$\mu(t)$	$\triangleq$	$\mu\alpha.[\widehat{tp}]\,((\lambda x.\mu\widehat{tp}.[\alpha]x)^*\ t)$
	$[t]$	$\triangleq$	$\mu\widehat{tp}.[\widehat{tp}]\,(\eta\ t)$
mutable reference	**read**	$\triangleq$	$\lambda().\mu\alpha.[\widehat{tp}]\,\lambda s.((\mu\widehat{tp}.[\alpha]s)\ s)$
	write	$\triangleq$	$\lambda s.\mu\alpha.[\widehat{tp}]\,\lambda\_.((\mu\widehat{tp}.[\alpha]())\ s)$

Outline of the talk

I- Introduction and background

- A review: Felleisen's $\lambda_{\mathcal{C}}$, Parigot's $\lambda\mu$, ...

- A foundation for control: $\lambda\mu$tp-calculus (tp is the toplevel continuation constant)

- A foundation for delimited control: $\lambda\mu\widehat{tp}$-calculus ($\widehat{tp}$ is a dynamically scoped variable)

- CBV $\lambda\mu\widehat{tp}$-calculus $\simeq$ `shift`/`reset` calculus

II- Our main results

- A remarkable connection between two a priori unrelated calculi:

  A call-by-name calculus of control known to satisfy observational (Böhm) completeness (namely de Groote and Saurin's $\Lambda\mu$-calculus) is the exact canonical call-by-name variant of $\lambda\mu\widehat{tp}$-calculus.

- A uniform presentation of four calculi of delimited control

Towards our results

Call-by-name $\lambda\mu\widehat{tp}$-calculus?

CBV

$$
\begin{array}{llll}
V & ::= & x \mid \lambda x.t & \text{(values)} \\
t,u & ::= & V \mid t\,u \mid \mu\alpha.c \mid \mu\widehat{tp}.c & \text{(terms)} \\
c & ::= & [\beta]t \mid [\widehat{tp}]t & \text{(commands or states)}
\end{array}
$$

$$
\begin{array}{llcll}
\beta_v : & (\lambda x.\,t)\;V & \to & t[V/x] & \\
\mu_{app} : & (\mu\alpha.\,c)\;t & \to & \mu\beta.c[[\beta](\Box\;t)/\alpha] & \beta \text{ fresh} \\
\mu^{v}_{app} : & V\,(\mu\alpha.\,c) & \to & \mu\beta.c[[\beta](V\;\Box)/\alpha] & \beta \text{ fresh} \\
\mu_{var} : & [\beta]\mu\alpha.c & \to & c[\beta/\alpha] & \\
\mu^{\widehat{tp}}_{var} : & [\widehat{tp}]\mu\alpha.c & \to & c[\widehat{tp}/\alpha] & \\
\eta^{v}_{\widehat{tp}} : & \mu\widehat{tp}.[\widehat{tp}]\;V & \to & V & \text{even if } \widehat{tp} \text{ occurs in } V
\end{array}
$$

Call-by-name $\lambda\mu\widehat{tp}$-calculus?

CBV

not	V	$::=$	$x \mid \lambda x.t$	(values)
mod	t, u	$::=$	$V \mid t\,u \mid \mu\alpha.c \mid \mu\widehat{tp}.c$	(terms)
	c	$::=$	$[\beta]t \mid [\widehat{tp}]t$	(commands or states)

mod	β_v :	$(\lambda x.\, t)\ V$	$\rightarrow$	$t[V/x]$	
	μ_{app} :	$(\mu\alpha.\, c)\ t$	$\rightarrow$	$\mu\beta.c[[\beta](\Box\ t)/\alpha]$	β fresh
not	μ^{v}_{app} :	$V\,(\mu\alpha.\, c)$	$\rightarrow$	$\mu\beta.c[[\beta](V\ \Box)/\alpha]$	β fresh
	μ_{var} :	$[\beta]\mu\alpha.c$	$\rightarrow$	$c[\beta/\alpha]$	
not	$\mu^{\widehat{tp}}_{var}$:	$[\widehat{tp}]\mu\alpha.c$	$\rightarrow$	$c[\widehat{tp}/\alpha]$	
mod	$\eta^{v}_{\widehat{tp}}$:	$\mu\widehat{tp}.[\widehat{tp}]\ V$	$\rightarrow$	V	even if $\widehat{tp}$ occurs in V

CBN

t, u	$::=$	$x \mid \lambda x.t \mid t\,u \mid \mu\alpha.c \mid \mu\widehat{tp}.c$	(terms)
c	$::=$	$[\beta]t \mid [\widehat{tp}]t$	(commands or states)

β :	$(\lambda x.\, t)\ u$	$\rightarrow$	$t[u/x]$	
μ_{app} :	$(\mu\alpha.\, c)\ t$	$\rightarrow$	$\mu\beta.c[[\beta](\Box\ t)/\alpha]$	β fresh
μ_{var} :	$[\beta]\mu\alpha.c$	$\rightarrow$	$c[\beta/\alpha]$	
$\eta_{\widehat{tp}}$:	$\mu\widehat{tp}.[\widehat{tp}]\ t$	$\rightarrow$	t	even if $\widehat{tp}$ occurs in t

$\lambda\mu$-calculus

Parigot [1992] - computational interpretation of classical natural deduction

$$\begin{array}{lll} t & ::= & x \mid \lambda x.t \mid t\,t \mid \mu\alpha.c \quad \text{(terms)} \\ c & ::= & [\alpha]t \qquad\qquad\qquad\qquad \text{(commands)} \end{array}$$

$$\boxed{\begin{array}{llll} \beta : & (\lambda x.t)\,u & \to & t[u/x] \\ \mu_{app} : & (\mu\alpha.c)\,u & \to & \mu\beta.c[[\beta](\Box\, u)/\alpha] \quad \beta \text{ fresh} \\ \mu_{var} : & [\beta]\mu\alpha.c & \to & c[\beta/\alpha] \end{array}}$$

de Groote [1994] - alternative syntax of $\lambda\mu$-calculus

$$t \; ::= \; x \mid \lambda x.t \mid t\,t \mid \mu\alpha.t \mid [\alpha]t \quad \text{(terms)}$$

David and Py [2001] - Parigot's $\lambda\mu$-calculus DOES NOT satisfy Böhm's separability
2 not equal normal forms with non-separable observational behaviour.

Saurin [2005] - de Groote's $\lambda\mu$-calculus SATISFIES Böhm's separability

$\lambda\mu$-calculus (Parigot)

Parigot [1992] - computational interpretation of classical natural deduction

$$\begin{array}{rcll} t & ::= & x \mid \lambda x.t \mid t\,t \mid \mu\alpha.c & \text{(terms)} \\ c & ::= & [\alpha]t & \text{(commands)} \end{array}$$

$$\boxed{\begin{array}{llcll} \beta : & (\lambda x.t)\,u & \to & t[u/x] & \\ \mu_{app} : & (\mu\alpha.c)\,u & \to & \mu\beta.c[[\beta](\Box\; u)/\alpha] & \beta \text{ fresh} \\ \mu_{var} : & [\beta]\mu\alpha.c & \to & c[\beta/\alpha] & \end{array}}$$

de Groote [1994] - alternative syntax of $\lambda\mu$-calculus

$$t \; ::= \; x \mid \lambda x.t \mid t\,t \mid \mu\alpha.t \mid [\alpha]t \quad \text{(terms)}$$

David and Py [2001] - Parigot's $\lambda\mu$-calculus DOES NOT satisfy Böhm's separability
2 not equal normal forms with non-separable observational behaviour.

Saurin [2005] - de Groote's $\lambda\mu$-calculus SATISFIES Böhm's separability.

$\Lambda\mu$-calculus (de Groote - Saurin)

CBN $\lambda\mu\widehat{tp}$ vs $\Lambda\mu$ - equational correspondence

$\Lambda\mu$ is derived from $\lambda\mu$ by relaxing the syntax and keeping the same theory.
$\Lambda\mu$ can be contrastingly restated as a strict extension of $\lambda\mu$.

This extension is precisely our call-by-name variant of $\lambda\mu\widehat{tp}$.

Equational correspondence $\Lambda\mu$ and CBN $\lambda\mu\widehat{tp}$

- $\Pi \;:\; \Lambda\mu \longrightarrow \lambda\mu\widehat{tp}$
- $\Sigma \;:\; \lambda\mu\widehat{tp} \longrightarrow \Lambda\mu$

$$\begin{array}{lcl} \Pi(\mu\alpha.M) & \triangleq & \mu\alpha.[\widehat{tp}]\Pi(M) \\ \Pi([\alpha]M) & \triangleq & \mu\widehat{tp}.[\alpha]\Pi(M) \end{array}$$

$$\begin{array}{lcl} \Sigma(\mu\alpha.[\widehat{tp}]M) & \triangleq & \mu\alpha.(\Sigma(M)) \\ \Sigma(\mu\widehat{tp}.[\alpha]M) & \triangleq & [\alpha]\Sigma(M) \\ \Sigma(\mu\widehat{tp}.[\widehat{tp}]M) & \triangleq & \Sigma(M) \end{array}$$

Observational completeness of call-by-name $\lambda\mu\widehat{tp}$.

Classification of the reduction semantics of $\lambda\mu\widehat{tp}$-calculus

(two calculi)

the fundamental critical pair of computation

$(\lambda x.t)\ (\mu\alpha.c)$

$\swarrow$ (CBV) (CBN) $\searrow$

CBV:

subsidiary choice

$(\lambda x.t)\ (\mu\widehat{tp}.c)$

($\widehat{\mu}$ value) $\swarrow$ $\searrow$ ($\widehat{\mu}$ not value)

`shift/reset`

(Danvy-Filinski)

cps-completion (Kameyama-Hasegawa)

typed "domain"-completion (Sitaram-Felleisen)

CBN:

subsidiary choice

$[\widehat{tp}]\mu\alpha.c$

($\widehat{tp}$ co-value) $\swarrow$ $\searrow$ ($\widehat{tp}$ not co-value)

$\Lambda\mu$

(de Groote/Saurin)

Böhm-completion (Saurin)

Classification of the reduction semantics of $\lambda\mu\widehat{tp}$-calculus

(two NEW calculi)

the fundamental critical pair of computation

$(\lambda x.t)\ (\mu\alpha.c)$

$\swarrow$ (CBV) (CBN) $\searrow$

CBV:

subsidiary choice

$(\lambda x.t)\ (\mu\widehat{tp}.c)$

- ($\widehat{\mu}$ value) $\swarrow$
  `shift/lazy reset`
  (Sabry)
  cps-completion (Sabry)
- $\searrow$ ($\widehat{\mu}$ not value)
  `shift/reset`
  (Danvy-Filinski)
  cps-completion (Kameyama-Hasegawa)
  typed "domain"-completion (Sitaram-Felleisen)

CBN:

subsidiary choice

$[\widehat{tp}]\mu\alpha.c$

- ($\widehat{tp}$ co-value) $\swarrow$
  CBN `shift/reset`
  (Danvy, Kyseliov)
  (see also: de Groote's ϵ)
- $\searrow$ ($\widehat{tp}$ not co-value)
  $\Lambda\mu$
  (de Groote/Saurin)
  Böhm-completion (Saurin)

Ongoing and future work

- A uniform approach to CBV and CBN delimited control (4 calculi)
  - Syntax and reduction rules
  - Equational theory
  - Simple typing
  - CPS semantics (SPS)
  - Equational correspondence with known calculi
  - Operational semantics
  - Expressiveness
- Interpretation from the duality of computation point of view $\overline{\lambda}\mu\tilde{\mu}\widehat{tp}$