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*

← have operators that capture and modify the flow of control

\[
#1 + 5 \times \text{Abort} 2 \rightarrow 2
\]

*Languages with delimited control*

\[
#1 + #5 \times \text{Abort} 2 \rightarrow 3
\]

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 $\lambda\mu\hat{t}p$-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 $\lambda\mu\hat{t}p$-calculus.

- A uniform presentation of four calculi of delimited control
Introduction and background
Frameworks to reason about call-by-value (non delimited) control

\( \lambda_c \) [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**

\[
E ::= \square \mid V \ E \mid E \ t
\]

**expected semantics**

\[
E[\text{callcc}(\lambda k. t)] \\
\downarrow \\
\downarrow
\]

\[
E[t[\lambda x. A(E[x])/k]] \\
\downarrow \quad \rightarrow \text{reified occurrence} \\
\text{not reified}
\]
The operational semantics of call-with-current-continuation (how to simulate it in $\lambda_C$?)

Abbreviations in $\lambda_C$

\[
callcc(\lambda k.t) \triangleq \mathcal{C}(\lambda k.k\ t)
\]
\[
A(t) \triangleq \mathcal{C}(\lambda_.t)
\]

Simulation

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

should not be reified as a function!
Consequences of the *reification* of continuations in $\lambda C$

- 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 \texttt{call-with-current-continuation}
\hspace{1em} (how to simulate it in $\lambda\mu$?)

\textit{abbreviations in $\lambda\mu$}

\begin{align*}
\text{callcc}(\lambda k.t) &\triangleq \mu \alpha. [\alpha](t[\lambda x. \mu_.[\alpha]x/k]) \\
\mathcal{A}(t) &\triangleq \mu_.[?].t
\end{align*}

$\mathcal{A}$ discards the current evaluation context and jumps to the toplevel:
we need a \texttt{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 $tp$*

$$\text{callcc}(\lambda k.t) \triangleq \mu\alpha.[\alpha](t[\lambda x.\mu_.[\alpha]x/k])$$

$$A(t) \triangleq \mu_.[tp].t$$

*simulation*

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

$$= E[\mu\alpha.[\alpha](t[\lambda x.\mu_.[\alpha]x/k])]$$

$$\downarrow$$

$$E[t[\lambda x.\mu_.[tp](E[x])/k]]$$

$$= E[t[\lambda x.A(E[x])/k]]$$
A foundation for control: $\lambda\mu tp$-calculus

\begin{align*}
V & ::= x \mid \lambda x.t & \text{(values)} \\
\tau, \upsilon & ::= V \mid \tau \upsilon \mid \mu \alpha. c & \text{(terms)} \\
c & ::= [\beta]t \mid [\tau]t & \text{(commands or states)}
\end{align*}

$\lambda\mu tp$-calculus satisfies:

- **Faithful** simulation of call-with-current-continuation, $A$, $C$, ...
- **Observationally** equivalent to $\lambda C$ (but not operationally equivalent)
- Confluence, termination in simply-typed case, standardisation, ...
- Internal notion of state: evaluations are of the unique form $\tau p V$
How to make \( \lambda \mu tp \) suitable for delimited control?
A foundation for delimited control: $\lambda\mu\hat{p}$-calculus

Let’s turn $tp$ into a dynamically bound variable $\hat{tp}$:

$$
V ::= x | \lambda x.t \quad \text{(values)} \\
t, u ::= V | t u | \mu \alpha.c | \mu\hat{tp}.c \quad \text{(terms)} \\
c ::= [\beta]t | [\hat{tp}]t \quad \text{(commands or states)}
$$

The new operator $\mu\hat{tp}.c$ 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 $\lambda \mu \hat{t}p$-calculus

- **Danvy and Filinski’s delimited control**
  - **reset** $t$ \[\equiv \mu t\hat{p}[] t\]
  - **shift** \[\equiv \lambda y.\mu \alpha.[] (y \lambda x.\mu t\hat{p}[] x)\]

- **exception handling**
  - **raise** $t$ \[\equiv \mu_.[] t\]
  - **$t$ handle patterns** \[\equiv \text{case } \mu t\hat{p}[] (\text{Val } t) \text{ of}
    \begin{align*}
      &| \text{Val } x \Rightarrow x \\
      &| \text{patterns} \\
      &| x \Rightarrow \mu_.[] x
    \end{align*}\]

- **monads in direct style**
  - **$\mu(t)$** \[\equiv \mu \alpha.[] ((\lambda x.\mu t\hat{p}[] x)^* t)\]
  - **$[t]$$\equiv \mu t\hat{p}[] (\eta t)\]

- **mutable reference**
  - **read** \[\equiv \lambda().\mu \alpha.[] \lambda s.((\mu t\hat{p}[] s) s)\]
  - **write** \[\equiv \lambda s.\mu \alpha.[] \lambda_.((\mu t\hat{p}[] s) s)\]
Outline of the talk

I- Introduction and background

- A review: Felleisen’s $\lambda_C$, Parigot’s $\lambda\mu$, ...
- A foundation for control: $\lambda\mu tp$-calculus (tp is the toplevel continuation constant)
- A foundation for delimited control: $\lambda\mu\hat{tp}$-calculus ($\hat{tp}$ is a dynamically scoped variable)
- CBV $\lambda\mu\hat{tp}$-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 $\lambda\mu\hat{tp}$-calculus.

- A uniform presentation of four calculi of delimited control
Towards our results
Call-by-name $\lambda\mu\hat{t}p$-calculus?

CBV

\[
V ::= x \mid \lambda x.t \quad \text{(values)}
\]
\[
t, u ::= V \mid t u \mid \mu\alpha.c \mid \mu\hat{t}p.c \quad \text{(terms)}
\]
\[
c ::= [\beta]t \mid [\hat{t}p]t \quad \text{(commands or states)}
\]

\[
\beta_v : \ (\lambda x.t) V \to t[V/x]
\]
\[
\mu_{app} : \ (\mu\alpha.c) t \to \mu\beta.c[[\beta](\Box t)/\alpha] \quad \beta \text{ fresh}
\]
\[
\mu_v^{app} : \ V (\mu\alpha.c) \to \mu\beta.c[[\beta](V \Box)/\alpha] \quad \beta \text{ fresh}
\]
\[
\mu_{var} : \ [\beta]\mu\alpha.c \to c[\beta/\alpha]
\]
\[
\mu_{var}^{tip} : \ [\hat{t}p]\mu\alpha.c \to c[\hat{t}p/\alpha]
\]
\[
\eta_{ip}^{v} : \mu\hat{t}p.[\hat{t}p] V \to V \quad \text{even if } \hat{t}p \text{ occurs in } V
\]
Call-by-name $\lambda\mu\hat{t}p$-calculus?

CBV

\[
\begin{align*}
\text{not} & \quad V ::= x \mid \lambda x. t \quad \text{(values)} \\
\text{mod} & \quad t, u ::= V \mid t \cdot u \mid \mu \alpha. c \mid \mu \hat{t}p. c \quad \text{(terms)} \\
\quad c ::= [\beta]t \mid [\hat{t}p]t \quad \text{(commands or states)}
\end{align*}
\]

\[
\begin{align*}
\text{mod} & \quad \beta_v : \quad (\lambda x. t) V \to t[V/x] \\
\quad \mu_{app} : \quad (\mu \alpha. c) t \to \mu \beta. c[[\beta](\square t)/\alpha] \quad \beta \text{ fresh} \\
\text{not} & \quad \mu_{app}^v : \quad V (\mu \alpha. c) \to \mu \beta. c[[\beta](V \square)/\alpha] \quad \beta \text{ fresh} \\
\quad \mu_{var} : \quad [\beta] \mu \alpha. c \to c[\beta/\alpha] \\
\text{not} & \quad \mu_{var}^v : \quad [\hat{t}p] \mu \alpha. c \to c[\hat{t}p/\alpha] \\
\text{mod} & \quad \eta_{tp}^v : \quad \mu \hat{t}p. [\hat{t}p] V \to V \quad \text{even if } \hat{t}p \text{ occurs in } V
\end{align*}
\]

CBN

\[
\begin{align*}
\quad t, u ::= x \mid \lambda x. t \mid t \cdot u \mid \mu \alpha. c \mid \mu \hat{t}p. c \quad \text{(terms)} \\
\quad c ::= [\beta]t \mid [\hat{t}p]t \quad \text{(commands or states)}
\end{align*}
\]

\[
\begin{align*}
\quad \beta : \quad (\lambda x. t) u \to t[u/x] \\
\quad \mu_{app} : \quad (\mu \alpha. c) t \to \mu \beta. c[[\beta](\square t)/\alpha] \quad \beta \text{ fresh} \\
\quad \mu_{var} : \quad [\beta] \mu \alpha. c \to c[\beta/\alpha] \\
\quad \eta_{tp} : \quad \mu \hat{t}p. [\hat{t}p] t \to t \quad \text{even if } \hat{t}p \text{ occurs in } t
\end{align*}
\]
**λµ-calculus**

Parigot [1992] - computational interpretation of classical natural deduction

\[
\begin{align*}
t & ::= x \mid \lambda x.t \mid tt \mid \mu\alpha.c \quad \text{(terms)} \\
c & ::= [\alpha]t \quad \text{(commands)}
\end{align*}
\]

\[\beta : (\lambda x.t) u \rightarrow t[u/x]\]

\[\mu_{\text{app}} : (\mu\alpha.c) u \rightarrow \mu\beta.c[[\beta](\Box u)/\alpha] \quad \beta \text{ fresh}\]

\[\mu_{\text{var}} : [\beta]\mu\alpha.c \rightarrow c[\beta/\alpha]\]

de Groote [1994] - alternative syntax of λµ-calculus

\[
\begin{align*}
t & ::= x \mid \lambda x.t \mid tt \mid \mu\alpha.t \mid [\alpha]t \quad \text{(terms)}
\end{align*}
\]

David and Py [2001] - Parigot’s λµ-calculus DOES NOT satisfy Böhm’s separability

2 not equal normal forms with non-separable observational behaviour.

Saurin [2005] - de Groote’s λµ-calculus SATISFFIES Böhm’s separability
\[\lambda\mu\text{-calculus (Parigot)}\]

Parigot [1992] - computational interpretation of classical natural deduction

\[
t ::= \ x \mid \lambda x.t \mid tt \mid \mu\alpha.c \quad \text{(terms)}
\]
\[
c ::= [\alpha]t \quad \text{(commands)}
\]

\[
\beta : (\lambda x.t)u \rightarrow t[u/x]
\]
\[
\mu_{app} : (\mu\alpha.c)u \rightarrow \mu\beta.c[[\beta](\square u)/\alpha] \quad \beta \text{ fresh}
\]
\[
\mu_{var} : [\beta]\mu\alpha.c \rightarrow c[\beta/\alpha]
\]

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

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

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.

\[\Lambda\mu\text{-calculus (de Groote - Saurin)}\]
CBN $\lambda\mu\hat{t}p$ 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 $\lambda\mu\hat{t}p$.

Equational correspondence $\Lambda\mu$ and CBN $\lambda\mu\hat{t}p$

- $\Pi : \Lambda\mu \rightarrow \lambda\mu\hat{t}p$
- $\Sigma : \lambda\mu\hat{t}p \rightarrow \Lambda\mu$

\[
\Pi(\mu\alpha.M) \triangleq \mu\alpha.[\hat{t}p]\Pi(M) \\
\Pi([\alpha]M) \triangleq \mu\hat{t}p.[\alpha]\Pi(M)
\]

\[
\Sigma(\mu\alpha.[\hat{t}p]M) \triangleq \mu\alpha.(\Sigma(M)) \\
\Sigma(\mu\hat{t}p.[\alpha]M) \triangleq [\alpha]\Sigma(M) \\
\Sigma(\mu\hat{t}p.[\hat{t}p]M) \triangleq \Sigma(M)
\]

Observational completeness of call-by-name $\lambda\mu\hat{t}p$. 
Classification of the reduction semantics of $\lambda\mu\hat{\mu}$-calculus
(two calculi)

the fundamental critical pair of computation
$(\lambda x.t) (\mu\alpha.c)$

✓ (CBV)  (CBN) ✓

subsidary choice
$(\lambda x.t) (\mu\hat{\mu}.c)$

(μ value) ✓  (μ not value)
shift/reset
(Danvy-Filinski)
cps-completion (Kameyama-Hasegawa)
typed “domain”-completion (Sitaram-Felleisen)

subsidary choice
$[\hat{\mu}][\mu\alpha.c]$  

(μ not co-value) ✓
Λμ
(de Groote/Saurin)
Böhm-completion (Saurin)
Classification of the reduction semantics of $\lambda\mu\hat{t}p$-calculus
(two NEW calculi)

*the fundamental critical pair of computation*

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

- $\checkmark$ (CBV)
- $\rightarrow$ (CBN)

<table>
<thead>
<tr>
<th>Subsidiary choice</th>
<th>Subsidiary choice</th>
</tr>
</thead>
<tbody>
<tr>
<td>$$(\hat{\mu} \text{ value}) \checkmark$$</td>
<td>$$(\hat{t}p \text{ co-value}) \checkmark$$</td>
</tr>
<tr>
<td>Shift/lazy reset</td>
<td>CBN shift/reset</td>
</tr>
<tr>
<td>(Sabry)</td>
<td>(Danvy-Filinski)</td>
</tr>
<tr>
<td>Cps-completion (Sabry)</td>
<td>Cps-completion (Kameyama-Hasegawa)</td>
</tr>
<tr>
<td>Typed “domain”-completion (Sitaram-Felleisen)</td>
<td>(see also: de Groote’s $\epsilon$)</td>
</tr>
<tr>
<td>(Sabry)</td>
<td></td>
</tr>
</tbody>
</table>

- $$(\hat{\mu} \text{ not value})$$
- $$(\hat{t}p \text{ not co-value})$$
- $\Lambda\mu$
- (Danvy, Kyseliov)
- (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 $\bar{\lambda}_\mu \bar{\mu} \breve{\nu} \bar{p}$