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

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

Languages with delimited control

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

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

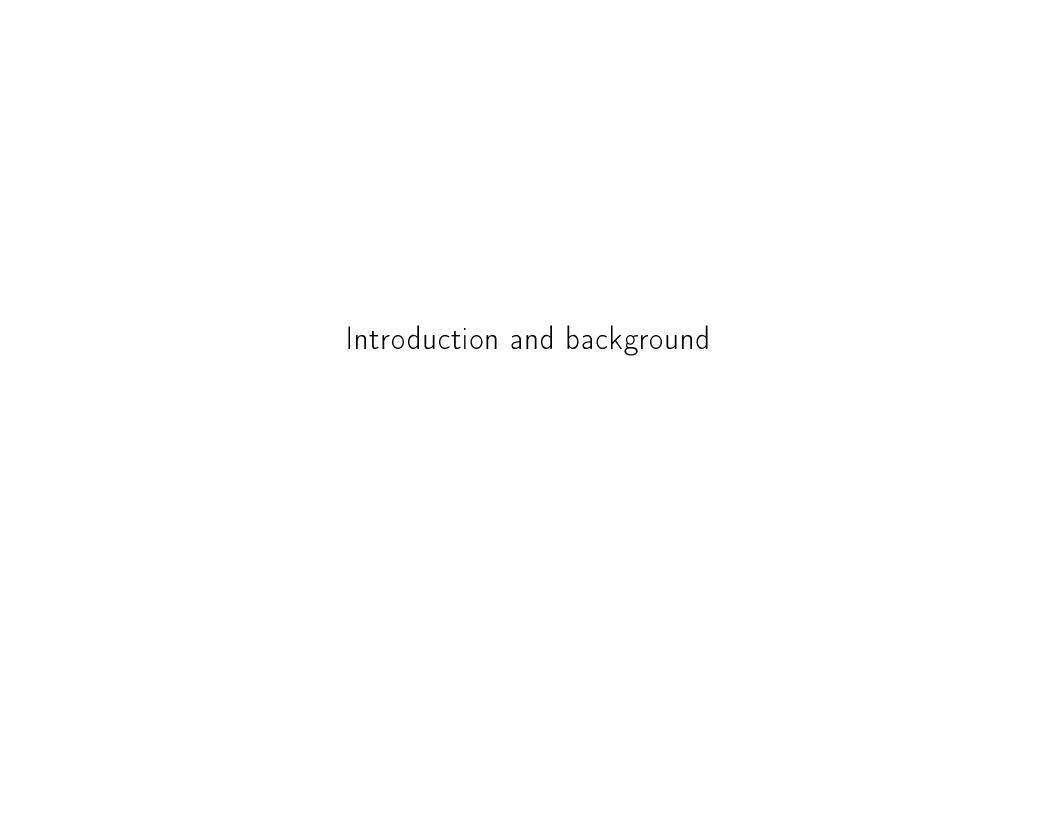
- Drawbacks of previous calculi of control
- A better foundation for control: $\lambda \mu$ tp-calculus
- A foundation for call-by-value delimited control: call-by-value $\lambda \mu \widehat{tp}$ -calculus

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



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

λ_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[\operatorname{callcc}(\lambda k.t)] \\ \downarrow \\ E[t[\lambda x.\mathcal{A}(E[x])/k]] \\ \downarrow \qquad \hookrightarrow \text{reified occurrence} \\ \text{not reified}
```

The operational semantics of call-with-current-continuation (how to simulate it in $\lambda_{\mathcal{C}}$?)

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

$$\begin{array}{ccc} \mathtt{callcc}(\lambda k.t) & \triangleq & \mathcal{C}(\lambda k.k \ t) \\ \mathcal{A}(t) & \triangleq & \mathcal{C}(\lambda _.t) \\ \end{array}$$

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 $\lambda\mu$

$$\begin{array}{lll} \mathtt{callcc}(\lambda k.t) & \triangleq & \mu \alpha.[\alpha](t[\lambda x.\mu_.[\alpha]x/k]) \\ \mathcal{A}(t) & \triangleq & \mu_.[?].t \end{array}$$

 ${\cal A}$ 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

$$\begin{array}{lll} \mathtt{callcc}(\lambda k.t) & \triangleq & \mu \alpha.[\alpha](t[\lambda x.\mu_.[\alpha]x/k]) \\ \mathcal{A}(t) & \triangleq & \mu_.[\mathtt{tp}].t \end{array}$$

simulation

$$E[\operatorname{callcc}(\lambda k.t)] \\ = \\ E[\mu \alpha.[\alpha](t[\lambda x.\mu_.[\alpha]x/k])] \\ \downarrow \\ E[t[\lambda x.\mu_.[\operatorname{tp}](E[x])/k]] \\ = \\ E[t[\lambda x.\mathcal{A}(E[x])/k]]$$

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

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

$\lambda\mu$ tp-calculus satisfies:

- Faithful simulation of call-with-current-continuation, \mathcal{A} , \mathcal{C} , ...
- Observationally equivalent to λC (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: $\lambda \mu \hat{tp}$ -calculus

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

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

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 $\lambda \mu \hat{tp}$ -calculus

```
 \triangleq \mu \widehat{tp}.[\widehat{tp}] t 
 \triangleq \lambda y.\mu \alpha.[\widehat{tp}] (y \lambda x.\mu \widehat{tp}.[\alpha]x) 
Danvy and Filinski's \
                                                                        {	t reset} \; t
delimited control
                                                                         shift
                                                                        \begin{array}{ll} \mathbf{raise} \; t & \triangleq & \mu\_.[\widehat{tp}] \, t \\ t \; \mathbf{handle} \; patterns & \triangleq & \mathsf{case} \; \mu \widehat{tp}.[\widehat{tp}] \, (\mathsf{Val} \; t) \; \mathsf{of} \end{array}
exception }
handling }
                                                                                                                                                        \begin{array}{c|c} | & \text{Val } x \Rightarrow x \\ | & patterns \end{array}
                                                                                                                                                          | x \Rightarrow \mu . [\widehat{tp}] x
                                                                                                                                        \triangleq \mu\alpha.[\widehat{tp}]((\lambda x.\mu\widehat{tp}.[\alpha]x)^*t)
monads in direct style
                                                                        \mu(t)
[t]
                                                                                                                                        \triangleq \quad \mu \widehat{tp}.[\widehat{tp}]\left(\eta \ t\right)
                                                                                                                                         \triangleq \lambda().\mu\alpha.[\widehat{tp}] \lambda s.((\mu\widehat{tp}.[\alpha]s) s) 
 \triangleq \lambda s.\mu\alpha.[\widehat{tp}] \lambda_{-}.((\mu\widehat{tp}.[\alpha]()) s) 
mutable reference
                                                                         read
                                                                         write
```

Outline of the talk

I- Introduction and background

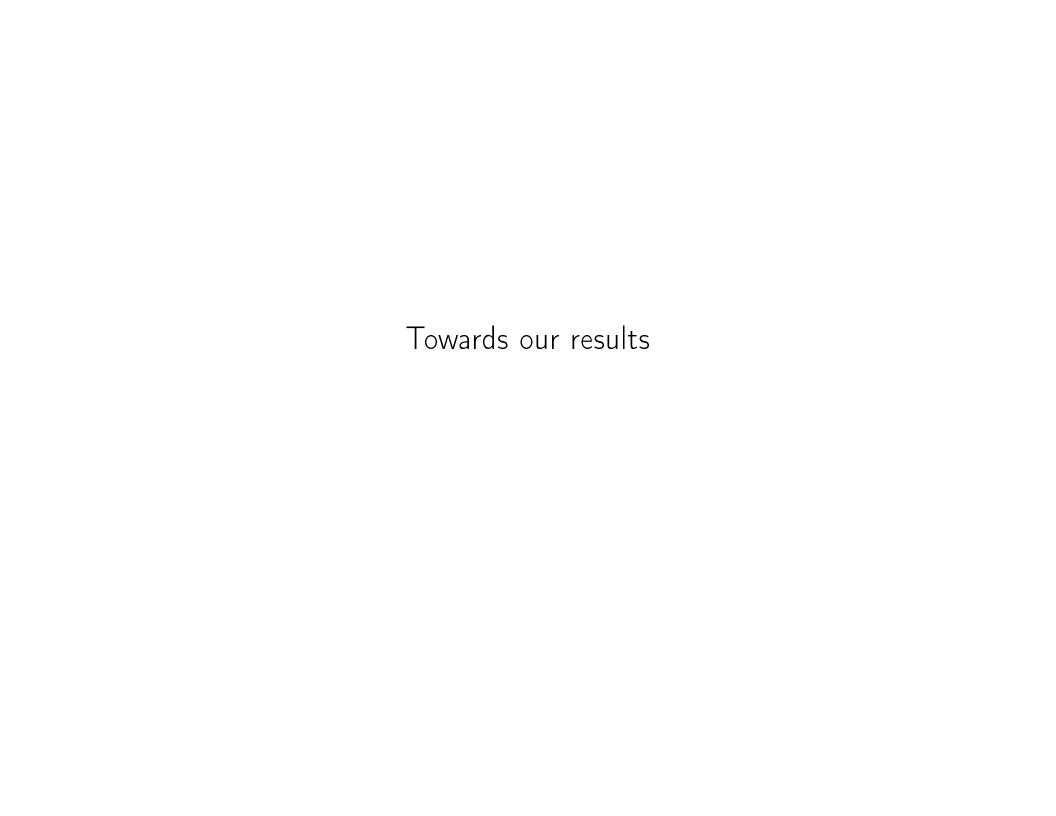
- A review: Felleisen's $\lambda_{\mathcal{C}}$, Parigot's $\lambda \mu$, . . .
- A foundation for control: $\lambda \mu$ tp-calculus (tp is the toplevel continuation constant)
- A foundation for delimited control: $\lambda \mu \hat{tp}$ -calculus (\hat{tp} is a dynamically scoped variable)
- CBV $\lambda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

- A uniform presentation of four calculi of delimited control



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

CBV

```
\begin{array}{lll} 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)} \\ \\ \beta_v : & (\lambda x.t) \, V & \rightarrow & t[V/x] \\ \mu_{app} : & (\mu \alpha.c) \, t & \rightarrow & \mu \beta.c[[\beta](\square \, t)/\alpha] \quad \beta \text{ fresh} \\ \mu^v_{app} : & V (\mu \alpha.c) & \rightarrow & \mu \beta.c[[\beta](V \, \square)/\alpha] \quad \beta \text{ fresh} \\ \mu_{var} : & [\beta]\mu \alpha.c & \rightarrow & c[\beta/\alpha] \\ \mu^{\widehat{tp}}_{var} : & [\widehat{tp}]\mu \alpha.c & \rightarrow & c[\widehat{tp}/\alpha] \\ \eta^v_{\widehat{tp}} : & \mu \widehat{tp}.[\widehat{tp}] \, V & \rightarrow & \text{even if } \widehat{tp} \text{ occurs in } V \end{array}
```

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

CBV

CBN

```
c ::= [\beta]t \mid [\widehat{tp}]t (commands or states)
 \operatorname{mod} \beta_v : (\lambda x. t) V \rightarrow t[V/x]
           \mu_{app}: \quad (\mu\alpha.\,c)\;t \quad \to \quad \mu\beta.c[[\beta](\Box\;t)/\alpha] \quad \beta \; {\rm fresh}
not \mu^v_{app}: V(\mu\alpha.c) \rightarrow \mu\beta.c[[\beta](V\square)/\alpha] \quad \beta fresh
          \mu_{var}: [\beta]\mu\alpha.c \rightarrow c[\beta/\alpha]
\begin{array}{lll} & \text{not} & \mu_{\widehat{var}}^{\widehat{\mathfrak{p}}} : & [\widehat{tp}]\mu\alpha.c & \rightarrow & c[\widehat{tp}/\alpha] \\ & \text{mod} & \eta_{\widehat{\mathfrak{p}}}^v : & \mu\widehat{tp}.[\widehat{tp}] \ V & \rightarrow & V & \text{even if } \widehat{tp} \text{ occurs in } V \end{array}
         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)
        \beta: (\lambda x. t) u \rightarrow t[u/x]
         \mu_{app}: (\mu\alpha.c) t \rightarrow \mu\beta.c[[\beta](\Box t)/\alpha] \quad \beta \text{ fresh}
        \mu_{var}: [\beta]\mu\alpha.c \rightarrow c[\beta/\alpha]
         \eta_{\widehat{ip}}: \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

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

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

$$t ::= x \mid \lambda x.t \mid tt \mid \mu \alpha.t \mid [\alpha]t$$
 (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

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

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

$$t ::= x \mid \lambda x.t \mid tt \mid \mu \alpha.t \mid [\alpha]t$$
 (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 $\lambda \mu \hat{tp}$.

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

•
$$\Pi$$
 : $\Lambda\mu \longrightarrow \lambda\mu \widehat{tp}$

 $\bullet \ \Sigma \ : \ \lambda \mu \widehat{tp} \ \longrightarrow \ \Lambda \mu$

$$\Pi(\mu\alpha.M) \triangleq \mu\alpha.[\widehat{tp}]\Pi(M)$$

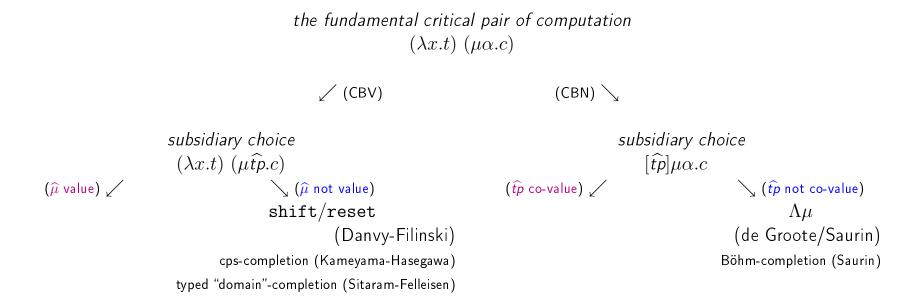
$$\Pi([\alpha]M) \triangleq \mu\widehat{tp}.[\alpha]\Pi(M)$$

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

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

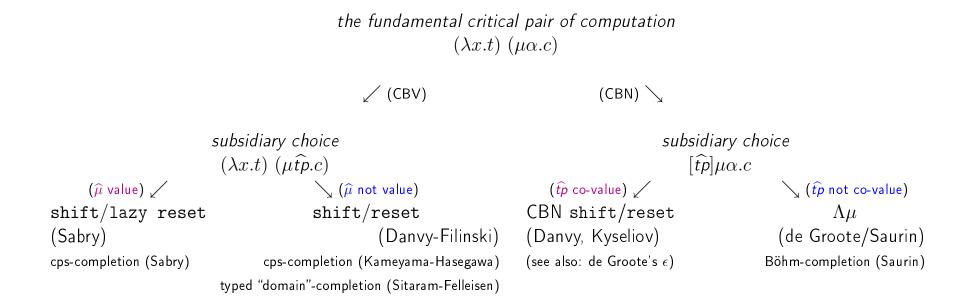
Classification of the reduction semantics of $\lambda \mu \hat{tp}$ -calculus

(two calculi)



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

(two NEW calculi)



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
- ullet Interpretation from the duality of computation point of view $\overline{\lambda}\mu ilde{\mu}\hat{t}p$