Closed Reduction and Director Strings

François-Régis Sinot[1] Maribel Fernández[2] Ian Mackie[2]

[1]LIX, École Polytechnique

[2]DCS, King's College London

Journées Coq+Réécriture, 2004

Introduction

- efficient computation of β-reduction in Coq ?

Introduction

- efficient computation of β-reduction in Coq ?
- possible answers:

Introduction

- efficient computation of β-reduction in Coq ?
- possible answers:
  - compilation

Introduction

- efficient computation of β-reduction in Coq ?
- possible answers:
  - compilation
  - optimisations

Introduction

- efficient computation of β-reduction in Coq ?
- possible answers:
  - compilation
  - optimisations
  - ...

Introduction

- efficient computation of β-reduction in Coq ?
- possible answers:
  - compilation
  - optimisations
  - ...
- *Don't put the cart before the horse...*

Introduction

- efficient computation of β-reduction in Coq ?
- possible answers:
  - compilation
  - optimisations
  - ...
- *Don't put the cart before the horse...*
- In Coq, terms are strongly normalisable, so why should we abide to the *"do not ever reduce under lambdas"* credo ?

Introduction

- efficient computation of β-reduction in Coq ?
- possible answers:
  - compilation
  - optimisations
  - ...
- *Don't put the cart before the horse...*
- In Coq, terms are strongly normalisable, so why should we abide to the *"do not ever reduce under lambdas"* credo ?
- Well, because this is our playground (*eg.* reuse compiler).

Introduction

- efficient computation of β-reduction in Coq ?
- possible answers:
  - compilation
  - optimisations
  - ...
- *Don't put the cart before the horse...*
- In Coq, terms are strongly normalisable, so why should we abide to the *"do not ever reduce under lambdas"* credo ?
- Well, because this is our playground (*eg.* reuse compiler).
- Let's instead take a fresh start and think first about an efficient **strategy**

Some intuitions

- Why reduce under lambdas ?

Some intuitions

- Why reduce under lambdas ?
- $(\lambda x.x\,x)(\lambda y.\underline{(\lambda z.z)\,y})$

Some intuitions

- Why reduce under lambdas ?
- $(\lambda x.x\,x)(\lambda y.\underline{(\lambda z.z)\,y})$

$\Rightarrow$ the underlined redex should be reduced first, whereas standard strategies **will duplicate** it

Some intuitions

- Why reduce under lambdas ?
- $(\lambda x.x\,x)(\lambda y.\underline{(\lambda z.z)\,y})$
- the underlined redex should be reduced first, whereas standard strategies **will duplicate** it
- Is it always better to reduce under lambdas ?

Some intuitions

- Why reduce under lambdas ?
- $(\lambda x.x\,x)(\lambda y.\underline{(\lambda z.z)\,y})$
- the underlined redex should be reduced first, whereas standard strategies **will duplicate** it
- Is it always better to reduce under lambdas ?
- $(\lambda x.x\,(\lambda z.z))(\lambda f.\underline{(\lambda y.y\,y)(f\,(\lambda z.z))})$

Some intuitions

- Why reduce under lambdas ?
- $(\lambda x.x\,x)(\lambda y.\underline{(\lambda z.z)\,y})$
- the underlined redex should be reduced first, whereas standard strategies **will duplicate** it
- Is it always better to reduce under lambdas ?
- $(\lambda x.x\,(\lambda z.z))(\lambda f.\underline{(\lambda y.y\,y)(f\,(\lambda z.z))})$

$\Rightarrow$ now it is better **not** to reduce the inner redex, because f will be duplicated and it is a **potential redex**

Some intuitions

- Why reduce under lambdas ?
- $(\lambda x.x\,x)(\lambda y.\underline{(\lambda z.z)\,y})$
- the underlined redex should be reduced first, whereas standard strategies **will duplicate** it
- Is it always better to reduce under lambdas ?
- $(\lambda x.x\,(\lambda z.z))(\lambda f.\underline{(\lambda y.y\,y)(f\,(\lambda z.z))})$
- now it is better **not** to reduce the inner redex, because f will be duplicated and it is a **potential redex**
- **Morale**: reduce under abstractions, but avoid as much as possible to duplicate open terms (*i.e.* terms with free variables)

Some intuitions

- Why reduce under lambdas ?
- $(\lambda x.x\,x)(\lambda y.\underline{(\lambda z.z)\,y})$
- the underlined redex should be reduced first, whereas standard strategies **will duplicate** it
- Is it always better to reduce under lambdas ?
- $(\lambda x.x\,(\lambda z.z))(\lambda f.\underline{(\lambda y.y\,y)(f\,(\lambda z.z))})$
- now it is better **not** to reduce the inner redex, because f will be duplicated and it is a **potential redex**
- **Morale**: reduce under abstractions, but avoid as much as possible to duplicate open terms (*i.e.* terms with free variables)
- Hence the name of **closed** reduction

The framework

- We want a framework:

The framework

- We want a framework:
  - simple

The framework

- We want a framework:
  - simple

    $\Rightarrow$ with names

The framework

- We want a framework:
  - simple
    - with names
  - with a realistic notion of cost

The framework

- We want a framework:
  - simple
    - with names
  - with a realistic notion of cost

    $\Rightarrow$ explicit substitutions

The framework

- We want a framework:
  - simple
    - with names
  - with a realistic notion of cost
    - explicit substitutions

    $\Rightarrow$ without implicit α-conversion

The framework

- We want a framework:
  - simple
    - with names
  - with a realistic notion of cost
    - explicit substitutions
    - without implicit α-conversion
- We want to talk freely about *strategies*, hence:

The framework

- We want a framework:
  - simple
    - with names
  - with a realistic notion of cost
    - explicit substitutions
    - without implicit α-conversion
- We want to talk freely about *strategies*, hence:
  - (ground) confluence and PSN are useful

The framework

- We want a framework:
  - simple
    - with names
  - with a realistic notion of cost
    - explicit substitutions
    - without implicit α-conversion
- We want to talk freely about *strategies*, hence:
  - (ground) confluence and PSN are useful
  - full simulation of β is *not* required

First step: the syntax

- λ-calculus

$$t, u \ ::= x | \lambda x.t | t\ u$$

First step: the syntax

- λ-calculus
- with explicit substitutions

$$t, u \ ::= x | \lambda x.t | t\ u \\ | t[x/u]$$

First step: the syntax

- λ-calculus
- with explicit substitutions
- and *"explicit resource management"* (inspired by linear logic)

$$
\begin{aligned}
t, u \ ::= \ & x \mid \lambda x.t \mid t\ u \\
& \mid t[x/u] \\
& \mid \epsilon_x.t \\
& \mid \delta_x^{y,z}.t
\end{aligned}
$$

and we impose linearity

Closed reduction

Name		Reduction	Condition
Beta	$(\lambda x.t)u$	$\to t[x/u]$	
Var_1	$x[x/v]$	$\to v$	
Var_2	$y[x/v]$	$\to y$	$x \neq y$
App	$(t\,u)[x/v]$	$\to (t[x/v])(u[x/v])$	
Lam_1	$(\lambda x.t)[x/v]$	$\to (\lambda x.t)$	
Lam_2	$(\lambda y.t)[x/v]$	$\to \lambda y.t[x/v]$	$x \notin \mathsf{fv}(t) \vee y \notin \mathsf{fv}(v)$
Lam_3	$(\lambda y.t)[x/v]$	$\to \lambda z.t[z/y][x/v]$	$x \in \mathsf{fv}(t), y \in \mathsf{fv}(v), z$ fresh
Comp	$t[y/w][x/v]$	$\to t[x/v][y/w[x/v]]$	$y \notin \mathsf{fv}(v)$

Closed reduction

Name		Reduction	Condition
Beta	$(\lambda x.t)u$	$\longrightarrow t[x/u]$	
Var_1	$x[x/v]$	$\longrightarrow v$	
Var_2	$y[x/v]$	$\longrightarrow y$	$x \neq y$
App	$(t\,u)[x/v]$	$\longrightarrow (t[x/v])(u[x/v])$	
Lam_1	$(\lambda x.t)[x/v]$	$\longrightarrow (\lambda x.t)$	
Lam_2	$(\lambda y.t)[x/v]$	$\longrightarrow \lambda y.t[x/v]$	$x \notin \mathrm{fv}(t) \vee y \notin \mathrm{fv}(v)$
Lam_3	$(\lambda y.t)[x/v]$	$\longrightarrow \lambda z.t[z/y][x/v]$	$x \in \mathrm{fv}(t), y \in \mathrm{fv}(v), z$ fresh
Comp	$t[y/w][x/v]$	$\longrightarrow t[x/v][y/w[x/v]]$	$y \notin \mathrm{fv}(v)$

Closed reduction

Name	Reduction	Condition
Beta	$(\lambda x.t)u \longrightarrow t[x/u]$	
Var	$x[x/v] \longrightarrow v$	
App	$(t\,u)[x/v] \longrightarrow (t[x/v])(u[x/v])$	
Lam	$(\lambda y.t)[x/v] \longrightarrow \lambda y.t[x/v]$	$\mathrm{fv}(v) = \emptyset$
Comp	$t[y/w][x/v] \longrightarrow t[x/v][y/w[x/v]]$	

Closed reduction

Name	Reduction	Condition
Beta	$(\lambda x.t)u \longrightarrow t[x/u]$	
Var	$x[x/v] \longrightarrow v$	
App	$(t\ u)[x/v] \longrightarrow (t[x/v])(u[x/v])$	
Lam	$(\lambda y.t)[x/v] \longrightarrow \lambda y.t[x/v]$	$\mathrm{fv}(v) = \emptyset$
Comp	$t[y/w][x/v] \longrightarrow t[x/v][y/w[x/v]]$	

Closed reduction

Name	Reduction	Condition
Beta	$(\lambda x.t)u \longrightarrow t[x/u]$	
Var	$x[x/v] \longrightarrow v$	
App_1	$(t\,u)[x/v] \longrightarrow (t[x/v])\,u$	$x \in \mathrm{fv}(t)$
App_2	$(t\,u)[x/v] \longrightarrow t\,(u[x/v])$	$x \in \mathrm{fv}(u)$
Lam	$(\lambda y.t)[x/v] \longrightarrow \lambda y.t[x/v]$	$\mathrm{fv}(v) = \emptyset$
Comp	$t[y/w][x/v] \longrightarrow t[y/w[x/v]]$	$x \in \mathrm{fv}(w)$

Closed reduction

Name	Reduction	Condition
Beta	$(\lambda x.t)u \longrightarrow t[x/u]$	
Var	$x[x/v] \longrightarrow v$	
App_1	$(t\,u)[x/v] \longrightarrow (t[x/v])\,u$	$x \in \mathrm{fv}(t)$
App_2	$(t\,u)[x/v] \longrightarrow t\,(u[x/v])$	$x \in \mathrm{fv}(u)$
Lam	$(\lambda y.t)[x/v] \longrightarrow \lambda y.t[x/v]$	$\mathrm{fv}(v) = \emptyset$
Comp	$t[y/w][x/v] \longrightarrow t[y/w[x/v]]$	$x \in \mathrm{fv}(w)$
Copy_1	$(\delta_x^{y,z}.t)[x/v] \longrightarrow t[y/v][z/v]$	$\mathrm{fv}(v) = \emptyset$
Copy_2	$(\delta_{x'}^{y;z}.t)[x/v] \longrightarrow \delta_{x'}^{y;z}.t[x/v]$	
Erase_1	$(\epsilon_x.t)[x/v] \longrightarrow t$	$\mathrm{fv}(v) = \emptyset$
Erase_2	$(\epsilon_{x'}.t)[x/v] \longrightarrow \epsilon_{x'}.t[x/v]$	

Closed reduction

Name	Reduction	Condition
Beta	$(\lambda x.t)u \longrightarrow t[x/u]$	$\mathrm{fv}(\lambda x.t) = \emptyset$
Var	$x[x/v] \longrightarrow v$	
App_1	$(t\,u)[x/v] \longrightarrow (t[x/v])\,u$	$x \in \mathrm{fv}(t)$
App_2	$(t\,u)[x/v] \longrightarrow t\,(u[x/v])$	$x \in \mathrm{fv}(u)$
Lam	$(\lambda y.t)[x/v] \longrightarrow \lambda y.t[x/v]$	$\mathrm{fv}(v) = \emptyset$
Comp	$t[y/w][x/v] \longrightarrow t[y/w[x/v]]$	$x \in \mathrm{fv}(w)$
Copy_1	$(\delta_x^{y,z}.t)[x/v] \longrightarrow t[y/v][z/v]$	$\mathrm{fv}(v) = \emptyset$
Copy_2	$(\delta_{x'}^{y,z}.t)[x/v] \longrightarrow \delta_{x'}^{y,z}.t[x/v]$	
Erase_1	$(\epsilon_x.t)[x/v] \longrightarrow t$	$\mathrm{fv}(v) = \emptyset$
Erase_2	$(\epsilon_{x'}.t)[x/v] \longrightarrow \epsilon_{x'}.t[x/v]$	

Closed reduction

Name	Reduction		Condition
Beta	$(\lambda x.t)u$	$\to t[x/u]$	$\mathrm{fv}(\lambda x.t) = \emptyset$
Var	$x[x/v]$	$\to v$	
App_1	$(t\,u)[x/v]$	$\to (t[x/v])\,u$	$x \in \mathrm{fv}(t)$
App_2	$(t\,u)[x/v]$	$\to t\,(u[x/v])$	$x \in \mathrm{fv}(u)$
Lam	$(\lambda y.t)[x/v]$	$\to \lambda y.t[x/v]$	$\mathrm{fv}(v) = \emptyset$
Comp	$t[y/w][x/v]$	$\to t[y/w[x/v]]$	$x \in \mathrm{fv}(w)$
Copy_1	$(\delta_x^{y,z}.t)[x/v]$	$\to t[y/v][z/v]$	$\mathrm{fv}(v) = \emptyset$
Copy_2	$(\delta_{x'}^{y,z}.t)[x/v]$	$\to \delta_{x'}^{y,z}.t[x/v]$	
Erase_1	$(\epsilon_x.t)[x/v]$	$\to t$	$\mathrm{fv}(v) = \emptyset$
Erase_2	$(\epsilon_{x'}.t)[x/v]$	$\to \epsilon_{x'}.t[x/v]$	

Zooming back

- Comp: two rules

Zooming back

- Comp: two rules
  - $t[y/w][x/v] \to t[y/w[x/v]]$ $(x \in \mathrm{fv}(w))$

Zooming back

- Comp: two rules
  - $t[y/w][x/v] \to t[y/w[x/v]]$ $(x \in \mathrm{fv}(w))$

  $\Rightarrow$ good: more sharing and does not endanger PSN
  (thanks to explicit erasing — *cf.* David & Guillaume)

Zooming back

- Comp: two rules
  - $t[y/w][x/v] \to t[y/w[x/v]]$ $(x \in \mathrm{fv}(w))$
  - good: more sharing and does not endanger PSN (thanks to explicit erasing — *cf.* David & Guillaume)
  - $t[y/w][x/v] \to t[x/v][y/w]$ $(x \in \mathrm{fv}(t))$

Zooming back

- Comp: two rules
  - $t[y/w][x/v] \to t[y/w[x/v]]$ $(x \in \mathrm{fv}(w))$
  - good: more sharing and does not endanger PSN (thanks to explicit erasing — *cf.* David & Guillaume)
  - $t[y/w][x/v] \to t[x/v][y/w]$ $(x \in \mathrm{fv}(t))$

  $\Rightarrow$ thrown away (may loop for ever)

Zooming back

- Comp: two rules
  - $t[y/w][x/v] \to t[y/w[x/v]]$ $(x \in \mathrm{fv}(w))$
  - good: more sharing and does not endanger PSN (thanks to explicit erasing — *cf.* David & Guillaume)
  - $t[y/w][x/v] \to t[x/v][y/w]$ $(x \in \mathrm{fv}(t))$
  - thrown away (may loop for ever)
- Beta: critical pair Beta/App_1

$$\begin{array}{ccccc} ((\lambda x.t)u)[y/v] & \longrightarrow & ((\lambda x.t)[y/v])u & \longrightarrow & (\lambda x.t[y/v])u \\ \downarrow & & & & \downarrow \\ t[x/u][y/v] & & \overset{?}{\longleftrightarrow} & & t[y/v][x/u] \end{array}$$

where $x, y \in \mathrm{fv}(t)$

Zooming back

- Comp: two rules
  - $t[y/w][x/v] \to t[y/w[x/v]]$ $(x \in \mathrm{fv}(w))$
  - good: more sharing and does not endanger PSN (thanks to explicit erasing — *cf.* David & Guillaume)
  - $t[y/w][x/v] \to t[x/v][y/w]$ $(x \in \mathrm{fv}(t))$
  - thrown away (may loop for ever)
- Beta: critical pair Beta/App$_1$

$$\begin{array}{ccccc} ((\lambda x.t)u)[y/v] & \longrightarrow & ((\lambda x.t)[y/v])u & \longrightarrow & (\lambda x.t[y/v])u \\ \downarrow & & & & \downarrow \\ t[x/u][y/v] & & \overset{?}{\longleftrightarrow} & & t[y/v][x/u] \end{array}$$

where $x, y \in \mathrm{fv}(t)$

$\Rightarrow$ $\mathrm{fv}(\lambda x.t) = \emptyset$ in Beta cuts the left branch

Director strings: first intuitions

- $(t\ u)[x/v] \longrightarrow (t[x/v])\ u$ if $x \in \mathrm{fv}(t)$ (and $x \notin \mathrm{fv}(u)$)

Director strings: first intuitions

- $(t\,u)[x/v] \to (t[x/v])\,u$ if $x \in \mathrm{fv}(t)$ (and $x \notin \mathrm{fv}(u)$)
- How to do that **efficiently** ?

Director strings: first intuitions

- $(t\,u)[x/v] \to (t[x/v])\,u$ if $x \in \mathrm{fv}(t)$ (and $x \notin \mathrm{fv}(u)$)
- How to do that **efficiently** ?

$\Rightarrow$ $(t\,u)\curvearrowleft[v] \to (t[v])\,u$

Director strings: first intuitions

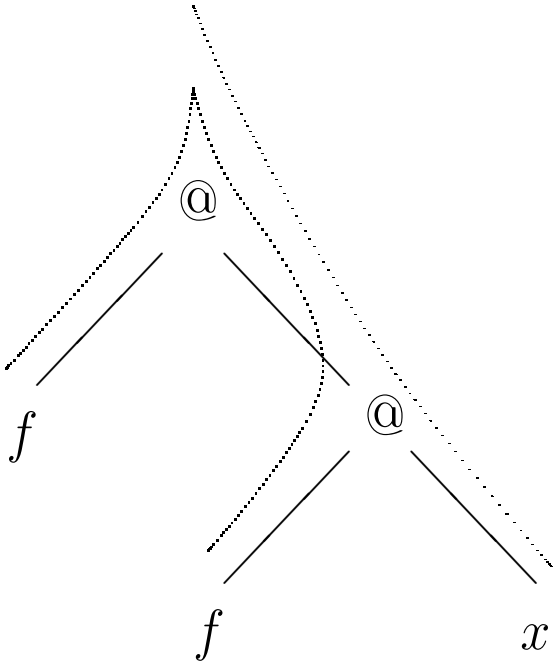
- $(t\,u)[x/v] \to (t[x/v])\,u$ if $x \in \mathrm{fv}(t)$ (and $x \notin \mathrm{fv}(u)$)
- How to do that **efficiently** ?
- $(t\,u)^{\curvearrowleft}[v] \to (t[v])\,u$
- **director strings** *internalize* the info about free variables in the syntax

Director strings: first intuitions

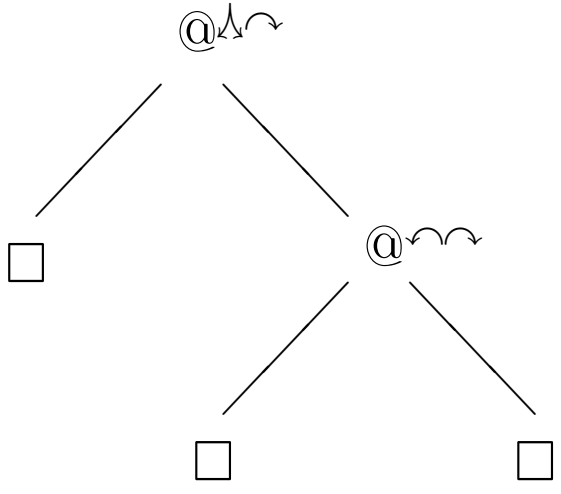
- $(t\,u)[x/v] \to (t[x/v])\,u$ if $x \in \mathrm{fv}(t)$ (and $x \notin \mathrm{fv}(u)$)
- How to do that **efficiently** ?
- $(t\,u)^{\curvearrowleft}[v] \to (t[v])\,u$
- **director strings** *internalize* the info about free variables in the syntax
- an old idea: Kennaway and Sleep (for combinators)

An example

$(f\ (f\ x))[f/F][x/X]$



$(\square\ (\square\ \square)\curvearrowright\curvearrowright)\curvearrowleft\curvearrowright$



Syntax

- **directors**: $\curvearrowleft$, $\curvearrowright$, $\bigwedge$, $\downarrow$

Syntax

- **directors**: $\curvearrowleft$, $\curvearrowright$, Λ, $\downarrow$
- **director strings**: words on directors

Syntax

- **directors**: $\curvearrowleft$, $\curvearrowright$, Λ, $\downarrow$
- **director strings**: words on directors
- **preterms**:

$$t, u ::= \Box \mid (\lambda t)^{\sigma} \mid (\lambda^{-} t)^{\sigma} \mid (t\ u)^{\sigma} \mid (t[k/u])^{\sigma}$$

Syntax

- **directors**: $\curvearrowleft$, $\curvearrowright$, Λ, $\downarrow$
- **director strings**: words on directors
- **preterms**:

$$t, u ::= \Box \mid (\lambda t)^{\sigma} \mid (\lambda^{-} t)^{\sigma} \mid (t\ u)^{\sigma} \mid (t[k/u])^{\sigma}$$

- **variables** (a place holder)

Syntax

- **directors**: $\curvearrowleft$, $\curvearrowright$, $\Lsh$, $\downarrow$
- **director strings**: words on directors
- **preterms**:

$$t, u ::= \Box \mid (\lambda t)^\sigma \mid (\lambda^- t)^\sigma \mid (t\ u)^\sigma \mid (t[k/u])^\sigma$$

- **variables** (a place holder)
- explicit **substitution** (k is the position of the variable)
  notation: $(t[u])^\sigma = (t[1/u])^\sigma$

Syntax

- **directors**: $\curvearrowleft$, $\curvearrowright$, Λ, $\downarrow$
- **director strings**: words on directors
- **preterms**:

$$t, u ::= \Box \mid (\lambda t)^\sigma \mid (\lambda^- t)^\sigma \mid (t\ u)^\sigma \mid (t[k/u])^\sigma$$

- **variables** (a place holder)
- explicit **substitution** (k is the position of the variable)
  notation: $(t[u])^\sigma = (t[1/u])^\sigma$
- two kinds of **abstraction**:

Syntax

- **directors**: $\curvearrowleft$, $\curvearrowright$, $\bigtriangleup$, $\downarrow$
- **director strings**: words on directors
- **preterms**:

$$t, u ::= \Box \mid (\lambda t)^{\sigma} \mid (\lambda^{-} t)^{\sigma} \mid (t\ u)^{\sigma} \mid (t[k/u])^{\sigma}$$

- **variables** (a place holder)
- explicit **substitution** (k is the position of the variable)
  notation: $(t[u])^{\sigma} = (t[1/u])^{\sigma}$
- two kinds of **abstraction**:
  - $(\lambda t)^{\sigma}$: the bound variable occurs free in t

Syntax

- **directors**: $\curvearrowleft$, $\curvearrowright$, Λ, $\downarrow$
- **director strings**: words on directors
- **preterms**:

$$t, u ::= \Box \mid (\lambda t)^\sigma \mid (\lambda^- t)^\sigma \mid (t\ u)^\sigma \mid (t[k/u])^\sigma$$

- **variables** (a place holder)
- explicit **substitution** (k is the position of the variable)
  notation: $(t[u])^\sigma = (t[1/u])^\sigma$
- two kinds of **abstraction**:
  - $(\lambda t)^\sigma$: the bound variable occurs free in t
  - $(\lambda^- t)^\sigma$: the bound variable does not appear free in t

Syntax

- **directors**: $\curvearrowleft$, $\curvearrowright$, $\Lsh$, $\downarrow$
- **director strings**: words on directors
- **preterms**:

$$t, u ::= \Box \mid (\lambda t)^{\sigma} \mid (\lambda^{-} t)^{\sigma} \mid (t\ u)^{\sigma} \mid (t[k/u])^{\sigma}$$

- **variables** (a place holder)
- explicit **substitution** (k is the position of the variable)
  notation: $(t[u])^{\sigma} = (t[1/u])^{\sigma}$
- two kinds of **abstraction**:
  - $(\lambda t)^{\sigma}$: the bound variable occurs free in t
  - $(\lambda^{-} t)^{\sigma}$: the bound variable does not appear free in t
- annotated by a director string
  notation: $\Box = \Box^{\downarrow}$

Syntax

- **directors**: $\curvearrowleft$, $\curvearrowright$, $\Lsh$, $\downarrow$
- **director strings**: words on directors
- **preterms**:

$$t, u ::= \square \mid (\lambda t)^\sigma \mid (\lambda^- t)^\sigma \mid (t\ u)^\sigma \mid (t[k/u])^\sigma$$

- **variables** (a place holder)
- explicit **substitution** (k is the position of the variable)
  notation: $(t[u])^\sigma = (t[1/u])^\sigma$
- two kinds of **abstraction**:
  - $(\lambda t)^\sigma$: the bound variable occurs free in t
  - $(\lambda^- t)^\sigma$: the bound variable does not appear free in t
- annotated by a director string
  notation: $\square = \square^\downarrow$
- **terms** have to satisfy well-formedness conditions

Open calculus

- Can we write reduction rules that:

Open calculus

- Can we write reduction rules that:
  - are **consistent** w.r.t. the syntax

Open calculus

- Can we write reduction rules that:
  - are **consistent** w.r.t. the syntax
  - **fully simulate** the λ-calculus

Open calculus

- Can we write reduction rules that:
  - are **consistent** w.r.t. the syntax
  - **fully simulate** the λ-calculus
  - are **confluent** and **PSN**

Open calculus

- Can we write reduction rules that:
  - are **consistent** w.r.t. the syntax
  - **fully simulate** the λ-calculus
  - are **confluent** and **PSN**
- **Yes !**

Open calculus

- Can we write reduction rules that:
  - are **consistent** w.r.t. the syntax
  - **fully simulate** the λ-calculus
  - are **confluent** and **PSN**
- Yes !
- *eg.* App_1: $((t\ u)^{\rho}[i/v])^{\sigma} \to ((t[j/v])^{\upsilon}\ u)^{\tau}$ when $\rho_i = \curvearrowleft$

Open calculus

- Can we write reduction rules that:
  - are **consistent** w.r.t. the syntax
  - **fully simulate** the λ-calculus
  - are **confluent** and **PSN**
- Yes !
- *eg.* App$_1$: $((t\,u)^{\rho}[i/v])^{\sigma} \to ((t[j/v])^{\upsilon}\,u)^{\tau}$ when $\rho_i = \curvearrowleft$
- but computation of τ and υ is inefficient

Open calculus

- Can we write reduction rules that:
  - are **consistent** w.r.t. the syntax
  - **fully simulate** the λ-calculus
  - are **confluent** and **PSN**
- Yes !
- *eg.* App$_1$: $((t\,u)^{\rho}[i/v])^{\sigma} \to ((t[j/v])^{\upsilon}\,u)^{\tau}$ when $\rho_i = \curvearrowleft$
- but computation of τ and υ is inefficient
- again, we are looking for an **efficient strategy**, so we may give up full simulation

Open calculus

- Can we write reduction rules that:
  - are **consistent** w.r.t. the syntax
  - **fully simulate** the λ-calculus
  - are **confluent** and **PSN**
- Yes !
- *eg.* App$_1$: $((t\,u)^{\rho}[i/v])^{\sigma} \to ((t[j/v])^{\upsilon}\,u)^{\tau}$ when $\rho_i = \curvearrowleft$
- but computation of τ and υ is inefficient
- again, we are looking for an **efficient strategy**, so we may give up full simulation
- rules are **simpler** for closed reduction (note that we associate erasing with abstraction as λ^- and copy with application thanks to $\curlywedge$)

Closed reduction with director strings

Name			
Beta	$((\lambda t)^{\epsilon}\ u)\curvearrowright^{n}$	$\longrightarrow$	$(t[u])\curvearrowright^{n}$
BetaE	$((\lambda^{-} t)^{\epsilon}\ u^{\epsilon})^{\epsilon}$	$\longrightarrow$	t
Var	$(\Box[v])\curvearrowright^{n}$	$\longrightarrow$	v
App_1	$((t\ u)\curvearrowleft\rho[v^{\epsilon}])\curvearrowleft^{n}$	$\longrightarrow$	$((t[v^{\epsilon}])\curvearrowleft^{\lvert\rho\rvert_{\mathrm{l}}}\ u)\rho$
App_2	$((t\ u)\curvearrowright\rho[v^{\epsilon}])\curvearrowleft^{n}$	$\longrightarrow$	$(t\ (u[v^{\epsilon}])\curvearrowleft^{\lvert\rho\rvert_{\mathrm{r}}})\rho$
App_3	$((t\ u)\downdownarrows\rho[v^{\epsilon}])\curvearrowleft^{n}$	$\longrightarrow$	$((t[v^{\epsilon}])\curvearrowleft^{\lvert\rho\rvert_{\mathrm{l}}}\ (u[v^{\epsilon}])\curvearrowleft^{\lvert\rho\rvert_{\mathrm{r}}})\rho$
Lam	$((\lambda t)^{\downarrow}\rho[v^{\epsilon}])\curvearrowleft^{n}$	$\longrightarrow$	$(\lambda(t[v^{\epsilon}])\curvearrowleft^{n+1})^{\downarrow n}$
LamE	$((\lambda^{-} t)^{\downarrow}\rho[v^{\epsilon}])\curvearrowleft^{n}$	$\longrightarrow$	$(\lambda^{-}(t[v^{\epsilon}])\curvearrowleft^{n})^{\downarrow n}$
Comp	$((t[w])\curvearrowright^{n+1}[v^{\epsilon}])\curvearrowleft^{n}$	$\longrightarrow$	$(t[(w[v^{\epsilon}])\curvearrowleft^{n}])\curvearrowright^{n}$

Closed reduction with director strings

Name			Reduction
Beta	$((\lambda t)^{\epsilon}\ u)\curvearrowright^{n}$	$\longrightarrow$	$(t[u])\curvearrowright^{n}$
BetaE	$((\lambda^{-} t)^{\epsilon}\ u^{\epsilon})^{\epsilon}$	$\longrightarrow$	t
Var	$(\Box[v])\curvearrowright^{n}$	$\longrightarrow$	v
App_1	$((t\ u)\curvearrowleft\rho[v^{\epsilon}])\curvearrowleft^{n}$	$\longrightarrow$	$((t[v^{\epsilon}])\curvearrowleft^{\lvert\rho\rvert_{\mathrm{l}}}\ u)\rho$
App_2	$((t\ u)\curvearrowright\rho[v^{\epsilon}])\curvearrowleft^{n}$	$\longrightarrow$	$(t\ (u[v^{\epsilon}])\curvearrowleft^{\lvert\rho\rvert_{\mathrm{r}}})\rho$
App_3	$((t\ u)\downdownarrows\rho[v^{\epsilon}])\curvearrowleft^{n}$	$\longrightarrow$	$((t[v^{\epsilon}])\curvearrowleft^{\lvert\rho\rvert_{\mathrm{l}}}\ (u[v^{\epsilon}])\curvearrowleft^{\lvert\rho\rvert_{\mathrm{r}}})\rho$
Lam	$((\lambda t)^{\downarrow}\rho[v^{\epsilon}])\curvearrowleft^{n}$	$\longrightarrow$	$(\lambda(t[v^{\epsilon}])\curvearrowleft^{n+1})^{\downarrow n}$
LamE	$((\lambda^{-} t)^{\downarrow}\rho[v^{\epsilon}])\curvearrowleft^{n}$	$\longrightarrow$	$(\lambda^{-}(t[v^{\epsilon}])\curvearrowleft^{n})^{\downarrow n}$
Comp	$((t[w])\curvearrowright^{n+1}[v^{\epsilon}])\curvearrowleft^{n}$	$\longrightarrow$	$(t[(w[v^{\epsilon}])\curvearrowleft^{n}])\curvearrowright^{n}$

Closed reduction with director strings

Name			Reduction
Beta	$((\lambda t)^{\epsilon}\ u)\curvearrowright^{n}$	$\longrightarrow$	$(t[u])\curvearrowright^{n}$
BetaE	$((\lambda^{-} t)^{\epsilon}\ u^{\epsilon})^{\epsilon}$	$\longrightarrow$	t
Var	$(\Box[v])\curvearrowright^{n}$	$\longrightarrow$	v
App_1	$((t\ u)\curvearrowleft\rho[v^{\epsilon}])\curvearrowleft^{n}$	$\longrightarrow$	$((t[v^{\epsilon}])\curvearrowleft^{\lvert\rho\rvert_{\mathrm{l}}}\ u)\rho$
App_2	$((t\ u)\curvearrowright\rho[v^{\epsilon}])\curvearrowleft^{n}$	$\longrightarrow$	$(t\ (u[v^{\epsilon}])\curvearrowleft^{\lvert\rho\rvert_{\mathrm{r}}})\rho$
App_3	$((t\ u)\downarrow\rho[v^{\epsilon}])\curvearrowleft^{n}$	$\longrightarrow$	$((t[v^{\epsilon}])\curvearrowleft^{\lvert\rho\rvert_{\mathrm{l}}}\ (u[v^{\epsilon}])\curvearrowleft^{\lvert\rho\rvert_{\mathrm{r}}})\rho$
Lam	$((\lambda t)^{\downarrow}\rho[v^{\epsilon}])\curvearrowleft^{n}$	$\longrightarrow$	$(\lambda(t[v^{\epsilon}])\curvearrowleft^{n+1})^{\downarrow n}$
LamE	$((\lambda^{-} t)^{\downarrow}\rho[v^{\epsilon}])\curvearrowleft^{n}$	$\longrightarrow$	$(\lambda^{-}(t[v^{\epsilon}])\curvearrowleft^{n})^{\downarrow n}$
Comp	$((t[w])\curvearrowright^{n+1}[v^{\epsilon}])\curvearrowleft^{n}$	$\longrightarrow$	$(t[(w[v^{\epsilon}])\curvearrowleft^{n}])\curvearrowright^{n}$

Closed reduction with director strings

Name			Reduction
Beta	$((\lambda t)^{\epsilon}\ u)\curvearrowright^{n}$	$\longrightarrow$	$(t[u])\curvearrowright^{n}$
BetaE	$((\lambda^{-} t)^{\epsilon}\ u^{\epsilon})^{\epsilon}$	$\longrightarrow$	t
Var	$(\square[v])\curvearrowright^{n}$	$\longrightarrow$	v
App_1	$((t\ u)\curvearrowleft^{\rho}[v^{\epsilon}])\curvearrowleft^{n}$	$\longrightarrow$	$((t[v^{\epsilon}])\curvearrowleft^{\lvert\rho\rvert_{\mathrm{l}}}\ u)\rho$
App_2	$((t\ u)\curvearrowright^{\rho}[v^{\epsilon}])\curvearrowleft^{n}$	$\longrightarrow$	$(t\ (u[v^{\epsilon}])\curvearrowleft^{\lvert\rho\rvert_{\mathrm{r}}})\rho$
App_3	$((t\ u)\downdownarrows^{\rho}[v^{\epsilon}])\curvearrowleft^{n}$	$\longrightarrow$	$((t[v^{\epsilon}])\curvearrowleft^{\lvert\rho\rvert_{\mathrm{l}}}\ (u[v^{\epsilon}])\curvearrowleft^{\lvert\rho\rvert_{\mathrm{r}}})\rho$
Lam	$((\lambda t)^{\downarrow\rho}[v^{\epsilon}])\curvearrowleft^{n}$	$\longrightarrow$	$(\lambda(t[v^{\epsilon}])\curvearrowleft^{n+1})^{\downarrow n}$
LamE	$((\lambda^{-} t)^{\downarrow\rho}[v^{\epsilon}])\curvearrowleft^{n}$	$\longrightarrow$	$(\lambda^{-}(t[v^{\epsilon}])\curvearrowleft^{n})^{\downarrow n}$
Comp	$((t[w])\curvearrowright^{n+1}[v^{\epsilon}])\curvearrowleft^{n}$	$\longrightarrow$	$(t[(w[v^{\epsilon}])\curvearrowleft^{n}])\curvearrowright^{n}$

Closed reduction with director strings

Name			Reduction
Beta	$((\lambda t)^{\epsilon}\ u)^{\curvearrowright^n}$	$\longrightarrow$	$(t[u])^{\curvearrowright^n}$
BetaE	$((\lambda^{-} t)^{\epsilon}\ u^{\epsilon})^{\epsilon}$	$\longrightarrow$	t
Var	$(\square[v])^{\curvearrowright^n}$	$\longrightarrow$	v
App_1	$((t\ u)^{\curvearrowleft\rho}[v^{\epsilon}])^{\curvearrowleft^n}$	$\longrightarrow$	$((t[v^{\epsilon}])^{\curvearrowleft^{\lvert\rho\rvert_{\mathrm{l}}}}\ u)^{\rho}$
App_2	$((t\ u)^{\curvearrowright\rho}[v^{\epsilon}])^{\curvearrowleft^n}$	$\longrightarrow$	$(t\ (u[v^{\epsilon}])^{\curvearrowleft^{\lvert\rho\rvert_{\mathrm{r}}}})^{\rho}$
App_3	$((t\ u)^{\curlywedge\rho}[v^{\epsilon}])^{\curvearrowleft^n}$	$\longrightarrow$	$((t[v^{\epsilon}])^{\curvearrowleft^{\lvert\rho\rvert_{\mathrm{l}}}}\ (u[v^{\epsilon}])^{\curvearrowleft^{\lvert\rho\rvert_{\mathrm{r}}}})^{\rho}$
Lam	$((\lambda t)^{\downarrow\rho}[v^{\epsilon}])^{\curvearrowleft^n}$	$\longrightarrow$	$(\lambda(t[v^{\epsilon}])^{\curvearrowleft^{n+1}})^{\downarrow^n}$
LamE	$((\lambda^{-} t)^{\downarrow\rho}[v^{\epsilon}])^{\curvearrowleft^n}$	$\longrightarrow$	$(\lambda^{-}(t[v^{\epsilon}])^{\curvearrowleft^n})^{\downarrow^n}$
Comp	$((t[w])^{\curvearrowright^{n+1}}[v^{\epsilon}])^{\curvearrowleft^n}$	$\longrightarrow$	$(t[(w[v^{\epsilon}])^{\curvearrowleft^n}])^{\curvearrowright^n}$

Closed reduction with director strings

Name			Reduction
Beta	$((\lambda t)^{\epsilon}\ u)^{\curvearrowright^n}$	$\longrightarrow$	$(t[u])^{\curvearrowright^n}$
BetaE	$((\lambda^{-} t)^{\epsilon}\ u^{\epsilon})^{\epsilon}$	$\longrightarrow$	t
Var	$(\square[v])^{\curvearrowright^n}$	$\longrightarrow$	v
App_1	$((t\ u)^{\curvearrowleft\rho}[v^{\epsilon}])^{\curvearrowleft^n}$	$\longrightarrow$	$((t[v^{\epsilon}])^{\curvearrowleft^{\lvert\rho\rvert_{\mathrm{l}}}}\ u)^{\rho}$
App_2	$((t\ u)^{\curvearrowright\rho}[v^{\epsilon}])^{\curvearrowleft^n}$	$\longrightarrow$	$(t\ (u[v^{\epsilon}])^{\curvearrowleft^{\lvert\rho\rvert_{\mathrm{r}}}})^{\rho}$
App_3	$((t\ u)^{\downdownarrows\rho}[v^{\epsilon}])^{\curvearrowleft^n}$	$\longrightarrow$	$((t[v^{\epsilon}])^{\curvearrowleft^{\lvert\rho\rvert_{\mathrm{l}}}}\ (u[v^{\epsilon}])^{\curvearrowleft^{\lvert\rho\rvert_{\mathrm{r}}}})^{\rho}$
Lam	$((\lambda t)^{\downarrow\rho}[v^{\epsilon}])^{\curvearrowleft^n}$	$\longrightarrow$	$(\lambda(t[v^{\epsilon}])^{\curvearrowleft^{n+1}})^{\downarrow^n}$
LamE	$((\lambda^{-} t)^{\downarrow\rho}[v^{\epsilon}])^{\curvearrowleft^n}$	$\longrightarrow$	$(\lambda^{-}(t[v^{\epsilon}])^{\curvearrowleft^n})^{\downarrow^n}$
Comp	$((t[w])^{\curvearrowright^{n+1}}[v^{\epsilon}])^{\curvearrowleft^n}$	$\longrightarrow$	$(t[(w[v^{\epsilon}])^{\curvearrowleft^n}])^{\curvearrowright^n}$

Closed reduction with director strings

Name			Reduction
Beta	$((\lambda t)^{\epsilon}\ u)^{\curvearrowright^n}$	$\longrightarrow$	$(t[u])^{\curvearrowright^n}$
BetaE	$((\lambda^{-} t)^{\epsilon}\ u^{\epsilon})^{\epsilon}$	$\longrightarrow$	t
Var	$(\square[v])^{\curvearrowright^n}$	$\longrightarrow$	v
App_1	$((t\ u)^{\curvearrowleft\rho}[v^{\epsilon}])^{\curvearrowleft^n}$	$\longrightarrow$	$((t[v^{\epsilon}])^{\curvearrowleft^{\lvert\rho\rvert_{\mathrm{l}}}}\ u)^{\rho}$
App_2	$((t\ u)^{\curvearrowright\rho}[v^{\epsilon}])^{\curvearrowleft^n}$	$\longrightarrow$	$(t\ (u[v^{\epsilon}])^{\curvearrowleft^{\lvert\rho\rvert_{\mathrm{r}}}})^{\rho}$
App_3	$((t\ u)^{\downdownarrows\rho}[v^{\epsilon}])^{\curvearrowleft^n}$	$\longrightarrow$	$((t[v^{\epsilon}])^{\curvearrowleft^{\lvert\rho\rvert_{\mathrm{l}}}}\ (u[v^{\epsilon}])^{\curvearrowleft^{\lvert\rho\rvert_{\mathrm{r}}}})^{\rho}$
Lam	$((\lambda t)^{\downarrow\rho}[v^{\epsilon}])^{\curvearrowleft^n}$	$\longrightarrow$	$(\lambda(t[v^{\epsilon}])^{\curvearrowleft^{n+1}})^{\downarrow^n}$
LamE	$((\lambda^{-} t)^{\downarrow\rho}[v^{\epsilon}])^{\curvearrowleft^n}$	$\longrightarrow$	$(\lambda^{-}(t[v^{\epsilon}])^{\curvearrowleft^n})^{\downarrow^n}$
Comp	$((t[w])^{\curvearrowright^{n+1}}[v^{\epsilon}])^{\curvearrowleft^n}$	$\longrightarrow$	$(t[(w[v^{\epsilon}])^{\curvearrowleft^n}])^{\curvearrowright^n}$

Properties

- completely **nameless** (no indices either)

Properties

- completely **nameless** (no indices either)
- **intuitive**: directors correspond to paths

Properties

- completely **nameless** (no indices either)
- **intuitive**: directors correspond to paths
- **simple** rules with a **constant** algorithmic cost

Properties

- completely **nameless** (no indices either)
- **intuitive**: directors correspond to paths
- **simple** rules with a **constant** algorithmic cost
- closed substitutions can **propagate under** λ**'s** in a very **simple** way (compared to named or de Bruijn λ-calculi)

Properties

- completely **nameless** (no indices either)
- **intuitive**: directors correspond to paths
- **simple** rules with a **constant** algorithmic cost
- closed substitutions can **propagate under** λ**'s** in a very **simple** way (compared to named or de Bruijn λ-calculi)
- free variables are **never duplicated**

Properties

- completely **nameless** (no indices either)
- **intuitive**: directors correspond to paths
- **simple** rules with a **constant** algorithmic cost
- closed substitutions can **propagate under** λ**'s** in a very **simple** way (compared to named or de Bruijn λ-calculi)
- free variables are **never duplicated**
- **confluent**

Properties

- completely **nameless** (no indices either)
- **intuitive**: directors correspond to paths
- **simple** rules with a **constant** algorithmic cost
- closed substitutions can **propagate under** λ**'s** in a very **simple** way (compared to named or de Bruijn λ-calculi)
- free variables are **never duplicated**
- **confluent**
- **PSN**

Properties

- completely **nameless** (no indices either)
- **intuitive**: directors correspond to paths
- **simple** rules with a **constant** algorithmic cost
- closed substitutions can **propagate under** λ**'s** in a very **simple** way (compared to named or de Bruijn λ-calculi)
- free variables are **never duplicated**
- **confluent**
- **PSN**
- adequate for reduction of **closed** terms to **whnf**

The **C**losed **R**eduction **A**bstract **M**achine

- weak head normal form:

The Closed Reduction Abstract Machine

- weak head normal form:
  - at the top-level, do not reduce under λ's

The Closed Reduction Abstract Machine

- weak head normal form:
  - at the top-level, do not reduce under λ's
  - reduce under λ's before copying (easily identified)

The Closed Reduction Abstract Machine

- weak head normal form:
  - at the top-level, do not reduce under λ's
  - reduce under λ's before copying (easily identified)
  - use Comp

The Closed Reduction Abstract Machine

- weak head normal form:
  - at the top-level, do not reduce under λ's
  - reduce under λ's before copying (easily identified)
  - use Comp
- full normal form:

The Closed Reduction Abstract Machine

- weak head normal form:
  - at the top-level, do not reduce under λ's
  - reduce under λ's before copying (easily identified)
  - use Comp
- full normal form:
  - compute **successive whnf's**

The Closed Reduction Abstract Machine

- weak head normal form:
  - at the top-level, do not reduce under λ's
  - reduce under λ's before copying (easily identified)
  - use Comp
- full normal form:
  - compute **successive whnf's**
  - well-known trick: successively **freeze** free variables

Experimental results

term	CRAM	CBV	BOHM
2 2 I I	61 (9)	82 (11)	40 (9)
2 2 2 I I	140 (19)	302 (42)	100 (16)
3 2 2 I I	248 (33)	3508 (531)	184 (21)
5 5 I I	217 (33)	26669 (3913)	229 (33)
5 2 2 I I	832 (109)	–	847 (31)
2 2 2 2 2 I I	1507714 (196655)	–	1074037060 (61)

Conclusion

- an **efficient strategy**

Conclusion

- an **efficient strategy**
- supported by a **good theoretical framework**

Conclusion

- an **efficient strategy**
- supported by a **good theoretical framework**
- but still a **prototype**

Conclusion

- an **efficient strategy**
- supported by a **good theoretical framework**
- but still a **prototype**
- techniques of **compilation**, *etc.* still to be investigated (non-trivial work)

Conclusion

- an **efficient strategy**
- supported by a **good theoretical framework**
- but still a **prototype**
- techniques of **compilation**, *etc.* still to be investigated (non-trivial work)
- seems well-suited for **proof assistants** (strongly normalising terms, higher-order functions, impredicativity)

Conclusion

- an **efficient strategy**
- supported by a **good theoretical framework**
- but still a **prototype**
- techniques of **compilation**, *etc.* still to be investigated (non-trivial work)
- seems well-suited for **proof assistants** (strongly normalising terms, higher-order functions, impredicativity)
- for **functional languages**, should be associated with a weaker strategy either in an *ad hoc* way (*cf.* if␣then␣else in ML) or with *optimistic evaluation*

That's all, folks

Questions ?