λ-calculus (part 1)

Introduction

$f(x)=3*x + 1$ $f(8)=3*8+1$

$$x \longmapsto 3*x + 1$$

$\lambda x.3*x+1$ $(\lambda x.(3*x+1)\ 8) = 3*8+1$

$$(\lambda x.B\ A) \longrightarrow_\beta B[x:=A]$$

Already un many languages

Maths

$x \mapsto x$ $g \circ f$ $(g,f) \mapsto (x \mapsto g\ (f\ x))$

ML

```
fun x -> x                    fun g f -> fun x -> g (f x)
```

Scheme

```
(lambda (x) x)        (lambda (g f) (lambda (x) (g (f x))))
```

JavaScript

```
function (x) {return(x);}
```

λ calculus

λx.x λg.λf.λx.(g (f x))

Abstract syntax

Term ≜		Variable	
	|	λ Variable.Term	*abstraction*
	|	(Term Term)	*application*

In λx.B, B is called the *body* of the function

Church wanted	$\hat{x}$.B
Typewritters do	^x.B
Then	Λx.B
Finally	λx.B

Notations

Curryfication

$$\lambda x.(\lambda y.B) = \lambda x.\lambda y.B = \lambda xy.B$$

$$((F\ A)\ C) = (F\ A\ C)$$

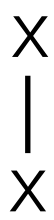
Example $(g,f) \mapsto (x \mapsto g\ (f\ x))$ is $\lambda gfx.(g\ (f\ x))$

Redex : Reducible expression

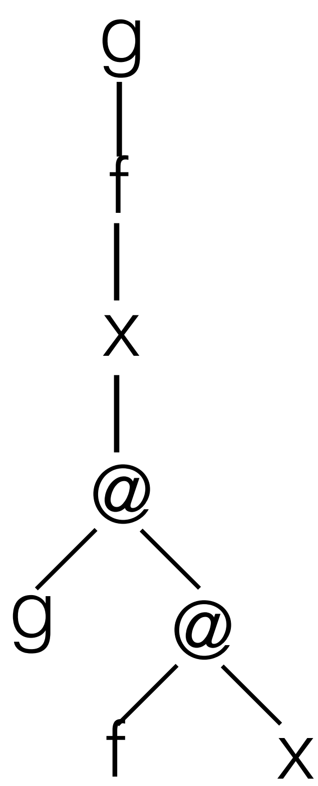
$$\text{let } x{=}A \text{ in } B \triangleq (\lambda x.B\ A)$$

Tree representation

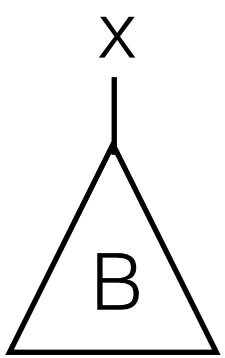
λx.x



λgfx.g (f x)



λx.B



β-reduction rule

$$(\lambda x.B\ A) \longrightarrow_{\beta} B[x:=A]$$

Substitution

Bound variables

$$\mathrm{Bound}(x) = \emptyset$$
$$\mathrm{Bound}(\lambda x.B) = \mathrm{Bound}(B) \cup \{x\}$$
$$\mathrm{Bound}(F\ A) = \mathrm{Bound}(F) \cup \mathrm{Bound}(A)$$

$$\mathrm{Bound}(\lambda x.x) = \{x\}$$

$$\mathrm{Bound}(\lambda xy.x) = \{x,y\}$$

$$\mathrm{Bound}(\lambda x.xy) = \{x\}$$

$$\mathrm{Bound}(\lambda x.(x\ \lambda y.(f\ y)))=\{x,y\}$$

Free variables

$$\mathrm{Free}(x) = \{x\}$$
$$\mathrm{Free}(\lambda\, x.B) = \mathrm{Free}(B) - \{x\}$$
$$\mathrm{Free}(F\ A) = \mathrm{Free}(F) \cup \mathrm{Free}(A)$$

$\mathrm{Free}(\lambda x.x) = \emptyset$

$\mathrm{Free}(\lambda xy.x) = \emptyset$

$\mathrm{Free}(\lambda x.(x\ y)) = \{y\}$

$\mathrm{Free}(\lambda x.(x\ \lambda y.(f\ y)))=\{f\}$

T is a *closed* term $\Leftrightarrow$ $\mathrm{Free}(T)= \emptyset$

sometimes called *combinator*

Bound and Free are not related

~~$Bound(T) \cap Free(T) = \emptyset$~~

$Bound(x\ \lambda x.x) = Free(x\ \lambda x.x) = \{x\}$

Closing a term by adding abstraction on the outside

Substitution

B[x:=N] : the substitution in a term B of all free occurrences of the variable x by N

x[x:=N]	→ N	
y[x:=N]	→ y	(x≠y)
(F A)[x:=N]	→ (F[x:=N] A[x:=N])	
~~(λ y.B)[x:=N]~~	~~→ λ y.(B[x:=N])~~	

Substitution (cont)

$$(\lambda y.x)[x:=y] \rightarrow (\lambda y.x[x:=y])$$
$$\rightarrow (\lambda y.y)$$

$(\lambda y.B)[x:=N] \rightarrow \lambda y.B \qquad x \notin \{y\} \cup Free(B)$

$(\lambda y.B)[x:=N] \rightarrow \lambda y.(B[x:=N]) \qquad y \notin \{x\} \cup Free(N)$

And when $x \in Free(B) \;\wedge\; y \in Free(N)$?

$(\lambda y.B)[x:=N] \rightarrow \lambda z.(B[y:=z][x:=N])$
with z a *fresh* variable

$z \notin \{x,y\} \cup Free(B) \cup Free(N)$

Barendregt convention

Convention : For all sub terms, a variable cannot be both free and bound

Bound(x λx.x) = Free(x λx.x) = {x}

λx.(x λx.x) for *all* sub terms

alpha equivalence

$$\frac{P}{Q} \quad \text{means } P \Rightarrow Q$$

$$x \equiv_\alpha x$$

$$\frac{F_1 \equiv_\alpha A_1 \;\wedge\; F_2 \equiv_\alpha A_2}{(F_1\ A_1) \equiv_\alpha (F_2\ A_2)}$$

$$\frac{B_1 \equiv_\alpha B_2[x_2{:=}x_1]}{\lambda x_1.B_1 \equiv_\alpha \lambda x_2.B_2}$$

$$\frac{x_2 \notin \mathrm{Free}(B)}{\lambda x_1.B \equiv_\alpha \lambda x_2.B[x_1{:=}x_2]}$$

$$\frac{B_1 \equiv_\alpha B_2}{\lambda x.B_1 \equiv_\alpha \lambda x.B_2}$$

$$\frac{B_1[x_1{:=}z] \equiv_\alpha B_2[x_2{:=}z]}{\lambda x_1.B_1 \equiv_\alpha \lambda x_2.B_2}$$

with z a *fresh* variable

de Bruijn

Term := Nat
| λ.Term
| Term Term

λx.x → λ.0

λxy.x → λ.λ.1

λgfx.(g (f x)) → λ.λ.λ.(2 (1 0))

What about free variables ?

What about the substitution ?

Strategy

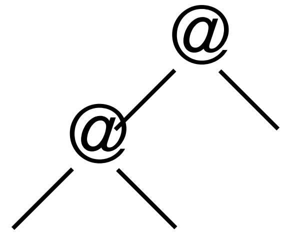
Reduction strategies

λx.(I I) is not *directly* a redex

$$\frac{B_1 \longrightarrow_\beta B_2}{\lambda x.B_1 \longrightarrow_\beta \lambda x.B_2}$$

Strong reduction

((I I) I) is not *directly* a redex

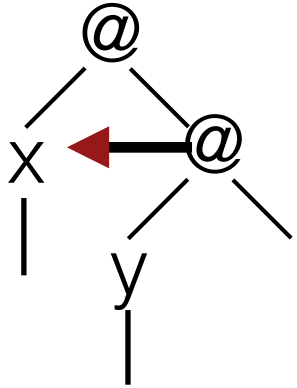


(weak) Call by need

$$\frac{F_1 \longrightarrow_\beta F_2}{(F_1\ A) \longrightarrow_\beta (F_2\ A)}$$

Reduction strategies

(I (I I)) have 2 redexes



outmost -> call by name

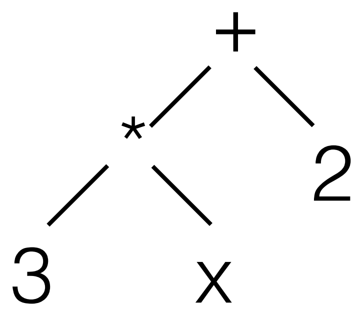
(I (I x)) have no cbv redex

$$(\lambda x.B\ v) \longrightarrow_{\beta_v} B[x{:=}v]$$ Call by value

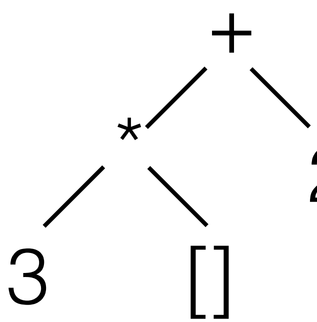
Contexts

A context is tree (AST) where a sub-tree is replaced by a hole

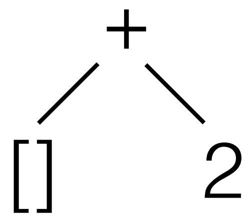
3*x + 2



3*[] + 2



[] + 2



Ast and Context

T := n | x | …
  | Op T T
  | …

C := []
  | OpL C T
  | OpR T C
  | …

C[E] fill the hole of the context with E

3*x + 2 = (3*[] + 2)[x] = C[x]

C[5] = 3*5 + 2

C[x+y] = 3*(x+y) + 2

reduction contexts

$$\frac{e_1 \longrightarrow_? e_2}{C[e_1] \longrightarrow_? C[e_2]}$$

$C \triangleq [] \mid \lambda x.C \mid C\ Term \mid Term\ C$ $\longrightarrow_\beta$

$C \triangleq [] \mid C\ Term \mid Term\ C$ $\longrightarrow_{\beta v}$

$C \triangleq [] \mid C\ Term$ $\longrightarrow_{\beta n}$

Transitive closure

$$\frac{M_1 \longrightarrow_\beta M_2}{M_1 \longrightarrow_{\beta^+} M_2}$$

$$\frac{M_1 \longrightarrow_{\beta^+} M_2 \wedge M_2 \longrightarrow_{\beta^+} M_3}{M_1 \longrightarrow_{\beta^+} M_3}$$

dessin avec extension sur un graphe

Reflexive transitive closure

$$M \longrightarrow_{\beta}^{*} M \qquad\qquad \frac{M_1 \longrightarrow_{\beta}^{+} M_2}{M_1 \longrightarrow_{\beta}^{*} M_2}$$

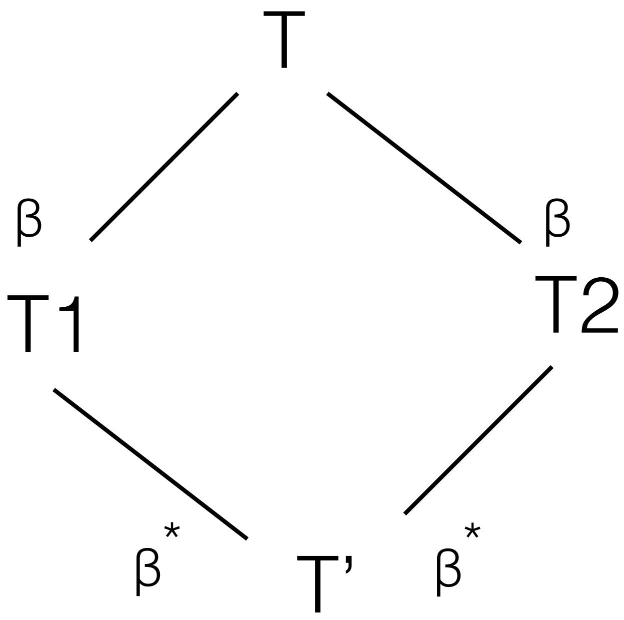
$A \longrightarrow_{\beta}^{*} B$ means that B can be reach from A with 0 or more beta-reduction

T is said in normal form if no more reduction can be applied

λx.x and λx.(x x) are in normal form

λx.(λy.y x) is in normal wrt the cvb strategy (head normal form) but not with a strong reduction

Confluence



Thus, the normal form of any term T, if exists, is unique

Operators

Fixpoint operator

$\triangle = \lambda f.(f\ f)$ $\qquad\qquad$ $\mathbf{Y} = \lambda f.(\triangle\ \lambda g.(f\ (g\ g)))$

$!n = n{*}!(n\text{-}1)$ $\qquad$ $f = \lambda n.({*}\ (f\ \underline{n})) = \lambda n.({*}\ ({*}\ (f\ \underline{n}))) = \ldots$

$$\lambda rn.({*}\ (r\ \underline{n}))$$

$$(\lambda rn.({*}\ (r\ r\ \underline{n}))\ \lambda rn.({*}\ (r\ r\ \underline{n})))$$

$$(\triangle\ \lambda gn.({*}\ (g\ g\ \underline{n}))) = (\triangle\ \lambda g.\lambda n.({*}\ ((g\ g)\ \underline{n})))$$

$$\leftarrow\ (\triangle\ \lambda g.(\lambda rn.({*}\ (r\ \underline{n}))\ (g\ g)))$$

$$\leftarrow\ (\lambda f.(\triangle\ \lambda g.(f\ (g\ g)))\ \lambda rn.({*}\ (r\ \underline{n}))) = (\mathbf{Y}\ \lambda rn.({*}\ (r\ \underline{n})))$$

Y property

$\mathbf{Y} = \lambda f.(\lambda g.(f\ (g\ g))\ \lambda g.(f\ (g\ g)))$

$$
\begin{aligned}
(Y\ F) &= (\lambda f.(\lambda g.(f\ (g\ g))\ \lambda g.(f\ (g\ g)))\ F) \\
&\longrightarrow (\lambda g.(F\ (g\ g))\ \ \lambda g.(F\ (g\ g))) \\
&\longrightarrow (F\ (\lambda g.(F\ (g\ g))\ \ \ \lambda g.(F\ (g\ g)))) \\
&\longleftarrow (F\ (Y\ F))
\end{aligned}
$$

Booleans

true = λxy.x

false = λxy.y

if = λbte.bte

(**if true** T E) = (λbte.bte λxy.x T E)
→ (λxy.x T E)
→ T

(**if false** T E) = (λbte.bte λxy.y T E)
→ (λxy.y T E)
→ E

Cartesian product

car = λp.(p **true**)

cdr = λp.(p **false**)

cons = λadb.(**if** b a d) = λadb.(b a d)

(**car** (**cons** X Y)) = (**car** (λadb.(b a d) X Y)
→ (**car** λb.(b X Y))
= (λp.(p λxy.x) λb.(b X Y))
→ (λb.(b X Y) λxy.x)
→ (λxy.x X Y)
→ X

Natural numbers

0 = λfx.x **1** = λfx.(f x) **2** = λfx.(f (f x))

0? = λn.(n λx.**false true**) **1+** = λnfx.(f (n f x))

+ = λmn.(m **1+** n) ***** = λmn.(n (**+** m) **0**)

^ = λmn.(n (***** m) **1**)

exo : **1+** = λnfx.(n f (f x))

exo **+** = λmnfx.(m f (n f x)) exo ***** = λmnf.(m (n f))

Natural numbers

Z? = λn.(n **true**)

P = λn.(n **false**)

Z = λb.(**if** b **true** ?)

S = λnb.(**if** b **false** n)

(**P** (**S** N)) = (λn.(n **false**) (λnb.(b **false** n) N))
→ (λn.(n **false**) λb.(b **false** N))
→ (λb.(b **false** N) **false**)
→ (**false false** N)
→ N

Lists

nil = λx.**true**

nth = λnl.(car (n cdr l))

null? = λp.(p λxy.**false**)

(**null? nil**) = (λp.(p λxy.**false**) λx.**true**)
→ (λx.**true** λxy.**false**)
→ **true**

(**null?** (cons X Y)) = (λp.(p λxy.**false**) (cons X Y))
→ ((λadb.bad X Y) λxy.**false**)
→ (λb.(b X Y) λxy.**false**)
→ (λxy.**false** X Y)
→ **false**