Lambda Calculus

$M \to x \mid \lambda x.M \mid M\,M$

Semantics usually defined in terms of equivalence

$$\frac{y \notin FV(M)}{\lambda x.M \equiv_\alpha \lambda y.[y/x]M}(\alpha) \qquad \frac{}{(\lambda x.M)\,N \equiv_\beta [N/x]M}(\beta)$$

$$\frac{x \notin FV(M)}{\lambda x.M\,x \equiv_\eta M}(\eta)$$

Reduction

$\lambda x.M \longleftrightarrow \lambda y.[y/x]M$ α-conversion
$(\lambda x.M)N \longrightarrow [N/x]M$ β-reduction

$$\lambda x.M\,x \underset{\eta\text{-reduction}}{\overset{\eta\text{-expansion}}{\rightleftarrows}} M$$

β-normal form - No more β reductions are possible

Not every term has a β-normal form
⇒ Can perform an infinite seq. of β-reductions!
If it does, it's unique and we only have to do β-reductions

"Computing" w/ λ-calculus = doing β-reductions

$$\frac{M_1 \mapsto M_1'}{M_1 M_2 \mapsto M_1' M_2}\ (1)$$

$$\frac{M_2 \mapsto M_2'}{(\lambda x.M_1)\, M_2 \mapsto (\lambda x.M_1)\, M_2'}\ (2)$$

$$\frac{}{(\lambda x.M_1)(\lambda y.M_2) \mapsto [\lambda y.M_2/x]M_1}\ (3)$$

Call-by-value

$$\frac{}{(\lambda x.M_1)\, M_2 \mapsto [M_2/x]M_1}\ (4)$$ – Call-by-name

Call-by-value

$(\lambda x.\ M)$ (loops forever) $\quad x \notin FV(M)$

$\mapsto \ldots$

May have a β-normal form. (M) but we'll never get there!

Call-by-name

$(\lambda x.\ x\,x\,x\,x)$ (takes a long time)

$\mapsto$ (takes a long time) (takes a long time) (takes a long time) (takes...

Theorem [Church-Rosser]

If $M_1 \mapsto^* M_2$ and $M_1 \mapsto M_3$, then there exists M_4 such that $M_2 \mapsto^* M_4$ and $M_3 \mapsto^* M_4$

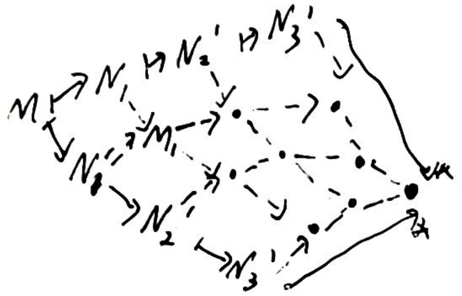
$$\begin{tikzcd}
 & M_2 \arrow[dr, dashed, "*"] & \\
M_1 \arrow[ur, "*"] \arrow[dr, "*"'] & & M_4 \\
 & M_3 \arrow[ur, dashed, "*"'] &
\end{tikzcd}$$

A lang has the "Church-Rosser property" if diff. evaluation orders lead to the same answer

Diamond property: If $M \mapsto N_1$ and $M \mapsto N_2$ then
$\exists M'$ s.t. $N_1 \mapsto M'$ and $N_2 \mapsto M'$

$$\begin{tikzcd} & N_1 \arrow[dr, mapsto] & \\ M \arrow[ur, mapsto] \arrow[dr, mapsto] & & M' \\ & N_2 \arrow[ur, mapsto] & \end{tikzcd}$$

Diamond property implies Church-Rosser



Does β reduction have the diamond property?
No. But this does:

$$\frac{}{M \mapsto_{||} M} \qquad \frac{M \mapsto_{||} M' \quad N \mapsto_{||} N'}{M N \mapsto_{||} M' N'} \qquad \frac{M \mapsto_{||} M' \quad N \mapsto_{||} N'}{(\lambda x. M) N \mapsto_{||} [N'/x] M'}$$

Notice that $M \mapsto^* M'$ iff $M \mapsto_{||}^* M'$

"Programming" w/ the λ-calculus

Multiple Arguments

$\lambda x.\ \lambda y.\ \lambda z.\ x$ ← 1-argument func that returns a 1-argument func that returns a 1-argument func that returns 1st arg

This approach is called currying after Haskell Curry (though he didn't invent it)

Booleans (Church Booleans)

if b then e_1 else e_2
if true then e_1 else $e_2 \equiv e_1$
if false then e_1 else $e_2 \equiv e_2$

Can only use application
Try: if b then e_1 else $e_2 \triangleq b\ e_1\ e_2$
What are true and false?
true e_1 e_2 must $\equiv e_1$ $\Rightarrow$ true $\triangleq \lambda t.\ \lambda f.\ t$
false e_1 e_2 must $\equiv e_2$ $\Rightarrow$ false $\triangleq \lambda t.\ \lambda f.\ f$

Pairs

fst $(x, y) \equiv x$ snd $(x, y) \equiv y$

$(x, y) \triangleq$ $\lambda s.\ s\ x\ y$

Which one do you want?

fst $\triangleq \lambda x.\ \lambda y.\ x$
snd $\triangleq \lambda x.\ \lambda y.\ y$

Recursion

$\lambda x. xx$ – Applies x to itself.
Interesting.

$$(\lambda x. xx)(\lambda x. xx)$$
$$\mapsto (\lambda x. xx)(\lambda x. xx)$$
$$\mapsto (\lambda x. xx)(\lambda x. xx)$$
$$\mapsto \ldots$$

Recursion, part 2

Let's say we have numbers (yeah, those can be programmed in λ too)

$fact \triangleq \lambda n.$ if $n=0$ then 1 else $n * fact\ (n-1)$

oops, not defined

no "let rec" in λ-calculus

Let's take another fact function as an argument

$fact' \triangleq \lambda f\ \lambda n.$ if $n=0$ then 1 else $n * f\ (n-1)$

$fact \triangleq fact'\ fact$

oops, same problem

Fixed point of a function f = value x such that $fx = x$

Fixed point combinator: A function "fix" such that $fix\ f \equiv f\ (fix\ f)$

Let's say we have a "fix"

$fact \triangleq fix\ fact'$

$\equiv fact'\ (fix\ fact')$ $(\equiv fact'\ fact)$

Is this good enough?

$fact'\ (fix\ fact')$

$\equiv \lambda n.$ if $n=0$ then 1 else $n * \underbrace{fact'\ (fix\ fact')}_{\equiv\ fix\ fact'\ \triangleq\ fact}(n-1)$

Looks good

$Y = \lambda f. (\lambda x. f(x\ x))(\lambda x. f(x\ x))$ - most famous fixed pt. comb.

$Y f \equiv_\beta (\lambda x. f(x\ x))(\lambda x. f(x\ x))$

$\equiv_\beta f((\lambda x. f(x\ x))\ (\lambda x. f\ (x\ x)))$

$= f\ (Y f)$ ✓