λ-Calculus.

purely mathematical language.

— Alonzo Church 1930's.

what does it mean to apply function (program) to argument (input).

Can't cover all C.S. no hardware I/O.

f(t) (f t)

— a program that expects input is a λ-abstraction.

λx.x+x. substitution-

(λx.x+x) 3 → 3+3 → (6) ⇐ normal form

(λx.+ x x)

---

but need to be careful when defining substitution.

```
int f(int x) {        f(1).
   { int x = 2   ← can't substitute for this x.
     x+x
   }
     x
}

int y = 1
int f(int x) {                 f(y)
   int y = 2
   return x+y                  should return 3
}
         (y+y) →               but this will return 4.
```

Formal Definitions

a λ-term is inductively defined as follows.

- a variable x, y, z
- if A and B are λ-terms, then (A B) is also
  Application-
- if x is a variable and A is a λ-term then λx.A
  Abstraction.
- constants 3, 4, 5. (extraneous).
  — includes constant functions such as +-*.

λx.(xy) is λ term λx.(x y) (x y) z is a λ-term.

(73) — don't worry about meaning. think mechanically, — like a computer.

Syntactic conventions.

1. applications associate to the left $a\,b\,c = (a\,b)\,c \neq a\,(b\,c)$
2. application binds tighter than abstraction.

$$\lambda x.\, x\,y \;\equiv\; \lambda x.(x\,y) \neq (\lambda x.x)\,y.$$

— the scope of a λ only ends with ( ).

$\lambda x.(x\,y)\,z$ — superfluous. $(\lambda x.\, x\,y)\,z$ — required.

$3 + 4*5$ ← binds tighter. $5-3-1 = 1$ ← assoc left.

Free and bound variables.

$\lambda x.\, M$ — binder; body.

$\lambda x.\, y\, x\, y$ — x bound, y free.

— always relative to a term.

$\lambda y.\, (\lambda x.\, x\,y)\,y$ — y is free in body, x is bound. x, y both bound for entire term.

$\lambda x. \,|(\lambda x.x)\,x\,|$ — bound, free.

$\lambda x.\, x+x$.

Substitution and Beta Reduction.

$$(\lambda x.\,M)\,N \Rightarrow_\beta M[N/x].$$

$M[N/x]$ means replace all free occurrences of x in M with N.

provided: the free variables of N are not bound in M.

— harkens back to C example.

$(\lambda x.\,M)\,N$ is a beta-redex — a place where β-reduction can occur

Examples:

$(\lambda x.\, x\,(\lambda x.x))\,y \rightarrow y\,(\lambda x.x)$

$(\lambda x.\, x\,y)\,(\lambda z.z) \rightarrow (\lambda z.z)\,y \rightarrow y.$ — put a parens here.

$(\lambda x.y)\,z \rightarrow y.$ — nothing to replace — a dummy λ is called a vacuous binder

$(\lambda x\,\lambda y.\; x\,x\,y)\;(\lambda u.u)\;v$ Curried Application.

$\to\; (\lambda y.\; (\lambda u.u)\;(\lambda u.u)\;y)\;v.$

$((\lambda u.u)\;(\lambda u.u))\;v.$

$(\lambda u.u)\;v \;\longrightarrow\; v.$

$(\lambda x.\lambda y.\; y\,x)\;v\;(\lambda u.u)$

$\to (\lambda y.\; y\,v)\;(\lambda u.u) \to (\lambda u.u)\,v \to v.$

$(\lambda y.\lambda x.\; (\lambda y.\; y+x)\;8)\;4.$

$\lambda x.\;(\lambda y.\; y+x)\;8 \;\to\; \lambda x.\; 8+x.$

---

do all terms reduce to β-normal forms?

$(\lambda x.xx)\;(\lambda x.xx)$ $\quad (\overrightarrow{\lambda x.xx})\;(\lambda x.xx).$

Bad examples

1. $(\lambda y.\;\lambda x.\;x\,y)\,v \;\to\; (\lambda x.\;x)\,v \to v.$

x $\quad \lambda x.xv$ already normal form.

---

2. $(\lambda x.\,y\,x)\,(\lambda u.u)\,z \longrightarrow (\lambda x.\,y\,x)\,z \to y\,z$

application associates to the left.

$(y\;\lambda u.u)\;z \;\to$ already normal form.

3. $(\lambda y.\;\lambda y.\;y\,y)\;v \to \lambda y.\;v\,v \longrightarrow$ swi'ed for bound var.

4. $(\lambda y.\;\lambda x.\;x\,y)\;x \;\to\; \lambda x.\;x\,x$!!

↑ free var occurs bound in $\lambda x.\;x\,y.$

what to do? α-conversion.

α-Conversion:

$(\lambda x.\;M) =_\alpha \lambda y.\; M[y/x]$

y is fresh (doesn't appear in M).

— α-convert only bound vars, not free vars!

$(\lambda y.\;\lambda z.\;z\,y)\;x \to \lambda z.\;z\,x.$

another example.

$(\lambda y.\ (\lambda x.(\lambda y.\ y+x)\,8)\ y)\ 1$

$(\lambda y.\ (\lambda x.(\lambda y'.\ y'+x)\,8)\ y)\ 1$

$(\lambda x.(\lambda y'.\ y'+x)\,8)\ y.$

$(\lambda y.\ ((\lambda y'.\ y'+y)\,8)\ y)\ 1.$

$(\lambda y.\ 8+y)\ 1$

$\to 8+1$

Basic Combinators:

$I = \lambda x.\ x$

$K = \lambda x \lambda y.\ x$

$S = \lambda x \lambda y \lambda z.\ x\ z\ (y\ z)$

Axioms

$IA = A$

$KAB = A$

$KIAB = B$

Axiom $\boxed{IA = A}$ $(\lambda x.\ x)A \to A$

$KI = \lambda x \lambda y.\ y.$

$KAB = A$:

$(\lambda x \lambda y.\ x)\ A\ B$

$\to \lambda y.\ A$ but there's no free y's in A else α-convert.

$(\lambda y.\ A)\ B = A$ because y must be a vacuous binder

$((KI)A)B = (KIA)\ B$

$= IB = B.$

$(\lambda x.\lambda y.\ x)(\lambda u.\ u)\ A\ B$

KI

$= (\lambda y.\lambda u.\ u)\ A\ B$

KI

$\to (\lambda u.\ u)\ B \to B.$

SKI

$(\lambda x \lambda y \lambda z.\ x\ z\ (y\ z))\ K\ I$

$(\lambda y \lambda z.\ K\ z\ (y\ z))\ I$

$= \lambda z.\ K\ z\ (I\ z)$

$\to \lambda z.\ z$ by axioms

CBN.

CBV

$\lambda z.\ K\ z\ z$

$\lambda z.\ z$

Class exercise SII.

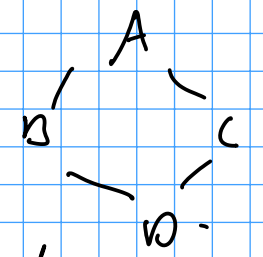
Call by value : CBV - reduce leftmost innermost redex.
Call by Name : reduce leftmost outermost redex.

$I\,(SKI)\;\; I\,(\lambda z.z) \to \lambda z.z.$ CBV

$SKI \to \lambda z.z.$ CBN.

can't have a print statement-

Church - Rosser theorem.



normal forms are unique

$(\lambda x.\, xxxxx)\,(SKI)$ → only reduce once.

$(\lambda x.y)\,((\lambda x.xx)\,(\lambda x.xx)) \longrightarrow$?

Theorem: if a term has a normal form, it can be reached via CBN. ↓ doesn't contradict Church-Rosser.

Booleans.

$KAB = A \quad KIAB = B$-

$T = K = \lambda x\lambda y.x$

$F = KI \;\; \lambda x\lambda y.y$

$Ifelse = \lambda c\lambda a\lambda b.\, c\ a\ b.$

$Ifelse\ K\ A\ B \to A$

$\to (K\ A\ B)$

$Ifelse\ (KI)\ A\ B \to B$-

---

boolean or · $\lambda a\lambda b.\ Ifelse\ a\ T\ b.$

and $\lambda a\lambda b.\ Ifelse\ a\ b\ F$-

Not $\lambda a.\ Ifelse\ a\ F\ T.$

Church Numerals-

$\emptyset = \lambda f\lambda x.\ x$

$1 = \lambda f\lambda x.\ f\ x$

$2 = \lambda f\lambda x\ f\ (f\ x)$

$3 = \lambda f\lambda x.\ f\ (f\ (f\ x))$

⋮

$\emptyset$ who would want to represent them like this?

$\{\}$ 1

$\{\{\}\}$ = 2

$\{\{\{\}\}\}$

← admissible w/r. to λ-calculus.

plus : $\lambda m\lambda n\lambda f\lambda x.\ m\ f\ (n\ f\ x)$

Times : $\lambda m\lambda n\lambda f\lambda x.\ m\ (n\ f)\ x$.

Expt : $\lambda m\lambda n.\,(n\ m)$-

plus 1 1 - $\lambda f\lambda x.\ 1\ f\ (1\ f\ x)$

$\lambda f\lambda x.\ f\ (1\ f\ x) \to \lambda f\lambda x.\ f\ (f\ x)$ -

Times zero two :
$\lambda f \lambda x.$ zero (two f) x. since zero = KI.
→ $\lambda f \lambda x. x$

Data Structures :

Cons pair = $\lambda a \lambda b \lambda c. c\ a\ b$
(cons 2 3) = $\lambda c. c\ 2\ 3.$

Car = $\lambda p. p\ T$
Cdr = $\lambda p. p\ F$

NIL = $F = \lambda x \lambda y. x.$ if p is $\lambda c. c\ a\ b.$

ISNIL = $\lambda p. p\ (\lambda x \lambda y \lambda z. F)\ T$

$(\lambda c. c\ a\ b)\ (\lambda x \lambda y \lambda z. F)\ T \rightarrow$
$(\lambda x \lambda y \lambda z. F)\ a\ b\ T \rightarrow F.$

cons 2 (cons 3 (cons 5 NIL)) 2-3-5-1.

— other structures with nested pairs.
cons x (cons left right)
— consistent with Abstract data type concepts.
— access through interface using functions.

Repetition / Recursion

Need term with the following behaviour:
(FIX m) = m (fix m)

m (m (fix m)).
= m (m (m fix m))

Fix = $\lambda m. (\lambda x. m\ (x\ x))\ (\lambda x. m\ (x\ x))$ ☆

verify that FIX m = m (Fix m)

$(\lambda x. m\ (x\ x))\ (\lambda x. m\ (x\ x)) \rightarrow$
$m\ ((\lambda x. m\ (x\ x))\ (\lambda x. m\ (x\ x)))$
$= m\ (FIX\ m).$

Sum = Fix $\lambda f \lambda n.$ Ifelse (ISNIL n) zero (plus (CAR n) (f (CDR n)))

Sum (cons 2 (cons 3 NIL))
= m (fix m) (cons 2 (cons 3 NIL))
= plus 2 (Fix m) cons 3 NIL
m (fix m) cons 3 NIL.

"plus 3 (Fxm) NiL.
m F x m ,
→ zero -