3-1

Determinism of Language

$$\forall P.\ \forall A_1, A_2.\quad eval(P) = A_1 \wedge eval(P) = A_2 \rightarrow A_1 = A_2$$

All programs have an answer

$$\forall P.\ \exists A.\quad eval(P) = A$$

This program doesn't finish (P)

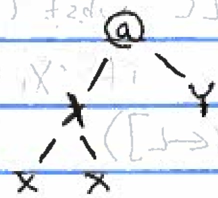
$$\forall M.\quad P \twoheadrightarrow M \Rightarrow \exists N.\quad M \rightarrow N$$

The Lambda Calculus by Alonzo Church

$M, N, L := X$ — variable reference
$\quad | \ (\lambda X.\ M)$ — abstractions (functions)
$\quad | \ (M\ \ N)$ — applications

interface LC
class VarRef impl LC
VarRef (string X)
class Abs impl LC
Abs (string x, LC m)
class App impl LC
App (LC M, LC N)

$((\lambda X. X)\ Y)$



new App (new Abs ("X",
new VarRef ("x")),
new VarRef ("Y"))

$f(x) = x + 5$
$f(10) = ? = 10 + 5 = 15$

1. defined fun $(f(x) = x + 5)$
2. applied fun $f(10)$
3. substituted $x = x + 5$
4. rewrote/transcribe $10 + 5$
5. computed 15

3-2

$((\lambda X. X)\ Y) = Y$ replaced w/ Y

$f(X) = X$
$f(Y) = ?$ $= Y$

$((\lambda X. (X\ X))\ Y) = (Y\ Y)$

$f(x) = g(y) = x$
$f(A)(B)$

$(((\lambda X. (\lambda Y. X))\ A)\ B) \rightarrow ((\lambda Y. A)\ B) \rightarrow A$

β — beta rule

$$(\lambda X. M)\ N \longrightarrow M\,[X \leftarrow N]$$

"Find all Xs inside M and replace with N"

$(\lambda X. \underbrace{(\lambda X. X)}_{M})\ \underbrace{A}_{N} \longrightarrow M\,[X \leftarrow N]$

int f(int x)

$(\lambda X. X)\,[X \leftarrow A] = (\lambda X. A)$

$= (\lambda X. X) \leftarrow$ correct

$(\lambda A. A)$ $(\lambda S. S)$ X
$(\lambda X. S)$ X

$(\lambda Y. (\lambda X. X))\ A$

$M[X \leftarrow N]$ subst (LC M, String X, LC N)

$X_1\ [X_1 \leftarrow N] = N$ class VarRef (String X_2)

$X_2\ [X_1 \leftarrow N] = X_2$ LC subst (String X_1, LC N)

if $X_1 == X_2$, N, o.w. this

$(M_1\ M)\,[X \leftarrow L] = (M[X \leftarrow L]\ N[X \leftarrow L])$

$(\lambda X_1 . M)\ [X_1 \leftarrow N] = (\lambda X_1 . M)$

$(\lambda X_2 . M)\ [X_1 \leftarrow N] = \cancel{(\lambda X_2 . M[X_1 \leftarrow N])}$

$(\lambda X_3 . M[X_2 \leftarrow X_3]\,[X_1 \leftarrow N])$

where X_3 is "new"

capture-avoiding substitution

-3

$\beta \qquad (\lambda X.\, M)\ N \ \rightarrow\ M[X \leftarrow N]$

$\alpha \qquad (\lambda X.\, M) \ \rightarrow\ (\lambda Y.\, M[X \leftarrow Y])$ (Y can't be used in M)

$\eta \qquad (\lambda X.\, (M\ X)) \ \rightarrow\ M$ (X isn't in M)

$(\lambda X.\, (M\ X))\ N \ \rightarrow\ (M\ X)[X \leftarrow N] = (M\ N)$

$M \quad N$

Booleans — OO way of thinking

$\text{true} := \lambda X.\lambda Y.\, X$ int choose (int x, int y)

$\text{false} := \lambda X.\lambda Y.\, Y$

$\text{if}\ C\ T\ F :=$

$(((C\ (\lambda X.\, T))\ (\lambda X.\, F))\ X)$

$\text{if true}\ A\ B \ \rightarrow\ (((\text{true}\ (\lambda X.\, A))\ (\lambda X.\, B))\ X)$

$= (((\lambda X.(\lambda Y.X))\ (\lambda X.A))\ (\lambda X.B))\ X)$

$\rightarrow (((\lambda Y.\, (\lambda X.A))\ (\lambda X.B))\ X)$

$\rightarrow ((\lambda X.\, A)\ X) \ \rightarrow\ A$

Pair $\qquad \text{fst}\ (\text{pair}\ A\ B) = A$

$\qquad \text{snd}\ (\text{pair}\ A\ B) = B$

$\text{pair} := \lambda A.\lambda B.\lambda C.\ ((C\ A)\ B)$

$\text{fst} := \lambda P.\ P\ \text{true}$

$\text{snd} := \lambda P.\ P\ \text{false}$

3-4 / Numbers as OO

n as a LC term does something n times

2 does it 2 times

0 does it 0 times

$\text{zero} := \lambda F. \lambda Z. \; Z$

$\text{one} := \lambda F. \lambda Z. \; F(Z)$

$\text{two} := \lambda F. \lambda Z. \; F(F(Z))$

$Y = \text{two} \Rightarrow F(F(X \; F \; Z))$

$X = \text{two} \Rightarrow F(F(F(F(Z))$

$\text{plus} := \lambda X. \lambda Y. \lambda F. \lambda Z. \left((Y \; F)((X \; F) \; Z)\right)$

$\text{iszero} := \lambda X. \; X \; (\lambda Y. \text{false}) \; (\text{true})$