Programming constructs in λ calculus

$\lambda x . M$ ← function that takes a single argument

multiple arguments

$f(x, y) = \ldots M \ldots$

$f = \lambda x . (\lambda y . M)$

$(f \; a) \; b$

"Currying" after mathematician Haskell Curry

$(\lambda x . \lambda y . x + y) \; 3 \; 2$

$(\lambda y . 3 + y) \; 2$

$3 + 2 = 5$

function specialization

Church Booleans

$tru = \lambda t.\lambda f.\ t$
$fls = \lambda t.\lambda f.\ f$

} functions that take two arguments

$NOT = \lambda x.\ \underbrace{(x\ \underline{fls})\ \underline{tru}}_{\text{apply } x \text{ to fls then tru}}$

tru/fls

if x is "true" it returns the first argument fls
if x is "false" it returns the second tru

---

conditional

if (b) then v else w

$\quad\ \ l \qquad\quad\ m \qquad\ \ n$

$cond = \lambda l.\lambda m.\lambda n.(l\ \ m)\ n$

cond tru v w

$= (\lambda l.\lambda m.\lambda n.\ l\ m\ n)\ tru\ \ v\ \ w$

$\rightarrow (\lambda m.\lambda n.\ tru\ m\ n)\ v\ \ w$

$\rightarrow tru\ \ v\ \ w$ $\qquad\qquad$ fls v w

$\rightarrow v$ $\qquad\qquad\qquad\quad\ \ \rightarrow w$

$$\text{AND} = \lambda p.\, \lambda q.\, (p\ q)\ p$$
$$(p\ q)\ \text{fls}$$

$$\text{OR} = \lambda p.\, \lambda q.\, (p\ p)\ q$$

$$\text{pair} = \lambda f.\, \lambda s.\, \underbrace{\lambda b.\ b\ f\ s}_{\text{"getter"}}$$

$$\text{fst} = \lambda p.\ p\ \text{tru}$$
$$\text{snd} = \lambda p.\ p\ \text{fls}$$

---

create(v, w)
pair

$$\text{pair}\ v\ w \longrightarrow \lambda b.\ b\ v\ w$$

$$\text{fst}\ (\text{pair}\ v\ w)$$
$$\longrightarrow \text{fst}\ (\lambda b.\ b\ v\ w)$$
$$= (\lambda p.\ p\ \text{tru})\ (\lambda b.\ b\ v\ w)$$
$$\longrightarrow (\lambda b.\ b\ v\ w)\ \text{tru}$$
$$\longrightarrow \text{tru}\ v\ w \longrightarrow v$$

Church Numerals

$$c_0 = \lambda \underset{\uparrow \atop \text{successor}}{s} . \lambda \underset{\uparrow \atop \text{zero}}{z} . z$$

$$c_1 = \lambda s. \lambda z. s\ z$$

$$c_2 = \lambda s. \lambda z. s\ (s\ z)$$

$$c_3 = \lambda s. \lambda z. s\ (s\ (s\ z))$$

$$\vdots$$

$$\text{inc} = \lambda n. \underbrace{\lambda s. \lambda z.}_{\text{add } \lambda s \text{ back}}\ \underbrace{s}_{\text{insert } s}\ \underbrace{(n\ s\ z)}_{\text{remove lambdas}}$$

$$\text{inc}\ (\lambda s'. \lambda z'. z')$$

$$\to \lambda s. \lambda z. s\ (\underbrace{(\lambda s'. \lambda z'. z')\ s\ z})$$

$$\to \lambda s. \lambda z. s\ \ z$$

$$\text{plus} = \lambda m.\lambda n.\underbrace{\lambda s.\lambda z.}\,(\underline{m}\ \underline{s})\ (\underbrace{n\,s\,z})$$

$$\underbrace{s\,s\,s\,s}_{m}\ \underbrace{s\,s\,s\ z}_{n}$$

$$\text{times} = \lambda m.\lambda n.\ \underbrace{m\ (\underbrace{\text{plus}\ n})\ \ c_0}_{\lambda z.\ s\,s\,s\,s\ldots z}$$

plus

$$\overbrace{s\ (s\ \ldots\ (s}^{m\ s\text{'s}}\ \ z))$$

n s's

$$\lambda s.\lambda z.\ s(s(s\ \ldots\ z)\ldots)$$

c_0

$$iszero = \lambda m.\ m\ \underbrace{(\lambda x.\ fls)}_{s}\ \underbrace{tru}_{z}$$

$$\lambda s.\ \lambda z.\ \underbrace{s\,(s\,(\cdots z))}_{m}$$

$$z$$

$$\underbrace{(\lambda s.\ \lambda z.\ z)}_{c_0}\ (\lambda x.\ fls)\ tru$$

$$\rightarrow tru$$

$$\underbrace{(\lambda s.\ \lambda z.\ s\ z)}_{c_1}\ \underbrace{(\lambda x.\ fls)}\ \underbrace{tru}$$

$$\rightarrow (\lambda x.\ fls)\ tru$$

$$\rightarrow fls$$

Exponentiation

$$m^n$$

$$exp = \lambda m . \lambda n . \; n \; m$$