

## Programming constructs in $\lambda$ calculus

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

multiple arguments

$$f(x, y) = M.$$

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

$$(f \ a) \ b$$

"Currying" after mathematician  
Haskell Curry

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

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

$$3 + 2 = 5$$

} function  
specialization

## Church Booleans

$\text{tru} = \lambda t. \lambda f. t$   
 $\text{fls} = \lambda t. \lambda f. f$  } functions that take two arguments

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

if  $x$  is "true" it returns the first argument fls

if  $x$  is "false" it returns the second tru

---

condition =

if  $(b)$  then  $v$  else  $w$

$\text{cond} = \lambda l. \lambda m. \lambda n. (l \ m) \ n$

$\text{cond tru } v \ w$

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

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

$\rightarrow \text{tru } v \ w$

$\rightarrow v$

$\text{fls } v \ w$

$\rightarrow w$

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

$(p \ q) \ \text{fls}$

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

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

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

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

$$\text{create}(v, w)$$

$\text{pair}$

$$\text{pair} \ v \ w \rightarrow \lambda b. \underbrace{b \ v \ w}_{\uparrow}$$

$$\text{fst} (\text{pair} \ v \ w)$$

$$\rightarrow \text{fst} (\lambda b. b \ v \ w)$$

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

$$\rightarrow (\lambda b. b \ v \ w) \ \text{tru}$$

$$\rightarrow \text{tru} \ v \ w \rightarrow v$$

## Church Numerals

$$C_0 = \lambda s. \lambda z. z$$

↑      ↑  
successor    zero

$$C_1 = \lambda s. \lambda z. s \ z$$

$$C_2 = \lambda s. \lambda z. s \ (s \ z)$$

$$C_3 = \lambda s. \lambda z. s \ (s \ (s \ z))$$

⋮

$$inc = \lambda n. \lambda s. \lambda z. \underbrace{s \ (n \ s \ z)}$$

↑  
add  
 $\lambda s$  back

↑  
insert  
 $s$

↑  
remove lambdas

$$inc \ (\lambda s'. \lambda z'. z')$$

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

$$\rightarrow \lambda s. \lambda z. s \ z$$

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

---


$$\text{times} = \lambda m. \lambda n. \underbrace{m \ (\text{plus } n)}_{\lambda z. s \ s \ s \ s \dots z} C_0$$

$$\underbrace{s \ (s \dots (s \ z))}_{\substack{m \text{ } s's \\ \uparrow \\ n \text{ } s's}} \quad \uparrow \text{plus}$$

$$\lambda s. \lambda z. s \ (s \ (s \dots z) \dots) \quad \uparrow C_0$$

$$\text{isZero} = \lambda m. m (\lambda x. \text{fls}) \text{tru}$$

$$\lambda s. \lambda z. \underbrace{s(s(\dots z))}_m$$

$$\begin{array}{c} z \\ \left( \lambda s. \lambda z. z \right) (\lambda x. \text{fls}) \text{tru} \\ \underbrace{\hspace{10em}}_{C_0} \\ \rightarrow \text{tru} \end{array}$$

$$\begin{array}{c} \left( \lambda s. \lambda z. s z \right) (\lambda x. \text{fls}) \text{tru} \\ \underbrace{\hspace{10em}}_{C_1} \quad \underbrace{\hspace{5em}} \\ \rightarrow (\lambda x. \text{fls}) \text{tru} \\ \rightarrow \text{fls} \end{array}$$

## Exponentiation

$$m^n$$

$$\exp = \ln m \cdot \ln n \cdot n \cdot m$$