



Constructions in λ -Calculus

- As λ -calculus is a general model of computation, we need to show that it has the expressive power of an ordinary programming language.
- We will show how some common programming language features can be constructed in the λ -calculus.
- Note that these constructions are *embedded within* the calculus; we do not need to add any extra feature to the calculus itself in order to define these.
- The most obvious things missing from the λ -calculus are the usual constants and operators that we associate with an ordinary programming language; this is due to the absence of types.
- We will now see how these constants and operators can be constructed in the λ -calculus.

Booleans

- We can define Booleans as follows:

$$\text{TRUE} = \lambda t. \lambda f. t$$

$$\text{FALSE} = \lambda t. \lambda f. f$$

The conditional if E then E_1 else E_2 can therefore be defined as follows:

$$\text{IF} = \lambda b. \lambda x. \lambda y. (b x y)$$

For example:

$$\begin{aligned} \text{if true then } E_1 \text{ else } E_2 &= \text{IF TRUE } E_1 E_2 \\ &= \underline{(\lambda b. \lambda x. \lambda y. (b x y)) (\lambda t. \lambda f. t) E_1 E_2} \\ &\Rightarrow \underline{(\lambda x. \lambda y. ((\lambda t. \lambda f. t) x y)) E_1 E_2} \\ &\Rightarrow \underline{(\lambda y. ((\lambda t. \lambda f. t) E_1 y)) E_2} \\ &\Rightarrow \underline{(\lambda t. \lambda f. t) E_1 E_2} \\ &\Rightarrow \underline{(\lambda f. E_1) E_2} \\ &\Rightarrow E_1 \end{aligned}$$

Booleans

■ Similarly:

$$\begin{aligned} \text{if false then } E_1 \text{ else } E_2 &= \text{IF FALSE } E_1 E_2 \\ &= \frac{(\lambda b.\lambda x.\lambda y.(b x y)) (\lambda t.\lambda f.f) E_1 E_2}{(\lambda x.\lambda y.((\lambda t.\lambda f.f) x y)) E_1 E_2} \\ &\Rightarrow \frac{(\lambda y.((\lambda t.\lambda f.f) E_1 y)) E_2}{(\lambda t.\lambda f.f) E_1 E_2} \\ &\Rightarrow \frac{(\lambda t.\lambda f.f) E_1 E_2}{(\lambda f.f) E_2} \\ &\Rightarrow E_2 \end{aligned}$$

■ From the definition of the conditional, it is easy to define all the usual logical operations:

$$\begin{aligned} \text{NOT} &= \lambda p.\text{IF } p \text{ FALSE TRUE} \\ \text{AND} &= \lambda p.\lambda q.\text{IF } p \ q \ \text{FALSE} \\ \text{OR} &= \lambda p.\lambda q.\text{IF } p \ \text{TRUE } q \end{aligned}$$



Natural Numbers

- Natural numbers can be encoded as *Church numerals*:

$$n = \lambda s. \lambda z. s^n z$$

- This means:

$$0 = \lambda s. \lambda z. z$$

$$1 = \lambda s. \lambda z. s z$$

$$2 = \lambda s. \lambda z. s (s z)$$

$$3 = \lambda s. \lambda z. s (s (s z))$$

$$4 = \lambda s. \lambda z. s (s (s (s z)))$$

and so on.

Successor

- The following operation adds one to a Church numeral:

$$\text{SUCC} = \lambda n. \lambda s. \lambda z. s (n s z)$$

- For example:

$$\begin{aligned} \text{SUCC } 2 &= \underline{(\lambda n. \lambda s. \lambda z. s (n s z)) (\lambda f. \lambda x. f (f x))} \\ &\Rightarrow \lambda s. \lambda z. s (\underline{(\lambda f. \lambda x. f (f x)) s z}) \\ &\Rightarrow \lambda s. \lambda z. s (\underline{(\lambda x. s (s x)) z}) \\ &\Rightarrow \lambda s. \lambda z. s (s (s z)) \end{aligned}$$

- We can also test if a Church numeral is zero:

$$\text{ISZERO} = \lambda n. n (\lambda x. \text{FALSE}) \text{TRUE}$$

Addition

- The addition of Church numerals is defined as follows:

$$\text{ADD} = \lambda m. \lambda n. \lambda s. \lambda z. m \ s \ (n \ s \ z)$$

- For example:

$$\begin{aligned} \text{ADD } 1 \ 2 &= \underline{(\lambda m. \lambda n. \lambda s. \lambda z. x. m \ s \ (n \ s \ z))} \ (\lambda f. \lambda x. f \ x) \ (\lambda f. \lambda x. f \ (f \ x)) \\ &\Rightarrow \underline{(\lambda n. \lambda s. \lambda z. (\lambda f. \lambda x. f \ x) \ s \ (n \ s \ z))} \ (\lambda f. \lambda x. f \ (f \ x)) \\ &\Rightarrow \lambda s. \lambda z. \underline{(\lambda f. \lambda x. f \ x) \ s} \ ((\lambda f. \lambda x. f \ (f \ x)) \ s \ z) \\ &\Rightarrow \lambda s. \lambda z. \underline{(\lambda x. s \ x)} \ ((\lambda f. \lambda x. f \ (f \ x)) \ s \ z) \\ &\Rightarrow \lambda s. \lambda z. s \ (\underline{(\lambda f. \lambda x. f \ (f \ x))} \ s \ z) \\ &\Rightarrow \lambda s. \lambda z. s \ (\underline{(\lambda x. s \ (s \ x))} \ z) \\ &\Rightarrow \lambda s. \lambda z. s \ (s \ (s \ z)) \end{aligned}$$

Multiplication

- The multiplication of Church numerals is defined as follows:

$$\text{MULT} = \lambda m. \lambda n. \lambda s. \lambda z. m (n s) z$$

- For example:

$$\begin{aligned} \text{MULT } 1 \ 2 &= \frac{(\lambda m. \lambda n. \lambda s. \lambda z. x. m (n s) z) (\lambda f. \lambda x. f x) (\lambda f. \lambda x. f (f x))}{(\lambda n. \lambda s. \lambda z. (\lambda f. \lambda x. f x) (n s) z) (\lambda f. \lambda x. f (f x))} \\ &\Rightarrow \lambda s. \lambda z. \frac{(\lambda f. \lambda x. f x) ((\lambda f. \lambda x. f (f x)) s) z}{(\lambda x. ((\lambda f. \lambda x. f (f x)) s) x) z} \\ &\Rightarrow \lambda s. \lambda z. \frac{((\lambda f. \lambda x. f (f x)) s) z}{(\lambda x. s (s x)) z} \\ &\Rightarrow \lambda s. \lambda z. s (s z) \end{aligned}$$

Pairs

- We can represent ordered pairs as follows:

$$\text{PAIR} = \lambda f.\lambda s.\lambda p.p f s$$

- Given this definition, the corresponding destructors for pairs can be defined as:

$$\text{FST} = \lambda p.p (\lambda x.\lambda y.x)$$

$$\text{SND} = \lambda p.p (\lambda x.\lambda y.y)$$

- For example:

$$\begin{aligned} \text{FST (PAIR } a b) &= \underline{(\lambda p.p (\lambda x.\lambda y.x)) ((\lambda f.\lambda s.\lambda p.p f s) a b)} \\ &\Rightarrow \underline{(\lambda f.\lambda s.\lambda p.p f s) a b (\lambda x.\lambda y.x)} \\ &\Rightarrow \underline{(\lambda s.\lambda p.p a s) b (\lambda x.\lambda y.x)} \\ &\Rightarrow \underline{(\lambda p.p a b) (\lambda x.\lambda y.x)} \\ &\Rightarrow \underline{(\lambda x.\lambda y.x) a b} \\ &\Rightarrow \underline{(\lambda y.a) b} \\ &\Rightarrow a \end{aligned}$$

Lists

- We can represent lists as follows:

$$\begin{aligned}\text{NIL} &= \lambda c.\lambda n.n \\ \text{CONS} &= \lambda h.\lambda t.\lambda c.\lambda n.c\ h\ t\end{aligned}$$

- Given this definition, the corresponding destructors for lists are as follows:

$$\begin{aligned}\text{HEAD} &= \lambda l.l\ (\lambda h.\lambda t.h)\ \text{NIL} \\ \text{TAIL} &= \lambda l.l\ (\lambda h.\lambda t.t)\ \text{NIL}\end{aligned}$$

- For example:

$$\begin{aligned}\text{HEAD (CONS } a\ b) &= \underline{(\lambda l.l\ (\lambda h.\lambda t.h)\ (\lambda c.\lambda n.n))\ ((\lambda h.\lambda t.\lambda c.\lambda n.c\ h\ t)\ a\ b)} \\ &\Rightarrow \underline{((\lambda h.\lambda t.\lambda c.\lambda n.c\ h\ t)\ a\ b)\ (\lambda h.\lambda t.h)\ (\lambda c.\lambda n.n)} \\ &\Rightarrow \underline{((\lambda t.\lambda c.\lambda n.c\ a\ t)\ b)\ (\lambda h.\lambda t.h)\ (\lambda c.\lambda n.n)} \\ &\Rightarrow \underline{(\lambda n.(\lambda h.\lambda t.h)\ a\ b)\ (\lambda c.\lambda n.n)} \\ &\Rightarrow \underline{(\lambda h.\lambda t.h)\ a\ b} \\ &\Rightarrow \underline{(\lambda t.a)\ b} \\ &\Rightarrow a\end{aligned}$$