

Lambda Calculus Cheat Sheet – continued

CSC 131

September 12, 2006

1 Booleans

`true` = $\lambda u. \lambda v. u.$
`false` = $\lambda u. \lambda v. v.$
`cond` = $\lambda u. \lambda v. \lambda w. u v w.$

2 Data structures

`Pair` = $\lambda m. \lambda n. \lambda b. \text{cond } b m n.$
`fst` = $\lambda p. p \text{ true}$
`snd` = $\lambda p. p \text{ false}$

3 Natural numbers and arithmetic

`0` = $\lambda s. \lambda z. z.$
`1` = $\lambda s. \lambda z. s z.$
`2` = $\lambda s. \lambda z. s (s z).$
...
`n` = $\lambda s. \lambda z. s (s (s (\dots (s z) \dots)))$,

where the right hand side of the last line includes `n` occurrences of `s`.

Notice that $\mathbf{n} f x = f^n(x)$

The successor function adds an extra application of successor to a number.

`Succ` = $\lambda n. \lambda s. \lambda z. s (n s z).$

Thus `Succ n = n+1`. With successor, it is easy to define addition, multiplication, and a test for zero:

`Plus` = $\lambda n. \lambda m. \lambda s. \lambda z. m s (n s z).$

`Mult` = $\lambda n. \lambda m. m (\text{Plus } n) 0.$

`isZero` = $\lambda n. n (\lambda x. \text{false}) \text{true}.$

Thus

`Plus n m` = $\lambda s. \lambda z. m s (n s z)$
= $\lambda s. \lambda z. s^n(s^m(z))$
= $\lambda s. \lambda z. s^{n+m}(z)$
= `n+m`.

while

$$\begin{aligned}
\text{Mult } n \ m &= m \ (\text{Plus } n) \ 0 \\
&= (\text{Plus } n)^m \ 0 \\
&= n * m.
\end{aligned}$$

Also, `isZero 0 = true` because the constant function $\lambda x. \text{false}$ is never applied, while `isZero n = false` when $n > 0$ because the constant function will be applied at least once.

Finding the predecessor of a number is tricky. Start by providing a new encoding of numbers as pairs:

$$\begin{aligned}
\text{PZero} &= \langle 0, 0 \rangle = \text{Pair } 0 \ 0 \\
\text{PSucc} &= \lambda n. \text{Pair } (\text{snd } n) \ (\text{Succ}(\text{snd } n))
\end{aligned}$$

Therefore n is encoded as `Pair (n-1) n`.

Now define the predecessor function:

$$\text{Pred} = \lambda n. \text{fst } (n \ \text{PSucc} \ \text{PZero}).$$

The definition of subtraction is easily obtained.

4 Recursion

Define

$$Y = \lambda f. (\lambda x. f(xx))(\lambda x. f(xx))$$

Y is called the fixed-point combinator because for all g , we get $Y \ g = g(Y \ g)$.

Thus $Y \ g$ is a *fixed point* of g for any function g .

Recall the definition of factorial:

$$\text{fact} = \lambda n. \text{cond } (\text{isZero } n) \ 1 \ (\text{Mult } n \ (\text{fact } (\text{Pred } n)))$$

Define F by

$$F = \lambda f. \lambda n. \text{cond } (\text{isZero } n) \ 1 \ (\text{Mult } n \ (f \ (\text{Pred } n)))$$

then $F(\text{fact}) = \text{fact}$. Because fact is a fixed point of F , define

$$\text{fact} = Y \ F.$$

Here are some values of `fact`.

$$\begin{aligned}
\text{fact } 0 &= (F \ (\text{fact})) \ 0 && \text{because fact is a fixed point of F} \\
&= \text{cond } (\text{isZero } 0) \ 1 \ (\text{Mult } 0 \ (\text{fact } (\text{Pred } 0))) && \text{expanding F} \\
&= 1 && \text{by the definition of cond}
\end{aligned}$$

$$\begin{aligned}
\text{fact } 1 &= (F \ (\text{fact})) \ 1 && \text{because fact is a fixed point of F} \\
&= \text{cond } (\text{isZero } 1) \ 1 \ (\text{Mult } 1 \ (\text{fact } (\text{Pred } 1))) && \text{expanding F} \\
&= \text{Mult } 1 \ (\text{fact } (\text{Pred } 1)) && \text{by the definition of cond} \\
&= \text{fact } 0 && \text{by the definition of Mult and Pred} \\
&= 1 && \text{by the above calculation}
\end{aligned}$$

$$\begin{aligned}
\text{fact } 2 &= (F \ (\text{fact})) \ 2 && \text{because fact is a fixed point of F} \\
&= \text{cond } (\text{isZero } 2) \ 1 \ (\text{Mult } 2 \ (\text{fact } (\text{Pred } 2))) && \text{expanding F} \\
&= \text{Mult } 2 \ (\text{fact } (\text{Pred } 2)) && \text{by the definition of cond} \\
&= \text{Mult } 2 \ (\text{fact } 1) && \text{by the definition of Pred} \\
&= \text{Mult } 2 \ 1 && \text{by the above calculation} \\
&= 2 && \text{by the definition of Mult}
\end{aligned}$$