COSE212: Programming Languages

Lecture 14 — Lambda Calculus (2)

Hakjoo Oh
2015 Fall

Programming in the Lambda Calculus

- boolean values
- natural numbers
- pairs
- recursion
- ...

Church Booleans

- Boolean values:

$$\begin{array}{rcl} \text{true} & = & \lambda t.\lambda f.t \\ \text{false} & = & \lambda t.\lambda f.f \end{array}$$

- Conditional test:

$$\text{test} = \lambda l.\lambda m.\lambda n.l\ m\ n$$

- Then,

$$\text{test}\ b\ v\ w = \begin{cases} v & \text{if } b = \text{true} \\ w & \text{if } b = \text{false} \end{cases}$$

- Example:

$$\begin{array}{rcl} \text{test true}\ v\ w & = & (\lambda l.\lambda m.\lambda n.l\ m\ n)\ \text{true}\ v\ w \\ & = & (\lambda m.\lambda n.\text{true}\ m\ n)\ v\ w \\ & = & \text{true}\ v\ w \\ & = & (\lambda t.\lambda f.t)\ v\ w \\ & = & (\lambda f.v)\ w \\ & = & v \end{array}$$

Church Boolean

Logical operators:

- Logical "and":

$$\text{and} = \lambda b.\lambda c.(b\ c\ \text{false})$$

$$\begin{aligned}\text{and true true} &= \text{true}\\ \text{and true false} &= \text{false}\\ \text{and false true} &= \text{false}\\ \text{and false false} &= \text{false}\end{aligned}$$

- (exercise) Logical "or" and "not"?

$$\begin{aligned}\text{or true true} &= \text{true}\\ \text{or true false} &= \text{true}\\ \text{or false true} &= \text{true}\\ \text{or false false} &= \text{false}\end{aligned}$$

$$\begin{aligned}\text{not true} &= \text{false}\\ \text{not false} &= \text{true}\end{aligned}$$

Pairs

$$\begin{array}{rcl} \text{pair } v\ w & : & \text{create a pair of } v \text{ and } w \\ \text{fst } p & : & \text{select the first component of } p \\ \text{snd } p & : & \text{select the second component of } p \end{array}$$

- Definition:

$$\begin{array}{rcl} \text{pair} & = & \lambda f.\lambda s.\lambda b.b\ f\ s \\ \text{fst} & = & \lambda p.p \text{ true} \\ \text{snd} & = & \lambda p.p \text{ false} \end{array}$$

- Example:

$$\begin{array}{rcl} \text{fst (pair } v\ w) & = & \text{fst } ((\lambda f.\lambda s.\lambda b.b\ f\ s)\ v\ w) \\ & = & \text{fst } (\lambda b.b\ v\ w) \\ & = & (\lambda p.p \text{ true})\ (\lambda b.b\ v\ w) \\ & = & (\lambda b.b\ v\ w) \text{ true} \\ & = & \text{true } v\ w \\ & = & v \end{array}$$

Church Numerals

$$\begin{array}{rcl} c_0 & = & \lambda s.\lambda z.z \\ c_1 & = & \lambda s.\lambda z.(s\ z) \\ c_2 & = & \lambda s.\lambda z.s\ (s\ z) \\ & \vdots & \\ c_n & = & \lambda s.\lambda z.s^n\ z \end{array}$$

Church Numerals

- Successor:

$$\text{succ } c_i = c_{i+1}$$

Definition:

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

Example:

$$\begin{array}{rcl} \text{succ } c_0 & = & \lambda n.\lambda s.\lambda z.(s\ (n\ s\ z))\ c_0 \\ & = & \lambda s.\lambda z.(s\ (c_0\ s\ z)) \\ & = & \lambda s.\lambda z.(s\ z) \\ & = & c_1 \end{array}$$

Church Numeral

- Addition:

$$\text{plus } c_n\ c_m = c_{n+m}$$

Definition:

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

Example:

$$\begin{array}{rcl} \text{plus } c_1\ c_2 & = & \lambda s.\lambda z.c_2\ s\ (c_1\ s\ z) \\ & = & \lambda s.\lambda z.c_2\ s\ (s\ z) \\ & = & \lambda s.\lambda z.s\ (s\ (s\ z)) \\ & = & c_3 \end{array}$$

Church Numerals

- Multiplication:

$$\text{mult } c_n \ c_m = c_{n*m}$$

Definition:

$$\text{mult} = \lambda m.\lambda n.m \ (\text{plus } n) \ c_0$$

Example:

$$\begin{aligned}
\text{mult } c_1 \ c_2 &= (\lambda m.\lambda n.m \ (\text{plus } n) \ c_0) \ c_1 \ c_2 \\
&= c_1 \ (\text{plus } c_2) \ c_0 \\
&= (\text{plus } c_2) \ c_0 \\
&= (\lambda m.\lambda s.\lambda z.m \ s \ (c_2 \ s \ z)) \ c_0 \\
&= \lambda s.\lambda z.c_0 \ s \ (c_2 \ s \ z) \\
&= \lambda s.\lambda z.c_2 \ s \ z \\
&= \lambda s.\lambda z.s \ (s \ z)
\end{aligned}$$

- Power (n^m):

$$\text{power} = \lambda m.\lambda n.m \ (\text{mult } n) \ c_1$$

Church Numerals

- Testing zero:

$$\begin{aligned} \text{zero? } c_0 &= \text{true} \\ \text{zero? } c_1 &= \text{false} \end{aligned}$$

Definition:

$$\text{zero?} = \lambda m.m\ (\lambda x.\text{false})\ \text{true}$$

Recursion

- In lambda calculus, recursion is realized via Y-combinator:

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

- For example, the factorial function

$$\text{fact}(n) = \text{if } n = 0 \text{ then } 1 \text{ else } n * \text{fact}(n-1)$$

is encoded by

$$\text{fac} = Y(\lambda f.\lambda n.\text{if } n = 0 \text{ then } 1 \text{ else } n * f(n-1))$$

Recursion

Let $F = \lambda f.\lambda n.\text{if } n = 0 \text{ then } 1 \text{ else } n * f(n-1)$ and $G = \lambda x.F(x\ x)$.

$$
\begin{aligned}
&\text{fac } 1\\
&= (Y\ F)\ 1\\
&= ((\lambda x.F(x\ x))(\lambda x.F(x\ x)))\ 1\\
&= (G\ G)\ 1\\
&= (F\ (G\ G))\ 1\\
&= (\lambda n.\text{if } n = 0 \text{ then } 1 \text{ else } n * (G\ G)(n-1))\ 1\\
&= \text{if } 1 = 0 \text{ then } 1 \text{ else } 1 * (G\ G)(1-1))\\
&= \text{if false then } 1 \text{ else } 1 * (G\ G)(1-1))\\
&= 1 * (G\ G)(1-1)\\
&= 1 * (G\ G)(1-1)\\
&= 1 * (F\ (G\ G))(1-1)\\
&= 1 * (\lambda n.\text{if } n = 0 \text{ then } 1 \text{ else } n * (G\ G)(n-1))(1-1)\\
&= 1 * \text{if } (1-1) = 0 \text{ then } 1 \text{ else } (1-1) * (G\ G)((1-1)-1)\\
&= 1 * 1
\end{aligned}
$$