Kaushik Raj V Nadar
Roll No.: 200499
CS350A: Principles of Programming Languages
October 27, 2022

# Assignment 2

**Question 1.**
Derive the list of free variables in the following $\lambda$ term. Outline your derivation according to the rules given in the notes.

$$(\lambda x.y(xx))(\lambda y.x(yy))(\lambda z.y)$$

**Solution :**
Following are the rules for identifying free variables :

$$\begin{aligned} FV[x] &= \{x\} \\ FV[MN] &= FV[M] \cup FV[N] \\ FV[(\lambda x \cdot M)] &= FV[M] \setminus \{x\} \end{aligned}$$

Intially, we will apply the $\alpha$-renaming rule to simplify the $\lambda$-term,

$$(\lambda x \cdot y(xx))(\lambda y \cdot x(yy))(\lambda z \cdot y) \overset{\alpha}{\equiv} (\lambda a \cdot y(aa))(\lambda b \cdot x(bb))(\lambda z \cdot y)$$

The set of free variables in the given $\lambda$-term is:

$$\begin{aligned} FV((\lambda x \cdot y(xx))(\lambda y \cdot x(yy))(\lambda z \cdot y)) &= FV((\lambda a \cdot y(aa))(\lambda b \cdot x(bb))(\lambda z \cdot y)) \\ &= FV(\lambda a \cdot y(aa)) \cup FV(\lambda b \cdot x(bb)) \cup FV(\lambda z \cdot y) \\ &= (FV(y(aa)) \setminus \{a\}) \cup (FV(x(bb)) \setminus \{b\}) \cup (FV(y) \setminus \{z\}) \\ &= \{y\} \cup \{x\} \cup \{y\} \\ &= \{x, y\} \end{aligned}$$

**Question 2.**
Evaluate the following $\lambda$ expressions using $\alpha$ and $\beta$ reduction rules to obtain the normal form. Please stop the reduction when you first obtain the normal form.

(a) $(\lambda ab \cdot ba)ab$
(b) $(\lambda x \cdot xx)(\lambda a \cdot a)$.
(c) $(\lambda x \cdot xx)(\lambda x \cdot xx)$.

**Solution :**
(a)

$$\begin{aligned}(\lambda ab \cdot ba)ab &\overset{\alpha}{\Rightarrow} (\lambda xy \cdot yx)ab \\ &\overset{\beta}{\Rightarrow} (yx)\,[x := a, y := b] \\ &\equiv ba\end{aligned}$$

(b)

$$\begin{aligned}(\lambda x \cdot xx)(\lambda a \cdot a) &\overset{\beta}{\Rightarrow} (xx)\,[x := (\lambda a \cdot a)] \\ &\equiv (\lambda a \cdot a)(\lambda a \cdot a) \\ &\overset{\alpha}{\Rightarrow} (\lambda a \cdot a)(\lambda b \cdot b) \\ &\overset{\beta}{\Rightarrow} a\,[a := (\lambda b \cdot b)] \\ &\equiv (\lambda b \cdot b)\end{aligned}$$

(c)

$$\begin{aligned}(\lambda x \cdot xx)(\lambda x \cdot xx) &\overset{\alpha}{\Rightarrow} (\lambda a \cdot aa)(\lambda x \cdot xx) \\ &\overset{\beta}{\Rightarrow} (aa)\,[a := (\lambda x \cdot xx)] \\ &\equiv (\lambda x \cdot xx)(\lambda x \cdot xx)\end{aligned}$$

The expression doesn't reduce further and does not terminate.
$\therefore$ The $\lambda$ expression is already in normal form.

**Question 3.**
Construct a $\lambda$ term that does not have a normal form - *i.e.* construct a term which can always be $\beta$ reduced further. Explain why this term has this property in one or two sentences.

**Solution :**
Consider the following $\lambda$ term:

$$(\lambda x \cdot xxx)(\lambda x \cdot xxx)$$

The above $\lambda$ term doesn't have a normal form as the expression does not contain a projection function which can terminate the process of $\beta$ reductions. In this case, performing $\beta$ reductions increases the size of the expression instead of terminating to a normal form.

$$(\lambda x \cdot xxx)(\lambda x \cdot xxx) \overset{\beta}{\Rightarrow} (\lambda x \cdot xxx)(\lambda x \cdot xxx)(\lambda x \cdot xxx)$$

**Question 4.**
Based on the Church representation of Boolean values given in the notes, define the $\lambda$ term which computes the "or" of Boolean values - *i.e.* a term which takes two arguments, and evaluates to the Boolean representation of True if either of them is True, and to False if both of them are False.

**Solution :**
In the Church representation of Boolean values, True is denoted by $T = \lambda xy \cdot x$ and False is denoted by $F = \lambda xy \cdot y$.
The logical or function can be defined as :

$$or \equiv (\lambda xy \cdot x(\lambda uv \cdot u)y) \equiv (\lambda xy \cdot xTy)$$

Following are all the possible cases for the logical or :

$$orFF \equiv ((\lambda xy \cdot x(\lambda uv \cdot u)y)(\lambda xy \cdot y)(\lambda xy \cdot y) \overset{\beta}{\Rightarrow} (\lambda xy \cdot y)(\lambda uv \cdot u)(\lambda xy \cdot y) \Rightarrow (\lambda xy \cdot y) \equiv F$$

$$orFT \equiv ((\lambda xy \cdot x(\lambda uv \cdot u)y)(\lambda xy \cdot y)(\lambda xy \cdot x) \overset{\beta}{\Rightarrow} (\lambda xy \cdot y)(\lambda uv \cdot u)(\lambda xy \cdot x) \Rightarrow (\lambda xy \cdot x) \equiv T$$

$$orTF \equiv ((\lambda xy \cdot x(\lambda uv \cdot u)y)(\lambda xy \cdot x)(\lambda xy \cdot y) \overset{\beta}{\Rightarrow} (\lambda xy \cdot x)(\lambda uv \cdot u)(\lambda xy \cdot y) \Rightarrow (\lambda xy \cdot x) \equiv T$$

$$orTT \equiv ((\lambda xy \cdot x(\lambda uv \cdot u)y)(\lambda xy \cdot x)(\lambda xy \cdot x) \overset{\beta}{\Rightarrow} (\lambda xy \cdot x)(\lambda uv \cdot u)(\lambda xy \cdot x) \Rightarrow (\lambda xy \cdot x) \equiv T$$

**Question 5.**
What is the set of fixed points of the $\lambda$ term $(\lambda x \cdot x)$?

**Solution :**
We know that if $f$ is a fixed point of function $g$ then $f = gf$.
Here, $g = (\lambda x \cdot x)$.

$\therefore$ We need fixed point $f$ such that $f = (\lambda x \cdot x)f$.
$g = (\lambda x \cdot x)$ being an identity function, the above equation is satisfied by any $\lambda$-term $f$.
Hence, the set of fixed points of $(\lambda x \cdot x)$ must contain all possible $\lambda$-terms.

**Question 6.**
Consider an enriched $\lambda$ calculus which has natural numbers available, has a normal if-then-else construct, and has the operators +, - and ==. Using the Y-combinator, define the following recursive function to sum the first n numbers.
sum = $\lambda$ n $\cdot$ if n==0 then 0 else n+(sum n-1).

**Solution :**
Given, sum = ($\lambda$ n $\cdot$ if n==0 then 0 else n+(sum n-1))

Consider sum as the following $\lambda$ term :
M = ($\lambda$ g $\cdot$ $\lambda$ n $\cdot$ if n==0 then 0 else n+(g(n-1)))

Now, given a function g, the $\lambda$ term outputs :
Mg = ($\lambda$n $\cdot$ if n==0 then 0 else n+(g(n-1)))

We know that for any function h, Yh is a fixed point of h (where Y is the Y-combinator).
$\therefore$ By the fixed point property, M(YM) = YM.

Now, M(YM) = ($\lambda$ n $\cdot$ if n==0 then 0 else n+(YM(n-1)))

But, M(YM) = YM,
$\therefore$ we get, YM = ($\lambda$ n $\cdot$ if n==0 then 0 else n+(YM(n-1)))

Hence, sum := YM is the recursive function to sum the first n numbers.