**CS350A**

Course Instructor: Satyadev Nandakumar | Homework #2
Submitted by: Gurbaaz Singh Nandra | October 27, 2022

SOLUTIONS

**1.** The list of free variables in the given $\lambda$ term

$\text{FV}((\lambda x.y(xx))(\lambda y.x(yy))(\lambda z.y))$
$= \text{FV}(\lambda x.y(xx)) \cup \text{FV}(\lambda y.x(yy)) \cup \text{FV}(\lambda z.y)$ using the rule $[\text{FV}(MN) = \text{FV}(M) \cup \text{FV}(N)]$
$= (\text{FV}(y(xx)) \setminus \{x\}) \cup (\text{FV}(x(yy)) \setminus \{y\}) \cup (\text{FV}(y) \setminus \{z\})$ using the rule $[\text{FV}(\lambda x.M) = \text{FV}(M) \setminus \{x\}]$
$= \{y\} \cup \{x\} \cup \{y\}$ using the rule $[\text{FV}(x) = \{x\}]$
$= \{x, y\}$

**2.**

(a) $(\lambda ab.ba)ab$

$$\begin{aligned}(\lambda ab.ba)ab &= (\lambda b.ba[a := a])b \text{ using } \beta \text{ reduction}\\ &= (\lambda b.ba)b \text{ using substitution rule}\\ &= ba[b := b] \text{ using } \beta \text{ reduction}\\ &= ba\end{aligned}$$

(b) $(\lambda x.xx)(\lambda a.a)$

$$\begin{aligned}(\lambda x.xx)(\lambda a.a) &= xx[x := (\lambda a.a)] \text{ using } \beta \text{ reduction}\\ &= (\lambda a.a)(\lambda a.a) \text{ using substitution rule}\\ &= a[a := (\lambda a.a)] \text{ using } \beta \text{ reduction}\\ &= (\lambda a.a) \text{ using substitution rule}\end{aligned}$$

(c) $(\lambda x.xx)(\lambda x.xx)$

$$\begin{aligned}(\lambda x.xx)(\lambda x.xx) &= (\lambda x.xx)(\lambda a.aa) \text{ using } \alpha \text{ renaming}\\ &= xx[x := (\lambda a.aa)] \text{ using } \beta \text{ reduction}\\ &= (\lambda a.aa)(\lambda a.aa) \text{ using substitution rule}\\ &= (\lambda x.xx)(\lambda x.xx) \text{ using } \alpha \text{ renaming}\end{aligned}$$

We get the same $\lambda$-term after $\alpha$ and $\beta$ reduction rules. This $\lambda$-term can always be reduced further and does not have a normal form.

**3.** $M = (\lambda x.xx)(\lambda x.xx)$ is one such example of a *lambda*-term that does not have a normal form - i.e. it that can always be $\beta$ reduced further. If a $\lambda$-term has some normal form then there is atleast one path of a sequence of $\beta$-reductions which must end in a normal form. But there's only one way to reduce the provided $\lambda$-term, as shown below.

$$\begin{aligned}(\lambda x.xx)(\lambda x.xx) &= (\lambda x.xx)(\lambda a.aa) \text{ using } \alpha \text{ renaming}\\ &= xx[x := (\lambda a.aa)] \text{ using } \beta \text{ reduction}\\ &= (\lambda a.aa)(\lambda a.aa) \text{ using substitution rule}\\ &= (\lambda x.xx)(\lambda x.xx) \text{ using } \alpha \text{ renaming}\end{aligned}$$

As we are back to $M$, and it itself is not in normal form, so it does not reduce to any term in normal form.

**4.** We can translate "or" of two Boolean Values $p$ and $q$ as:- return True if $p$ is True else return False. We can simplify this further by stating that return $p$ if $p$ is True, else return $q$ (since if $q$ is True then their "or" is True else False) We know that [if $E$ is true then $M$ else $N$] can be encoded as

$$(\lambda xyz.xyz)$$

Hence, the following $\lambda$-term captures the "or" expression

$$\boxed{(\lambda xy.xxy)pq}$$

**5.** For any $\lambda$-term $M$, $(YM)$ will be a fixed point of $M$ such that $M(YM) = YM$. Then, for $M = (\lambda x.x)$, without loss of generality,

$$\begin{aligned} M(YM) &= (\lambda x.x)YM \\ &= x[x := YM] \text{ [}\beta\text{ Reduction]} \\ &= YM \text{ [Substitution Rule]} \end{aligned}$$

for any $\lambda$-term $Y$. Therefore, set of fixed points of $M$ is the set of all $\lambda$-terms.
**6.** Given

$$sum = \lambda n.\text{ if } n == 0 \text{ then } 0 \text{ else } n + (\text{sum } (n-1))$$

We consider the following $\lambda$ term:

$$(\lambda g.(\lambda n.\text{ if } n == 0 \text{ then } 0 \text{ else } n + (\text{sum } (n-1)))) \tag{1}$$

It is a $\lambda$ term that has "$g$" in place of the recursive call to *sum* recursive function. The above is a lambda term that, given a "function" $g$, outputs the "function"

$$(\lambda n.\text{ if } n == 0 \text{ then } 0 \text{ else } n + (\text{sum } (n-1))) \tag{2}$$

Let us denote the $\lambda$-term in (1) as $G$, and the $\lambda$-term in (2) as $g'$. Thus $G$ maps $g$ to $g'$.
Now we are interested in finding a function "$h$" such that $Gh = h$. We introduce $Y$-combinator here, such that $YG$ is a fixed point of $G$ and $G(YG) = YG$. Hence $G(YG)$ is

$$(\lambda n.\text{ if } n == 0 \text{ then } 0 \text{ else } n + (\text{YG}(n-1))) \tag{3}$$

By the fixed-point property, this is equal to $YG$, and hence $YG$ satisfies the equation

$$YG = (\lambda n.\text{ if } n == 0 \text{ then } 0 \text{ else } n + (\text{YG}(n-1))) \tag{4}$$

$\therefore$ $YG = g$ is our *sum* recursive function.