Gianna Julio

CMPT331 - Theory of Programming Languages

Lambda Calculus

---

1. **Beta-reduce the following expressions to their normal form:**

(a) $(\lambda a \lambda y.ya)(zz)$
$(\lambda y.ya)[(zz)/a]$
$\lambda y.yzz$

(b) $(\lambda x \lambda y.(xy))(\lambda z.y)$
$(\lambda x \lambda a.(xa))(\lambda z.y)$ — *Renaming bound variable y to a*
$(\lambda a.(xa))[(\lambda z.y)/x]$
$(\lambda a.(\lambda z.y)a)$
$(\lambda a.(y))[a/z]$ — *There is no z to substitute here*
$\lambda a.y$

(c) $(\lambda x.(xx))(\lambda y.(yy))$
$(xx)[(\lambda y.(yy))/x]$
$(\lambda y.(yy))(\lambda y.(yy))$
$(yy)[(\lambda y.(yy))/y]$
$(\lambda y.(yy))(\lambda y.(yy))$
$(yy)[(\lambda y.(yy))/y]$
...
self-application - infinite recursion

(d) $K\ x\ y$ — *$K \equiv \lambda ab.a$*
$(\lambda ab.a)xy$
$(\lambda b.a)[x/a]$
$(\lambda bx)y$
$(x)[y/b]$ — *There is no b to substitute here*
x

(e) $S\ K$ — *$S \equiv \lambda xyz.xz(yz)$*
$(\lambda xyz.xz(yz))K$
$(\lambda yz.xz(yz))[K/x]$
$\lambda yz.Kz(yz)$ — *K beta-reduces to only the first expression, z*
$\lambda yz.z \equiv K'$
K'

(f) $(S\ K)\ y\ y\ z$
$(K')yyz$ — *As found in part e, $SK \equiv K'$*
$(\lambda xy.y)yyz$ — *$K' \equiv \lambda xy.y$*
$(\lambda y.y)[y/x]$ — *There is no x to substitute, K' beta-reduces to only second expression, y*
$(\lambda y.y)yz$

$(y)[y/y]$
yz *No more functions to evaluate*

(g) $K'\ y\ y\ z$ *As shown in part f, K' beta-reduces to only the second expression, y*
yz *No more functions to evaluate*

2. **What is the normal form of $(K\ S)(K\ I)$?**
Evaluating (KS):
$(\lambda xy.x)S$ *K beta-reduces to only the first expression, S*
$(\lambda y.x)[S/x]$
$\lambda y.S$ *Given any input, this expression outputs S*
$\lambda y.S$

Evaluating $(\lambda y.S)(KI)$: *Given any input, this expression outputs S*
$(S)[(KI)/y]$ *There is no y to substitute here*
S

3. **Prove the following equivalencies by reducing each side to its normal form.**

(a) $I = S\ K\ K$
$I = (\lambda epz.ez(pz))KK$
$I = (\lambda pz.ez(pz))[K/e]$
$I = (\lambda pz.Kz(pz))K$
$I = (\lambda z.Kz(pz))[K/p]$
$I = (\lambda z.Kz(Kz))$
$I = \lambda z.K(z)(Kz)$ *K beta-reduces to only the first expression, z*
$I = \lambda z.z \equiv I$
$I = I$

(b) $S\ K\ K = K\ I\ I$
$I = KII$ *As shown in part a, $SKK = I$*
$I = (\lambda xy.x)II$ *K beta-reduces to only the first expression, I*
$I = (\lambda y.x)[I/x]$
$I = (\lambda y.I)I$
$I = (I)[I/y]$ *There is no y to substitute here*
$I = I$

4. **Given the definition of Church numerals below, what does (m n) do when m and n are Church numerals? For example** $(\bar{2}\ \bar{3})$**. It may be easier to work out as** $\lambda m \lambda n.(mn)$**. Show your work (or at least an example).**

Church Numerals

Let $\bar{0} = \lambda f\ x\ .\ x$
Let $\bar{1} = \lambda f\ x\ .\ (f\ x)$
Let $\bar{2} = \lambda f\ x\ .\ (f\ (f\ x))$
Let $\bar{3} = \lambda f\ x\ .\ (f\ (f\ (f\ x)))$
.
.
.
.
.
.
.
.
Let $\bar{n} = \lambda f\ x\ .\ (f^n.\ x)$

Let successor = $\overline{succ} = \lambda n\ f\ x\ .\ n\ f\ (f\ x)$

$(\ \overline{succ}\ \bar{0}\ ) \equiv (\lambda n\ f\ x\ .\ n\ f\ (f\ x))\ (\bar{0})$
$[\bar{0}/n]$ in $\lambda\ f\ x\ .\ n\ f\ (f\ x)$
$\rightarrow_\beta \lambda\ f\ x\ .\ \bar{0}\ f\ (f\ x)$

The result is going to be λ f x . *something*.
something is $\bar{0}\ f\ (f\ x)$
$\equiv\ (\lambda f\ x\ .\ x)\ f\ (f\ x)$
[f/f in λ x . x]
$\rightarrow_\beta \lambda\ x\ .\ x\ (f\ x)$
[(f x) / x in x]
$\rightarrow_\beta (f\ x)$

plugging *something* back into the above...
$\lambda f\ x\ .\ (f\ x)$
$\equiv \bar{1}$

m = 2, n = 3

Evaluating $(\bar{2}\ \bar{3})$**:**
$m \equiv \lambda fx.f^m(x)$

$(\bar{m}\ \bar{n}) = (\lambda fx.f^m(x))n$
$(\bar{2}\ \bar{3}) = (\lambda fx.f^2(x))3$
$= (\lambda fx.f(f(x)))3$
$= (\lambda x.f(f(x)))[3/f]$
$= (\lambda x.3(3(x)))$
$=$ 3 applied twice to x
$= 3^2$

Evaluating $(\bar{3}\ \bar{2})$**:**
$n \equiv \lambda fx.f^n(x)$

$(\bar{n}\ \bar{m}) = (\lambda fx.f^n(x))m$
$(\bar{3}\ \bar{2}) = (\lambda fx.f^3(x))2$
$= (\lambda fx.f(f(f(x))))2$
$= (\lambda x.f(f(f(x))))[2/f]$
$= (\lambda x.2(2(2(x))))$
$=$ 2 applied three times to x
$= 2^3$

Applying Church numerals to each other results in exponentiation.