

# Some solutions to the introductory exercises on the $\lambda$ -calculus

November 23rd 2007

Here are solutions to some of the questions from the first  $\lambda$ -calculus exercise sheet of this course. The important ones are questions (3) and (4): it is essential to understand  $\beta$ -reduction properly.

2. To show that if  $x \notin \text{FV}(M)$  then  $M[N/x] \equiv M$ , we work by induction on the structure of  $M$ .

If  $M$  is a variable, it must be some  $y \neq x$ , so by definition  $M[N/x] = y \equiv M$  as required.

If  $M$  is an abstraction  $\lambda y.M'$ , assuming the variable convention we have  $(\lambda y.M')[N/x] = \lambda y.(M'[N/x])$ . The variable  $x$  cannot be free in  $M'$  if it was not free in  $\lambda y.M'$  so by the inductive hypothesis,  $M'[N/x] = M'$ . Thus we can conclude that  $(\lambda y.M')[N/x] = \lambda y.M'$  as required.

If  $M$  is an application  $M_1M_2$  then by definition  $M[N/x] = M_1[N/x]M_2[N/x]$ . The variable  $x$  is not free in either of the  $M_i$  so by the inductive hypothesis each  $M_i[N/x] \equiv M_i$ . This means that  $M[N/x] = M_1M_2 \equiv M$ .

Note that what we do here is to reduce the problem for a big term into the same problem for its sub-terms, and then appeal to the inductive hypothesis; that's how structural inductions always work.

3. (a)  $\lambda x.x$  is in normal form.  
(b)  $x(\lambda x.x)(\lambda y.y)$  is in normal form because application associates left.  
(c)  $(\lambda x.x)(\lambda y.y) \rightarrow_{\beta} \lambda y.y$   
(d)  $(\lambda x.xx)(\lambda y.yy) \rightarrow_{\beta} (\lambda y.yy)(\lambda y.yy)$   
(e)  $x((\lambda x.xy)z)y \rightarrow_{\beta} x(zy)y$   
(f)  $(\lambda x.xy)xy \rightarrow_{\beta} xyy$   
(g)  $(\lambda x.(\lambda y.xy))y \rightarrow_{\beta} \lambda z.yz$ —note the  $\alpha$ -conversion taking place here, to avoid  $y$  being captured as it is substituted in.  
(h)  $(\lambda x.(\lambda y.xy))(\lambda y.y) \rightarrow_{\beta} \lambda y.(\lambda y.y)y$ —no need to  $\alpha$ -convert here, although of course you can if you like.

- (i)  $(\lambda x.(\lambda y.xy)x)z \rightarrow_{\beta} (\lambda x.xx)z$  if we reduce the innermost redex; reducing the outermost redex gives  $(\lambda x.(\lambda y.xy)x)z \rightarrow_{\beta} (\lambda y.zy)z$ . Notice that both of these will reduce to  $zz$  in one more step.
- (j)  $(\lambda x.y)(\lambda x.xx)(\lambda x.xx) \rightarrow_{\beta} y(\lambda x.xx)$ —remember that application associates left.
- (k)  $(\lambda x.y)((\lambda x.xx)(\lambda x.xx)) \rightarrow_{\beta} y$  if we reduce the outermost redex; if we choose the innermost redex, it reduces to itself.
- (l)  $(\lambda x.xxx)(\lambda x.xxx) \rightarrow_{\beta} (\lambda x.xxx)(\lambda x.xxx)(\lambda x.xxx)$ .
- (m)  $(\lambda xy.yx)(\lambda z.z) \rightarrow_{\beta} \lambda y.y(\lambda z.z)$
- (n)  $(\lambda xy.yxy)(\lambda xy.yxy)(\lambda xy.yxy) \rightarrow_{\beta} (\lambda y.y(\lambda xy.yxy)y)(\lambda xy.yxy)$ . After one more step we end up back where we started.
- (o)  $SKK$ , where  $S = \lambda xyz.xz(yz)$  and  $K = \lambda xy.x$ . Here  $SKK \rightarrow_{\beta} (\lambda yz.Kz(yz))K$ . Let's do some more reductions for fun:

$$\begin{aligned} (\lambda yz.Kz(yz))K &\rightarrow_{\beta} (\lambda yz.(\lambda y.z)(yz))K \\ &\rightarrow_{\beta} (\lambda yz.z)K \\ &\rightarrow_{\beta} \lambda z.z \end{aligned}$$

- (p)  $Y_1Y_1M$ , where  $Y_1 = \lambda f.\lambda x.x(ffx)$  and  $M$  is any  $\lambda$ -term. Here are two steps of reduction:

$$\begin{aligned} Y_1Y_1M &\rightarrow_{\beta} (\lambda x.x(Y_1Y_1x))M \\ &\rightarrow_{\beta} M(Y_1Y_1M). \end{aligned}$$

Oh look, it's a fixed point. The term  $Y_1Y_1$  is Turing's fixed-point combinator.

4. The term  $W$  is defined to be

$$\lambda x.\lambda y.xyy.$$

Give a complete analysis of the sequences of reductions that the term  $WWW$  can perform.

We worked through this in a lecture. The overall reduction graph looks like this. Notice that some of the arrows here go both ways.

$$\begin{array}{ccc} WWW & \longleftrightarrow & (\lambda y.Wyy)W \\ \uparrow & & \downarrow \\ (\lambda y.yyy)W & \longleftarrow & (\lambda y.(\lambda z.yzz)y)W \end{array}$$