1 Computation and equivalence in the λ-calculus

1.1 β-reduction and β-equivalence

Recall, a *redex*is a term having the following shape.

$$(\lambda x.M)N$$

i.e. it is the application of a lambda term to any other term.
We gave the following definitions in class.

Definition 1.1 (One-step β-reduction)

$$1.) \quad (\lambda x.M)N \rightarrow_\beta M[x := N]$$

We can extend this definition to allow one redex to be reduced anywhere in a term.

$$\begin{array}{ll} 2.) & M \rightarrow_\beta N \Rightarrow MZ \rightarrow_\beta NZ \\ 3.) & M \rightarrow_\beta N \Rightarrow ZM \rightarrow_\beta ZN \\ 4.) & M \rightarrow_\beta N \Rightarrow \lambda x.M \rightarrow_\beta \lambda x.N \end{array}$$

Note that $\rightarrow_\beta$ can be considered a a relation on terms.

$$\rightarrow_\beta = \{\langle M, N\rangle \mid M \rightarrow_\beta N\}$$

i.e. it is the set of all terms M and N such that $M \rightarrow_\beta N$.

Example 1.1. Note that clauses (2) and (3) allow reduction in either side of an application term, but **not in both** simultaneously. For example, the following two are valid *beta*-reductions by clause (1).

$$\begin{array}{l} (\lambda x.x)y \rightarrow_\beta y \\ (\lambda y.y)w \rightarrow_\beta w \end{array}$$

Clause (2) allows reduction on the left side of an application term, so, if $M = ((\lambda x.x)y)$ and $N = y$, for any term Z we know the following holds.

$$((\lambda x.x)y)Z \rightarrow_\beta yZ$$

If $Z = (\lambda y.y)w$, then,

$$((\lambda x.x)y)((\lambda y.y)w) \rightarrow_\beta y((\lambda y.y)w)$$

Similarly, by (3), if $Z = ((\lambda x.x)y)$ then,

$$((\lambda x.x)y)((\lambda y.y)w) \rightarrow_\beta ((\lambda x.x)y)w$$

It is **not the case** that the following is a reduction satisfying the relation $\rightarrow_\beta$.

$$((\lambda x.x)y)((\lambda y.y)w) \rightarrow_\beta yw$$

Which rule would you apply? The definition of $\rightarrow_\beta$ says one side must stay constant. You can't get there from here ... unless you apply two steps of β-reduction.

$$((\lambda x.x)y)((\lambda y.y)w) \;\rightarrow_\beta\; y((\lambda y.y)w) \;\rightarrow_\beta\; yw$$

or

$$((\lambda x.x)y)((\lambda y.y)w) \;\rightarrow_\beta\; ((\lambda x.x)y)w \;\rightarrow_\beta\; yw$$

Example 1.2. Clause (4) allows reduction in the body of an abstraction. Since $(\lambda x.x)y \rightarrow_\beta y$, by (4), $\lambda z.(\lambda x.x)y \rightarrow_\beta \lambda z.y$.

Definition 1.2 (β^*-reduction) The $\rightarrow^*_\beta$ relation is the reflexive transitive closure of the $\rightarrow_\beta$ relation.

$$\begin{array}{l} M \rightarrow_\beta N \Rightarrow M \rightarrow^*_\beta N \\ M \rightarrow^*_\beta M \\ M \rightarrow^*_\beta N \wedge N \rightarrow^*_\beta L \Rightarrow M \rightarrow^*_\beta L \end{array}$$

Problem 1.1. Beta-reduce the following lambda-terms by hand.

0.) $(\lambda y.y)z$
1.) $w\,((\lambda y.y)z)$
2.) $((\lambda y.y)z)w$
3.) $(\lambda x.\lambda y.x\ y)z$
4.) $((\lambda x.(\lambda y.(y\ x)))y)$
5.) $(\lambda x.((\lambda y.(y\ x))z))$

We can make $\rightarrow^*_\beta$ into an equivalence relation by making it symmetric.

Definition 1.3 (β-equivalence)

1.) $M \rightarrow^*_\beta N \Rightarrow M \equiv_\beta N$
2.) $M \equiv_\beta N \Rightarrow N \equiv_\beta M$
3.) $(M \equiv_\beta N \wedge N \equiv_\beta L) \Rightarrow M \equiv_\beta L$

Clause (1) just says two terms are $\equiv_\beta$ if they are already in the relation $\rightarrow_\beta$. Clause (2) ensures that $\equiv_\beta$ is symmetric. If you think of β-reduction as computation, then $\equiv_\beta$ allows *backward* steps of computation. Clause

(3) ensures $\equiv_\beta$ is transitive. Note that since $\rightarrow_\beta$ is reflexive, by clause (1), so is $\equiv_\beta$. You might think that since $\rightarrow_\beta$ is also transitive, clause (3) here is redundant. It's not, the transitivity in $\rightarrow_\beta$ only includes "forward" steps of computation, it does not allow a sequence containing forward *and* "backward" steps.

1.2 η-Reduction

Next we introduce a different kind of reduction, η-reduction[1].

Definition 1.4. (η-reduction)

$$(\lambda x.Mx) \rightarrow_\eta M \qquad \text{if } x \notin FV(M)$$

Look, because x is not free in M, $M[x := N] = M$ and so

$$(\lambda x.Mx)N \rightarrow_\beta (Mx)[x := N] = M[x := N]x[x := N] = MN$$

The idea of η-reduction is to say, why not just use M instead of $\lambda x.Mx$ since any term (say N) that I apply this function to is just the same as I will get by directly applying M.

An example of η-reduction you already know occurs in languages which allow functions to be defined in Curried form, like Haskell and OCaml. For example, if $f\,x\,y = x + y$, we can simply write $g = f\,5$ instead of $g\,y = f\,5\,y$. This is a form of η-reduction.

Note that $\rightarrow_\eta$ is a relation just like $\rightarrow_\beta$,

$$\rightarrow_\eta = \{\langle M, N\rangle \mid M \rightarrow_\eta N\}$$

Definition 1.5 ($\rightarrow_{\{\beta,\eta\}}$) Remember, when we consider $\rightarrow_\beta$ and $\rightarrow_\eta$ as relations, we are thinking of them as sets of pairs of terms. We can union sets together.

$$\rightarrow_{\{\beta,\eta\}} \stackrel{\text{def}}{=} \rightarrow_\beta \cup \rightarrow_\eta$$

1.3 Recursion in the lambda Calculus

Recursion in the λ-calculus is performed with so-called *fixedpoint combinators.* A *combinator* is just a term with no free variables, also called a *closed term.* defined as follows:

$$Y \stackrel{\text{def}}{=} \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$$

[1] η is the Greek letter named "eta". Also, η reduction is described in chapter 6 (pg 167) of the Schmidt text.

Definition 1.6 (The Fixedpoint property) A term (say F) has the fixed-point property if, for every term M,

$$FM \equiv_\beta M(FM)$$

Theorem 1.1. We prove that Y has the fixed point property, that is, for every term M

$$YM \equiv_\beta M(YM)$$

The proof is as follows:

$$\begin{aligned} YM = \quad & (\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)))M \\ & \rightarrow_\beta (\lambda x.M(xx))(\lambda x.M(xx)) \\ & \rightarrow_\beta M((\lambda x.M(xx))(\lambda x.M(xx))) \\ & \leftarrow_\beta M((\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)))M) \\ & = M(YM) \end{aligned}$$

□

Now, recall that application associates to the left, so xyz means $(xy)z$. And so, regarding η-reduction in particular, and this will be useful to solve the next problem.

$$\lambda y.NNy \rightarrow_\eta NN$$

If N is something complicated like $(\lambda x.M(\lambda y.xxy))$, then

$$\lambda y.(\lambda x.M(\lambda y.xxy))(\lambda x.M(\lambda y.xxy))y \rightarrow_\eta (\lambda x.M(\lambda y.xxy))(\lambda x.M(\lambda y.xxy))$$

Definition 1.7. The Z combinator is defined as follows:

$$Z \stackrel{\text{def}}{=} \lambda f.(\lambda x.f(\lambda y.xxy))(\lambda x.f(\lambda y.xxy))$$

Problem 1.2. Prove[2] that, like Y, Z enjoys the fixed-point property when we consider the equivalence $\equiv_{\beta\eta}$ (the relation that allows us to use both β and η reduction.) You must prove that for any term M, that

$$ZM \equiv_{\beta\eta} M(ZM)$$

[2]You will need to use one instance of η-reduction to get the proof to go through.