Computer Science 131, Fall 2002
Assignment 11: λ-Calculus

Out: Wednesday, April 10

Due: Wednesday, April 17

1 Predecessor (30%)

For practice working with λ-calculus, show that the definition of predecessor that you saw in class is correct:

1. **Prove that** $\ulcorner 0 \urcorner\, M_1\, M_2 \longleftrightarrow_\beta^* M_2$ **and that** $\ulcorner n{+}1 \urcorner\, M_1\, M_2 \longleftrightarrow_\beta^* M_1(\ulcorner n \urcorner\, M_1\, M_2)$**.**

   - $\ulcorner 0 \urcorner\, M_1\, M_2$
     $= (\lambda f.\lambda b.b)\, M_1\, M_2 \rightarrow_\beta (\lambda b.b)M_2 \rightarrow_\beta M_2$
   - $\ulcorner n+1 \urcorner\, M_1\, M_2$
     $= (\lambda f.\lambda b.f^{(n+1)}(b))\, M_1\, M_2 \rightarrow_\beta (\lambda b.M_1^{(n+1)}b)M_2 \rightarrow_\beta M_1^{(n+1)}(M_2)$
     $= M_1(M_1^{(n)}(M_2)) \leftarrow_\beta M_1((\lambda b.(M_1)^{(n)}b)M_2) \leftarrow_\beta M_1((\lambda f.\lambda b.f^{(n)}(b))\, M_1\, M_2) = M_1(\ulcorner n \urcorner\, M_1\, M_2)$

2. **In class the predecessor function was defined as**

$$\begin{aligned} \mathbf{pred}' &= \lambda m.m\ \langle \ulcorner 0 \urcorner, \ulcorner 0 \urcorner \rangle\ (\lambda p.\langle \mathbf{snd}\ p, \mathbf{succ}(\mathbf{snd}\ p)\rangle) \\ \mathbf{pred} &= \lambda n.\mathbf{fst}\ (\mathbf{pred}'\ n) \end{aligned}$$

   Prove that $\forall m \geq 0.\ \mathbf{pred}' \ulcorner m{+}1 \urcorner \longleftrightarrow_\beta^* \langle \ulcorner m \urcorner, \ulcorner m{+}1 \urcorner \rangle$**.**

   To show: $\forall m \geq 0.\ \mathbf{pred}' \ulcorner m + 1 \urcorner \longleftrightarrow_\beta^* \langle \ulcorner m \urcorner, \ulcorner m + 1 \urcorner \rangle$.

   By induction.

   - Case: $m = 0$. Then
     $\mathbf{pred}' \ulcorner 1 \urcorner$
     $= (\lambda m.m\ \langle \ulcorner 0 \urcorner, \ulcorner 0 \urcorner \rangle\ (\lambda p.\langle \mathbf{snd}\ p, \mathbf{succ}(\mathbf{snd}\ p)\rangle))(\ulcorner 1 \urcorner)$
     $\rightarrow_\beta \ulcorner 1 \urcorner\ \langle \ulcorner 0 \urcorner, \ulcorner 0 \urcorner \rangle\ (\lambda p.\langle \mathbf{snd}\ p, \mathbf{succ}(\mathbf{snd}\ p)\rangle))$
     $\rightarrow_\beta{}^2\ (\lambda p.\langle \mathbf{snd}\ p, \mathbf{succ}(\mathbf{snd}\ p)\rangle))\langle \ulcorner 0 \urcorner, \ulcorner 0 \urcorner \rangle$
     $\rightarrow_\beta{}^*\ \langle \ulcorner 0 \urcorner, \ulcorner 1 \urcorner \rangle$.

- Inductive step: Assume $\mathbf{pred'}\ulcorner m+1 \urcorner \longleftrightarrow^*_\beta \langle \ulcorner m \urcorner, \ulcorner m+1 \urcorner\rangle$; we need to show that $\mathbf{pred'}\ulcorner m+2 \urcorner \longleftrightarrow^*_\beta \langle \ulcorner m+1 \urcorner, \ulcorner m+2 \urcorner\rangle$.
Then
$\mathbf{pred'}\ulcorner m+2 \urcorner \rightarrow_\beta \ulcorner m+2 \urcorner\ \langle \ulcorner 0 \urcorner, \ulcorner 0 \urcorner\rangle\ (\lambda p.\langle \mathbf{snd}\ p, \mathbf{succ}(\mathbf{snd}\ p)\rangle)$.
By the previous part, this
$\rightarrow_\beta{}^* (\lambda p.\langle \mathbf{snd}\ p, \mathbf{succ}(\mathbf{snd}\ p)\rangle)(\ulcorner m+1 \urcorner\ \langle \ulcorner 0 \urcorner, \ulcorner 0 \urcorner\rangle\ (\lambda p.\langle \mathbf{snd}\ p, \mathbf{succ}(\mathbf{snd}\ p)\rangle))$
$\leftarrow_\beta (\lambda p.\langle \mathbf{snd}\ p, \mathbf{succ}(\mathbf{snd}\ p)\rangle)(\mathbf{pred'}\ \ulcorner m+1 \urcorner)$. By the inductive hypothesis, this $\longleftrightarrow_\beta (\lambda p.\langle \mathbf{snd}\ p, \mathbf{succ}(\mathbf{snd}\ p)\rangle)\langle \ulcorner m \urcorner, \ulcorner m+1 \urcorner\rangle \rightarrow_\beta{}^* \langle \ulcorner m+1 \urcorner, \ulcorner m+2 \urcorner\rangle$.

3. **Prove that $\forall m \geq 0.\ \mathbf{pred}\ulcorner m+1 \urcorner \longleftrightarrow^*_\beta \ulcorner m \urcorner$.**

$\mathbf{pred}\ulcorner m+1 \urcorner \rightarrow_\beta \mathbf{fst}\ (\mathbf{pred'}\ \ulcorner m+1 \urcorner) \longleftrightarrow_\beta{}^* \mathbf{fst}\ \langle \ulcorner m \urcorner, \ulcorner m+1 \urcorner\rangle \longleftrightarrow_\beta{}^* \ulcorner m \urcorner$.

2 Lambda Calculus Encodings (50%)

For this problem you will devise an encoding for lists within the untyped λ-calculus. Recall that a list is either empty or it has a head (first element) and a tail (a list containing the rest of the elements, possibly empty).

1. **Find a lambda term nil to represent the empty list and a lambda term cons such that cons $M\ N$ represents the non-empty list with head M and with tail N. Then a finite list $[M_1, \ldots, M_n]$ would be represented as cons M_1 (cons M_2 $(\cdots$ (cons M_n Explain how to define the following functions for your encoding:**

   - **The function isnil that returns tt if given nil and returns ff if given a non-empty list.**
   - **The function hd that returns the head of a non-empty list.**
   - **The function tl that returns the tail of a non-empty list.**

   Verify that

$$\begin{array}{l}\mathbf{isnil\ \ nil} \longleftrightarrow^*_\beta \mathbf{tt}\\ \mathbf{isnil}\ (\mathbf{cons}\, M_1\ \ M_2) \longleftrightarrow^*_\beta \mathbf{ff}\\ \mathbf{hd}\ (\mathbf{cons}\, M_1\ \ M_2) \longleftrightarrow^*_\beta M_1\\ \mathbf{tl}\ (\mathbf{cons}\, M_1\ \ M_2) \longleftrightarrow^*_\beta M_2\end{array}$$

   There are many, many possible encodings; here's a fairly direct encoding using pairing and booleans as defined in class. A list a pair whose first element is a boolean saying whether it is nil or not; if not, the second element is a pair containing the head and tail of the list:

$$\begin{array}{l}\mathbf{nil} := \langle tt, M_0\rangle\\ \mathbf{cons} := \lambda h.\lambda t.\langle f\!f, \langle h, t\rangle\rangle\\ \mathbf{isnil} := \lambda l.(\mathbf{fst}\ l)\\ \mathbf{hd} := \lambda l.(\mathbf{fst}\ (\mathbf{snd}\ l))\\ \mathbf{tl} := \lambda l.(\mathbf{snd}\ (\mathbf{snd}\ l))\end{array}$$

   where M_0 in the definition of **nil** is arbitrary and can be any term in the lambda-calculus.

Here's another; the idea is that lists are "things that one can do `case`-like operation on; if M_1 is a term representing a list, then $M_1\ M_2\ (\lambda x.\lambda y.M_3)$ will act like `case` M_1 `of nil =>` M_2 `| x::y =>` M_3. That is, a list value is a lambda calculus that takes two arguments, and returns the first of the two if the the list value is nil, and otherwise returns the second argument applied to the head and tail of the list.

$$
\begin{aligned}
&\mathbf{nil} := (\lambda n.\lambda c.n)\\
&\mathbf{cons} := \lambda h.\lambda t.(\lambda n.\lambda c.c\ h\ t)\\
&\mathbf{isnil} := \lambda l.l\ tt\ (\lambda h.\lambda t.\mathit{ff})\\
&\mathbf{hd} := \lambda l.l\ M_0\ (\lambda h.\lambda t.h)\\
&\mathbf{tl} := \lambda l.l\ M_0\ (\lambda h.\lambda t.t)
\end{aligned}
$$

where M_0 is arbitrary. Then

- $\mathbf{isnil}\ \mathbf{nil} \to_\beta{}^* (\lambda n.\lambda c.n)\ tt\ (\lambda h.\lambda t.\mathit{ff}) \to_\beta{}^2\ tt.$
- $\mathbf{isnil}\,(\mathbf{cons}\,M_1\ M_2) \to_\beta{}^* \mathbf{isnil}(\lambda n.\lambda c.c\,M_1\ M_2) \to_\beta (\lambda n.\lambda c.c\,M_1\ M_2)\ tt\ (\lambda h.\lambda t.\mathit{ff}) \to_\beta{}^2 (\lambda h.\lambda t.\mathit{ff})\ M_1\ M_2 \to_\beta{}^2\ \mathit{ff}.$
- $\mathbf{hd}\,(\mathbf{cons}\,M_1\ M_2) \to_\beta{}^* \mathbf{hd}(\lambda n.\lambda c.c\,M_1\ M_2) \to_\beta (\lambda n.\lambda c.c\,M_1\ M_2)M_0(\lambda h.\lambda t.h) \to_\beta{}^2 (\lambda h.\lambda t.h)\ M_1\ M_2 \to_\beta{}^2\ M_1.$
- $\mathbf{tl}\ (\mathbf{cons}\,M_1\ M_2) \to_\beta{}^* \mathbf{tl}(\lambda n.\lambda c.c\ M_1\ M_2) \to_\beta (\lambda n.\lambda c.c\ M_1\ M_2)M_0(\lambda h.\lambda t.t) \to_\beta{}^2 (\lambda h.\lambda t.t)\ M_1\ M_2 \to_\beta{}^2\ M_2.$

2. **Find a lambda term length such that length $M \longleftrightarrow_\beta^* \ulcorner n \urcorner$ when M represents a finite list with n elements.**

   Let $\mathbf{length} := Y(\lambda f.\lambda l.(\mathbf{isnil}\ \ l)\ulcorner 0\urcorner(\mathbf{succ}(f(\mathbf{tl}\ \ l)))).$

3. **Find a lambda term zeros that represents an infinite list whose elements are all $\ulcorner 0 \urcorner$. Prove that it satisfies $\mathbf{hd}(\mathbf{tl}^{(n)}\,\mathbf{zeros}) \longleftrightarrow_\beta^* \ulcorner 0\urcorner$ for every $n \geq 0$.**

   Observing that if we had such at term it would satisfy the equation

   $$\mathbf{zeros} = \mathbf{cons}\ \ulcorner 0\urcorner\ \mathbf{zeros}$$

   we get the answer $\mathbf{zeros} := Y(\lambda l.\,\mathbf{cons}\ \ulcorner 0\urcorner\ l)$.

   To show that this is correct, note that $\mathbf{zeros} \longleftrightarrow_\beta ((\lambda l.\,\mathbf{cons}\ \ulcorner 0\urcorner\ l)\mathbf{zeros}) \longleftrightarrow_\beta \mathbf{cons}\,\ulcorner 0\urcorner\,\mathbf{zeros}$. Thus $\mathbf{hd}\ \mathbf{zeros} \longleftrightarrow_\beta \mathbf{hd}(\mathbf{cons}\,\ulcorner 0\urcorner\,\mathbf{zeros}) \longleftrightarrow_\beta \ulcorner 0\urcorner$ and $\mathbf{tl}\ \mathbf{zeros} \longleftrightarrow_\beta \mathbf{tl}(\mathbf{cons}\ \ulcorner 0\urcorner\ \mathbf{zeros}) \longleftrightarrow_\beta \mathbf{zeros}$. Hence $\mathbf{hd}(\mathbf{tl}^{(n)}\,\mathbf{zeros}) \longleftrightarrow_\beta^* \ulcorner 0\urcorner$ for every $n \geq 0$.

   The definition $Y(\mathbf{cons}\,\ulcorner 0\urcorner)$ also works and is simpler, while more convoluted definitions are possible as well.

3 Fixed Points (20%)

In class you saw the Y combinator for finding fixed points, which satisfies $YM \longleftrightarrow_\beta^* M(YM)$ for any term M. There are actually many terms other than Y which compute fixed points.

1. **The Turing fixed-point combinator Θ is defined to be**

$$(\lambda x.\lambda y.y(xxy))(\lambda x.\lambda y.y(xxy))$$

   Prove that for any term M we have not just $\Theta M \longleftrightarrow_\beta^* M(\Theta M)$, but $\Theta M \rightarrow_\beta^* M(\Theta M)$.

   Put $D := (\lambda x.\lambda y.y(xxy))$, so that $\Theta = DD$. Then $\Theta M = (DD)M = ((\lambda x.\lambda y.y(xxy))D)(M) \rightarrow_\beta (\lambda.y.y(DDy))(M) \rightarrow_\beta M(DDM) = M(\Theta M)$.

2. **For curried functions with many arguments, it is common to write terms leaving out all but the first lambda, for example abbreviating $\lambda x.\lambda y.\lambda z.\lambda w.M$ as $\lambda xyzw.M$. Define**

$$\begin{aligned} T &:= \lambda abcdefghijklmnopqstuvwxyzr.r(thisisafixedpointcombinator) \\ U &:= TTTTTTTTTTTTTTTTTTTTTTTTTT \end{aligned}$$

   Show that U satisfies $UM \longleftrightarrow_\beta^* M(UM)$ for every term M.

   T is a curried function that takes twenty-six arguments, and U is an application of T to twenty-five arguments. Therefore, apply twenty-five beta reductions to substitute in these arguments (and hence replacing every variable here except r with T) we have

$$U \rightarrow_\beta^* \lambda r.r(TTTTTTTTTTTTTTTTTTTTTTTTTTr) = \lambda r.r(Ur)$$

   Hence $UM \rightarrow_\beta^* (\lambda r.r(Ur))M \rightarrow_\beta M(UM)$.