COMP6463: Untyped λ-Calculus Assignment

4 September 2013

Due: 9am, Monday, 16 September 2013 **20 marks total**

Submit your assignment electronically, preferably by e-mailing a PDF to `michael.norrish@nicta.com.au`. I'm perfectly happy to receive a PDF scan of hand-written work.

1. Write down four rules for *strict* or *applicative*, left-to-right order evaluation of λ-calculus terms in the style of the rules given for normal order evaluation in the lectures. In other words, define a binary relation $\rightarrow_a$ that ensures that all functions and arguments are evaluated before substitutions are performed. Here's one rule for free:

$$\frac{N \rightarrow_a N' \quad M \text{ is in } \beta\text{-normal form}}{M\ N \rightarrow_a M\ N'}$$

   This rule guarantees that evaluation moves from left to right because N is not allowed to be evaluated before M has been reduced to normal form. You need to write three more rules. **[5 marks]**

2. Prove the following "inequalities" by giving a derivation of $\boldsymbol{\lambda} \vdash X = Y$ for arbitrary terms X and Y when the inequality is assumed to be an equation:

   (a) $K\ I\ \#\ K$ **[2 marks]**

   (b) $x\ \#\ y$ (with x and y distinct λ-calculus variables) **[3 marks]**

3. Define the λ-term corresponding to the following recursive function f *without* using the Y (or any other recursion) combinator:

$$
\begin{aligned}
f(0) &= 3 \\
f(n+1) &= 2 \times f(n) + 3
\end{aligned}
$$

Use primitive recursion and the λ-terms corresponding to 2, 3, $+$ and $\times$. *Don't* solve the recurrence relation and define the function with a closed form using exponentiation. **[5 marks]**

4. Again, using primitive recursion, define a function on lists that adds 1 to each element of a list. Thus:

$$
\begin{aligned}
f([]) &= [] \\
f(h :: t) &= (h+1) :: f(t)
\end{aligned}
$$

Recall that $h :: t$ is the list consisting of the element h followed by the list t (a "cons" cell). **[5 marks]**