

Exercise Sheet 4

For these exercises, we use an applied lambda calculus with booleans, integers and cons-pairs as primitives. The δ -conversion rules for the cons-pairs are the following:

$$\begin{aligned}\text{null nil} &= \text{true} \\ \text{null (cons } x \ y) &= \text{false} \\ \text{car (cons } x \ y) &= x \\ \text{cdr (cons } x \ y) &= y\end{aligned}$$

We will utilise the normal infix notation for arithmetic operators, e.g. $n + (m * 2)$, instead of the official notation $+ n (* m 2)$.

Attempt as many questions as you can during the exercise class, and work on the remainder at home. Hand in your solutions to the tutor in the Friday exercise class, on 19 Feb, 2010.

Exercise 1: Recursive functions

Recall the definition of the **Y** combinator:

$$\mathbf{Y} t = (\lambda x. t (x x)) (\lambda x. t (x x))$$

and the fact that beta-reduction gives $\mathbf{Y} t \rightarrow_{\beta} t (\mathbf{Y} t)$. We have dubbed this reduction the “DNA” rule.

- a. Consider the function:

$$N = \lambda f. \lambda n. \text{cons } n (f (n + 1))$$

Carry out a few beta reductions for the term $\mathbf{Y} N 1$ and describe its computational effect. To keep your derivation manageable, it can be useful to immediately apply the appropriate δ -conversion rule to any primitive operation that is applied only to constants, for example $1 + 2 \rightarrow_{\delta} 3$.

Also, you can try to perform multiple β -reductions simultaneously when the function being applied takes several arguments. Since we can think of N as taking 2 arguments, we can consume 2 arguments in one step, as in:

$$N (\mathbf{Y} N) 3 \longrightarrow \text{cons } 3 (\mathbf{Y} N 4)$$

- b. Translate the following Java-like pseudocode into a recursive lambda calculus function:

```
int sum (List<int> l) {
  int result = 0;
  while (!l.isEmpty()) {
    result += l.head();
    l = l.tail();
  }
  return result;
}
```

Exercise 2: Infinite data structures

- a. Consider the following functions:

$$\begin{aligned}A &= \lambda f. \lambda y. \lambda z. \text{cons } ((\text{car } y) + (\text{car } z)) (f (\text{cdr } y) (\text{cdr } z)) \\ F &= \lambda l. \text{cons } 1 (\text{cons } 1 (\mathbf{Y} A l (\text{cdr } l)))\end{aligned}$$

Here are the first couple of reduction steps for the term $\mathbf{Y} F$, always choosing the outermost redex.

$$\begin{aligned}\mathbf{Y} F &\longrightarrow F(\mathbf{Y} F) \\ &\longrightarrow \text{cons } 1 (\text{cons } 1 (\mathbf{Y} A (\mathbf{Y} F) (\text{cdr } (\mathbf{Y} F))))\end{aligned}$$

Carry out enough beta reductions for the remaining term $(\mathbf{Y} A (\mathbf{Y} F) (\text{cdr } (\mathbf{Y} F)))$ until you can see a couple of elements of this list. Can you guess what $\mathbf{Y} F$ reduces to, from these observations?

- b. Define a function called *series* which takes arguments k and n , and generates an infinite series of the integers $n, n + k, n + 2k, \dots$ as an infinite list.
- c. Define a function called *common*, which takes as arguments two infinite lists x and y of integers in ascending order, and produces another infinite list of all the integers that occur in both x and y .
- d. Describe the effect of the expression:

$$\text{common } (\text{series } 2 \ 0) (\text{series } 3 \ 0)$$