TP 1

17/11/2009

The object of this TP is the *pure untyped lambda calculus*.

Exercise 1 (Warm-up)**.** Let $\mathcal{X}$ be a countably infinite set of *variables*. We denote by $x, y, \ldots$ members of $\mathcal{X}$. Then *lambda terms* are defined inductively as follows:

$$\begin{array}{lcll} t & ::= & & \textit{lambda-terms:} \\ & & x & \textit{variable} \\ & & \lambda x.t & \textit{abstraction} \\ & & t\,t & \textit{application} \end{array}$$

1. Which are the bound and which are the free variables in $((\lambda x.x)x)y$?
2. Is $((\lambda x.x)\ x)\ y =_\alpha ((\lambda z.z)\ z)\ y$?
3. Is $\lambda x.(x\ y) =_\alpha \lambda y.(y\ x)$?
4. Is $\lambda x.(x\ y) =_\alpha \lambda y.(y\ y)$?

Exercise 2 (Programming with booleans)**.**

1. Give three lambda terms denoted *true*, *false* and *ifthenelse* such that

   $$\textit{ifthenelse}\ \textit{true}\ u\ v \to^* u$$

   and such that

   $$\textit{ifthenelse}\ \textit{false}\ u\ v \to^* v.$$

2. Give lambda terms to performs the logical operations *not*, *and* and *or* on the above representation of booleans.

Exercise 3 (Programming with pairs)**.**

Define lambda terms *pair*, *fst* and *snd* such that

$$\textit{fst}(\textit{pair}\ u\ v) \to^* u$$

and such that

$$\textit{snd}(\textit{pair}\ u\ v) \to^* v.$$

Define a “function” which computes the logical or of a pair of booleans using the above representations of pairs and respectively of booleans.

Exercise 4 (Programming with naturals). 1. We define the terms

$$\begin{array}{lcl} c_0 & = & \lambda s.\lambda z.z \\ c_1 & = & \lambda s.\lambda z.(s\ z) \\ c_2 & = & \lambda s.\lambda z.(s\ (s\ z)) \\ \ldots & & \end{array}$$

which will represent the natural numbers. Define the term *succ* such that

$$\mathit{succ}\, c_i \to^* c_{i+1}.$$

2. Define the term *plus* such that

$$\mathit{plus}\, c_i\ c_j \to c_{i+j}.$$

3. Define the term *mult* such that

$$\mathit{mult}\, c_i\ c_j \to c_{i\times j}.$$

4. Define the term *iszero* which returns *true* if its argument is c_0 and *false* otherwise.

5. Define the term *pred* which returns c_0 if its argument is c_0 and c_{i-1} if its argument is c_i for some $i > 0$.

Exercise 5 (Emulating recursive functions). 1. A lambda-term with no free variables is called a *combinator*. One of the most “famous” combinators is:

$$Y = \lambda f.(\lambda x.(f\ (x\ x))\ (\lambda x.f\ (x\ x)))$$

which is a fixpoint operator (see next). Prove that

$$Y\ g = g\ (Y\ g).$$

2. Define a lambda-term *fact* which returns the factorial of its argument.

3. Define a lambda-term *isprime* which returns true iff its argument is a prime number.

4. Explain why the Y-combinator cannot be used in a call-by-value setting.