Theory of Computation Homework 5 clarification

November 22, 2004

There have been several requests for more information on the lambda-calculus. This note is an attempt to provide more information.

1 Syntax

Terms in the lambda calculus are either variables, lambda-abstractions, or applications.

Application associates to the left, that is mno is associated $((mn)o)$.

The dot in lambda abstraction is a left parenthesis that extends as far to the right as possible. Hence, $\lambda x.\lambda y.mno$ is $(\lambda x.(\lambda y.((mn)o)))$.

The first rule (α) states formally that variable names do not matter:

$$\lambda x.M = \lambda y.M[y/x] \quad \text{provided } y \text{ does not occur free in } M.$$

The second rule (β) embodies the basic mechanism for function application, in which the formal parameter is replaced by the actual parameter.

$$(\lambda x.M)N = M[N/x]$$

Both rules rely on substitution, which is defined as follows:

$$\begin{array}{rcll} z[M/x] & = & z & \text{if } x \neq z \\ x[M/x] & = & M & \text{if } x = z \\ (\lambda y.N)[M/x] & = & \lambda z.(N[z/y][M/x]) & \text{where } z \text{ is a new variable that is distinct from } x \text{ and does not occur in } N \text{ and is not free in } M. \\ (N_1 N_2)[M/x] & = & (N_1[M/x])(N_2[M/x]) & \end{array}$$

The treatment of variable renaming prevents capture of bound variables. It is sometimes necessary to do all of the renaming suggested by the definition. Often in calculation it is useful to preserve names if they do not cause conflict.

2 Calculating with Church numerals

First, note that the Church numerals iterate application of the first argument to the second. Examples are:

$$\begin{aligned} 0 &- \lambda s.\lambda z.z \\ 1 &- \lambda s.\lambda z.sz \\ 2 &- \lambda s.\lambda z.s(sz) \\ 3 &- \lambda s.\lambda z.s(s(sz)) \end{aligned}$$

Note how the successor function acts on the numeral 2:

$$\begin{gathered} (\lambda n.\lambda s.\lambda z.s(nsz))(\lambda s.\lambda z.s(sz)) \\ (\lambda s'.\lambda z'.s'((\lambda s.\lambda z.s(sz))s'z')) \\ (\lambda s'.\lambda z'.s'(\lambda z.s'(s'z))z')) \\ (\lambda s'.\lambda z'.s'(s'(s'z'))) \end{gathered}$$

When programming with Church numerals it is important to think of each numeral as capable of driving a loop the iterates that number of times.

To understand pairing and projection note that we expect:

$$\begin{aligned} \pi_1(\mathit{mkpair}\ a\ b) &= a \\ \pi_2(\mathit{mkpair}\ a\ b) &= b \end{aligned}$$

For problem 3, in many cases the function definition schema has parameters that vary with the arity. It is acceptable to give a family of lambda terms to implement the schema. For example the constant function zero of arity k is given by the family of terms:

$$\lambda x_1.\ldots.\lambda x_k.\lambda s.\lambda z.z$$

My implementation of the primitive recursion schema uses some of the same techniques as the implementation of predecessor. If it helps, pick a fixed arity of PR to implement.

My implementation of the minimization schema makes essential use of the fixed point combinator used in the factorial example.