Com S 641 — Programming Languages 1 February 19, 2000

Homework 2: Untyped Calculi

Due: February 29, 2000.
In this homework, will explore Abadi and Cardelli's untyped ς-calculus, and variations on it.
Read Chapter 6 of *A Theory of Objects*.

1. Briefly summarize the treatment of inheritance described in section 6.6. Are you convinced that this encoding of inheritance is faithful to languages like Smalltalk? Why why not?
2. Using either λProlog or Haskell, encode the operational semantics of Abadi and Cardelli's untyped ς-calculus. In what you hand in, be sure to include a printout of your code and some testing, preferably on some of the examples from section 6.5. You do not have to implement any of the syntactic sugars, although you may do so to ease your testing. Your testing input can be abstract syntax; that is, you do not need to implement a parser.
3. Consider a version of the ς-calculus, let us call it the **self**-calculus, in which methods do not have an explicitly-written self parameter, but instead we introduce the reserved word **self** to stand for the (now implicit) self parameter. Suppose we also allow methods to be declared to either take several named parameters, or none (a field). In other words, we use the following syntax.

a,b ::=	terms
x	variable
| **self**	self parameter
| $[l_i = m_i{}^{i \in 1..n}]$	object formation (l_i distinct)
| $a.l(b_1, \ldots, b_n)$	method invocation ($n \geq 1$)
| $a.l$	field selection
| $a.l \Leftarrow m$	field update / method update
m ::=	methods
$\mathbf{method}(x_1, \ldots, x_n)b$	method (x_i distinct, $n \geq 1$)
| **field** b	fields

Write down the following for the **self**-calculus:

(a) a primitive semantics, as in section 6.1.2,
(b) a formal definition of free variables and substitution, as in section 6.2.1,
(c) a formal definition of the reduction relations, as in section 6.2.2,
(d) a formal definition of an equational theory, as in section 6.2.3,
(e) a formal definition of the big-step operational semantics, as in section 6.2.4,
(f) a translation of the untyped lambda calculus into the **self**-calculus,
(g) a fixed-point operator in the **self**-calculus, along with a verification that it has the fixpoint property.

4. Compare the **self**-calculus with the ς-calculus. Can you translate one into the other and vice versa? If so, give the appropriate translation(s). Use examples to show any advantages and disadvantages of one over the other.

   If the ς-calculus has any advantages over the **self**-calculus, can may be repaired by making some minor changes? If so, describe how to do that, and give some examples that show how your repairs work to alleviate the disadvantages. If not, give a convincing argument why they cannot be repaired without going back to something like the original ς-calculus.