CS 520 - Principles of Programming Languages
Solutions for Problem Set 2

Michael Long

Sep. 24, 2004

Problem 1:

In Figure 5-3 in [Pierce], variables are not values. This means that the term t = ((lambda x.x) y) is in normal form according to call-by-value evaluation. For brevity, we say t is in cbv-normal form. Suppose we adjust the call-by-value semantics of Figure 5-3 to now include variables among values, so that t can be further reduced to y. We refer to this adjusted call-by-value by call-by-value'. Accordingly, we can say t is not in cbv'-normal form. Note that a term in cbv'-normal form is also in cbv-normal form, but not the other way around. Consider arbitrary terms u, u1 and u2 such that:

1. u reduces to u1 according to cbv evaluation, and u1 is in cbv-normal form.
2. u reduces to u2 according to cbv' evaluation, and u2 is in cbv'-normal form.

Show that if u is closed, then u1 = u2. (This means that, as far as closed terms are concerned, cbv and cbv' produce exactly the same results.)

Solution: We are given that $u \rightarrow u_1$ under cbv and $u \rightarrow u_2$ under cbv' where u_1 and u_2 are each in normal form (completely reduced). Since u_1 and u_2 are also specified as closed terms, then by definition we know there are no free variables present in either u_1 or u_2. The only difference between the evaluation strategies cbv and cbv' is that cbv' allows evaluation of terms with free variables, however we know that there are no free variables in these terms. Thus we can conclude that the rules of evaluation that have been taken in cbv' are identical to those of cbv, and that $u_1 = u_2$.

Problem 2: Exercise 5.2.8

Solution in [Pierce, page 500].

Problem 3: Exercise 5.2.10

Solution in [Pierce, page 501].

Problem 4: Exercise 5.3.6

Solution in [Pierce, page 502] contains an error. Rule (E-AppAbs) should read:

$$(\lambda x.t_{12})v_2 \rightarrow [x \rightarrow v_2]\ t_{12}$$

Problem 5: Exercise 3.3.7

Evaluation of expressions in the λ**NB** may reach `wrong` states in the following three ways:

i. The use of λ expressions as arguments in $\mathbb{BN}$ [Pierce, Fig. 3-1 & 3-2]
Solution:we define:

$$\texttt{lambdaterm} ::= \\ \lambda x.t$$

and add the `lambdaterm` to the existing `badbool` and `badnat` defined in [Pierce, Exercise 3.5.16]. The existing evaluation rules in [Pierce, Exercise 3.5.16] will now catch instances of lambda expressions being used as arguments to terms expecting either boolean or natural numbers.

ii. An attempt to use either boolean or numerical terms as an abstraction.
Solution: we define:

$$\begin{array}{r} \texttt{badabstraction} ::= \\ \texttt{true} \\ \texttt{false} \\ \texttt{nv} \\ \texttt{wrong} \end{array}$$

and include the evaluation rule:

$$\texttt{badabstraction}\ t \rightarrow \texttt{wrong}$$

iii. The use of a term deemed `wrong` as an argument to an abstraction.
Solution: we add the evaluation rule:

$$t_1\texttt{wrong} \rightarrow \texttt{wrong}$$