Homework 4: Fixpoint Semantics

Consider this homework to be "due" on November 12.

1. Some basic exercises in fixpoint semantics.

   (a) Do exercise 4 on page 71 of Schmidt's text. As you simplify, show the steps involved. (This problem is good for being able to read standard style semantic descriptions and for formal manipulations of such semantic descriptions. Refer to chapters 3 and 4 for Schmidt's notation. If you want practice in such work, try problem 1 on page 70, or problem 3 on page 55.)

   (b) Do exercise 2, parts a and b, on page 132 of Schmidt's text. (The fixpoint construction.)

2. Some analysis of the langauge of **while**-programs as found in Gunter's notes. (See Schmidt's 6.6.5 for an alternative treatment.)

   (a) Compute, using the operational semantics of **while**-programs given in Gunter's table 3.1, the final state reached by the following program:

   k := 3;
   while k > 1
   do if k = 3
       then k := 5
       else if k = 5 **then** k := 8 **else** k := 1 **fi**
       fi
   od; output := k

   (It is clear what the answer is, the problem is to show it rigorously.)

   (b) What is the bottom element of the domain **States** = **Indentifiers** $\rightharpoonup$ **Nat**? What is the bottom element of the domain **Indentifiers** $\to$ $\mathbf{Nat}_\perp$?

   (c) Compute, using the fixpoint semantics of **while**-programs, given on page 37 and pages 41-42 of Gunter's notes, the meaning of the above program (as a partial function from **States** to **States**).

   (d) Is there a denotation of the following **while**-program, as a partial function from **States** to **States**?

   while true do skip od

   If so, what is it?

   (e) Prove or disprove the following for all **while**-programs, C and all **States** s. If $(C, s) \to^* s'$, then $[\![C]\!]s$ is defined and equals s', where the meaning of $[\![C]\!]$ is given by the fixpoint semantics of **while**-programs. (That is, the fixpoint semantics is sound.)

   What about the converse? (i.e., completeness)

3. Do Gunter's exercise 3.2 on page 38 of his notes.

   If you are interested in the theory of cpos per se, you might want to try some of Gunter's exercises on page 41, or Schmidt's exercise 20 on page 136.

4. Some analysis of PCF.

   (a) Do Gunter's exercise 3.25 on page 52 of his notes. (Programming in PCF.)

   (b) Consider the following structural operational semantics for PCF (based on Plotkin's paper "LCF Considered as a Programming Langauge"). The set of configurations Γ is just the set of typed PCF expressions. The transition relation is written $\rightarrow$, and the set of terminal configurations T is the set of configurations that are terminal according to the following rules.

$$\begin{gathered}
\mathbf{pred}(0) \rightarrow 0 \\
\mathbf{pred}(\mathbf{succ}(e)) \rightarrow e \\
\mathbf{zero?}(0) \rightarrow \mathbf{true} \\
\mathbf{zero?}(\mathbf{succ}(0)) \rightarrow \mathbf{false} \\
\mathbf{zero?}(\mathbf{succ}(\mathbf{succ}(e))) \rightarrow \mathbf{zero?}(\mathbf{succ}(0)) \\
((\lambda x : s \,.\, e)e') \rightarrow [e'/x]e \\
(\mathbf{if\ true\ then}\ e_2\ \mathbf{else}\ e_3) \rightarrow e_2 \\
(\mathbf{if\ false\ then}\ e_2\ \mathbf{else}\ e_3) \rightarrow e_3 \\
(\mu x : t.e) \rightarrow [(\mu x : t.e)/x]e
\end{gathered}$$

$$\frac{e_1 \rightarrow e_1'}{(e_1 e_2) \rightarrow (e_1' e_2)}$$

$$\frac{e \rightarrow e'}{(\mathbf{if}\ \mathrm{e}\ \mathbf{then}\ e_2\ \mathbf{else}\ e_3) \rightarrow (\mathbf{if}\ e'\ \mathbf{then}\ e_2\ \mathbf{else}\ e_3)}$$

$$\frac{e \rightarrow e'}{\mathbf{succ}(e) \rightarrow \mathbf{succ}(e')}$$

$$\frac{e \rightarrow e'}{\mathbf{pred}(e) \rightarrow \mathbf{pred}(e')}$$

$$\frac{e \rightarrow e'}{\mathbf{zero?}(e) \rightarrow \mathbf{zero?}(e')}$$

   Show that if $e \rightarrow^* c$, then $\mathcal{C}[\![e]\!] = \mathcal{C}[\![c]\!]$. (That is, show that the standard fixpoint semantics of PCF is sound with respect to this operational semantics.)

   (c) How would you define canonical terms with the above operational semantics?

   (d) How would you show the equivalence of the above operational semantics for PCF and Gunter's? Is there more than one way?

5. The fixpoint construction gives a principle of reasoning for recursive definitions known as fixpoint induction. This is discussed in Schmidt's section 6.7. Read this section and then do exercises 3, 4, and 5 on page 132 of Schmidt's text.

6. Fixpoint semantics can be used to show the soundness of other kinds of reasoning methods. Read Schmidt's secton 6.6.6 and do exercise 9 on page 133.