Homework 3: Operational Semantics for the Declarative Computation Model

Due: Tuesday, September 26, 2006.

In this homework you will learn about operational semantics for the declarative computation model.

Don't hesitate to contact the staff if you are stuck at some point.

Read Chapter 2, and start reading Chapter 3, of the textbook [RH04]. See the course lecture notes for the terminal transition system details, at the following URL.

`http://www.cs.iastate.edu/~cs541/lectures/declarative/kernel.txt`

You may also want to refer Matthew Hennessy's book [Hen90], or other texts, for more background on operational semantics.

Operational Semantics Exercises

I suggest doing the following exercises on a computer, so that you can copy configurations from one step to the next. However, you don't have to typeset your answers, a simple encoding such as that shown in the lecture note files is fine.

(If you want to do fancy typesetting, it may take quite a bit longer. However, if you want to do that with LaTeX, then you may want to use my calculation macros, which are at the following URL.

`http://www.cs.iastate.edu/~leavens/include/calculation.tex`

There is documentation in the file, with examples.)

To get you started, the following describes an example similar to one we did in class. Consider the program in Figure 1. This is translated into kernel syntax in Figure 2. A calculation of how it executes using the terminal transition system given in class for the declarative kernel is shown in Figure 3.

1. (40 points) Consider the program given in Figure 5. Show all the steps in the execution of this program, using a format similar to that in Figure 3.

2. (40 points) Consider the program given in Figure 7. Show all the steps in the execution of this program, using a format similar to that in Figure 3.

3. (40 points; extra credit) Write a small example that uses exception handling (try, raise, and catch, and finally), and show how it executes in the terminal transition system, using a format similar to Figure 3.

4. (50 points; extra credit) The terminal transition we gave in the lecture notes uses an environment. As you can see now, this isn't so handy for hand calculation. Write up another small step operational semantics for the kernel of the declarative model that uses substitutions instead of an environment. You'll have to allow variables in places where the syntax only allows variable identifiers now. Show how your system works by redoing exercise 1 in your semantics. See section 3.3.1 of [RH04].

5. (50 points; extra credit) Suppose we also restrict the language to the strict functional model, as described in section 2.8.1 of [RH04]. In this model there are no unbound values that can appear in data structures. Give a small step operational semantics for this model, replacing the old syntax and rules for **`local`** with the two new syntactic forms shown in section 2.8.1, and giving new rules in the operational semantics for this syntax. Show how your system works by redoing exercise 1 in this semantics.

Other Problems

6. (50 points total; extra credit) Do the paper review problem at the end of homework 2.

References

[Ast91] Edigio Astesiano. Inductive and operational semantics. In E. J. Neuhold and M. Paul, editors, *Formal Description of Programming Concepts*, IFIP State-of-the-Art Reports, pages 51–136. Springer-Verlag, New York, NY, 1991.

[Hen90] Matthew Hennessy. *The Semantics of Programming Languages: an Elementary Introduction using Structural Operational Semantics*. John Wiley and Sons, New York, NY, 1990.

[Lan66] P. J. Landin. The next 700 programming languages. *Communications of the ACM*, 9(3):157–166, March 1966.

[Plo77] G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.

[Plo81] Gordon Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Aarhus University, September 1981.

[RH04] Peter Van Roy and Seif Haridi. *Concepts, Techniques, and Models of Computer Programming*. The MIT Press, Cambridge, Mass., 2004.

```
local Answer K in
   fun {K X} fun {$ Y} X end end
   Answer = {{K 3} 4}
end
```

Figure 1: A small Oz program using various sugars.

```
local Answer in
   local K in
      K = proc {$ X R}
             R = proc {$ Y R2} R2=X end
          end
      local Four in
         local Three in
            Three = 3
            Four = 4
            local Z in
               {K Three Z}
               {Z Four Answer}
            end
         end
      end
   end
end
```

Figure 2: Kernel translation of the program in Figure 1.

Let $E_0 = \{\}$ be the empty environment, let $\sigma_0 = \{\}$ be the empty store, and let S_0 be the program of Figure 2. The following calculation starts with the state resulting from the input of S_0. Note that this follows the book [RH04] in making closures be pairs of the form (P, E), where P is a procedure text and E an environment.

$((S_0, E_0) \mid \texttt{nil}, \sigma_0)$

$\rightarrow$ $\left\langle \begin{array}{l}\text{by [local], with } E_1 = E_0 + \{\texttt{Answer} \rightarrow x_1\} = \{\texttt{Answer} \rightarrow x_1\},\ x_1 = \mathit{next}(\sigma_0), \text{ and} \\ \sigma_1 = \mathit{alloc}(\sigma_0) = \{x_1\}\end{array} \right\rangle$

$((\textbf{local } \texttt{K } \textbf{in } \texttt{...} \textbf{ end}, E_1) \mid \texttt{nil}, \sigma_1)$

$\rightarrow$ $\left\langle \begin{array}{l}\text{by [local], with } E_2 = E_1 + \{\texttt{K} \rightarrow x_2\} = \{\texttt{Answer} \rightarrow x_1, \texttt{K} \rightarrow x_2\},\ x_2 = \mathit{next}(\sigma_1), \text{ and} \\ \sigma_2 = \mathit{alloc}(\sigma_1) = \{x_1, x_2\}\end{array} \right\rangle$

$((\texttt{K = } \textbf{proc } \texttt{...} \textbf{ end local } \texttt{Four } \textbf{in } \texttt{...} \textbf{ end}, E_2) \mid \texttt{nil}, \sigma_2)$

$\rightarrow$ $\langle$by [sequence]$\rangle$

$((\texttt{K = } \textbf{proc } \texttt{...} \textbf{ end}, E_2) \mid (\textbf{local } \texttt{Four } \textbf{in } \texttt{...} \textbf{ end}, E_2) \mid \texttt{nil}, \sigma_2)$

$\rightarrow$ $\left\langle \begin{array}{l}\text{by [value creation to unbound], with } E_\emptyset = E_2|_{\{\}} = \{\}, \\ \mathit{kv} = (\textbf{proc } \texttt{\{\$ X R\} R = } \textbf{proc } \texttt{\{\$ Y R2\} R2=X } \textbf{end end}, E_\emptyset) \\ \sigma_3 = \mathit{bind}(\sigma_2)(\{x_2\}, \mathit{kv}) = \{x_1, x_2 = \mathit{kv}\}\end{array} \right\rangle$

$((\textbf{local } \texttt{Four } \textbf{in } \texttt{...} \textbf{ end}, E_2) \mid \texttt{nil}, \sigma_3)$

$\rightarrow$ $\left\langle \begin{array}{l}\text{by [local], with } E_3 = E_2 + \{\texttt{Four} \rightarrow x_4\} = \{\texttt{Answer} \rightarrow x_1, \texttt{K} \rightarrow x_2, \texttt{Four} \rightarrow x_3\}, \\ x_3 = \mathit{next}(\sigma_3), \text{ and } \sigma_4 = \mathit{alloc}(\sigma_3) = \{x_1, x_3, x_2 = \mathit{kv}\}\end{array} \right\rangle$

$((\textbf{local } \texttt{Three } \textbf{in } \texttt{...} \textbf{ end}, E_3) \mid \texttt{nil}, \sigma_4)$

$\rightarrow$ $\left\langle \begin{array}{l}\text{by [local], with} \\ E_4 = E_3 + \{\texttt{Three} \rightarrow x_4\} = \{\texttt{Answer} \rightarrow x_1, \texttt{K} \rightarrow x_2, \texttt{Four} \rightarrow x_3, \texttt{Three} \rightarrow x_4\}, \\ x_4 = \mathit{next}(\sigma_4), \text{ and } \sigma_5 = \mathit{alloc}(\sigma_4) = \{x_1, x_3, x_4, x_2 = \mathit{kv}\}\end{array} \right\rangle$

$((\texttt{Three = 3 Four = 4 } \textbf{local } \texttt{Z } \textbf{in } \texttt{...} \textbf{ end}, E_4) \mid \texttt{nil}, \sigma_5)$

$\rightarrow$ $\langle$by [sequence]$\rangle$

$((\texttt{Three = 3}, E_4) \mid \texttt{Four = 4 } \textbf{local } \texttt{Z } \textbf{in } \texttt{...} \textbf{ end}, E_4) \mid \texttt{nil}, \sigma_5)$

$\rightarrow$ $\langle$by [value creation to unbound], with $\sigma_6 = \mathit{bind}(\sigma_5)(\{x_4\}, 3) = \{x_1, x_3, x_2 = \mathit{kv}, x_4 = 3\}$ $\rangle$

$(\texttt{Four = 4 } \textbf{local } \texttt{Z } \textbf{in } \texttt{...} \textbf{ end}, E_4) \mid \texttt{nil}, \sigma_6)$

$\rightarrow$ $\langle$by [sequence]$\rangle$

$((\texttt{Four = 4}, E_4) \mid (\textbf{local } \texttt{Z } \textbf{in } \texttt{...} \textbf{ end}, E_4) \mid \texttt{nil}, \sigma_6)$

$\rightarrow$ $\langle$by [value creation to unbound], with $\sigma_7 = \mathit{bind}(\sigma_6)(\{x_3\}, 4) = \{x_1, x_2 = \mathit{kv}, x_3 = 4, x_4 = 3\}$ $\rangle$

$((\textbf{local } \texttt{Z } \textbf{in } \texttt{...} \textbf{ end}, E_4) \mid \texttt{nil}, \sigma_7)$

$\rightarrow$ $\left\langle \begin{array}{l}\text{by [local], with} \\ E_5 = E_4 + \{\texttt{Z} \rightarrow x_5\} = \{\texttt{Answer} \rightarrow x_1, \texttt{K} \rightarrow x_2, \texttt{Four} \rightarrow x_3, \texttt{Three} \rightarrow x_4, \texttt{Z} \rightarrow x_5\}, \\ x_5 = \mathit{next}(\sigma_7), \text{ and } \sigma_8 = \mathit{alloc}(\sigma_7) = \{x_1, x_5, x_2 = \mathit{kv}, x_3 = 4, x_4 = 3\}\end{array} \right\rangle$

$((\texttt{\{K Three Z\} \{Z Four Answer\}}, E_5) \mid \texttt{nil}, \sigma_8)$

$\rightarrow$ $\langle$by [sequence]$\rangle$

$((\texttt{\{K Three Z\}}, E_5) \mid (\texttt{\{Z Four Answer\}}, E_5) \mid \texttt{nil}, \sigma_8)$

$\rightarrow$ $\left\langle \begin{array}{l}\text{by [application], since } E_5(\texttt{K}) = x_2 \text{ and } \sigma_8(x_2) = \mathit{kv} \text{ with} \\ E_6 = E_\emptyset + \{\texttt{X} \rightarrow E_5(\texttt{Three})\} + \{\texttt{R} \rightarrow E_5(\texttt{Z})\} = \{\texttt{X} \rightarrow x_4, \texttt{R} \rightarrow x_5\}\end{array} \right\rangle$

$((\texttt{R = } \textbf{proc } \texttt{\{\$ Y R2\} R2=X } \textbf{end}, E_6) \mid (\texttt{\{Z Four Answer\}}, E_5) \mid \texttt{nil}, \sigma_8)$

$\rightarrow$ $\left\langle \begin{array}{l}\text{by [value creation to unbound], with } E'_6 = E_6|_{\{\texttt{X}\}} = \{\texttt{X} \rightarrow x_4\} \\ \mathit{kyv} = (\textbf{proc } \texttt{\{\$ Y R2\} R2=X } \textbf{end}, E'_6), \\ \sigma_9 = \mathit{bind}(\sigma_8)(\{x_5\}, \mathit{kyv}) = \{x_1, x_2 = \mathit{kv}, x_3 = 4, x_4 = 3, x_5 = \mathit{kyv}\}\end{array} \right\rangle$

$((\texttt{\{Z Four Answer\}}, E_5) \mid \texttt{nil}, \sigma_9)$

$\rightarrow$ $\left\langle \begin{array}{l}\text{by [application], since } E_5(\texttt{Z}) = x_5 \text{ and } \sigma_9(x_5) = \mathit{kyv} \text{ with} \\ E_7 = E'_6 + \{\texttt{Y} \rightarrow E_5(\texttt{Four})\} + \{\texttt{R2} \rightarrow E_5(\texttt{Answer})\} = \{\texttt{X} \rightarrow x_4, \texttt{Y} \rightarrow x_3, \texttt{R2} \rightarrow x_1\}\end{array} \right\rangle$

$((\texttt{R2=X}, E_7) \mid \texttt{nil}, \sigma_9)$

$\rightarrow$ $\left\langle \begin{array}{l}\text{by [var-var binding], with } \sigma_{10} = \mathit{unify}(\sigma_9)(x_1, x_4) \text{ and so} \\ \sigma_{10} = \{x_1 = 3, x_2 = \mathit{kv}, x_3 = 4, x_4 = 3, x_5 = \mathit{kyv}\}\end{array} \right\rangle$

$(\texttt{nil}, \sigma_{10})$

Figure 3: A calculation of the TTS steps taken when executing the program of Figure 2.

```
local Answer Zero One in
   fun {Zero F} fun {$ X} X end end
   fun {One F} fun {$ X} {F X} end end
   Answer = {{{{One One} {One Zero}} true} false}
end
```

Figure 4: Program for Exercise 1 with sugars.

```
local Answer in
   local Zero in
      local One in
         Zero = proc {$ F R}
                   R = proc {$ X R2} R2=X end
                end
         One = proc {$ F R}
                  R = proc {$ X R2} {F X R2} end
               end
         local True in
            local False in
               True = true
               False = false
               local Z in
                  local Z2 in
                     local Z3 in
                        local Z4 in
                           {One One Z}
                           {One Zero Z2}
                           {Z Z2 Z3}
                           {Z3 True Z4}
                           {Z4 False Answer}
                        end
                     end
                  end
               end
            end
         end
      end
   end
end
```

Figure 5: Kernel translation of the program in Figure 4.

```
local Answer Plus in
   fun {Plus X Y}
      case X of succ(Xm1)
      then succ({Plus Xm1 Y})
      else Y
      end
   end
   Answer = {Plus succ(zero) succ(succ(zero))}
end
```

Figure 6: Program for Exercise 2 with sugars.

```
local Answer in
   local Plus in
      Plus = proc {$ X Y R}
                case X of succ(Xm1)
                then local Temp in
                        R = succ(Temp)
                        {Plus Xm1 Y Temp}
                     end
                else R = Y
                end
             end
      local Zero in
         local One in
            local Two in
               Zero = zero
               One = succ(Zero)
               Two = succ(One)         % (optimized :-)
               {Plus One Two Answer}
            end
         end
      end
   end
end
```

Figure 7: Kernel translation of the program in Figure 6.