COMP313-08A
Programming Languages

Coursework Four
Semantics coursework

1. Using the TINY interpreter as a starting point (there is a copy on the Moode site which has had a function $display$ added which you will need to answer the question), add code which allows us to see the state of the program when an error occurs. The state information should be output as an error string—suitably formatted—by the hugs interpreter when an error occurs in the TINY interpreter. The simplest thing to do is use the in-built hugs function

   $error :: String -> a$

   which gives a program error and causes the hugs interpreter to stop, having displayed the string given to $error$ as an argument.

   For example, when evaluating

   $> run\ (Output\ \text{“}z\text{“})\ [\,]$

   there is clearly an error since z is not in the current memory, so an error message like

   $Program\ error\ :\ \ Memory\ :\ x\ =\ Unbound, y\ =\ Unbound, z\ =\ Unbound; Input\ :\ [\,]; Output : [\,]$

   (assuming that the memory started empty, the input sequence was initially empty and the output sequence was initially empty too) needs to be produced by the hugs interpreter. This makes it clear that z is not in the memory.

   Or, when evaluating

   $> eval\ (Seq\ (Assign\ \text{“}x\text{“}\ Read)\ (Assign\ \text{“}y\text{“}\ Read))\ [Numeric\ 1]$

   when the store has just $[1]$ as the initial input sequence then we need to get the error message

   $Program\, error :\ Memory : x = Stored(Numeric\, 1), y = Unbound, z = Unbound; Input : [\,]; Output : [\,]$

   (assuming that the memory started empty, the input sequence was initially $[Numeric\ 1]$ and the output sequence was initially empty).

2. The solution above, using the original way of representing the memory, is not very flexible. This is because in order to print the error message we need to know what identifiers have so far been given a binding, and we cannot do this with the memory represented as just a function.

   However, if we represent the memory by a pair with its first element being the domain of the function, which is the second element and is just as it is currently, then we always know which identifiers are being used, so we can give an error message without having to fix the selection of identifiers, as we had to in the solution above.

   So, change the code so that the memory is now a pair of type

   $$([Ide], Ide -> MemVal)$$

   and now as each identifier is brought into use it is added to the list of identifiers (the first element of the pair) as well as being updated in the second element (just as it is currently).

   This change will require several changes throughout the code (to get the types working correctly, and to make the memory work correctly too).

   Run your solution on the same examples as you used for the first solution to make sure it works just as well.

3. (This is question Q4 from page 149 of Gordon's book.)

   Use the semantic clauses for SMALL (distributed already—it is chapter six of Gordon's book) to evaluate:

   (a) $P[\![program\ output\ 1; output\ 2]\!]()$

   (b) $P[\![program\ output\ (read\ +\ read)]\!]1.2$

   (c) $P[\![program\ begin\ var\ x\ =\ read; output\ x\ end]\!]()$

   Show your working in detail, in particular say, for each step in your calculations, which semantic equation you are using (e.g. P, C1, E4, D3, etc.). The worked examples on page 86 of Gordon's book show the sort of detail required.

As ever, please hand-in your solution on the Moodle site, as a plain text file which has your name included on it at the top.

The deadline for receipt of your solution is 1000 on Monday 16^{th} June 2008.