COMPX307-21B
Programming Languages

Coursework Four
Semantics coursework

1. **Background**

   Using the TINY interpreter as a starting point, I have developed a version which allows us to see the state of the TINY program being interpreted when an error occurs (this is in the file $CW4\_simple\_error\_TINY\_denot.lhs$ on the Moodle site for the paper).

   The state information is output as an error string—suitably formatted—by the Haskell interpreter when an error occurs in the TINY interpreter. This is done the simplest way by using the in-built ghci function

   $error :: String -> a$

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

   For example, when evaluating

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

   there is clearly an error since z is not in the current memory (assuming that the memory started empty, the input sequence was initially empty and the output sequence was initially empty too), so an error message like

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

   is produced by the 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 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).

Your first task

The file has some of the definitions for expressions and commands missing (as shown in comments in the file). Write Haskell definitions for the missing clauses so that we have an interpreter for the whole of the original language (see chapter two of Mike Gordon's book, as used in the lectures and as on the Moodle site to track down the missing clauses).

2. **Your next task**

   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 it currently is, 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 (which is just as this function 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, we need to 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. **Your final tasks**

   Write new semantic equations (in the same style as the equations in chapter two of Mike Gordon's book, as used in the lectures) for TINY that give semantics to the following:

   (a) A $skip$ command, which does nothing, i.e. there is no state change when it is executed;

   (b) An $if - then$ command, i.e. a command like the existing conditional in TINY, but without an $else$-branch;

   (c) A $repeat - until$ command, i.e. a command which executes the commands in its body, then does a Boolean test. If that test is false, then the whole $repeat - until$ command is done again (i.e. repeated). If the test is true then the whole $repeat - until$ command is completed and nothing more happens (i.e. no further state-change);

   (d) Add them to the end of the file as text.

4. Use the semantic clauses for TINY (distributed already—it is chapter two of Gordon's book) to evaluate:

   (a) $C[\![output\ 1; output\ 2]\!]$ where the initial state has empty input and output sequences

   (b) $C[\![output\ (read\ +\ read)]\!]$ where the initial state has 1 and 2 on the input sequence and an empty output sequence

(c) $C[\![x := read; output\ x]\!]$ where the initial state has 1 and 2 on the input sequence and 3 on the output sequence

Show your working in detail, in particular say, for each step in your calculations, which semantic equation you are using as justification for the step (as I did in some examples in lectures). You can type your answers as a text part of the .lhs file you're handing in.

As ever, please hand-in your solution on the Moodle site, as a literate Haskell script (i.e. a .lhs file) with your name at the top. Please explain what you are doing as fully and accurately as possible; **do not** simply hand in some equations and a program with no commentary or explanation.

The deadline for receipt of your solution is 2359 on Tuesday 19^{th} October 2021.