CS 81 Assignment 8 for Wed., April 6 Solutions

1. (abridged) Work out the details of simulating an arbitrary Turing machine in logic, specifically Otter. The representation should be pure logic, along the lines of the pegs puzzle presented in the lecture. The system should be able to produce a clause containing only the answer literal *iff* the TM will halt on the corresponding input. When the machine halts, the answer literal should contain an argument term representing the contents of the tape.

   Demonstrate your result by constructing a Turing machine that will accept the language, where acceptance is defined by having a 1 somewhere on the tape.

   $\{a^n b^n c^n \mid n \in N\}$

   $= \{\Lambda, abc, aabbcc, aaabbbccc, \ldots\}$

   See the attached sample solution.

2. Based on your result in problem 1, discuss what, if anything, can be concluded about the set of valid formulas in predicate calculus?

   There is no algorithm that determines whether or not a formula is valid (true for all interpretations). Part 1 effectively reduces the halting problem for Turing machines to the problem of determining whether a set of formulas is unsatisfiable. That problem, in turn, is equivalent to the general validity problem.

   Classify the following problems as to whether they are computable or not, and give justification, ideally a proof, for your answers.

3. $$f(x) = \begin{cases} 1 \text{ if x is the description of a Turing machine having more than 10 control states} \\ 0 \text{ otherwise} \end{cases}$$

   This is computable. Scan the description of the machine, maintaining a list of the names of states as they are encountered. If more than 10 states are encountered, output 1. Otherwise output 0.

4. $$g(x) = \begin{cases} 1 \text{ if x is the description of a Turing machine that visits at most 10 cells when started on a blank tape} \\ 0 \text{ otherwise} \end{cases}$$

   This is computable. Let n be the number of tape symbols and m be the

number of control states. The maximum number of distinct configurations that include visiting at most 10 cells is $n^{10}m$. Simulate the machine starting with a blank tape, and taking note whether more than 10 cells are visited. Also, keep track of all configurations reached. If more than 10 cells are visited, output 1. Further, if the same configuration is visited twice and 1 has not been output, then output 0. Finally, if $n^{10}m$ steps have occurred, and no output has been given by the above, then output 0, as the machine is in a loop and will never visit an 11^{th} cell.

5. $$h(x) = \begin{cases} 1 \text{ if x is the description of a Turing machine that eventually visits control state } q_{10} \text{ when started on a blank tape} \\ 0 \text{ otherwise} \end{cases}$$

This is not computable. Assume it were. Then we show the blank-tape halting problem is solvable, as follows: Given the argument description of a machine M, create M' by first renaming any occurrences of q_{10} to a new state symbol. Now for any states of the resulting machine that are halting, merge them and change their names to q_{10}. Also, if there are any (state, symbol) combinations that do not have defined transitions, create those transitions and direct them to q_{10}. Then h(<M'>) = 1 if M halts on blank tape, and 0 otherwise. But this was shown uncomputable in class.

6. $$k(x) = \begin{cases} 1 \text{ if x is the description of a Turing machine that eventually halts if started on an input tape containing 1111111111} \\ 0 \text{ otherwise} \end{cases}$$

This is not computable. Assume it were. We will reduce the blank tape halting problem to this problem. Let M be the description of a machine for which we would like to determine whether M halts on a blank tape. Modify the description of M to get machine M', which behaves as follows: M' *erases* anything on the tape. Then it behaves as M would have. So if M halts on a blank tape, then M' will halt on 1111111111, because it will erase the tape, then behave exactly as M on a blank tape. Conversely, if M does not halt on a blank tape, then M' will not halt on 1111111111. So k(M') would solve the blank tape halting problem for M.

Note: It is not correct to have M' *write* 1111111111 on its tape. This is because k is assumed to determine whether M' will halt on 1111111111, not on 1111111111 followed by 1111111111.

If M' were to first write on its tape, there is no reason to expect that it would behave as M on a blank tape, since M''s simulation would then be that of M on 1111111111, not on a blank tape.