Lec-5-HW-3-UTM-halting-SOLN

Reading:
TM Lecture notes and PP Chp. 3:

3.6(sequencial machines),
3.6.1(combination lock),
3.6.2(state),
3.6.3(FSM, clock),
3.6.4(danger light example).

Do the following problems from PP: 3.40 (tic-tac-toe FSM), 3.41 (soda machine FSM)

(PROBLEM) --------------------
A computer is in essence a Universal Turing Machine (UTM). We will know we have built a computer if what we build can do anything a UTM can do. What can a UTM do? NB-The question does not ask what TMs in general can or cannot do, the question asks what UTMs do.

(PROBLEM) --------------------
Consider a Turing Machine (TM) with symbol set, { 0, 1}. Recall that we refer to a TM's tape configuration as bounded if all 1s on the tape are within some finite distance from the initial head position. We saw in lecture that every bounded tape configuration can be viewed as an encoding of a binary number, and that a TM can be said to compute the integer function f if, for every n, starting that TM with a tape encoding n will result in its halting with the tape containing an encoding of f(n). Given TMs that compute f(n) and g(n), respectively, describe how to construct a TM that computes f(g (n)). You may use boxes to represent the two macines, indicating how to connect their halt and start states.

(PROBLEM) --------------------
A simulating UTM can read descriptions of machines written, i.e., encoded, in a way appropriate for that UTM. We call that encoding method a language. Here we deem that language the "UTM's ISA, Instruction Set Architecture". There can be many different UTMs with differing ISAs. Suppose we have two UTMs, UTM-1 and UTM-2, using the same symbol set, but with different ISAs, L1 and L2, respectively.

(a) Can a description of machine M written in L-1 be used as input to UTM-2? That is, if UTM-2 has this on its input tape,

... x desc-L1(M) ...

where "desc-L1(M)" is a description of M encoded in L1, what would happen? Will M be simulated correctly? (Make a general statement about the consequences of running UTM-2 with that input, assuming the R/W head was appropriately positioned initially: for both UTM-1 and UTM-2 the R/W head should be on the first cell to the right of the input.)

(b) Suppose we have know the design of a translating TM, Tr, that can translate desc-L2-(M) to desc-L1(M). We have a UTM-1, and we also desc-L2-(M). We want to have UTM-1 simulate Tr, which will translate desc-L2(M) to desc-L1 (M), and then have UTM-1 simulate M. What language would we use to describe Tr? Show the input tape to UTM-1. Assume we can restart UTM-1 after the translation is completed and Tr halts, having repositioned its R/W head so that it will simulate M. Tr expects its input to be to the left of its R/W head when it starts.

PROBLEM) --------------------
If we can build a Finite State Machine (FSM), a read/write tape, and a read/write head, and connect them to work together as a TM, we can build any TM, including a UTM. If we can build a UTM, we can build a computer. Let's say we are building such a thing. At what point in the process should we make a decision about what our UTM/computer's ISA should be? NB-"Build" means design and actually implement in physical hardware.

(PROBLEM)-------------------
Suppose we have a UTM, T_sim. T_sim's R/W head's correct initial position is on the first blank cell to the right of its input. Since T_sim is a TM, it can be described by some encoding scheme; for instance, by using unary encoding for states and symbols and laying out the rule table as we did in our prior assignment. For instance, if we have this on T_sim's tape initially,

```
   ... y desc(X) ...
```

with the R/W head positioned correctly, T_sim will simulate machine X reading y. Here, "desc()" means the description of a machine in an encoding suitable for T_sim, and "y" is the encoded tape for the simulated machine. Consider this input tape for T_sim,

```
    ... y desc(T_sim) ...
```

a) What are the consequences of running T_sim with this input?

b) Suppose y = [x desc(M)], where M is a turing machine and x is the input for that machine, encoded appropriately as input to T_sim. Explain what results from running T_sim with input,

```
    ... [ x desc(M) ] desc( T_sim ) ...
```

NB--There is a double-layered encoding of input and tapes here. That is, y = [x desc(M)] is encoded as an input tape suitable for the machine described, desc(T_sim), to read. Within that encoding, x is encoded appropriately as M's input tape.

(PROBLEM)-------------------
We proved in lecture that the function Halts(x,M), which determines whether M reading x halts, is not computable. For the following function, describe whether it is computable or not and explain your reasoning.

HaltsBefore(x,M,s)

If M reading x would halt within s steps, this machine halts after printing "1"; otherwise it halts after printing "0."

The input tape for the TM, HaltsBefore, looks like this,

... xMs ...

(Note that the input is written "xMs". This is just a convention to indicate that the input tape does not have separators between sections of tape. Consequently, the total input looks like a single string. Also note that we use "M" as shorthand for "desc(M)". The technical details of how our TM could "know" that x ends and descr(M) begins is a bit complicated. Suffice it to say that we can encode a machine in a such a way that it is possible to detect the begin and end of the encoding, regardless of what is adjacent to it. This is called a "self-delimiting encoding".)

(Problem)---------------------
Consider two FSMs. The first, M1, sends its output to the second, M2, where it is used as input. M2's output likewise goes to M1. Is there any problem with the timing of valid data in this arrangement?

(Problem)--------------
Here is the rule table for a FSM, M (a Moore machine, ignore output):

current-state	input	next-state
A	0	B
A	1	C
A	2	A
B	0	C
B	1	A
B	2	C
C	0	B
C	1	C
C	2	B

(1) Pick a 1-hot state encoding.
(2) Show the next state function, F, as a table, using that encoding.
(2) Implement M: show state elements, indicate the next state function as a circle.
Note: The next-state function will actually consist of several functions, one for each state element. Indicate each as a circle with a description of its branching logic.

(Problem)-------------------
Suppose we want a description language that works for TMs conceptualized in the controller/datapath model. In that model there is a controller FSM to describe, registers and RTL operations to describe, and functional data operations to describe. It was easy to devise a language for describing TMs in the Turing sense of description: it was just a rule table. Suggest a language approriate for composing descriptions in the controller/datapath model. In particular, suggest how to describe RTL operations and controller conditional branching.

(Problem)------------------------
Suppose we have a TM, M, with a built-in even-parity machine as part of its FSM. Let's suppose its symbol set includes "e": whenever it sees "e" as input, it enters the start state of the parity machine. Suppose it also has a register that serves to record which state it should enter after exiting the parity machine. (We'll think of this in our controller/datapath model.) Using the controller/datapath model, and using RTL for controller states, show two different states, X and Y, of M that cause a transfer to the parity machine. Also show the return from that hardware sub-routine. It returns to state A, if started from state X, and to state B, if started from state Y. Show how M sets up the return, and how the return branching is done. You will need more states than just X, Y, A, and B.