CMSC 303 Introduction to Theory of Computation, VCU Spring 2015, Assignment 5 Due: Tuesday, March 28, 2015 in class

Matthew Bowers

Total marks: 38 marks + 4 marks bonus for typing your solutions in LaTeX.

Unless otherwise noted, the alphabet for all questions below is assumed to be $\Sigma = \{0, 1\}$.

1. [6 marks] This question asks you to examine the formal definitions of a TM and related concepts closely. Based on these definitions, answer the following.

   (a) A *configuration* of a Turing Machine (TM) consists of three things. What are these three things?

      i. The tape with the input string written on it
      ii. A starting location of the head
      iii. The starting state

   (b) Can a Turing machine ever write the blank symbol $\sqcup$ on its tape?
   No the blank marks the end of the useful bit of the tape

   (c) Can the tape alphabet Γ be the same as the input alphabet Σ? No the tape alphabet must include at least $\sqcup$ which cannot be in the input alphabet. It is also useful to have additional characters available to mark sections of the tape.

   (d) Can a Turing machine's head *ever* be in the same location in two successive steps?
   Strictly no. This behavior can be simulated by having the head read and write nothing and move back to where it was, so for all intents and purposes it can stay.

   (e) Can a TM contain just a single state?
   No the machine must have at least an accept, reject, and some other state to keep running or operation would terminate at accept or reject

   (f) What is the difference between a decidable language and a Turing-recognizable language?
   If a language is decidable its TM will terminate one way or the other. If a language is only recognizable non-members may cause the machine to loop forever.

2. [8 marks] This question gets you to practice describing TM's at a semi-low level. Give an implementation-level description of a TM that decides the language

$$L = \{x \mid x \text{ contains twice as many 0s as 1s}\}.$$

By *implementation-level description*, we mean a description similar to Example 3.11 in the text (i.e. describe how the machine's head would move around, whether the head might mark certain tape cells, etc.... Please do *not* draw a full state diagram (for your sake and for ours)).

   (a) Sweep from left to right
   (b) if tape is empty accept

(c) reset head to start

(d) Sweep from left to right

(e) replace a 1 with X if end is reached reject

(f) reset head

(g) Sweep and replace two 0's with an X

(h) if end is reached before marking 2 X's reject

(i) Sweep left to right if a 1 is found go to 3 else

(j) Sweep left to right if a 0 is found reject

(k) If all symbols are X accept.

3. [9 marks] This question investigates a variant of our standard TM model from class. Our standard model included a tape which was infinite in one direction only. Consider now a TM whose tape is infinite in *both* directions (i.e. you can move left or right infinitely many spaces on the tape). We call this a TM with *doubly infinite tape*.

   (a) [3 marks] Show that a TM with doubly infinite tape can simulate a standard TM.
   Choose a point on the doubly infinite tape and mark it with a #. Copy the input of the single tape one the double tape beginning to the right of the #. Treat the # exactly as you would treat the left edge of the standard tape.

   (b) [5 marks] Show that a standard TM can simulate a TM with doubly infinite tape.
   pick a point on our doubly infinite tape. Copy all symbols to the left of this mark onto a singly infinite tape T1 in reverse order. Copy all sybols right of the symbol onto a second singly infinte tape T2. A left move on of left part of the tape becomes a right move on T1. A right move on the right side of the tape becomes a right move on T2. Only one tape will be active at a time so the inactive head will stay at a # placed at the first position on each tape. A move across the midpoint changes which tape is active. It has already been shown that k-tape TM's can be simulate by standard TM's.

   (c) [1 mark] What does this imply about the sets of languages recognized by both models?
   Neither machine is more powerful so the sets are equivilant.

4. [10 marks] This question studies closure properties of the decidable and Turing-recognizable languages.

   (a) [5 marks] Show that the set of decidable languages is closed under concatenation.
   Proof by construction. Take two TM's M1 and M2 which decide our two languages. Run the input on M1. After each step of M1 the last n-1 characters from the input to M2 and run it. Our combined machine will accept only if both machines get to accept.

   (b) [5 marks] Show that the set of Turing-recognizable languages is closed under concatenation. (Hint: This is trickier than part (a) because if a (deterministic) Turing machine decides to split an input string x as $x = yz$ and check if $y \in L_1$ and $z \in L_2$, i.e. to check if $x \in L_1 \circ L_2$, then running the *recognizer* for L_1 on y (say) may result in an infinite loop if $y \notin L_1$.)
   Begin with a simular set up to part a. If M1 gets to accept send the remaining input to M2. Each step of M2 causes a new M2 to be spawned which accepts the remaining input from its predicessor. This simulates cutting the input at every place beyond an initial string accepted by M1.

5. [5 marks] This question allows you to explore variants of the computational models we've defined in class. Let a k-PDA be a pushdown automaton that has k stacks. In this sense, a 0-PDA is an NFA and a 1-PDA is a conventional PDA. We know that 1-PDAs are more powerful (recognize a larger class of languages) than 0-PDAs. Show that 2-PDAs are more powerful than 1-PDAs. (Hint: Recall from A4 that the language $L = \{a^n b^n c^n \mid n \geq 0\}$ is not context-free.)
To show that a 2-PDA is more powerful we will construct one which will recognize L. Let M be a 2-PDA with stacks $s1$ and $s2$. From q_start to q_a push \$ to $s1$. Loop on q_a popping nothing and pushing a onto $s1$. On a

read of b transition to q_b1 pop an a from $s1$ and push a \$ to $s2$. Epsilon transition to q_b2 push b to $s2$ Loop on q_b2 popping a's and pushing b's. On a read of c you must pop \$ from $s1$ and b from $s2$ epsilon to q_accept when pop \$ from $s2$.