Theory of Computation: Assignment 6 Solutions

Arjun Chandrasekhar

1. There are infinitely many strings. So if we try to perform step 1 for every possible string, we will never get to step 2 for any string. Thus, we may not finish all of the accepting coputations, and we won't print out every string in the language.

2. There are infinitely many settings, so it is not valid to tell the machine to reject if none of the settings evaluate to 0. The best we can do is tell the machine to accept if any settings evaluate to 0, and accept that it may loop forever if there are no such settings.

3. Suppose L is regular. Then there is a DFA $D = (Q, \Sigma, \delta, S, F)$ that recognizes L. Following the hint, we'll construct a Turing machine that simulates D. The start state for M will be the start state for D. Whenever M reads a character, it erases it, moves right, and transitions to whatever state the DFA would have transitioned to. When the machine encounters a blank square, it checks if it is in one of the DFA's accepting states. If it is, then it accepts; otherwise it rejects.

   Formally, our TM will be defined as a 7-tuple $(Q_M, \Sigma_M, \Gamma_M, \delta_M, S_M, q_A, q_R)$ as follows:

   - $Q_M = Q \cup \{q_A, q_R\}$ - we have the same states as the DFA, along with a special accept and reject state.
   - $\Sigma_M = \Sigma$ - same input alphabet as the DFA
   - $\Gamma_M = \Sigma \cup \{\sqcup\}$ - the tape alphabet is the input alphabet along with the reserved "blank space" symbol
   - If $\delta(q_i, \sigma) = q_j$, then our TM will include the transition $\delta_M(q_i, \sigma) = (q_j, \sqcup, R)$. Every time the TM reads a symbol, it erases the symbol, transitions to whatever state the DFA would transition to, moves right.
   - For all states $q \in F$, we add the transition $\delta_M(q, \sqcup) = (q_A, \sqcup, R)$. Additionally, for all $q \notin F$, we add the transition $\delta_M(q, \sqcup) = (q_R, \sqcup, R)$. If we encounter a blank symbol, it means we have reached the end of the input. We check whether the DFA would have accepted. If we are in a DFA accept state, we transition to the TM accept state. Otherwise, we transition to the TM reject state.
   - $S_M = S$ - the TM starts in the same state that the DFA would start in.
   - q_A, q_R represent the special accept and reject state of the TM. These are not part of the DFA states.

4. This problem is taken from exercise 3.8 in Sipser. Give tape-level descriptions for Turing machines to recognize the following languages on the alphabet $\Sigma = \{0, 1\}$

   (a) The following machine will recognize L:

   1. If the tape is empty, accept
   2. Scan from left to right for a 0. If you don't find it, reject; otherwise, erase it and move to step 3
   3. Scan from left to right for a 1. If you don't find it, reject; otherwise, erase it and move to step 1

   (b) The following machine will recognize L:

1. If the tape is empty, accept
2. Scan from left to right for a 0. If you don't find it, reject; otherwise, erase it and move to step 3
3. Scan from left to right for a 0. If you don't find it, reject; otherwise, erase it and move to step 4
4. Scan from left to right for a 1. If you don't find it, reject; otherwwise, erase it and move to step 1

(c) For this language we can simply take the machine from part (b) and flip the accept/reject states.

5. We showed in question 2 that any language that can be recognized by a DFA can be recognized by a Turing machine. We showed in question 3 that Turing machines can recognize some languages that cannot be recognized by any DFA (due to the pumping lemma). Thus, Turing machines are strictly more powerful than

6. ($\Rightarrow$) First we'll prove the forwards direction. Suppose L is recognized by a regular Turing machine M. It may be tempting to say that M *is* a doubly infinite machine that simply chooses not to use part of its tape, but this is not exactly true. That's because when M is on its leftmost square, if it tries to move left it will stay in place, whereas a proper doubly infinite TM would have moved left.

   Instead, we will construct a doubly infinite machine M_2 that simulates M. At the start of the computation, M_2 will write a special symbol \$ to the left of the first input symbol. Then, during the computation, if M_2 ever encounters the \$ symbol, it knows that it moved left at a point where the original machine would have encountered its left boundary and been forced to stay in place. M_2 will immediately move right, thus staying within the original machine's left boundary. Thus, L can be recognized by a doubly infinite machine M_2.

   ($\Leftarrow$) Next we'll prove the backwards direction. Suppose L is recognized by a doubly-infinite machine M_2. We'll construct a regular Turing machine M_2 that simulates M. At the start of the computation, M will "shift" every symbol over by one square, and right the symbol \$ on the left most square. This lets the machine keep track of its left boundary, and know when it needs to make more room in the left direction.

   If M ever reads \$, it knows that it encountered its left boundary, and it needs to make more room in order to continue simulating M_2's infinite left hand side. M will once again "shift" all of the current tape symbols right by one square, to make more room on the left hand side, and then continue simulating M_2. Because the right side of the tape is infinite, we can perform these shifts as many times as we need to create as much room as we need on the left side.

   Thus, L is recognized by a regular TM, and it is Turing-recognizable.

   The following diagram illustrates how M simulates the doubly infinite machine M_2

Original Input

Tape Head

Add a left side marker

Tape Head

“Shift” tape when we encounter $

Tape Head