Homework 6

Ricky Hempel

April 3, 2018

Chapter 3: Exercises: 3.1-3.2, 3.5, 3.7, 3.8, 3.9, 3.11-3.14, 3.15b-c

3.1 (a.)$q_1 0$, $\sqcup q_2 \sqcup$, $\sqcup \sqcup q_{accept}$
(b.)$q_1 00$, $\sqcup q_2 0$, $\sqcup x q_3 \sqcup$, $\sqcup q_5 x \sqcup$, $q_5 \sqcup x \sqcup$, $\sqcup q_2 x \sqcup$, $\sqcup x q_2 \sqcup$, $\sqcup x \sqcup q_{accept}$
(c.)$q_1 000$, $\sqcup q_2 00$, $\sqcup x q_3 0$, $\sqcup x 0 q_4 \sqcup$, $\sqcup x 0 \sqcup q_{reject}$
(d.)$q_1 000000$, $\sqcup q_2 00000$, $\sqcup x q_3 0000$, $\sqcup x 0 q_4 000$, $\sqcup x 0 x q_3 00$, $\sqcup x 0 x 0 q_4 0$, $\sqcup x 0 x 0 x q_3 \sqcup$, $\sqcup x 0 x 0 q_5 x \sqcup$ $\sqcup x 0 x q_5 0 x \sqcup$, $\sqcup x 0 q_5 x 0 x \sqcup$, $\sqcup x q_5 0 x 0 x \sqcup$, $\sqcup q_5 x 0 x 0 x \sqcup$, $q_5 \sqcup x 0 x 0 x \sqcup$, $\sqcup q_2 x 0 x 0 x \sqcup$, $\sqcup x q_2 0 x 0 x \sqcup$, $\sqcup x x q_3 x 0 x \sqcup$, $\sqcup x x x q_3 0 x \sqcup$, $\sqcup x x x 0 q_4 x \sqcup$, $\sqcup x x x 0 x q_4 \sqcup$, $\sqcup x x x 0 x \sqcup q_{reject}$

3.2 (a.)$q_1 11$, $x q_3 1$, $x 1 q_3 \sqcup$, $x 1 \sqcup q_{accept}$
(b.)$q_1 1 \# 1$, $x q_3 \# 1$, $x \# \ q_5 1$, $x q_6 \# x$, $q_7 x \# x$, $x q_1 \# x$, $x \# q_8 x$, $x \# x q_8 \sqcup$, $x \# x \sqcup q_{accept}$
(c.)$q_1 1 \# \# 1$, $x q_3 \# \# 1$, $x \# q_5 \# 1$, $x \# \# q_{reject} 1$
(d.)$q_1 10 \# 11$, $x q_3 0 \# 11$, $x 0 q_3 \# 11$, $x 0 \# q_5 11$, $x 0 q_6 \# x 1$, $x q_7 0 \# x 1$, $q_7 x 0 \# x 1$, $x q_1 0 \# x 1$, $x x q_2 \# x 1$, $x x \# q_4 x 1$, $x x \# x \ q_4 1$, $x x \# x 1 q_{reject}$
(e.)$q_1 10 \# 10$, $x q_3 0 \# 10$, $x 0 q_3 \# 10$, $x 0 \# q_5 10$, $x 0 q_6 \# x 0$, $x q_7 0 \# x 0$, $q_7 x 0 \# x 0$, $x q_1 0 \# x 0$, $x x q_2 \# x 0$, $x x \# q_4 x 0$, $x x \# x q_4 0$, $x x \# q_6 x x$, $x x q_6 \# x x$, $x q_7 x \# x x$, $x x q_1 \# x x$, $x x \# q_8 x x$, $x x \# x q_8 x$, $x x \# x x q_8 \sqcup$, $x x \# x x \sqcup q_{accept} \sqcup$

3.5 (a.)Yes. The tape alphabet Γ contains $\sqcup$. Therefore, a Turing machine can write any characters in Γ on its tape.
(b.)No. Σ never contains $\sqcup$, but Γ always contains $\sqcup$. Therefore, they cannot be equal.
(c.)Yes. If a Turing machine attempts to move its head off the left-hand end of the tape, it remains on the same tape cell.
(d.)No.Any Turing machine must contain two distinct states: q_{accept} and q_{reject}. Therefore, a Turing machine contains at least two states.

3.7 The description is not a legitimate Turing machine because in step 1 to store all the values the TM would require an infinite tape. Also in step 2 a TM would need infinite time to require all the values for evaluation. Since the two steps are not possible step 3 is rejected. Thus, the TM M_{bad} could require infinite time and infinite steps to try all of them, but the TM description requires that every stage in the TM be completed in a finite number of steps.

3.8 (a.)"On input string w:

1. Scan the tape and mark the first 0 that has not been marked. If no unmarked 0 is found, go to stage 4. Otherwise, move the head back to the front of the tape.
2. Scan the tape and mark the first 1 that has not been marked. If no unmarked 1 is found, reject.
3. Move the head back to the front of the tape and go to stage 1.
4. Move the head back to the front of the tape. Scan the tape to see if any unmarked 1s remain. If none are found, accept; otherwise, reject."

(b.)"On input string w:

1. Scan from the beginning of the tape, mark the first 0.

2. Scan the tape and mark a second 0. If there is not any more 0s reject.
3. Scan from the beginning of the tape and mark a 1. If there are no more 1s reject.
4. Repeat steps 1,2 and 3 until there are no more 0s
5. Scan for 1s. If there are still any 1s reject and accept otherwise."

(c.)"On input string w:

1. Scan from the beginning of the tape, mark the first 0.
2. Scan the tape and mark a second 0. If there is not any more 0s accept.
3. Scan from the beginning of the tape and mark a 1. If there are no more 1s accept
4. Repeat steps 1,2 and 3 until there are no more 0s
5. Scan for 1s. If there are still any 1s accept and reject otherwise."

3.9 (a.)

Proof. Let $L = \{0^i 1^i 2^i \mid i \geq 0\}$.L is not context-free and there is not a 1-PDA that recognizes it.Let P be a 2-PDA that recognizes L. Push a $ on each stack to mark their bottoms. Then for each 0 seen in the input string, push a zero onto both stacks.Now, nondeterministically, guess when the 0s are finished. Then for each 1 seen, pop a 0 from the first stack. Nondeterministically guess when the 1s are finished. Then for each 2 seen, pop a 0 from the second stack. If the $ is on top of both stacks after the last 2 is read and the symbols were seen in the proper order, then accept.Otherwise reject.Since P accepts L since it keeps track of how many 0s were seen first and uses both stacks to ensure that the number of 1s and 2s match. Thus, 2-PDAs recognize every language 1-PDAs recognize, but also recognize at least one language that 1-PDAs cannot. Therefore, 2-PDAs are more powerful than 1-PDAs. ■

(b.)

Proof. A Turing machine by a 2-PDA can be simulation is done as follows:
Stack 1 would represent the tape contents to the left of the current head position of the TM, while stack 2 would be the tape contents to the right of the current head position with the current symbol on the top of the stack. The two stack contents would change accordingly as the head moved across the tape left or right.
Since above we showed that two stacks can simulate a TM, an extra stack does not lead to a more powerful automaton. The extra stack can easily be presented by another tape on the TM. By theorem 3.13, any k-tape TM is equivalent to a single tape TM, therefore 3-PDAs are not more powerful then 2-PDAs. ■

3.11 *Proof.* Let T be a TM with doubly infinite tape. T marks the left-hand end of the input to detect and prevent the head from moving off of that end. Let D be a 2-tape TM, which was already shown to be equivalent in power to an ordinary TM,by theorem 3.13.The first tape of D is written with the input string and the second tape is blank. We cut the tape of T into two parts, at the starting cell of the input string. The portion with the input string and all the blank spaces to its right appears on the first tape of D. The portion to the left of the input string appears on the second tape, in reverse order. ■

3.12 *Proof.* Let M be a Turing machine and let a Turing machine with a left reset be M_{LR}.
M_{LR} ="On input string w:

1. If the current state Q is the accept or reject state of M, go to step 4. Otherwise,M_{LR} will simulate a right or left transition of M.

2. Right Transition:
   i. If the current tape symbol is a, and the transtition function of M, $\delta_M(q, a) = (q', b, R)$, then replace the a with a b and RESET.
   ii. Scan right for a marked tape symbol. If none are found RESET and mark the first tape symbol and move the tape head to the right changing the state of M_{LR} to q' and go to (1). If a marked symbol is found remove the mark, move the tape head to the right.
   iii. Mark the symbol under the tape head and move to the right changing the state of M_{LR} to q' and go to (1).
3. Left Transition:
   i. If δ_M(q,a) = (q', b, L), replace the a with a b and RESET.
   ii. If the first symbol is marked, remove the mark, RESET and change the state of M_{LR} to q' and go to (1).
   iii. Otherwise, scan right for a marked tape symbol. If none is found reject.
   iv. If a marked symbol is found RESET and mark the first symbol.
   v. If the next symbol is marked unmark it, RESET, and move right to the second tape cell changing the state of M_{LR} to q' and go to (1).
   vi. If the second cell is not marked repeat the following loop:
      A. RESET. Move right to the first marked cell. Unmark it and move right.
      B. Mark the current cell and move right.
      C. If the cell is unmarked return to (i).
      D. Otherwise unmark it and RESET.
      E. Move right back to the first marked cell. Move right and change the state of M_{LR} to q' and go to (1).
4. If q' is the accept state of M, accept. If q' is the reject state of M, reject."

Since M_{LR} emulates the transitions of M and only accepts or rejects when M does.Therefore $L(M) = L(M_{LR})$, and a Turing machine with a left reset recognizes the class of Turing-recognizable languages. ■

3.13 *Proof.* Let Turing machine with a stay put instead of left M=$(Q, \Sigma, \Gamma, q_0, q_{accept}, q_{reject})$. Let a NFA N=$(Q', \Sigma, \delta', q_1, F)$ where Q'=Q$\times(\Gamma \cup \{rd\})$.Where rd is the symbol read. The state is a pair (q, X) where q is the state of M that is being kept track of and X is the symbol in the cell that M is reading currently; if X = rd it means that M moved right in the previous step and N must read the next symbol from input. $q_1 = (q_0, rd)$.Initially, M is in q_0, and we should read the first symbol.F=$\{(q_{accept}, a) \mid (q_{accept}, a) \in Q'\}$.So we accept whenever our simulation of M ends in an accept state.Lastly, $\Sigma = \Sigma$ and δ=.

$$\delta^{'}((q, X), a) = \begin{cases} \{(q', X')\} \text{if} X \neq \text{rd and} a = \varepsilon \text{and} \delta(q, X) = (q', X', S) \\ \{(q', rd)\} \text{if} X \neq \text{rd and} a = \varepsilon \text{and} \delta(q, X) = (q', X', R) \\ \{(q, a)\} \text{if} X = \text{rd and} a \neq \varepsilon \end{cases}$$

The language is recognized by a NFA. Therefore a TM with a stay put instead of left recognizes the regular languages. ■

3.14 *Proof.* 1. Write a $ sign on the queue to indicate the left-hand end of the TM tape. Consider the head of M always points at the right-hand of the queue. Suppose now x is on the right-hand end of the queue, then there are two operations of M to be simulated:

   i. x→y,L
   Do this simply by pulling x out of the queue, and push y into the queue. This will cycle-shift the content on the queue one bit to the right.

ii. x,→y,R
Write on top of the left-end element in the queue, in order to make it unique. Then, pull out x, and push y into the queue. Continue to pull out the element on the right-hand end of the queue, but this time we push the same element back to the queue. Repeat this procedure until the dotted element reaches the right-hand end of the queue, then we remove the dot.

In this way, we could simulate the Turing machine by using the operations defined by DQA.

2. Simulate the queue on the input tape of M. We write # on the left-hand end of the tape, to indicate the left-hand end. There two operations in DQA: push and pull.

i. push
In order to push a new element on the left-hand end of the tape, move all characters, except #, on the tape one cell to the right, then we make the head of the tape point to the blank right beside #, and put the element in this position.

ii. pull
Just cross out the elements that needed to be pulled out.

In this way, we could simulate DQA by using Turing machine. Because of (1) and (2), we say that a language can be recognized by a deterministic queue automaton iff the language is Turing-recognizable. ■

3.15 (b.)

Proof. Let L_1 and L_2 be two decidable langauges and let M_1 and M_2 be the (N)Turning Machinea that decides L_1 and L_2 respectively. Let M' be a Turing machine that decides the concatenate of L_1 and L_2. The description of M' is as follows:
M'="On input w:
1.Nondeterministically split w into two tapes w_1 and w_2 such that $w = w_1w_2$.
2.Run M_1 on w_1. If it halts and rejects, reject. If it accepts, go to stage 3.
3.Run M2 on w2. If it accepts, accept. If it halts and rejects, reject.
Try each and every possible cut of w. If the first part is accepted by M_1 and the second part is accepted by M_2 then w is accepted by M'.Else, w does not belong to the concatenation of languages and is rejected. Therefore $L(M') = L_1L_2$ ■

(c.)

Proof. Any Turing-recognizable language L, Let M be the TM that recognizes it. We construct a NTM N that recognizes the star of L:
"On input w:
1.Nondeterministically cut w into parts so that w=w1w2...wn
2.Run M on wi for all i. If M accepts all of them, accept. If it halts and rejects any of them, reject. "
Therefore If there is a way to cut w into substrings such M accepts all the substrings, w belongs to the star of L and M will accept w after a finite number of steps. ■