

Homework 6 Solutions

3.2

a. see book.

b.

$q_1 1 \# 1$
 $x q_3 \# 1$
 $x \# q_5 1$
 $x q_6 \# x$
 $q_7 x \# x$
 $x q_1 \# x$
 $x \# q_8 x$
 $x \# x q_8$
 q_{accept}

c.

$q_1 1 \# \# 1$
 $x q_3 \# \# 1$
 $x \# q_5 \# 1$
 q_{reject}

d.

$q_1 10 \# 11$
 $x q_3 0 \# 11$
 $x 0 q_3 \# 11$
 $x 0 \# q_5 11$
 $x 0 q_6 \# x 1$
 $x q_7 0 \# x 1$
 $q_7 x 0 \# x 1$
 $x q_1 0 \# x 1$
 $xx q_2 \# x 1$
 $xx \# q_4 x 1$
 $xx \# x q_4 1$
 q_{reject}

e.

$q_1 10 \# 10$
 $x q_3 0 \# 10$
 $x 0 q_3 \# 10$
 $x 0 \# q_5 10$
 $x 0 q_6 \# x 0$

$xq_70\#x0$
 $q_7x0\#x0$
 $xq_10\#x0$
 $xxq_2\#x0$
 $xx\#q_4x0$
 $xx\#xq_40$
 $xx\#q_6xx$
 $xxq_6\#xx$
 $xq_7x\#xx$
 $xxq_1\#xx$
 $xx\#q_8xx$
 $xx\#xq_8x$
 $xx\#xxq_8$
 q_{accept}

3.5

See book.

3.14

First, show that we can simulate a queue automaton with a Turing Machine.

Assume that the input string is on TM tape. Move to the end of the string and insert a new character, say #, to denote the beginning of the queue. Then shift this delimiter and each character of the input tape to the right one symbol. At the beginning of the string, insert another delimiter \$. The \$ symbol will denote which symbol is currently being read in the queue automaton simulation, and the queue itself will be maintained at the end of the string between the \$ and the end-of-tape character.

Now, run as a subroutine of the TM the queue automaton. Whenever the queue automaton would pop an element on the queue, run a subroutine instead for the TM that would move to the leftmost edge of the queue portion of the tape, read in that symbol and then shift each element of the queue to the left one space. Likewise, for push, move to the end of the queue portion of the tape, move the end-of-tape character one symbol farther, and then insert the target symbol at this new space.

To show that a queue automaton can simulate a Turing Machine, start with the input on a separate read-only tape as per the format, and an empty queue. To initialize the queue, push a delimiter \$ and then each character of the input string in succession. Then push another delimiter # to make the end of the input string. \$ will denote the current character being read on the tape encoded circularly in the queue, and the # will denote the end of the string.

The queue automaton, having already read in all of the input strings, will then perform epsilon transitions to simulate the TM states. To read in a character, it will be popped

from the queue and the appropriate write will be pushed at the end. However, merely reading the queue successively will simulate a TM that only reads to the right. To move to the left, the queue automaton will cycle (popping and pushing the same symbol) until it reaches the \$ symbol that will always be popped before the current symbol is read, and then pushed after the written value is pushed. To maintain the \$ symbol then, we must cycle through the entire queue each time we read: for if we read to the left in the TM, we will simulate it by cycling through our entire tape until we reach the \$ symbol pointing to the tape symbol just to the left. However, even reading to the right, we must maintain the \$ symbol as the delimiter just to the left of the symbol we are currently reading on the tape. By cycling through the queue as such, and altering the order in which the \$ symbol is pushed in relation to when the TM symbol to be written is pushed, we can simulate all tape operations.