Chapter 9

Turing Machines

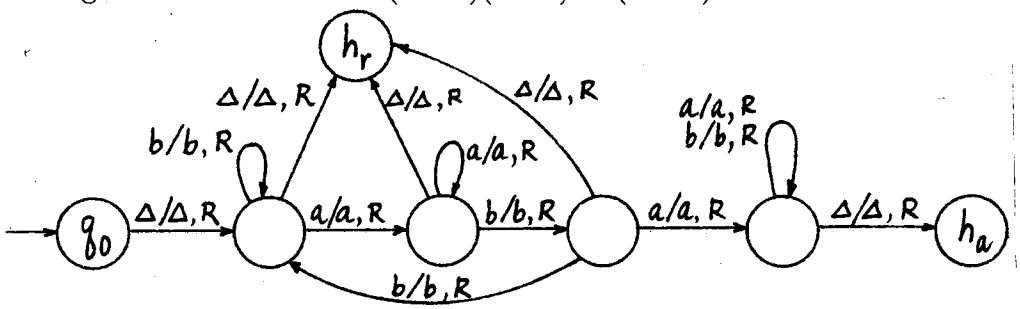
9.1.

$(q_0, \underline{\Delta}aaba) \vdash (q_1, \Delta\underline{a}aba) \vdash (q_2, \Delta A\underline{a}ba) \vdash (q_2, \Delta Aa\underline{b}a) \vdash (q_2, \Delta Aab\underline{a}) \vdash$
$(q_2, \Delta Aaba\underline{\Delta}) \vdash (q_3, \Delta Aab\underline{a}) \vdash (q_4, \Delta Aa\underline{b}A) \vdash (q_4, \Delta A\underline{a}bA) \vdash (q_4, \Delta\underline{A}abA) \vdash$
$(q_1, \Delta A\underline{a}bA) \vdash (q_2, \Delta AA\underline{b}A) \vdash (q_2, \Delta AAb\underline{A}) \vdash (q_3, \Delta AA\underline{b}A) \vdash (q_4, \Delta A\underline{A}BA) \vdash$
$(q_1, \Delta AA\underline{B}A) \vdash (q_5, \Delta A\underline{A}BA) \vdash (q_5, \Delta\underline{A}aBA) \vdash (q_5, \underline{\Delta}aaBA) \vdash (q_6, \Delta\underline{a}aBA) \vdash$
$(q_8, \Delta A\underline{a}BA) \vdash (q_8, \Delta Aa\underline{B}A) \vdash (h_r, \Delta AaB\underline{A})$

9.2. (c) Starting in the initial configuration $(q_0, \underline{\Delta}x)$, where x is an arbitrary string in $\{a, b\}^*$, this TM halts in the configuration $(h_a, \Delta\underline{\Delta}xx^r)$.

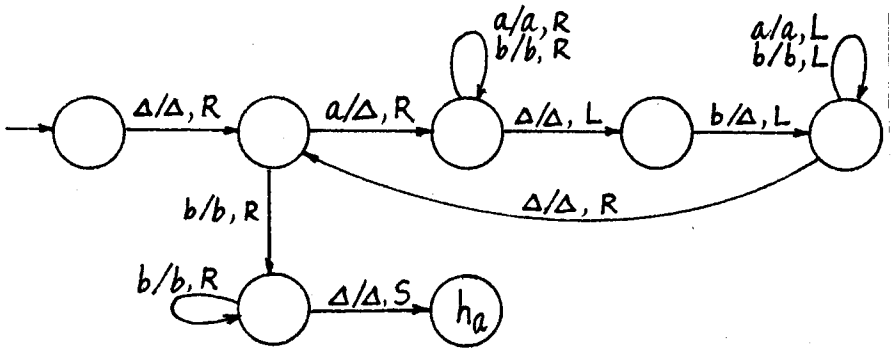
9.3. A configuration is determined by specifying the state, the tape contents, and the tape head position. There are $s + 2$ possibilities for the state, including h_a and h_r; $(t + 1)^{n+1}$ possibilities for the tape contents, since there are $n + 1$ squares, each of which can have an element of Σ or a blank; and $n + 1$ possibilities for the tape head position. The total number of configurations is therefore $(s + 2)(t + 1)^{n+1}(n + 1)$.

9.4.

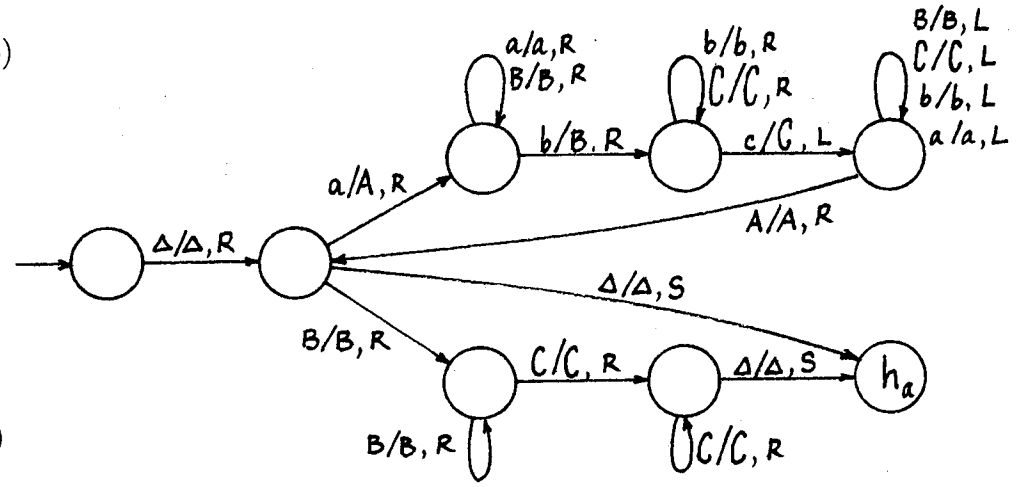


9.5. The technique used in Figure 9.3 can be used in general. There is an introductory move from q_0 to another state that represents the initial state of the FA, which leaves a blank on square 0 and moves the tape head right. The FA transitions are used without change, in the sense that any move on symbol a corresponds to a TM move that leaves a on the tape and moves the tape head right. Finally, from every state that is an accepting state in the FA, there is a transition on the symbol Δ to the state h_a that moves the tape head right.

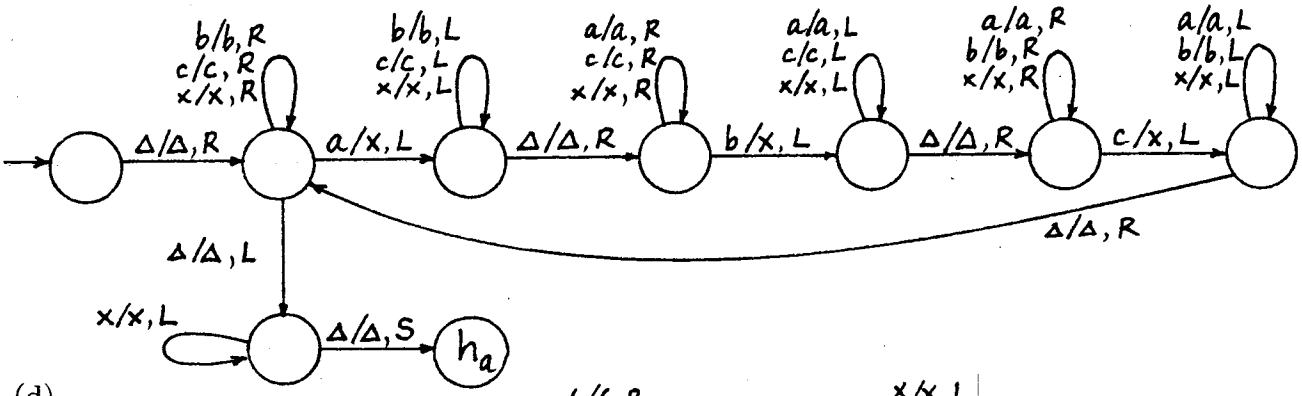
9.6. (a)



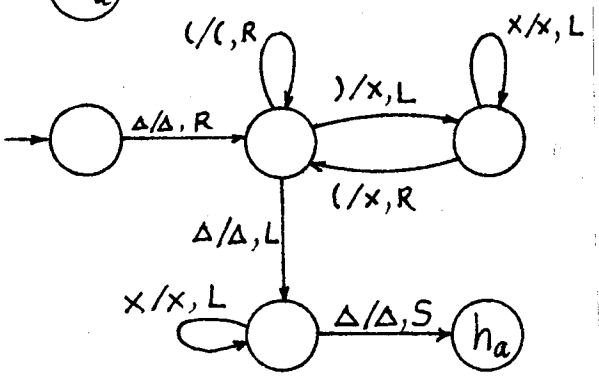
9.6 (b)


a/a,R
B/B,R
b/b,R
C/C,R
B/B,L
C/C,L
b/b,L
a/a,L
b/B,R
c/C,L
a/A,R
A/A,R
Δ/Δ,R
Δ/Δ,S
B/B,R
C/C,R
Δ/Δ,S
h_a
B/B,R
C/C,R

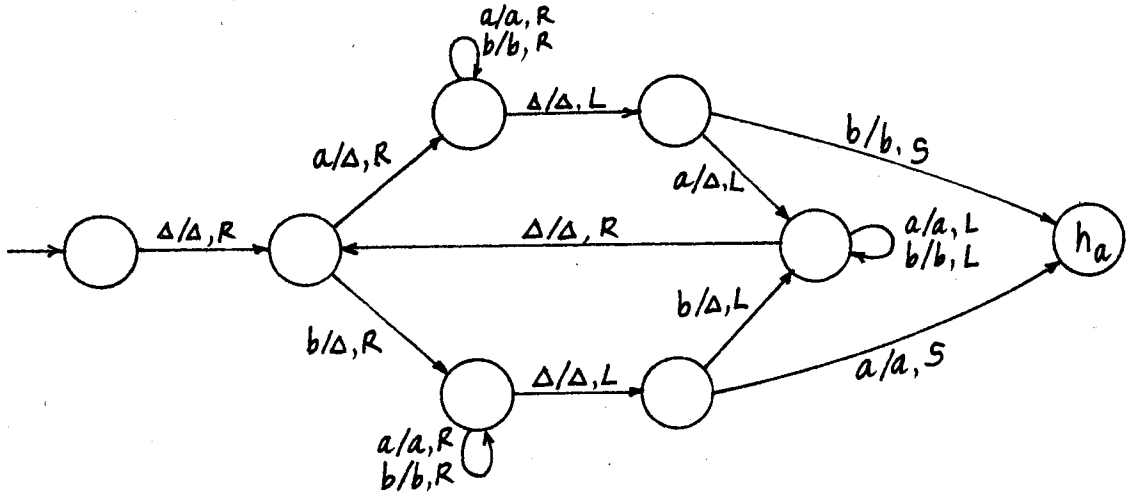
(c)


b/b,R
c/c,R
x/x,R
b/b,L
c/c,L
x/x,L
a/a,R
c/c,R
x/x,R
a/a,L
c/c,L
x/x,L
a/a,R
b/b,R
x/x,R
a/a,L
b/b,L
x/x,L
Δ/Δ,R
a/x,L
Δ/Δ,R
b/x,L
Δ/Δ,R
c/x,L
Δ/Δ,R
Δ/Δ,L
x/x,L
Δ/Δ,S
h_a

(d)


(/(,R
x/x,L
)/x,L
Δ/Δ,R
(/x,R
Δ/Δ,L
x/x,L
Δ/Δ,S
h_a

(e)


a/a,R
b/b,R
Δ/Δ,L
b/b,S
a/Δ,R
a/Δ,L
Δ/Δ,R
Δ/Δ,R
a/a,L
b/b,L
h_a
b/Δ,L
b/Δ,R
Δ/Δ,L
a/a,S
a/a,R
b/b,R

(f) The approach of Example 9.3 can be used, except that in the first phase the point one-third of the way through the string is located, by converting two symbols at the end of the string for every one at the beginning.

9.7. The language $\{1^{2^i} \mid i \geq 0\}$.

9.8. A Λ-transition in an NFA-Λ or a PDA makes it possible for the device to make a move without processing an input symbol. This is unnecessary in a TM because there is already the possibility of a move that leaves the tape head in the same position. (Moves that leave the tape head alone or move it left mean that a TM is not constrained to process the input left-to-right as the simpler machines are.)

9.9. The question is, what does "any obvious modification of it" mean? It is possible to consider a nondeterministic TM that arbitrarily picks a point in the middle of the input string, saves the second part while it processes the first part as T_1 would, and, if and when T_1 would accept, erases the portion of the tape used for that computation and then executes T_2 on the second portion of the original input. (This can also be done without nondeterminism, but it would be more complicated.) Short of this or something like this, however, the approach would not work, because T_1 might accept a string x but never reach the accepting state on a longer string of which x is a prefix, and it might fail to accept x by itself but accept some longer string beginning with x.

9.10. First we relabel states if necessary so that $Q_1 \cap Q_2 = \emptyset$. Then Q is the set $Q_1 \cup Q_2$, the initial state q_0 is q_1, the initial state of Q_1, Σ is Σ_1, and Γ is Γ_2. The transition function δ has the same values as δ_1 for any point of the form (q, a), where $q \in Q_1$ and $\delta_1(q, a) \neq h_a$. (In particular, if T_1 rejects, so does the composite machine.) For any point of the form (q, a) where $q \in Q_2$, $\delta(q, a) = \delta_2(q, a)$. Finally, if $q \in Q_1$ and $\delta_1(q, a) = h_a$, $\delta(q, a) = q_1$.

9.11. Let T_1 be a new TM, which differs from T as follows. It has a new state q not present in T, and for every symbol a, $\delta(q, a) = (q, a, \mathrm{S})$. (The effect is that once T_1 gets to q, it loops forever.) For every combination of state p and tape symbol σ (including Δ) for which T moves to h_r, $\delta(p, \sigma) = (q, \sigma, \mathrm{S})$.

The other way T might end up in h_r is to try to move its tape head off the left end of the tape. T_1 avoids this by executing a preliminary sequence of moves, during which it inserts a special tape symbol \$ in square 0, moving the input string and the preceding blank over by one square. After these preliminary moves, T_1 begins in the configuration $(q_0, \$\underline{\Delta}x)$, where x is the input string. If T_1 ever reads the symbol \$, it enters an infinite loop in which the tape head stays on square 0.

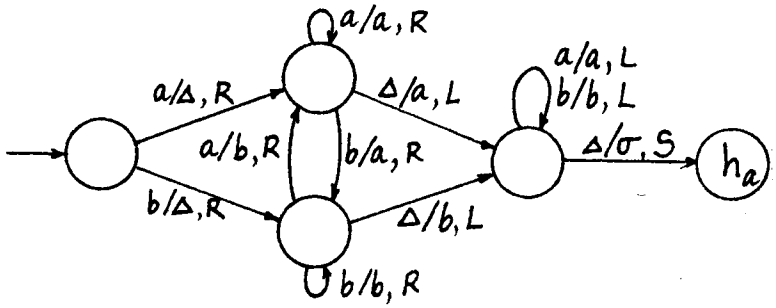
These modifications do not interfere with T's processing of any input string that it accepts, and they do not cause any additional strings to be accepted. Therefore, T_1 accepts the same language as T and never enters the state h_r.

9.12. (a) Suppose there is such a TM T_0, and consider a TM T_1 that halts in the configuration $(h_a, \underline{\Delta}1)$. Let n be the number of the highest-numbered tape square read by

T_0 in the composite machine T_1T_0. Now let T_2 be another TM, which halts in the configuration $(h_a, \underline{\Delta}1\Delta^n1)$. Since the first n tape squares are identical in the final configurations of T_1 and T_2, T_0 will not read the rightmost 1 left by T_2, and so T_2T_0 will not halt with the tape head positioned on the rightmost nonblank symbol left by T_2.

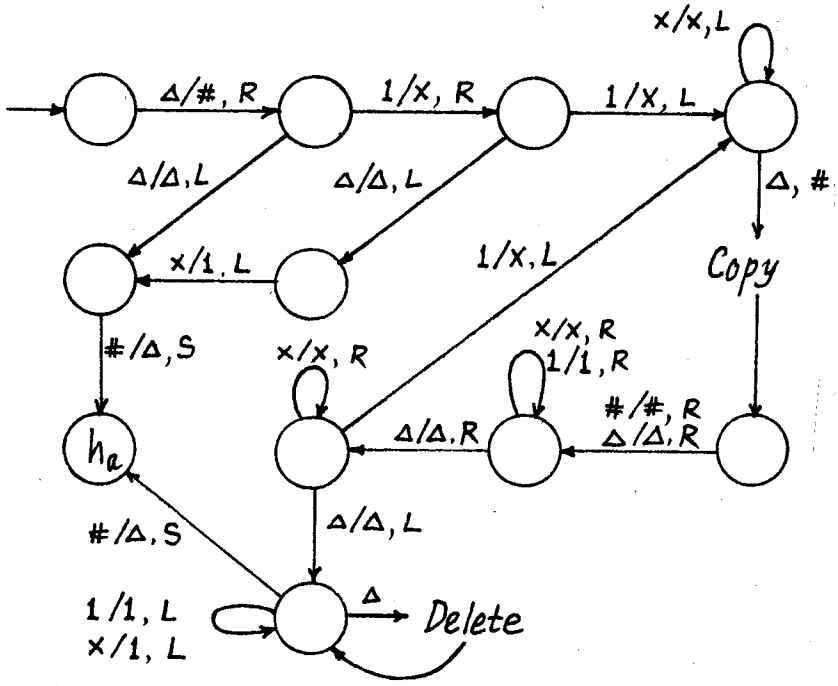
(b) R_T will simulate T, but in such a way that at the end of the processing, the rightmost nonblank symbol can be located. One way to do this is to start by placing a special marker symbol # at the right end of the input string and to return to square 0. During the processing, the marker # will be treated exactly like a blank, except that whenever the tape head reaches the square containing #, this symbol will be moved over one more square to the right. In this way, it continues to mark the rightmost edge of the portion of the tape that has been examined. Once the simulation of T is done, the marker can be erased and the tape head left at the rightmost nonblank symbol preceding it.

9.13.

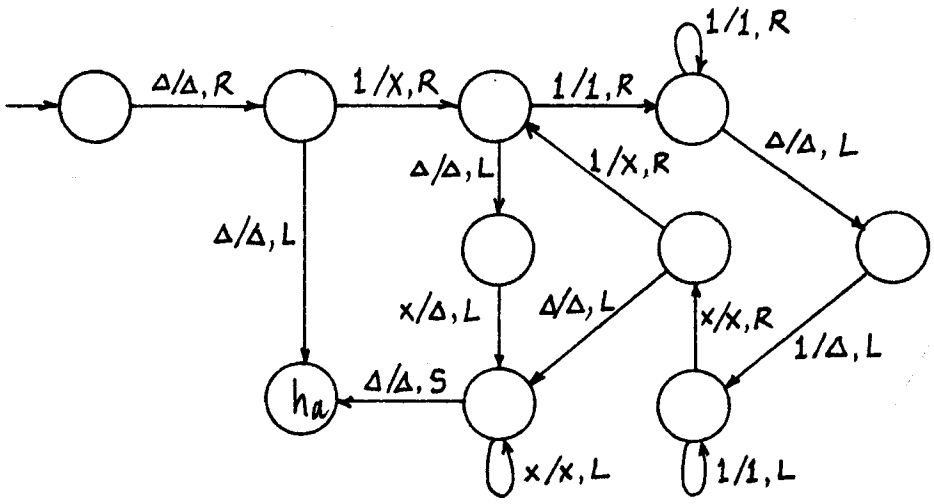


9.14. According to our definition, a TM that accepts an input string but does not leave the tape in the prescribed configuration does not compute a partial function.

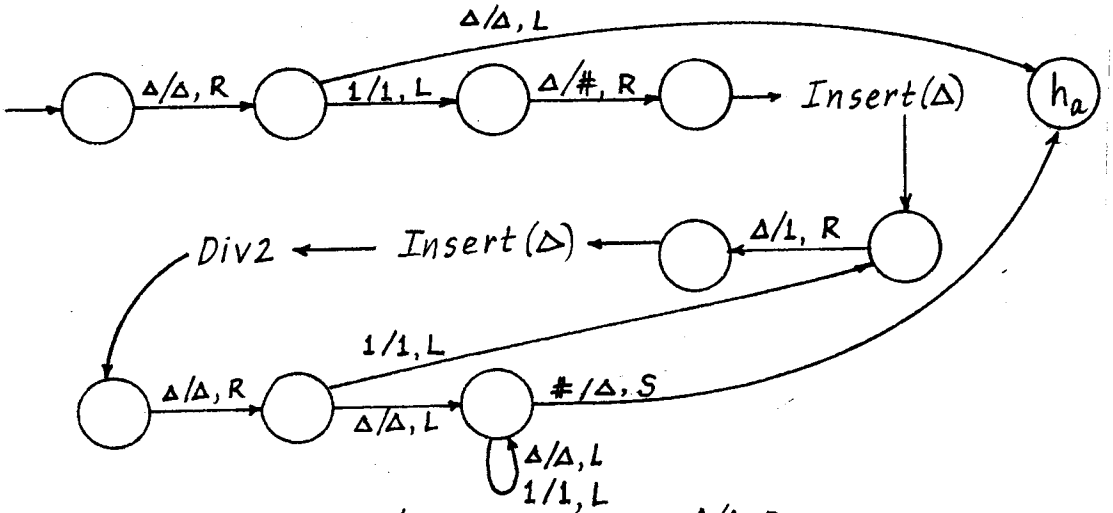
9.15. (c)



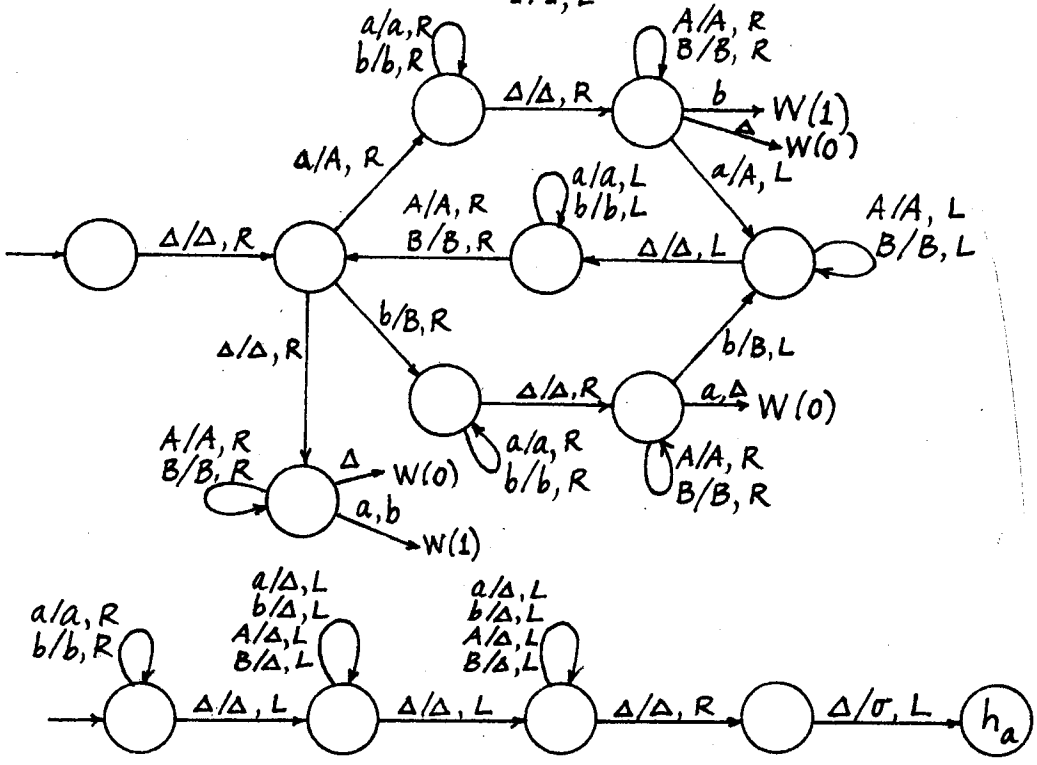
9.15 (d)


Δ/Δ, R
1/X, R
1/1, R
1/1, R
Δ/Δ, L
1/X, R
Δ/Δ, L
Δ/Δ, L
x/Δ, L
Δ/Δ, L
x/x, R
1/Δ, L
h_a
Δ/Δ, S
x/x, L
1/1, L

(e)


Δ/Δ, L
Δ/Δ, R
1/1, L
Δ/#, R
Insert(Δ)
h_a
Div2
Insert(Δ)
Δ/1, R
1/1, L
Δ/Δ, R
#/Δ, S
Δ/Δ, L
Δ/Δ, L
1/1, L

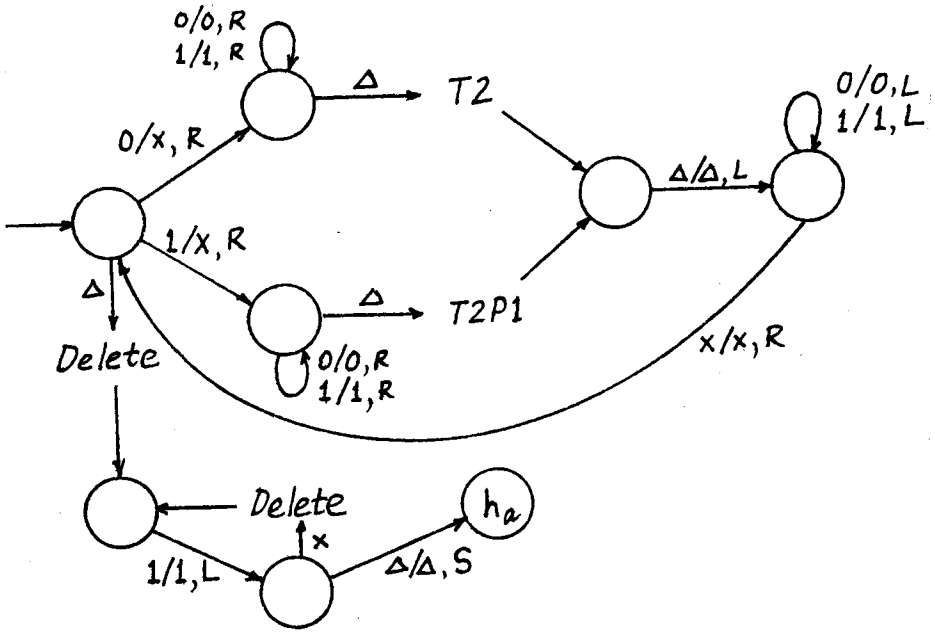
(g)


a/a, R
b/b, R
A/A, R
B/B, R
Δ/Δ, R
b
W(1)
Δ
W(0)
a/A, R
a/A, L
a/a, L
b/b, L
A/A, R
B/B, R
Δ/Δ, R
Δ/Δ, L
A/A, L
B/B, L
b/B, R
b/B, L
Δ/Δ, R
Δ/Δ, R
a, Δ
W(0)
A/A, R
B/B, R
Δ
W(0)
a, b
W(1)
a/a, R
b/b, R
A/A, R
B/B, R
a/Δ, L
b/Δ, L
A/Δ, L
B/Δ, L
a/Δ, L
b/Δ, L
A/Δ, L
B/Δ, L
a/a, R
b/b, R
Δ/Δ, L
Δ/Δ, L
Δ/Δ, R
Δ/σ, L
h_a

9.16. See the solution to Exercise 9.15(i).

9.17. In both (a) and (b), a composite TM can start by making a copy of the input, executing T_1 on the copy, interchanging the result with the original input string (which amounts to moving the blank that separates the two the appropriate distance left or right), and executing T_2 on the original input. The result is that the results of T_1 and T_2 are on the tape, separated by a blank. Now it is straightforward to calculate either their sum or their minimum.

9.18. *T2* and *T2P1* compute the functions $n \rightarrow 2n$ and $n \rightarrow 2n+1$, respectively. The TM below that uses these applies an algorithm that is essentially Horner's rule.



9.19. A pseudocode solution can be expressed as follows. Let `x` denote the input string (or the integer it represents), and `y` the answer.

```
y = 0 (i.e., the null string);
while x is not 0
{  insert the digit x mod 2 at the beginning of y;
   x = x / 2;
}
```

This can be adapted fairly easily to a TM: the string `y` is maintained at the beginning of the tape, and each iteration of the loop consists of a pass through the remaining input to see whether it is even or odd, the insertion of the appropriate symbol at the beginning of `y`, and the execution of the TM in Exercise 9.15(d) on the remaining input.

9.20. One way to modify the diagram is the following. (a) Have the TM insert the symbol # at the beginning of the tape (moving the other tape symbols over) as well as at the end, to simplify the problem of erasing the tape before writing the answer.

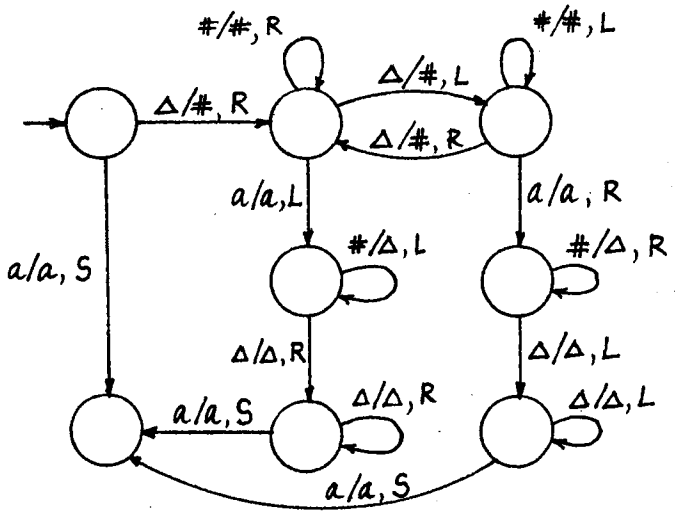
9.20. (b) Replacing the move to the halt state by a sequence of moves that erases the tape, prints 1 on square 1, and halts with the tape head on square 0.

(c) In five places in the diagram where the TM would crash for a string not in the language, add a sequence of moves that would erase the tape, print 0 on square 1, and halt with the tape head on square 0. These five places are: i) the state at the top of the diagram that is third from the right, if the symbol encountered is b or c when it should be a; ii) the state below this one, if the symbol is a or c when it should be b; iii) the state to the left of the one in i), if Δ is encountered instead of c; iv) the state just below this one, in the same situation; v) the state just above the halt state, if anything other than Δ or # is encountered in the loop.

9.21. Here is a verbal description of one solution. Start by inserting an additional blank at the beginning of the tape, to obtain $\Delta\underline{\Delta}x$ (where x is the input). Make a sequence of passes. The first moves the first symbol of x one square to the left and the last symbol one square to the right; the second moves the second symbol of x one square to the left and the next-to-last symbol one square to the right; etc. If x is of odd length, reject. The tape now looks like $\Delta x_1 \Delta\Delta x_2$, where x_1 and x_2 are the first and second halves of x. Now begin comparing symbols of x_1 and x_2, but replacing symbols that have been compared by blanks, rather than uppercase letters as in Example 9.3. In order for this to work, it is necessary to check before each pass to see whether the symbols that are about to be compared are the last symbols in the respective halves. (This prevents an attempt to perform another pass when there are no more nonblank symbols, which would attempt to move the tape head off the tape.)

9.22. The move $\delta(p, a) = (q, b, \mathrm{S})$ can be simulated by using the moves $\delta(p, a) = (p_1, b, R)$ and $\delta(p_1, \sigma) = (q, \sigma, L)$ for any $\sigma \in \Gamma \cup \{\Delta\}$. Here p_1 is a new state used only for this purpose, and the moves shown are the only ones in which it is involved.

9.24.



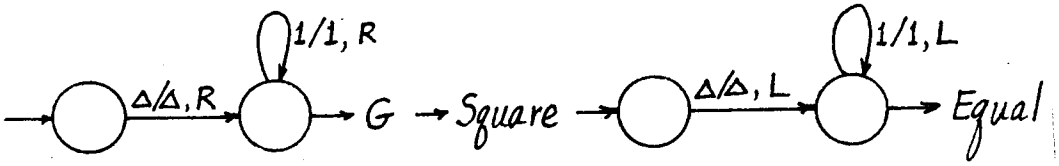
9.25. Given a TM T, a TM T_1 with a doubly infinite tape can be constructed to work as follows: starting in the initial configuration $(q_0, \underline{\Delta}x)$, with the tape head on square 0, it places a special symbol # in square -1, returns the tape head to square 0, and executes

T, except that if it ever reads the symbol # it rejects. It is easy to see that T_1 has the properties specified in the exercise.

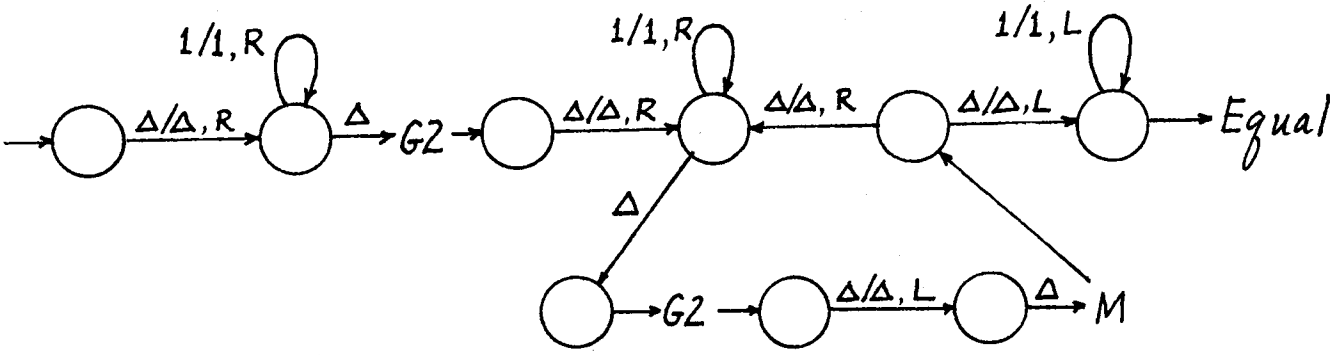
9.29. The TM begins moving its tape head to the right, writing a 0 or a 1 on each square as it goes. It either continues this forever or halts in the configuration $(h_a, \underline{\Delta}x)$, where x is some element of $\{0,1\}^*$. Here x is arbitrary (i.e., for any x, some computation of the TM causes it to halt with x on the tape).

9.30. The language $\{xx \mid x \in \{0,1\}^\}$.

9.31. In the diagram, G is the TM described in Exercise 9.29, *Square* is the TM of Exercise 9.15(c). and *Equal* is as in Exercise 9.30.

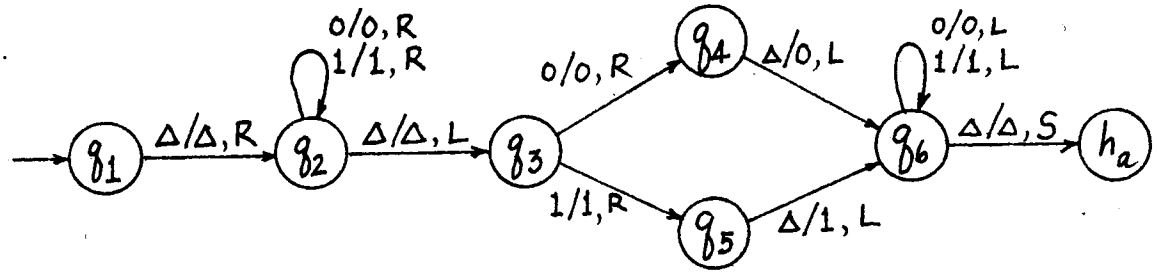


9.32. In the diagram, $G2$ is the same as G in Exercise 9.29 except that it always leaves at least two nonblank symbols on the tape. The TM M computes the multiplication function: it transforms the tape $\underline{\Delta}1^i\Delta 1^j$ to the tape $\underline{\Delta}1^{i*j}$. The TM shown is nondeterministic in that $G2$ is, but also because the steps shown on the lower level can be executed an arbitrary number of times after the first time.

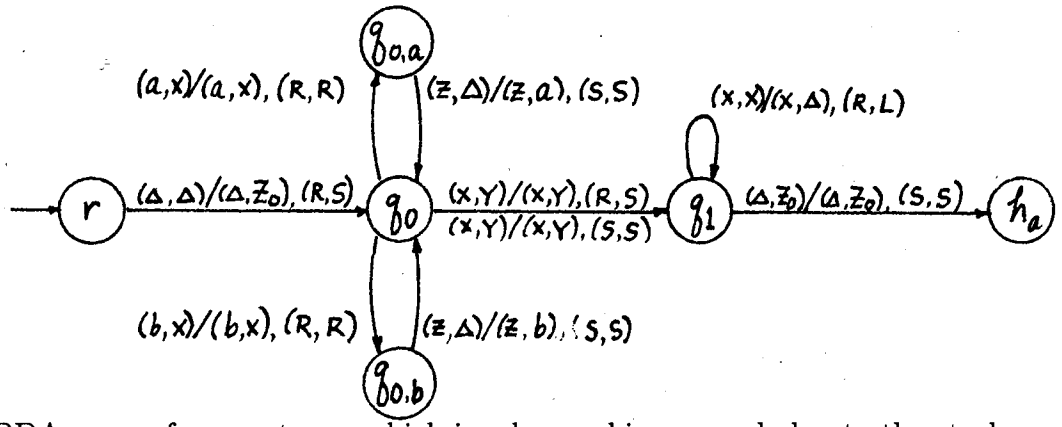


9.33. To accept the set of prefixes of elements of L, for example, a TM could begin by executing the nondeterministic TM G in Exercise 9.29, concatenating the result onto the end of the input string, and executing T on the result. A similar approach works for parts (b) and (c).

9.35.



9.37.



The PDA moves from q_0 to q_0, which involve pushing a symbol onto the stack, require two TM moves, since the TM needs to change a symbol other than the current one on the tape representing the stack. The TM begins by moving to the state corresponding to the initial PDA state q_0, moving the tape head on tape 1 one square to the right, and replacing Δ by Z_0 in the first square of tape 2. The labels on the transitions from q_0 to $q_{0,a}$ and $q_{0,b}$ actually each represent three labels: in both cases, the symbol x can be either a, b, or Z_0 (but is the same in both occurrences within the label). Similarly, on the labels from these two states back to q_0, z can be either a or b but is the same in both occurrences within the label. Each label on the transition from q_0 to q_1 represents six transitions: x can be either a or b, and y can be either a, b, or Z_0 (but all occurrences of x within a label represent the same symbol, and the same for y). Finally, the label on the transition from q_1 to q_1 represents two labels: x can be either a or b.

For example, the moves by which the input string aba is accepted are shown below.

$$(r, \underline{\Delta}aba, \underline{\Delta}) \vdash (q_0, \Delta\underline{a}ba, \underline{Z_0}) \vdash (q_{0,a}, \Delta a\underline{b}a, Z_0\underline{\Delta}) \vdash (q_0, \Delta a\underline{b}a, Z_0\underline{a})$$
$$(q_1, \Delta ab\underline{a}, Z_0\underline{a}) \vdash (q_1, \Delta aba\underline{\Delta}, \underline{Z_0}) \vdash (h_a, \Delta aba\underline{\Delta}, \underline{Z_0})$$

The moves by which aa is accepted are as follows.

$$(r, \underline{\Delta}aa, \underline{\Delta}) \vdash (q_0, \Delta\underline{a}a, \underline{Z_0}) \vdash (q_{0,a}, \Delta a\underline{a}, Z_0\underline{\Delta}) \vdash (q_0, \Delta a\underline{a}, Z_0\underline{a})$$
$$(q_1, \Delta a\underline{a}, Z_0\underline{a}) \vdash (q_1, \Delta aa\underline{\Delta}, \underline{Z_0}) \vdash (h_a, \Delta aa\underline{\Delta}, \underline{Z_0})$$

9.38. The iterative step can be described as follows. If x is the last palindrome written on the tape, then x is copied, so that the last part of the nonblank portion of the tape looks like $\Delta x\Delta x$. If x consists entirely of 1's, then the second copy is then changed to the string $0^{|x|+1}$. Otherwise, the last 0 in the first half of the copy is located. (We take the "first half" to include the middle symbol if $|x|$ is odd.) It is changed to 1, and all the 1's after it within the first half are changed to 0. The second half is then modified so as to make the string a palindrome.

9.39. (a) If the tape head does not move past square n, the possible number of nonhalting configurations is $s(t+1)^{n+1}(n+1)$, where s and t are the sizes of Q and Γ, respectively (see the solution to Exercise 9.3). Within that many moves, therefore, the TM will have

either halted or repeated a nonhalting configuration, and so if it has not halted you may conclude it is in an infinite loop.

(b) As in part (a), let s be the size of the set Q. Suppose you watch the TM until it moves to the first square after the input string, and continue to watch for s more moves. If the tape head has moved right each time, then either it has encountered the combination (q, Δ) twice, for some state q, or the tape head moves right in every possible combination (q, Δ). In either case, the TM is in an infinite loop.

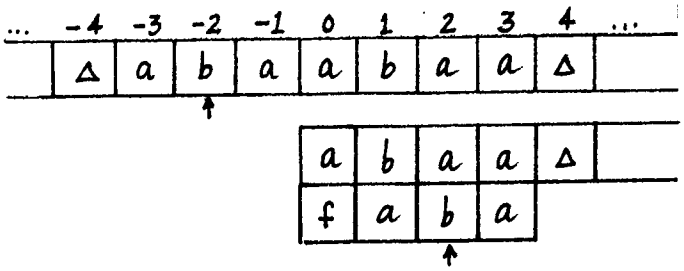
9.40. (a) In this case, the result for an input string x depends only on the first $n+1$ symbols of x. This means that if $S_1 = \{x \in L(T) \mid |x| < n+1\}$ and $S_2 = \{x \in L(T) \mid |x| = n+1\}$, then $L(T) = S_1 \cup S_2\Sigma^*$, which is a regular language.

(b) Once the tape head has passed the last symbol of the input, the TM will never examine the input string again. This means that whether it ever reaches the accepting state depends only on the state it is in when it reaches the first blank square after the input; i.e., there is a set A of states (possibly including h_a) so that for any x, x is accepted if and only if the TM is in a state in A when it reaches the first blank square after the input. Therefore, we may consider the finite automaton $M = (Q, \Sigma, q_1, A, \rho)$, where q_1 is the state T is in after the first move, and for any state $q \in Q$ and $a \in \Sigma$, $\rho(q, a)$ is the state to which T moves on state-symbol combination (q, a). (If $h_a \in A$, then we may add this state to Q and let $\rho(h_a, \sigma) = h_a$ for every $\sigma \in \Sigma$.) The strings accepted by T are precisely those accepted by M, and thus $L(T)$ is accepted by an FA.

9.41. Yes. It follows from the assumption on T that there is a fixed integer n so that for every i, once the tape head reaches square i, it cannot move farther to the left than square $i-n$. This means that we can build another TM T_1 that mimics T but never moves its tape head to the left, by allowing T_1 always to remember the contents of the n squares preceding the current one. Furthermore, it is unnecessary for T_1 to make a move that leaves the tape head stationary, because by examining the transition table for T we can predict the contents of the current tape square and the state to which it goes when it finally moves the tape head right. The conclusion is that there is a TM that accepts the same language as T but always moves its tape head to the right. Exercise 9.40(b) then implies that $L(T)$ is regular.

9.42. We use the first approach. At any point where T has moved its head k squares and no further beyond square 0, either to the left or to the right, T_1's tape will show the first k negative squares as "folded over," in the sense that the portion of the tape from 1 to k will contain two tracks, the first representing the squares from 1 to k and the second those from -1 to $-k$. Square 0 will also contain an extra symbol representing the fold, so that any time T's head crosses from one side of the tape to the other, T_1's head can change from one track to the other.

The following diagram shows what we imagine the two tapes to look like at some point. If T is currently scanning square -2, T_1 will be scanning square 2, paying attention to the lower part only.



The alphabet of T_1 contains not only symbols of Γ but pairs of symbols of Γ, as well as pairs of the form (σ, f), where $\sigma \in \Gamma$ and f represents the fold. (In fact, the alphabet will be even a little larger, as we will see.) The states will be of the form (q, r) and (q, l), where $q \in Q$. The second entries indicate whether the current square of T's tape is on the right side or the left.

T_1 begins by placing the fold marker f in square 0 and moving to state (q_0, r), where it begins to simulate the moves of T. During this simulation, whenever T would move the tape head, T_1 does also, except that if it is currently in a state (q, l) it moves in the opposite direction, and whenever it encounters f it may change tracks. Each time T_1 moves farther from square 0 than it has gone up to that point, it sees a single symbol rather than a pair; at this point it extends the folded portion by converting the single symbol to a pair before continuing.

T_1 starts with the move $\delta(q_1, \Delta) = ((q_0, \mathrm{r}), (\Delta, \mathrm{f}), \mathrm{S})$. For each move $\delta(p, X) = (q, Y, D)$ made by T, we must allow the following moves of T_1. To simplify the notation, let $-D$ denote the direction opposite to D. First, for every $Z \in \Gamma \cup \{\Delta\}$,

$$\begin{aligned} \delta_1((p, \mathrm{r}), (X, Z)) &= ((q, \mathrm{r}), (Y, Z), D) \\ \delta_1((p, \mathrm{l}), (Z, X)) &= ((q, \mathrm{l}), (Z, Y), -D) \end{aligned}$$

Second, $\delta_1((p, \mathrm{r}), (X, \mathrm{f}))$ and $delta_1((p, \mathrm{l}), (X, \mathrm{f})$ are both $((q, \mathrm{l}), (Y, \mathrm{f}), \mathrm{R})$ if $D = \mathrm{L}$ and $((q, \mathrm{r}), (Y, \mathrm{f}), D)$ otherwise. In addition, for each move $\delta(p, X) = (q, Y, D)$, T_1 has the moves $\delta_1((p, \mathrm{r}), X) = ((q, \mathrm{r}), (Y, \Delta), D)$ and (if $X = \Delta$) $\delta_1((p, \mathrm{l}), \Delta) = ((q, \mathrm{l}, (\Delta, Y), -D)$. These moves allow T_1 to simulate faithfully the moves of T up to the point where it halts.

If and when T accepts, T_1 moves instead to some state that still allows it to remember which track of the tape it is supposed to be looking at. Before it can halt, it must fix up its tape so that it is really a duplicate of T's, with the obvious exception that the nonblank symbols will begin in square 0 or somewhere to the right. This essentially means "unfolding" the tape. We can do this by adding extra alphabet symbols of the form (σ, γ'), (σ', γ), and (σ', f), where $\sigma, \gamma \in \Gamma \cup \{\Delta\}$. The primes indicate the current head position and simplify the problem of finding this position later. The unfolding can be done in a sequence of passes, where each pass deletes one symbol from the second track of the tape and inserts it at the beginning of the first. When the second track has been eliminated, the f is deleted, the final position of the tape head is located, and the symbol there is changed back to its original form.

9.44. If we assume that M_1 reads the entire input string, then we can assume that there will be no nonblank symbols farther to the right than the nth square. This means that the

preliminary insertion and final deletion of the symbol # can be done in a total of approximately $4n$ moves, the final pass converting symbol pairs to single symbols can be done in approximately $2n$ moves, and moving to the final tape position requires no more than n moves. The total so far is therefore about $7n$. To simulate a single move in the way that's described in the proof requires three iterations of the following process: move the tape head to the current location on one of the tracks, make some fixed number of moves, and move the tape head back. Since by the ith move, the current location on either track can't be farther right than square i, these three iterations require at most $6i$ moves, plus some fixed number independent of i. The total of these is therefore bounded by

$$\sum_{i=1}^{n}(6i + C) = 3n(n + 1) + Cn$$

The overall total is therefore a quadratic function of n: $3n^2 + 7n + Cn$, where C is a fairly small constant.

9.46. See the proof of Lemma 11.1.

9.47. See the proof of Theorem 10.2.

9.48. For each string of digits of length k that specifies a sequence of moves of the nondeterministic TM to be tried, the number of moves made by T_2 is roughly proportional to $n+k$: n because of copying the input string, k because of executing the sequence of moves, erasing the tape, and calculating the next sequence. For each k, there are 2^k sequences of k moves. So the total number of moves of T_2 is, very approximately, $\sum_{k=1}^{n_x}(n + k)2^k$. This is roughly proportional to $(n + n_x)2^{n_x+1}$.

9.49. We give an informal description of one such machine. For each initial a that it reads, it pushes an x onto the first stack. When it encounters a b, it processes each b by removing an x from stack 1 and placing it on stack 2. When it encounters a c, it checks that stack 1 is empty and transfers the contents of stack 2 to stack 1. It then processes c's the same way it processed b's. This approach can obviously be generalized to any number of distinct symbols.

9.51. The machine makes a sequence of passes. On the first pass, it moves symbols from the front to the rear, except that the first a, b, and c are replaced by A, B, and C, respectively. It crashes if it detects an illegal pair of symbols (ba, cb, ac, or ca), or if it finds one but not all three of the lowercase symbols. On each subsequent pass, it processes the first remaining a, b, and c similarly, crashing if it finds one but not all of the three. It accepts if on some pass there are no remaining a's, b's, or c's.

9.52. The idea that allows a Post machine to simulate a move to the left on the tape is to remember the last two symbols that have been removed from the front of the queue. We illustrate by an example. Suppose that the initial contents of the queue are $abaab$, with the

a at the front. The machine starts by placing a marker X at the rear, to produce $abaabX$. It proceeds to remove symbols from the front, remembering the last two, and placing the symbols at the rear but lagging behind by one. Thus after two steps, the contents are $aabXa$, and the machine remembers that the last symbol removed was b. After three more steps, the contents are $Xabaa$, and the machine remembers that the last symbol was b. When it reads X, it places X on the rear immediately, followed by b, to produce $abaaXb$. At this point, it begins simply moving symbols from the front to the rear, and continues until it has removed the X. If it fails to replace it, the contents of the queue are now $babaa$, and the move to the left has been effectively simulated.