EXAMPLE 9.10

Design a Turing machine that copies strings of 1's. More precisely, find a machine that performs the computation

q0w  *→M\* qf*ww,

for any w ∈ {1}+ .

To solve the problem, we implement the following intuitive process:

1. Replace every 1 by an x

2. Find the rightmost x and replace it with 1.

3. Travel to the right end of the current nonblank region and create a 1 there.

4. Repeat Steps 2 and 3 until there are no more x's.

The solution is shown in the transition graph in Figure 9.7 below. It may be a little hard to see at first that the solution is correct, so let us trace the program with the simple string 11. The computation performed in this case is given below after the figure.

q3

q2

q1

q0

b, 1, L

b, b, R

b, b, L

1, x, R

1, 1, L

1, 1, R

x, 1, R

FIGURE 9.7

q011 → xq01 →xxq0⊥ → xq1x →

→ x1q2⊥ → xq111 → q1x11→

→ 1q211 → 11q21 → 111q2⊥ →

→ 11q111 → 1q1111 → q11111 → q1⊥1111 → q31111

EXAMPLE 9.11

Let x and y be two positive integers represented in unary notation. Construct a TM that will halt in a final state qy if x ≥ y, and that will halt in a nonfinal state qn if x < y. More specifically, the machine is to perform the computation

q0w(x)0w(y) →\* qywx0w(y) if x ≥ y,

q0w(x)0w(y) →\* qnwx0w(y) if x < y.

To solve this problem, we can use the idea in Example 9.7 (where a TM accepts anbn) with some minor modifications. Instead of matching a's and b's, we match each 1 on the left of the dividing 0 with the I on the right. At the end of the matching, we will have on the tape either

xx … 110xx … x⊥

Or

xx … xx0xx … 11⊥

depending on whether x > y or y > x. In the first case, when we attempt to match another 1, we encounter the blank at the right of the working space. This can be used as a signal to enter the state qy. In the second case, we still find a 1 on the right when all I's on the left have been replaced. We use this to get into the other state qn. The complete program for this should be straightforward (a good exercise!).

This example makes the important point that a TM can be programmed to make decisions based on arithmetic comparisons. This kind of simple decision is common in the machine language of computers, where alternate instruction streams are entered, depending on the outcome of an arithmetic operation.