2.2 Examples of DFA’s

Figure 2.7: DFA for $(0 + 1)^*$.

(a)  
(b)  

Figure 2.8: Solution to Example 2.7.

(a)  
(b)  

e one edge and put both labels on the new edge. For instance, the transition diagram of Figure 2.7(a) can be simplified to that of Figure 2.7(b).

Example 2.7 The set of all binary strings beginning with prefix 01.

Solution. It is easy to find a regular expression $01(0 + 1)^*$ for this language. From the regular expression, we can immediately find its labeled digraph representation as shown in Figure 2.8(a). However, it is not the transition diagram of a DFA. Recall that in the transition diagram of a DFA $M = (Q, \Sigma, \delta, q_0, F)$, there is exactly one out-edge from state $q$ with label $a$ for each pair of $(q, a) \in Q \times \Sigma$. To satisfy this condition, we add a failure state $q_3$ and edges $q_0 \rightarrow q_3$ and $q_1 \rightarrow q_3$. The complete transition diagram is shown in Figure 2.8(b).

Example 2.8 The set of all binary strings having a substring 00.

Solution. First, let us note that the regular expression $(0 + 1)^*00(0 + 1)^*$ for this set does not indicate where the first pair 00 occurs in the string. On the other hand, the DFA must recognize the substring 00 at its first occurrence. Thus, the above regular expression is not helpful to this problem. Instead, we need to analyze the problem more carefully.
Suppose the string $x_1x_2\cdots x_n$ is stored on the tape, where each $x_i$ denotes one bit in $\{0,1\}$. Then, we may check each of the substrings $x_1x_2$, $x_2x_3$, $\ldots$, $x_{n-1}x_n$ in turn, to see whether it is equal to 00, and accept the input string as soon as a substring 00 is found. This suggests the following way to construct the required DFA.

Step 1. Build a checker as shown in Figure 2.9(a), with state $q_2$ being a success state, meaning that if two consecutive 0's are found then the input string is to be accepted. In particular, if $x_1x_2 = 00$, the string $x$ is accepted.

Step 2. If $x_1 = 1$, then we give up on substring $x_1x_2$ and continue to check $x_2x_3$. So, we need to go back to state $q_0$; that is, $\delta(q_0, 1) = q_0$. This action is shown in Figure 2.9(b).

Step 3. If $x_1 = 0$ and $x_2 = 1$, then neither $x_1x_2$ nor $x_2x_3$ is 00. So, we also need to go back to restart at $q_0$ to check $x_3x_4$. That is, we let $\delta(q_1, 1) = q_0$. Figure 2.9(c) shows the complete DFA.

In general, suppose that $x_1x_2\cdots x_n$ is the string on the tape and we want to check $x_1x_2x_3\cdots x_{k+1}x_{k+2}\cdots x_n$ in turn to match a substring $a_1a_2\cdots a_k$. Then, we set up $k + 1$ states $q_0, q_1, \ldots, q_k$, with $q_i$ standing for "found $a_1 a_2 \cdots a_i$". At state $q_i$, if $b = a_{i+1}$, then we define $\delta(q_i, b) = q_{i+1}$. If $b \neq a_{i+1}$, then we define $\delta(q_i, b) = q_j$, where $j$ is the maximum index $j$ such that

$$a_{i+1}a_{i+2}\cdots a_{i+j-1}a_{i+j} = a_1a_2\cdots a_j.$$

That is, we find the longest suffix $y$ of $a_1 \cdots a_j$ which is a prefix of $a_1 \cdots a_{i+j}$ and go to $q_j$. The following example explains this idea more clearly.

Example 2.9 The set of all binary strings having the substring 00101.

Solution. Following the above idea, we first construct a checker as shown in Figure 2.10(a).

Intuitively, for each $i = 0, 1, \ldots, 5$, state $q_i$ means "the past $i$ input symbols just read are $a_1a_2\cdots a_i$," where $a_1a_2\cdots a_5$ is the target substring 00101. Thus, at state $q_0$, if we read a new symbol 1, the new string $a_1a_2\cdots a_51 = 1$ (here, $i = 0$) is not a prefix of 00101, and so we need to go back to state $q_0$. Similarly, if a symbol 1 is read at state $q_1$, neither the string $a_11a_2a_3a_4a_5$ is a prefix of 00101, and so we let $\delta(q_1, 1) = q_0$. We upgrade the DFA as shown in Figure 2.10(b).

Now, consider $\delta(q_2, 0)$. The string $a_1a_20 = 00$ is not a prefix of 00101, but the last two symbols $a_30 = 00$ is a prefix of 00101. That is, if the next three input symbols are 1, 0 and 1, then we should accept the input. So, we define $\delta(q_2, 0) = q_3$ to indicate this partial success. This action is shown in Figure 2.10(c).

Based on the same idea, we define $\delta(q_3, 1) = q_0$ (neither $a_1a_2a_3a_4a_5 = 11$ nor any of its suffixes is a prefix of 00101), and $\delta(q_4, 0) = q_2$ (the last two bits of $a_1a_2a_3a_4a_50 = 00$ and form a prefix of 00101). The complete DFA is shown in Figure 2.10(d). □

Example 2.10 The set $A$ of all binary strings ending with 01.

Solution. Initially, we build a checker as shown in Figure 2.11(a). Again, states $q_0$ and $q_1$ indicate "found no prefix of 01" and "found prefix 0 01," respectively. Note, however, that although state $q_2$ is a final state, it is not a success state since a string must end at state $q_2$ to be accepted.

Now, following the idea of the last example, it is easy to see that we need to define $\delta(q_0, 1) = q_0$ and $\delta(q_1, 0) = q_1$. □
At state $q_2$, if we read more input symbols, then we need to follow the same idea to set $\delta(q_2,0) = q_1$ and $\delta(q_2,1) = q_0$. The complete DFA is shown in Figure 2.11(b).

Example 2.11 The set of all binary expansions of positive integers which are congruent to zero modulo 5.

Solution. The idea of this DFA is similar to that of the last two examples. We need to set up five states $q_0, q_1, \ldots, q_4$, with each state $q_i$ meaning "the prefix $y$ of the input string read so far has the property of $y \equiv i \pmod{5}$." That is, we need to define $\delta(q_0, x_1 x_2 \cdots x_k) = q_i$ if $x_1 x_2 \cdots x_k \equiv i \pmod{5}$.

How do we determine the edges between these five states from this idea? Recall that the transition function $\delta$ of a DFA has to satisfy

$$\delta(\delta(q_0, x), a) = \delta(q_0, xa),$$

for any $x \in \{0,1\}^*$ and any $a \in \{0,1\}$. Suppose $\delta(q_0, x) = q_i$ and $\delta(q_0, xa) = q_j$. Then, we must have $x \equiv i \pmod{5}$ and $xa \equiv j \pmod{5}$. Thus,

$$j \equiv xa \pmod{5}$$
$$\equiv 2 \cdot i + a \pmod{5}$$
$$\equiv 2 \cdot i + a \pmod{5}.$$

Therefore, we need to define $\delta(q_i, a) = q_j$, where $j \equiv 2i + a \pmod{5}$. For instance, $\delta(q_2,0) = q_4$ and $\delta(q_2,1) = q_0$. See Figure 2.12 for the other edges. In addition, state $q_0$ is the unique final state, since $\delta(q_0, x) = q_0$ means $x \equiv 0 \pmod{5}$.

Finally, we note that a binary expansion of a positive integer always begins with the symbol 1. So, we need to add a new initial state and a failure state, as shown in Figure 2.12.

4, with leading zeros allowed. Using the idea of the above example, we get a new DFA for set $A \cup \{1\}$ as shown in Figure 2.13. Note that the states $q_0$ and $q_2$ in this DFA can be merged into one, and the simplified DFA is just the one shown in Figure 2.11(b), except that the initial state has been changed (to state $q_1$ of Figure 2.11(b)).

Example 2.12 The set of all binary strings having a substring 00 or ending with 01.

Solution. This language is the union of two languages $(0+1)^*00(0+1)^*$ and $(0+1)^*01$. In Examples 2.8 and 2.10, we have already constructed two DFA’s for these two languages. To check whether an input string $x$ belongs to the union of these two languages, we can run these two DFA’s in parallel. For example, suppose $x = 0101$. In the first DFA $M_1$ shown in Figure 2.9(c), the computation path of $x$ is $(q_0, q_1, q_0, q_1, q_0)$, and in the second DFA $M_2$ in Figure 2.11(b), the computation path is $(q_0, q_1, q_2, q_1, q_2)$. Since the second path ends at a final state, $x$ is accepted in this parallel simulation.

One idea on how to build a DFA for the union of these two languages is, then, to combine the two DFA’s into one such that, at each step, the new DFA would keep track of the computation paths of both DFA’s. This suggests us to consider a product automaton $M = M_1 \times M_2$ as follows: Assume that the first DFA is $M_1 = (Q_1, \Sigma, \delta_1, q_0, F_1)$ and the second DFA is $M_2 = (Q_2, \Sigma, \delta_2, q_0, F_2)$. (Note that $Q_1$ and $Q_2$ may have states of the same name but playing different roles in two DFA’s.) Then, the state set $Q$ of $M$ is the cross product of the state sets of $M_1$ and $M_2$; that is, $Q = Q_1 \times Q_2$. We denote each member of $Q$ as $[q_i, q_j]$, where $q_i \in Q_1$ and $q_j \in Q_2$. The initial state of $M$ is $[q_0, q_0]$. At each state $[q_i, q_j]$ in $Q$, we simulate both computations of $M_1$ and $M_2$ in parallel by

$$\delta([q_i, q_j], a) = [\delta_1(q_i, a), \delta_2(q_j, a)].$$
Example 2.14 The set of all binary strings having a substring 00 but not ending with 01.

Solution. This language is the difference of language \((0 + 1)^*00\) minus language \((0 + 1)^*01\). Thus, we can use the same product DFA as we did in Examples 2.12 and 2.13, except that we need to choose the set of final states to consist of every state in which the first component is a final state of the first DFA and the second component is a nonfinal state of the second DFA; that is, our new final set is \(F = F_1 \times (Q_2 - F_2)\). Its transition diagram is like that of Figure 2.14, with the final states \([q_2, q_0], [q_2, q_1]\).

A special case of subtraction is complementation: \(\overline{A} = \Sigma^* - A\). There is a simpler construction in this case. Note that in DFA \(M = (Q, \Sigma, \delta, q_0, F)\), for any input string \(x, x \in L(M)\) if and only if \(\delta(q_0, x) \in F\). Equivalently, \(x \not\in L(M)\) if and only if \(\delta(q_0, x) \not\in F\). It follows that the DFA \((Q, \Sigma, \delta, q_0, Q - F)\) accepts the complement of \(L(M)\).

Example 2.15 The set \(L\) of all binary strings in which every block of four consecutive symbols contains a substring 01.

Solution. The condition "every block of four consecutive symbols contains a substring 01" is a global condition, which appears difficult to verify. By considering the complement \(\overline{L}\), we turn this condition into a simpler local condition: \(\overline{L}\) contains binary strings with a substring 0000, 1000, 1100, 1110, or 1111. We first construct a DFA accepting \(\overline{L}\) and then change all final states into nonfinal states and all nonfinal states into final states. A solution is shown in Figure 2.16.

The above four examples established the following properties of the class of languages accepted by DFA's.

Theorem 2.16 The class of languages accepted by DFA's is closed under union, intersection, subtraction, and complementation.

Exercise 2.2

1. For each of the following languages, construct a DFA that accepts the language:

   (a) The set of binary strings beginning with 010.
   (b) The set of binary strings ending with 101.
   (c) The set of binary strings beginning with 10 and ending with 01.
   (d) The set of binary strings having a substring 010 or 101.
   (e) The set of binary strings in which the last five symbols contain at most three 0's.
   (f) The set of binary strings \(w\) in which \(\#_0(w) + 2\#_a(w)\) is divisible by 5, where \(\#_a(w)\) is the number of occurrences of the symbol \(a\) in string \(w\).
   (g) The set of strings over the alphabet \(\{1, 2, 3\}\) in which the sum of all symbols is divisible by 5.
   (h) The set of strings over the alphabet \(\{0, 1, 2\}\) which are the ternary expansions (base-3 representations) of positive integers which are congruent to 2 modulo 7.
   (i) The set of binary strings in which every block of four symbols contains at least two 0's.
   (j) The set of binary strings in which every substring 010 is followed immediately by substring 111.

2. For each of the following languages, use the product automaton method to construct a DFA that accepts the language:

   (a) The set of binary strings beginning with 010 or ending with 101.
   (b) The set of binary strings having a substring 010 but not having a substring 101.
   (c) The set of binary strings beginning with 010, ending with 101 and having a substring 0000.

3. For each of the following languages, use the checker method to construct a DFA that accepts the language:

   (a) The set of Example 2.12.