Chap 2

Finite Automata

2.1 Deterministic Finite Accepters

[1] Deterministic Accepters and Transition Graphs

Definition 2.1

A **deterministic finite accepter** or **dfa** is defined by the quintuple

$$M = (Q, \Sigma, \delta, q_0, F),$$

where

Q is a finite set of **internal states**,
Σ is a finite set of symbols called the **input alphabet**,
$\delta : Q \times \Sigma \to Q$ is a total function called the **transition function**,
$q_0 \in Q$ is the **initial state**,
$F \subseteq Q$ is a set of **final states.**

Example 2.1 The graph in Figure 2.1 represents the dfa

$$M = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\}),$$

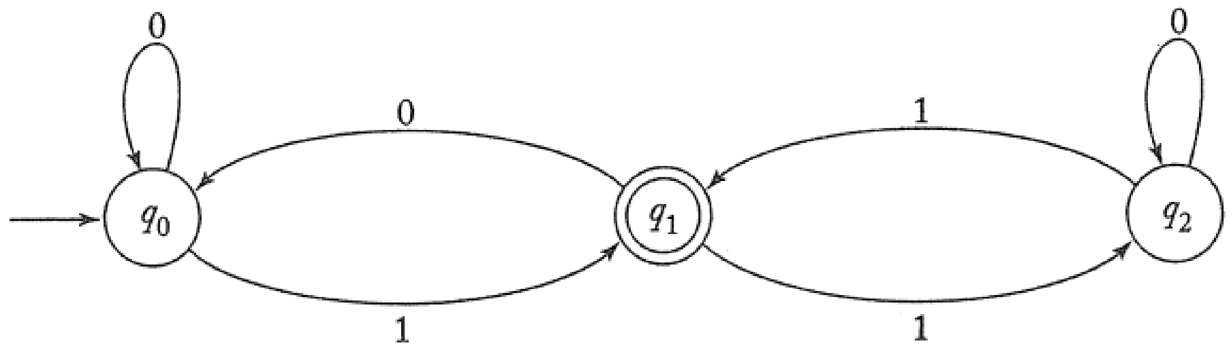
where δ is given by

$$\begin{aligned} \delta(q_0, 0) &= q_0, & \delta(q_0, 1) &= q_1, \\ \delta(q_1, 0) &= q_0, & \delta(q_1, 1) &= q_2, \\ \delta(q_2, 0) &= q_2, & \delta(q_2, 1) &= q_1. \end{aligned}$$

This dfa accepts the string 01. Starting in state q_0, the symbol 0 is read first. Looking at the edges of the graph, we see that the automaton remains in state q_0. Next, the 1 is read and the automaton goes into state q_1. We are now at the end of the string and, at the same time, in a final state q_1. Therefore, the string 01 is accepted. The dfa does not accept the string 00, since after reading two consecutive 0's, it will be in state q_0. By similar reasoning, we see that the automaton will accept the strings 101, 0111, and 11001, but not 100 or 1100.

More formally, if $M = (Q, \Sigma, \delta, q_0, F)$ is a deterministic finite accepter, then its associated transition graph G_M has exactly $|Q|$ vertices, each one labeled with a different $q_i \in Q$. For every transition rule $\delta(q_i, a) = q_j$, the graph has an edge (q_i, q_j) labeled a. The vertex associated with q_0 is called the **initial vertex**, while those labeled with $q_f \in F$ are the **final vertices**. It is a trivial matter to convert from the $(Q, \Sigma, \delta, q_0, F)$ definition of a dfa to its transition graph representation and vice versa.

Figure 2.1



It is convenient to introduce the extended transition function $\delta^* : Q \times \Sigma^* \rightarrow Q$. The second argument of δ^* is a string, rather than a single symbol, and its value gives the state the automaton will be in after reading that string. For example, if

$$\delta(q_0, a) = q_1$$

and

$$\delta(q_1, b) = q_2,$$

then

$$\delta^*(q_0, ab) = q_2.$$

Formally, we can define δ^* recursively by

$$\delta^*(q, \lambda) = q, \tag{2.1}$$

$$\delta^*(q, wa) = \delta(\delta^*(q, w), a), \tag{2.2}$$

for all $q \in Q$, $w \in \Sigma^*$, $a \in \Sigma$. To see why this is appropriate, let us apply these definitions to the simple case above. First, we use (2.2) to get

$$\delta^* (q_0, ab) = \delta (\delta^* (q_0, a) , b) . \tag{2.3}$$

But

$$\begin{aligned} \delta^* (q_0, a) &= \delta (\delta^* (q_0, \lambda) , a) \\ &= \delta (q_0, a) \\ &= q_1 . \end{aligned}$$

Substituting this into (2.3), we get

$$\delta^* (q_0, ab) = \delta (q_1, b) = q_2,$$

as expected.

[2]

Languages and Dfa's

Having made a precise definition of an accepter, we are now ready to define formally what we mean by an associated language. The association is obvious: The language is the set of all the strings accepted by the automaton.

Definition 2.2

The language accepted by a dfa $M = (Q, \Sigma, \delta, q_0, F)$ is the set of all strings on Σ accepted by M. In formal notation,

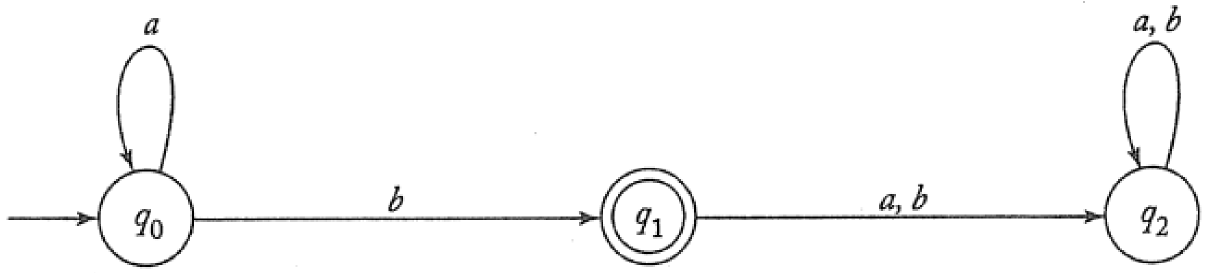
$$L(M) = \{w \in \Sigma^* : \delta^*(q_0, w) \in F\}.$$

$$\overline{L(M)} = \{w \in \Sigma^* : \delta^*(q_0, w) \notin F\}.$$

Example 2.2 Consider the dfa in Figure 2.2.

$$L = \{a^n b : n \geq 0\}.$$

Figure 2.2



Theorem 2.1 Let $M = (Q, \Sigma, \delta, q_0, F)$ be a deterministic finite accepter, and let G_M be its associated transition graph. Then for every $q_i, q_j \in Q$, and $w \in \Sigma^+$, $\delta^*(q_i, w) = q_j$ if and only if there is in G_M a walk with label w from q_i to q_j.

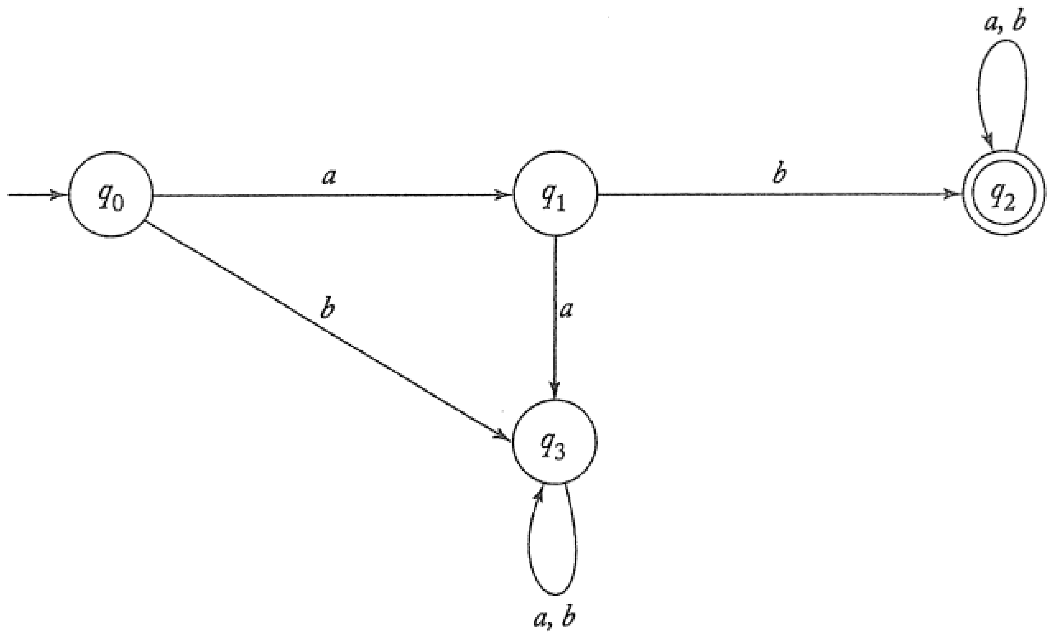
While graphs are convenient for visualizing automata, other representations are also useful. For example, we can represent the function δ as a table. The table in Figure 2.3 is equivalent to Figure 2.2. Here the row label is the current state, while the column label represents the current input symbol. The entry in the table defines the next state.

Figure 2.3

	a	b
q_0	q_0	q_1
q_1	q_2	q_2
q_2	q_2	q_2

Example 2.3 Find a deterministic finite accepter that recognizes the set of all strings on $\Sigma = \{a, b\}$ starting with the prefix ab.

Figure 2.4



Example 2.4 Find a dfa that accepts all the strings on $\{0, 1\}$, except those containing the substring 001.

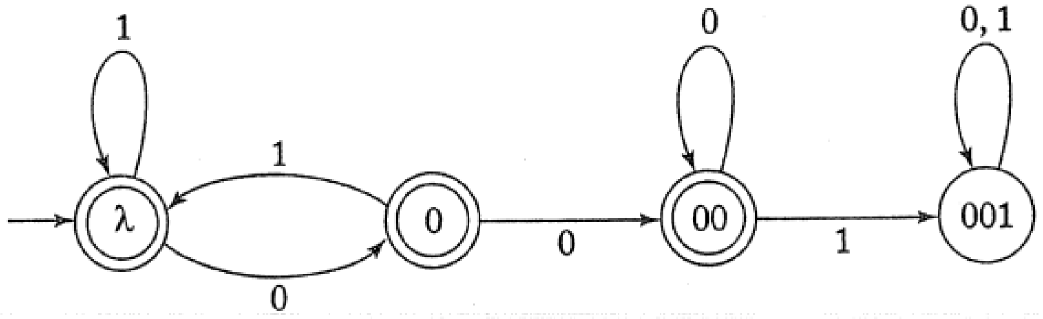


Figure 2.5

[3] Regular Languages

Definition 2.3

A language L is called **regular** if and only if there exists some deterministic finite accepter M such that

$$L = L(M).$$

Example 2.5 Show that the language

$$L = \{awa : w \in \{a,b\}^*\}$$

is regular.

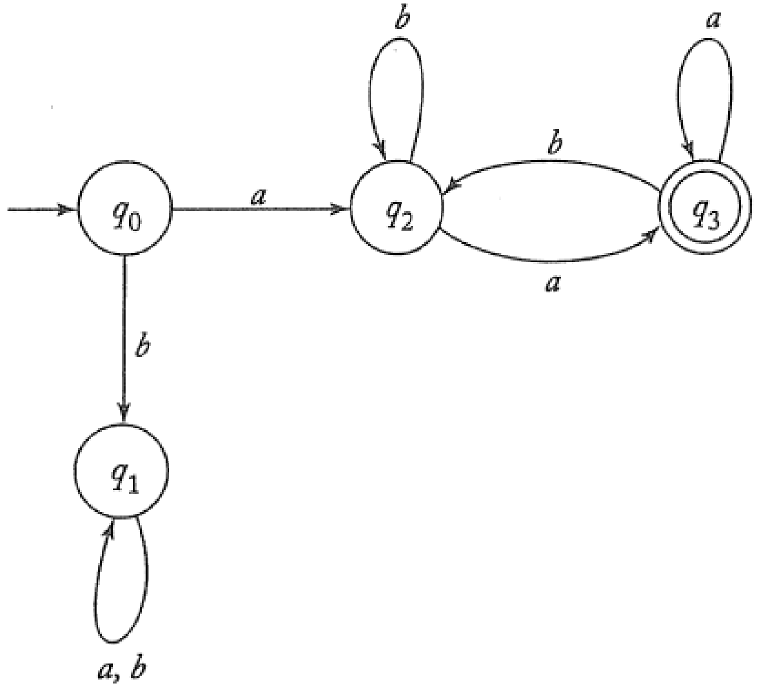
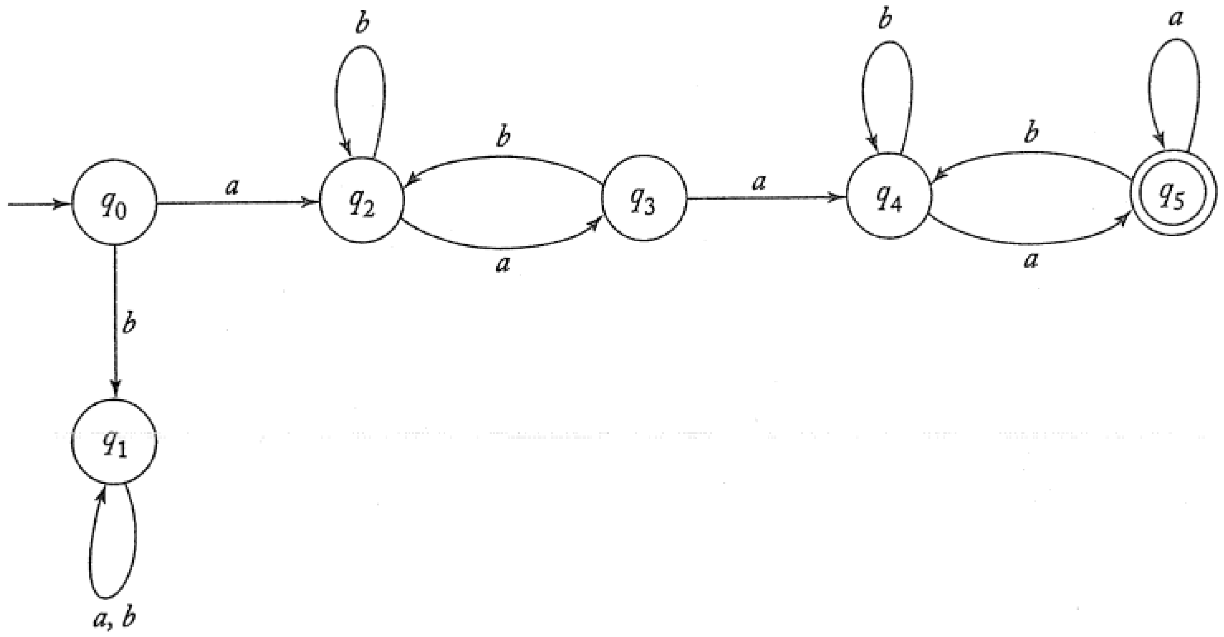


Figure 2.6

Example 2.6 Let L be the language in Example 2.5. Show that L^2 is regular. Again we show that the language is regular by constructing a dfa for it. We can write an explicit expression for L^2, namely,

$$L^2 = \left\{aw_1aaw_2a : w_1, w_2 \in \{a,b\}^*\right\}.$$

Figure 2.7



q_0
q_1
q_2
q_3
q_4
q_5
a
b
a, b