3.6 NONDETERMINISTIC FINITE STATE MACHINES

We explain the concept of nondeterministic finite automaton using a transition diagram (Fig. 3.7).

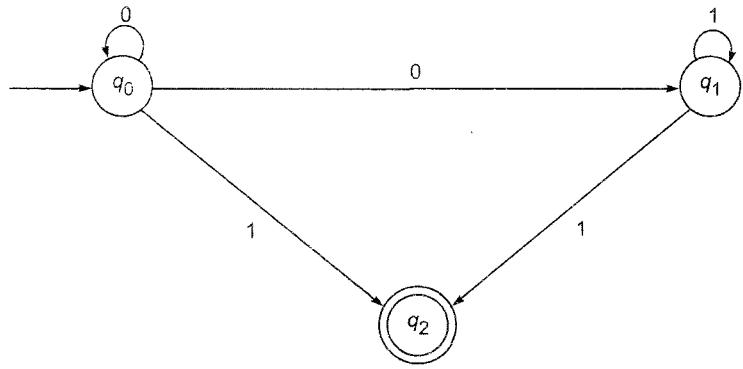


Fig. 3.7 Transition system representing nondeterministic automaton.

If the automaton is in a state $\{q_0\}$ and the input symbol is 0, what will be the next state? From the figure it is clear that the next state will be either $\{q_0\}$ or $\{q_1\}$. Thus some moves of the machine cannot be determined uniquely by the input symbol and the present state. Such machines are called nondeterministic automata, the formal definition of which is now given.

Definition 3.5 A nondeterministic finite automaton (NDFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

(i) Q is a finite nonempty set of states;
(ii) Σ is a finite nonempty set of inputs;
(iii) δ is the transition function mapping from $Q \times \Sigma$ into 2^Q which is the power set of Q, the set of all subsets of Q;
(iv) $q_0 \in Q$ is the initial state; and
(v) $F \subseteq Q$ is the set of final states.

We note that the difference between the deterministic and nondeterministic automata is only in δ. For deterministic automaton (DFA), the outcome is a state, i.e. an element of Q; for nondeterministic automaton the outcome is a subset of Q.

Consider, for example, the nondeterministic automaton whose transition diagram is described by Fig. 3.8.

The sequence of states for the input string 0100 is given in Fig. 3.9. Hence,

$$\delta(q_0, 0100) = \{q_0, q_3, q_4\}$$

Since q_4 is an accepting state, the input string 0100 will be accepted by the nondeterministic automaton.

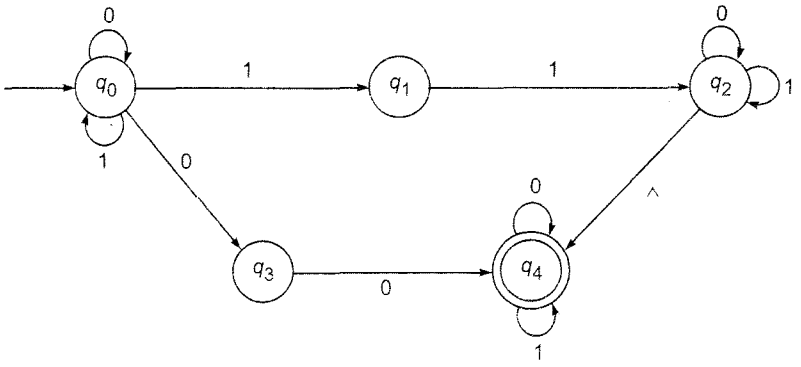


Fig. 3.8 Transition system for a nondeterministic automaton.

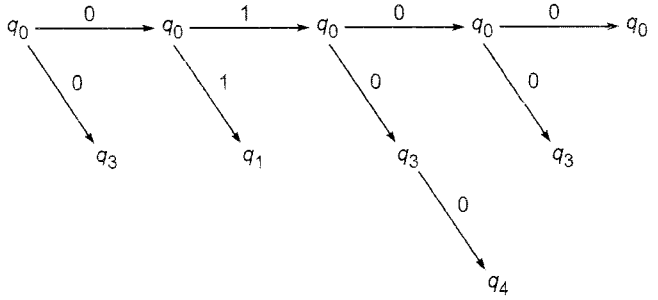


Fig. 3.9 States reached while processing 0100.

Definition 3.6 A string $w \in \Sigma^*$ is accepted by NDFA M if $\delta(q_0, w)$ contains some final state.

Note: As M is nondeterministic, $\delta(q_0, w)$ may have more than one state. So w is accepted by M if a final state is *one* among the possible states that M can reach on application of w.

We can visualize the working of an NDFA M as follows: Suppose M reaches a state q and reads an input symbol a. If $\delta(q, a)$ has n elements, the automaton splits into n identical copies of itself: each copy pursuing one choice determined by an element of $\delta(q, a)$. This type of parallel computation continues. When a copy encounters (q, a) for which $\delta(q, a) = \emptyset$, this copy of the machine 'dies'; however the computation is pursued by the other copies. If any one of the copies of M reaches a final state after processing the entire input string w, then we say that M accepts w. Another way of looking at the computation by an NDFA M is to assign a tree structure for computing $\delta(q, w)$. The root of the tree has the label q. For every input symbol in w, the tree branches itself. When a leaf of the tree has a final state as its label, then M accepts w.

Definition 3.7 The set accepted by an automaton M (deterministic or nondeterministic) is the set of all input strings accepted by M. It is denoted by $T(M)$.

3.7 THE EQUIVALENCE OF DFA AND NDFA

We naturally try to find the relation between DFA and NDFA. Intuitively we now feel that:

(i) A DFA can simulate the behaviour of NDFA by increasing the number of states. (In other words, a DFA $(Q, \Sigma, \delta, q_0, F)$ can be viewed as an NDFA $(Q, \Sigma, \delta', q_0, F)$ by defining $\delta'(q, a) = \{\delta(q, a)\}$.)

(ii) Any NDFA is a more general machine without being more powerful.

We now give a theorem on equivalence of DFA and NDFA.

Theorem 3.1 For every NDFA, there exists a DFA which simulates the behaviour of NDFA. Alternatively, if L is the set accepted by NDFA, then there exists a DFA which also accepts L.

Proof Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NDFA accepting L. We construct a DFA M' as:

$$M' = (Q', \Sigma, \delta, q'_0, F')$$

where

(i) $Q' = 2^Q$ (any state in Q' is denoted by $[q_1, q_2, \ldots, q_j]$, where $q_1, q_2, \ldots, q_j \in Q$);

(ii) $q'_0 = [q_0]$; and

(iii) F' is the set of all subsets of Q containing an element of F.

Before defining δ', let us look at the construction of Q', q'_0 and F'. M is initially at q_0. But on application of an input symbol, say a, M can reach any of the states $\delta(q_0, a)$. To describe M, just after the application of the input symbol a, we require all the possible states that M can reach after the application of a. So, M' has to remember all these possible states at any instant of time. Hence the states of M' are defined as subsets of Q. As M starts with the initial state q_0, q'_0 is defined as $[q_0]$. A string w belongs to $T(M)$ if a final state is one of the possible states that M reaches on processing w. So, a final state in M' (i.e. an element of F') is any subset of Q containing some final state of M.

Now we can define δ':

(iv) $\delta'([q_1, q_2, \ldots, q_i], a) = \delta(q_1, a) \cup \delta(q_2, a) \cup \ldots \cup \delta(q_i, a)$.

Equivalently,

$$\delta'([q_1, q_2, \ldots, q_i], a) = [p_1, \ldots, p_j]$$

if and only if

$$\delta(\{q_1, \ldots, q_i\}, a) = \{p_1, p_2, \ldots, p_j\}.$$

Before proving $L = T(M')$, we prove an auxiliary result

$$\delta'(q'_0, x) = [q_1, \ldots, q_i], \tag{3.4}$$

if and only if $\delta(q_0, x) = \{q_1, \ldots, q_i\}$ for all x in Σ^*.

We prove by induction on $|x|$, the 'if' part, i.e.

$$\delta'(q'_0, x) = [q_1, q_2, \ldots, q_i] \tag{3.5}$$

if $\delta(q_0, x) = \{q_1, \ldots, q_i\}$.

When $|x| = 0$, $\delta(q_0, \Lambda) = \{q_0\}$, and by definition of δ', $\delta'(q_0', \Lambda) = q_0' = [q_0]$. So, (3.5) is true for x with $|x| = 0$. Thus there is basis for induction.

Assume that (3.5) is true for all strings y with $|y| \leq m$. Let x be a string of length $m + 1$. We can write x as ya, where $|y| = m$ and $a \in \Sigma$. Let $\delta(q_0, y) = \{p_1, \ldots, p_j\}$ and $\delta(q_0, ya) = \{r_1, r_2, \ldots, r_k\}$. As $|y| \leq m$, by induction hypothesis we have

$$\delta'(q_0', y) = [p_1, \ldots, p_j] \tag{3.6}$$

Also,

$$\{r_1, r_2, \ldots, r_k\} = \delta(q_0, ya) = \delta(\delta(q_0, y), a) = \delta(\{p_1, \ldots, p_j\}, a)$$

By definition of δ',

$$\delta'([p_1, \ldots, p_j], a) = [r_1, \ldots, r_k] \tag{3.7}$$

Hence,

$$\begin{aligned} \delta'(q_0', ya) &= \delta'(\delta'(q_0', y), a) = \delta'([p_1, \ldots, p_j], a) && \text{by (3.6)} \\ &= [r_1, \ldots, r_k] && \text{by (3.7)} \end{aligned}$$

Thus we have proved (3.5) for $x = ya$.

By induction, (3.5) is true for all strings x. The other part (i.e. the 'only if' part), can be proved similarly, and so (3.4) is established.

Now, $x \in T(M)$ if and only if $\delta(q, x)$ contains a state of F. By (3.4), $\delta(q_0, x)$ contains a state of F if and only if $\delta'(q_0', x)$ is in F'. Hence, $x \in T(M)$ if and only if $x \in T(M')$. This proves that DFA M' accepts L. ∎

Note: In the construction of a deterministic finite automaton M_1 equivalent to a given nondeterministic automaton M, the only difficult part is the construction of δ' for M_1. By definition,

$$\delta'([q_1 \ldots q_k], a) = \bigcup_{i=1}^{k} \delta(q_i, a)$$

So we have to apply δ to (q_i, a) for each $i = 1, 2, \ldots, k$ and take their union to get $\delta'([q_1 \ldots q_k], a)$.

When δ for M is given in terms of a state table, the construction is simpler. $\delta(q_i, a)$ is given by the row corresponding to q_i and the column corresponding to a. To construct $\delta'([q_1 \ldots q_k], a)$, consider the states appearing in the rows corresponding to $q_1, \ldots, q_k$, and the column corresponding to a. These states constitute $\delta'([q_1 \ldots q_k], a)$.

Note: We write δ' as δ itself when there is no ambiguity. We also mark the initial state with $\rightarrow$ and the final state with a circle in the state table.

EXAMPLE 3.6

Construct a deterministic automaton equivalent to

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

where δ is defined by its state table (see Table 3.2).

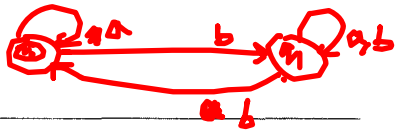


TABLE 3.2 State Table for Example 3.6

State/Σ	a	b
→ (q_0)	q_0	q_1
q_1	q_1	q_0, q_1

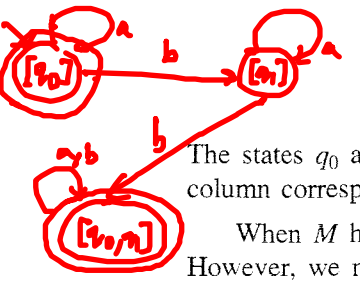
Solution

For the deterministic automaton M_1,

(i) the states are subsets of $\{q_0, q_1\}$, i.e. $\emptyset$, $[q_0]$, $[q_0, q_1]$, $[q_1]$;
(ii) $[q_0]$ is the initial state;
(iii) $[q_0]$ and $[q_0, q_1]$ are the final states as these are the only states containing q_0; and
(iv) δ is defined by the state table given by Table 3.3.

TABLE 3.3 State Table of M_1 for Example 3.6

State/Σ	0	1
→ $\emptyset$	$\emptyset$	$\emptyset$
A $[q_0]$	$[q_0]$	$[q_1]$
B $[q_1]$	$[q_1]$	$[q_0, q_1]$
C $[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$



The states q_0 and q_1 appear in the rows corresponding to q_0 and q_1 and the column corresponding to 0. So, $\delta([q_0, q_1], 0) = [q_0, q_1]$.

When M has n states, the corresponding finite automaton has 2^n states. However, we need not construct δ for all these 2^n states, but only for those states that are reachable from $[q_0]$. This is because our interest is only in constructing M_1 accepting $T(M)$. So, we start the construction of δ for $[q_0]$. We continue by considering only the states appearing earlier under the input columns and constructing δ for such states. We halt when no more new states appear under the input columns.

EXAMPLE 3.7

Find a deterministic acceptor equivalent to

$$M = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\})$$

where δ is as given by Table 3.4.

TABLE 3.4 State Table for Example 3.7

State/Σ	a	b
→ q_0	q_0, q_1	q_2
q_1	q_0	q_1
(q_2)		q_0, q_1

Solution

The deterministic automaton M_1 equivalent to M is defined as follows:

$$M_1 = (2^Q, \{a, b\}, \delta, [q_0], F')$$

where

$$F = \{[q_2], [q_0, q_2], [q_1, q_2], [q_0, q_1, q_2]\}$$

We start the construction by considering $[q_0]$ first. We get $[q_2]$ and $[q_0, q_1]$. Then we construct δ for $[q_2]$ and $[q_0, q_1]$. $[q_1, q_2]$ is a new state appearing under the input columns. After constructing δ for $[q_1, q_2]$, we do not get any new states and so we terminate the construction of δ. The state table is given by Table 3.5.

TABLE 3.5 State Table of M_1 for Example 3.7

State/Σ	a	b
→ $[q_0]$	$[q_0, q_1]$	$[q_2]$
$[q_2]$*	$\emptyset$	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_1, q_2]$
$[q_1, q_2]$*	$[q_0]$	$[q_0, q_1]$
→ $\emptyset$	$\emptyset$	$\emptyset$

EXAMPLE 3.8

Construct a deterministic finite automaton equivalent to

$$M = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}), \delta, q_0, \{q_3\})$$

where δ is given by Table 3.6.

TABLE 3.6 State Table for Example 3.8

State/Σ	a	b
→ q_0	q_0, q_1	q_0
q_1	q_2	q_1
q_2	q_3	q_3
(*) q_3		q_2

Solution

Let $Q = \{q_0, q_1, q_2, q_3\}$. Then the deterministic automaton M_1 equivalent to M is given by

$$M_1 = (2^Q, \{a, b\}, \delta, [q_0], F)$$

where F consists of:

$$[q_3], [q_0, q_3], [q_1, q_3], [q_2, q_3], [q_0, q_1, q_3], [q_0, q_2, q_3], [q_1, q_2, q_3]$$

and

$$[q_0, q_1, q_2, q_3]$$

and where δ is defined by the state table given by Table 3.7.

TABLE 3.7 State Table of M_1 for Example 3.8

State/Σ	a	b
$[q_0]$	$[q_0, q_1]$	$[q_0]$
$[q_0, q_1]$	$[q_0, q_1, q_2]$	$[q_0, q_1]$
$[q_0, q_1, q_2]$	$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_3]$
$[q_0, q_1, q_3]$	$[q_0, q_1, q_2]$	$[q_0, q_1, q_2]$
$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_2, q_3]$

3.8 MEALY AND MOORE MODELS

3.8.1 FINITE AUTOMATA WITH OUTPUTS

The finite automata which we considered in the earlier sections have binary output, i.e. either they accept the string or they do not accept the string. This acceptability was decided on the basis of reachability of the final state by the initial state. Now, we remove this restriction and consider the model where the outputs can be chosen from some other alphabet. The value of the output function $Z(t)$ in the most general case is a function of the present state $q(t)$ and the present input $x(t)$, i.e.

$$Z(t) = \lambda(q(t), x(t))$$

where λ is called the output function. This generalized model is usually called the *Mealy machine*. If the output function $Z(t)$ depends only on the present state and is independent of the current input, the output function may be written as

$$Z(t) = \lambda(q(t))$$

This restricted model is called the *Moore machine*. It is more convenient to use Moore machine in automata theory. We now give the most general definitions of these machines.

Definition 3.8 A Moore machine is a six-tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$, where

(i) Q is a finite set of states;
(ii) Σ is the input alphabet;
(iii) Δ is the output alphabet;
(iv) δ is the transition function $\Sigma \times Q$ into Q;
(v) λ is the output function mapping Q into Δ; and
(vi) q_0 is the initial state.

Definition 3.9 A Mealy machine is a six-tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$, where all the symbols except λ have the same meaning as in the Moore machine. λ is the output function mapping $\Sigma \times Q$ into Δ.

For example, Table 3.8 describes a Moore machine. The initial state q_0 is marked with an arrow. The table defines δ and λ.