CHAPTER 2 FINITE AUTOMATA

- Finite automata is a mathematical model with abstract inputs and outputs. This model is used for modeling many software and hardware systems. Therefore, finite automaton (FA) models are widely used for computer science and engineering.
- As mentioned earlier, a finite automaton consists of a set of states, and its ’control’ moves from state to state in response to external ‘inputs’. The most important distinction in the classification of finite automata is that the automaton is specified as a deterministic or non-deterministic model.

- The deterministic model means that the automaton can never be in more than one state, while the non-deterministic model means that the automaton can be in more than one state at a time. However, as we will learn in this chapter, the non-deterministic model can be converted into a deterministic model with various algorithms.

2.1 Deterministic Finite Automata

- The term ‘deterministic’ refers to the fact that the various inputs applied to the automaton correspond to only one state, and the outputs obtained by transitions from the state of the automaton. The deterministic model is defined as a quintuple as follows :

$$M =< Q, \Sigma, \delta, q_0, F >$$

Q: is a finite set of internal states

Σ: is a finite set of symbols called input alphabet

q_0: initial state $q_0 \in Q$

δ: $(Q\ x\Sigma) \longrightarrow Q$ is a total function called transition function

F: is a set of final states F$\subseteq Q$

- In the basic model, one (state) is mapped to each (state, input symbol) pair with the transition function. If the first of these states is called “current state” and the second is called “next state”, it can be said that one (next state) is mapped to each (current state, input symbol) pair with the state transition function. Mathematically, it is as follows;

$$\delta(q_i, a) = q_j\ , \qquad q_j\ \in Q$$

- As understood from the above definition, the basic FA model is a deterministic model. This model is called DFA (deterministic finite automata) for short. Therefore, when FA (finite automaton) is called, the DFA model is understood. According to this model, FA starts from state q_0 and moves to a new state with each input symbol applied. At any moment, the state of FA is clearly defined.
- For example, if

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

then if the dfa is in state q_0 and the current input symbol is a, the dfa will go into state q_1.

Example 2.1

$M_1 =< Q, \Sigma, \delta, q_0, F >$

$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{0,1\}$
$F = \{q_2\}$
$\delta(q_0, 0) = \{q_0\}$
$\delta(q_0, 1) = \{q_1\}$
$\delta(q_1, 0) = \{q_0\}$
$\delta(q_1, 1) = \{q_2\}$
$\delta(q_2, 0) = \{q_2\}$
$\delta(q_2, 1) = \{q_2\}$

Transition Diagram

- As seen in Example 2.1, the mathematical definition of the transition function is both long and difficult to understand. For this reason, a so-called "transition diagram" is often used for the definition of the transition function.
- The transition diagram, which is a directed graph, has a node for each state. State transitions are indicated by directional arcs and the input symbol that causes the transition is written on the arcs. In the transition graph, the initial state is indicated by “→” and the final states are indicated by a double circle. The transition diagram created for the FA in Example 2.1 is shown in Figure 2.1.

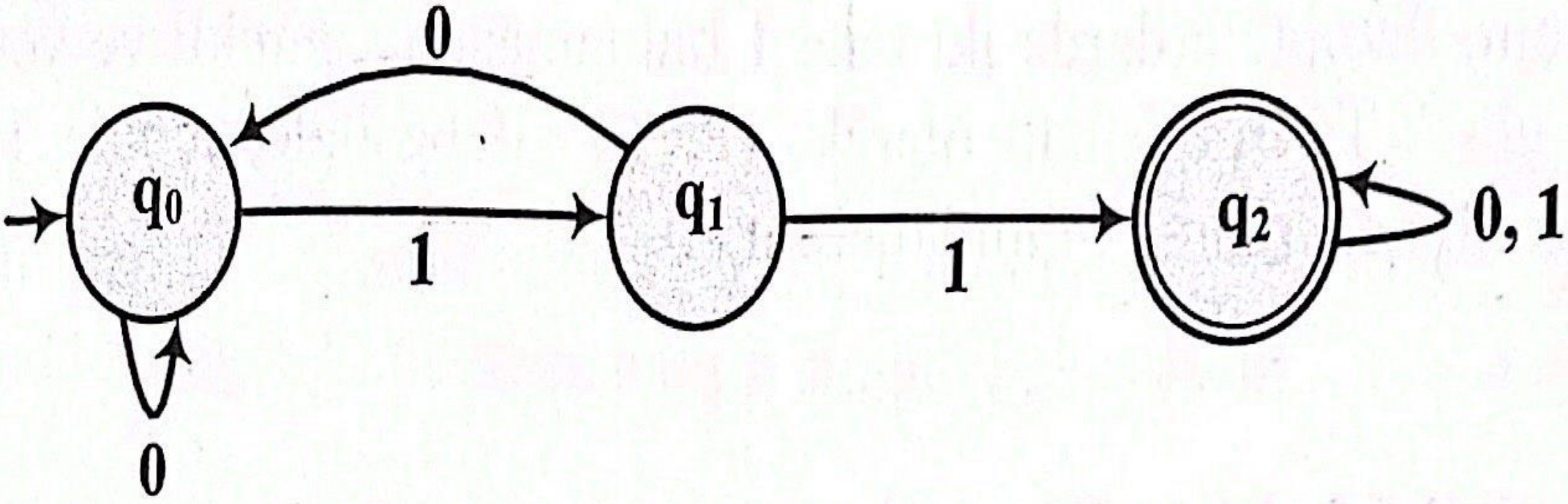


Figure 2.1 Transition graph for DFA model of M_1

Languages and DFA's

Definition 2.1: $M =< Q, \Sigma, \delta, q_0, F >$ is a DFA and the language accepted by DFA is the set of all strings accepted by M defined on Σ.

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

- The set of strings applied to DFA is divided into accepted and not accepted by DFA.

- The language accepted by the DFA is the set of strings that take the DFA from its initial state to a final state.

- If we examine the DFA described in Example 2.1, the string 0101 is not accepted by M_1. Because after reading 0101, it will be in state q_1 and q_1 is not a final state. However the string 0110 take M_1 from initial state q_0 to final state q_2. The set of strings that M_1 accepts is an infinite set.

$$L(M_1) = \{11,011,110,0110,01011, \dots\}$$

2.2 Nondeterministic Finite Automata

- The concept of a finite automaton is quite complex in its non-deterministic form. However, it is useful because it is not deterministic because it accommodates multiple movements in automatons.

- A non-deterministic finite automaton (NFA) is defined as a quintuple as follows:

$$M =< Q, \Sigma, \delta, q_0, F >$$

- In this definition, the meanings of Q,Σ, q_0 and F are the same as in the definition of the deterministic model. As in the deterministic model, in the non-deterministic model, Q represents the set of states, Σ the input alphabet, q_0 the initial state, and F the set of final states. The only difference between the two models is in the definition of the transition function. Transition function in deterministic model is defined as a mapping;

$$\text{from } (Q x \Sigma) \text{ to } Q$$

while in the non-deterministic model the transition function is a maping;

$$\text{from } (Q x \Sigma) \text{ to the subsets of } Q$$

- Accordingly, while in the deterministic model, one and only one state can be passed from each state with each input symbol, in the non-deterministic model, the number of states that can be passed from a state with an input symbol can be zero, one or many. Due to the difficulty of using the deterministic model, a non-deterministic model, which is easier to use and more flexible, has been developed.

- Consider the transition graph in Figure 2.2. It describes a nondeterministic accepter since there are two transitions labeled *a* out of q_0.

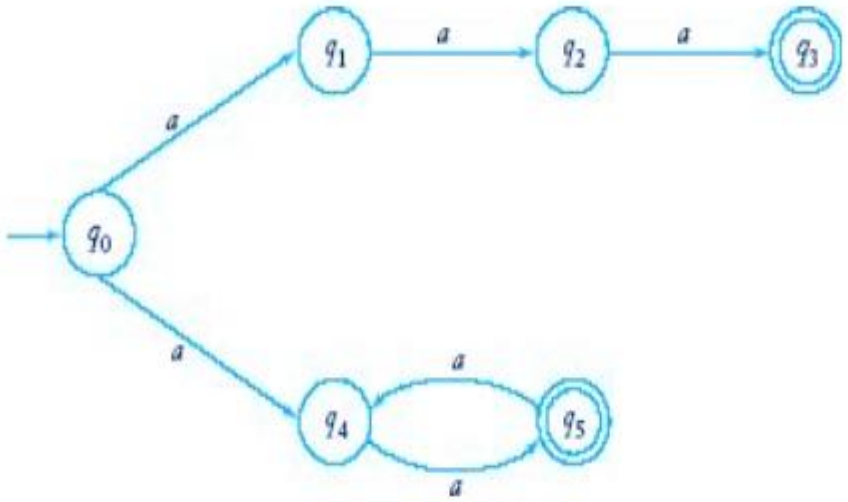


Figure 2.2

Example 2.2

$M_2 =< Q, \Sigma, \delta, q_0, F >$

$Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{0,1\}$

$F = \{q_3\}$

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

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

$\delta(q_1, 0) = \{q_3\}$

$\delta(q_1, 1) = \{\ \} = \phi$

$\delta(q_2, 0) = \phi$

$\delta(q_2, 1) = \{q_3\}$

$\delta(q_3, 0) = \{q_3\}$

$\delta(q_3, 1) = \{q_3\}$

- As seen in the definition of M_2, in the non-deterministic model, it is possible to switch from some states to more than one state with some input symbols. From some states, it is not possible to switch to any state with some input symbols. Therefore, in the deterministic model, it is precisely known what state the machine will be in when the initial state and the applied input string are known. Conversely, in a non-deterministic model, the initial state and the final state of the machine whose applied input string is known may be uncertain. For example, when the input string w=000 is applied to machine M_2 with initial state q_0, the final state of the machine can be any of q_0, q_1 and q_3.

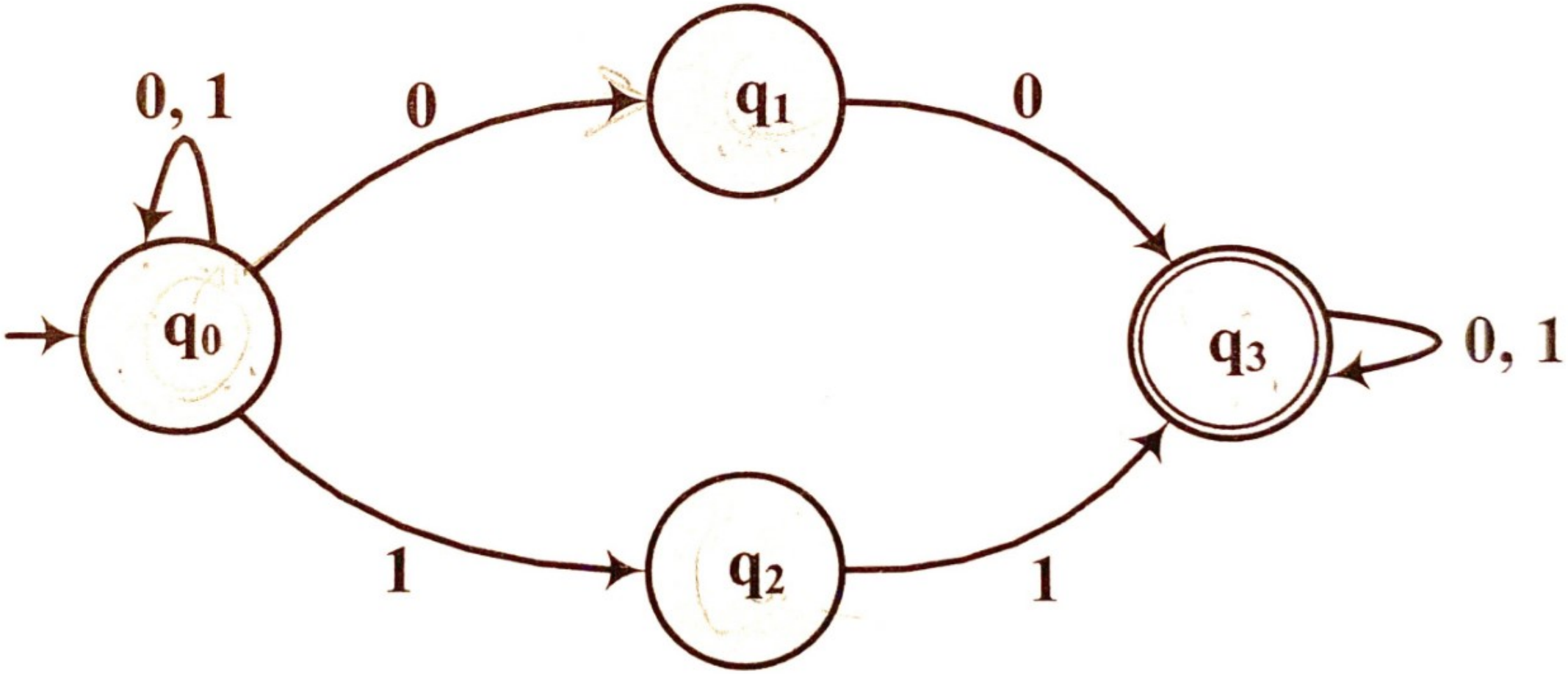


Figure 2.2 Transition graph for NFA model of M_2

- For a non-deterministic finite automaton (NFA) to recognize an input string, a path must be found in the transition graph starting from the initial state and reaching a final state with that input string. For example, M_3 recognizes the input string w=10001. Because this is a path corresponding to the input string, starting from q_0 and ending in the only final state q_3.

- In contrast, M_3 does not accept the input string w=010. Because between cases q_0 ve q_3 it is not possible to find a corresponding path to this input string.

Why Nondeterminism?

- Non-determinism is a difficult concept. Digital computers are completely deterministic; their state at any time is uniquely predictable from the input and the initial state. Thus it is natural to ask why we study nondeterministic machines at all. We are trying to model real systems, so why include such nonmechanical features as choice? We can answer this question in various ways.
- A nondeterministic algorithm that can make the best choice would be able to solve the problem without backtracking, but a deterministic one can simulate nondeterminism with some extra work. For this reason, nondeterministic machines can serve as models of search-and backtrack algorithms.
- Nondeterminism is an effective mechanism for describing some complicated languages concisely.
- As we will see, certain theoretical results are more easily established for nfa's than for dfa's

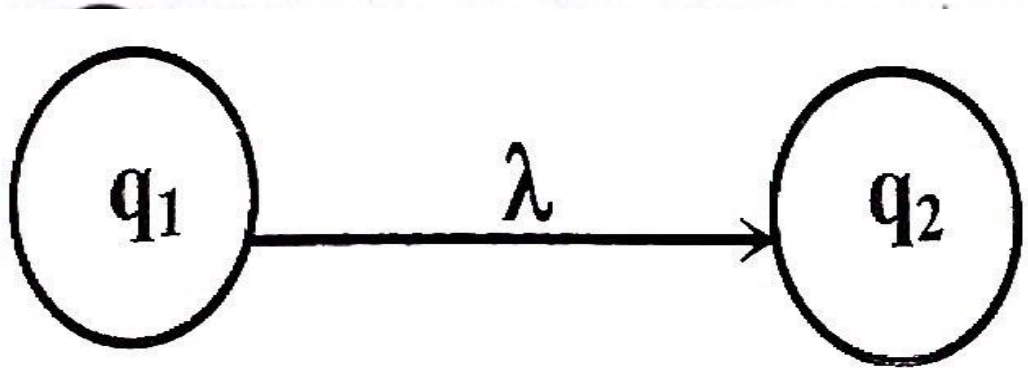
2.3 Lambda (λ) Transition

- In finite automata, state transitions occur by processing input symbols. While the machine is in a certain state, applying an input symbol causes the machine to switch to the next state. The "next state" defined by the transition function is a single state in the deterministic model. In the non-deterministic model, on the other hand, the "next state" is a subset of the set of states, which can contain zero, one, or many states. According to the definitions made so far, a machine conforming to the DFA or NFA model maintains its certain state unless the input symbol is applied.

- With the lambda (λ) transition, the definition of finite automata, which is extended and facilitated by the non-deterministic model, is further extended. Lambda and lambda-transition are abstract concepts. Lambda can be thought of as an empty symbol. Lambda-transition, on the other hand, opposes a state transition without any input symbols applied (or processed). Between the q_1 and q_2 states of a machine,

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

If there is a λ-transition defined as above , it is understood that the machine in the q_1 state can automatically switch to the q_2 state without any input symbols being processed.


q_1
λ
q_2

Example 2.3 To illustrate the flexibility that lambda transitions provide to the finite automata model, in the set {0, 1, 2}, Let's consider the machine as;

$$L(M_4) = \{0^{2n}1^{2m}\ 2^{2k} \ \big|\ n \geq 0, m \geq 0, k \geq 0\}$$

The transition plot of M_4, created without using λ-transition, is shown in Drawing 2.3.a, and the transition plot created using λ-transition is shown in Figure 2.3.b. When the λ-intransitive and λ-transitive transition schemes are examined, the flexibility and ease of use that λ-transitions bring to the finite automata model is clearly seen.

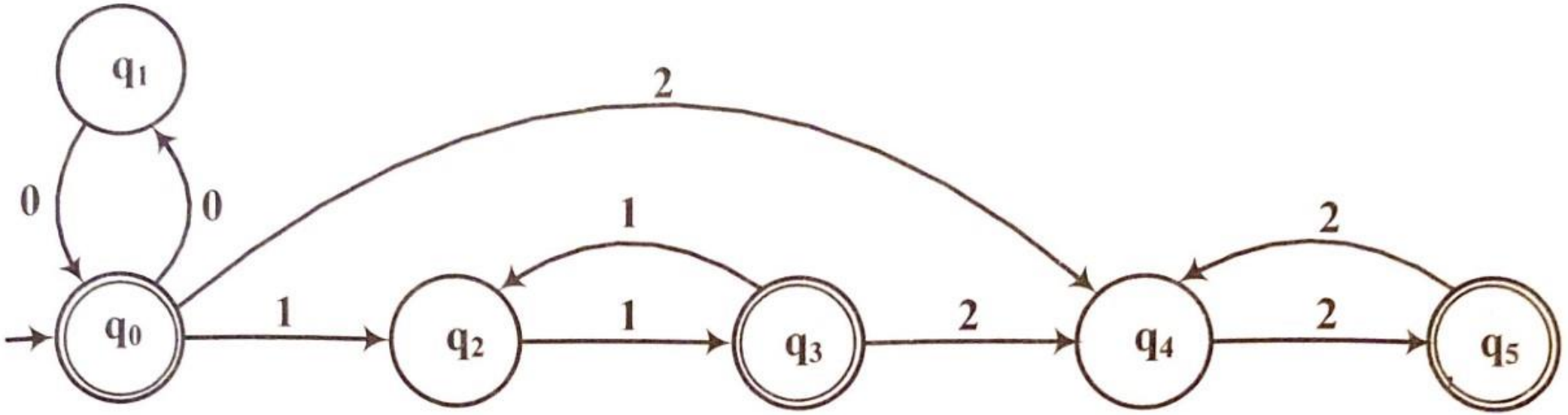


Figure 2.3a without λ − transition

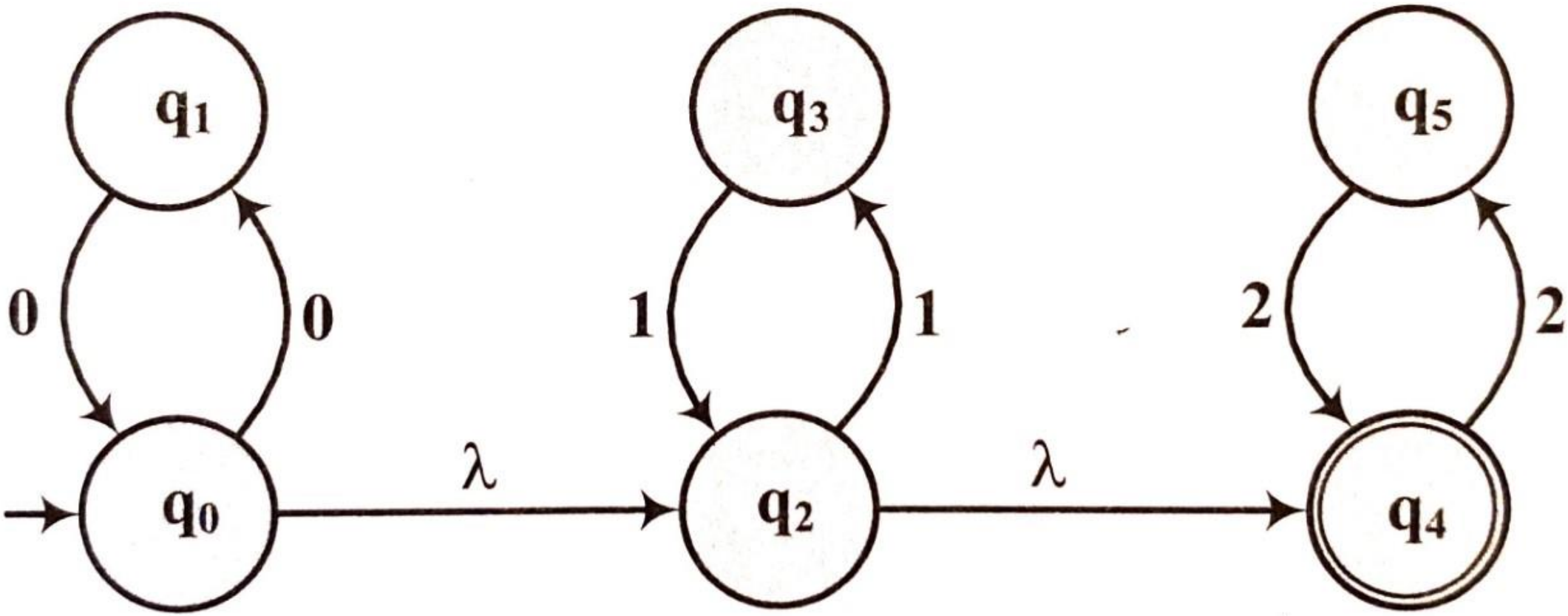


Figure 2.3b with λ – transition

Finding the λ-intransitive Equivalent Transition Graph

- The use of λ-transitions makes the transition graphs easier to use, but does not increase their power. In other words, for every transition graphs created using λ-transitions, an equivalent λ-intransitive transition graph can be found. The equivalence of the two transition schemes means the equality of the set they recognize. Accordingly, there is no set of strings recognized by a λ-transitive transition graph, but not recognized by any λ-intransitive transition graph. This indicates that using λ-transition in transition graph makes no change in the expressive power of the NFA model.

- Given a λ-transition graph, it is possible to eliminate λ-transitions one by one to obtain an equivalent transition graph. In this context, if there is a λ-transition from q_1 to q_2 between states q_1 and q_2;

1. For each state transition starting from the q_2 state, a state transition is added starting from the q_1 state and reaching the same state with the same input symbol.
2. if q_1 is the initial state, then q_2 is also made the initial state.
3. if q_2 is the final state, then q_1 is also made the final state.

After all, λ-transition can be deleted.

Example 2.4 Let the M_5 machine recognizes a set of strings in the alphabet {a,b,c}, starting with zero or two a or an even number of b and ending with cc. Here are a few example of strings in the set M_5 accepts;

$$L(M_5) = \{cc, aacc, bbcc, bbbbcc, bbbbbbcc, \ldots.\}$$

- The λ-transitive transition graph created for M_5 is shown in figure 2.4.a. In order to obtain the λ-intransitive transition graph equivalent to this diagram, the λ-transitions between q_2 and q_4 , and then between q_0 and q_2 are eliminated. In order to eliminate the λ-transition between q_2 and q_4 , a c-transition is added between q_2, ile q_5 and the transition diagram in Figure 2.4.b is obtained. In order to eliminate the λ-transition between q_0 and q_2 , b-transition between $q_0$and q_3 is added, c-transition is added between q_0 ile q_5 and the initial state of q_2 is made, and the λ-intransitive transition diagram in Figure 2.4.c is obtained.

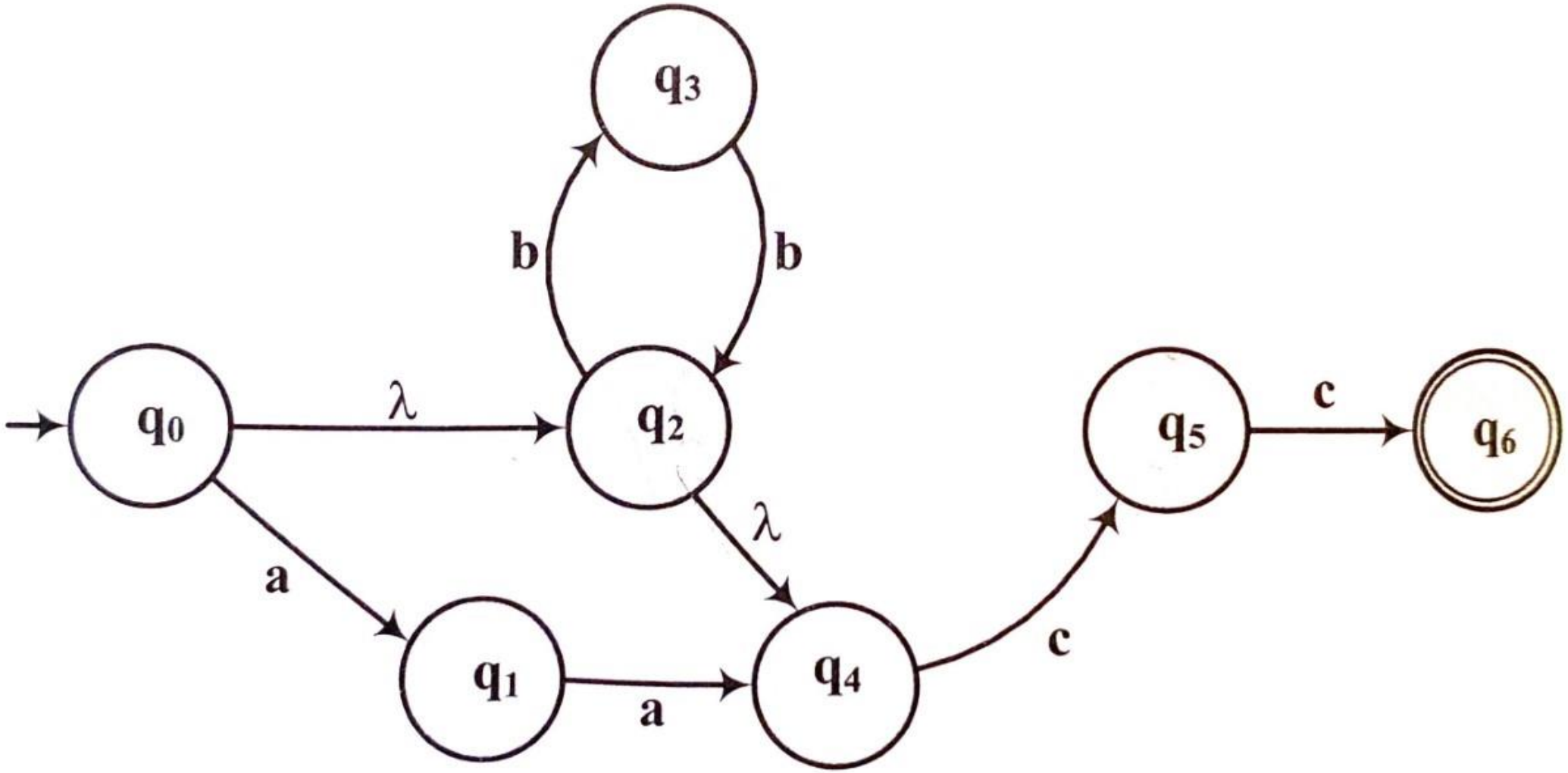


Figure 2.4a

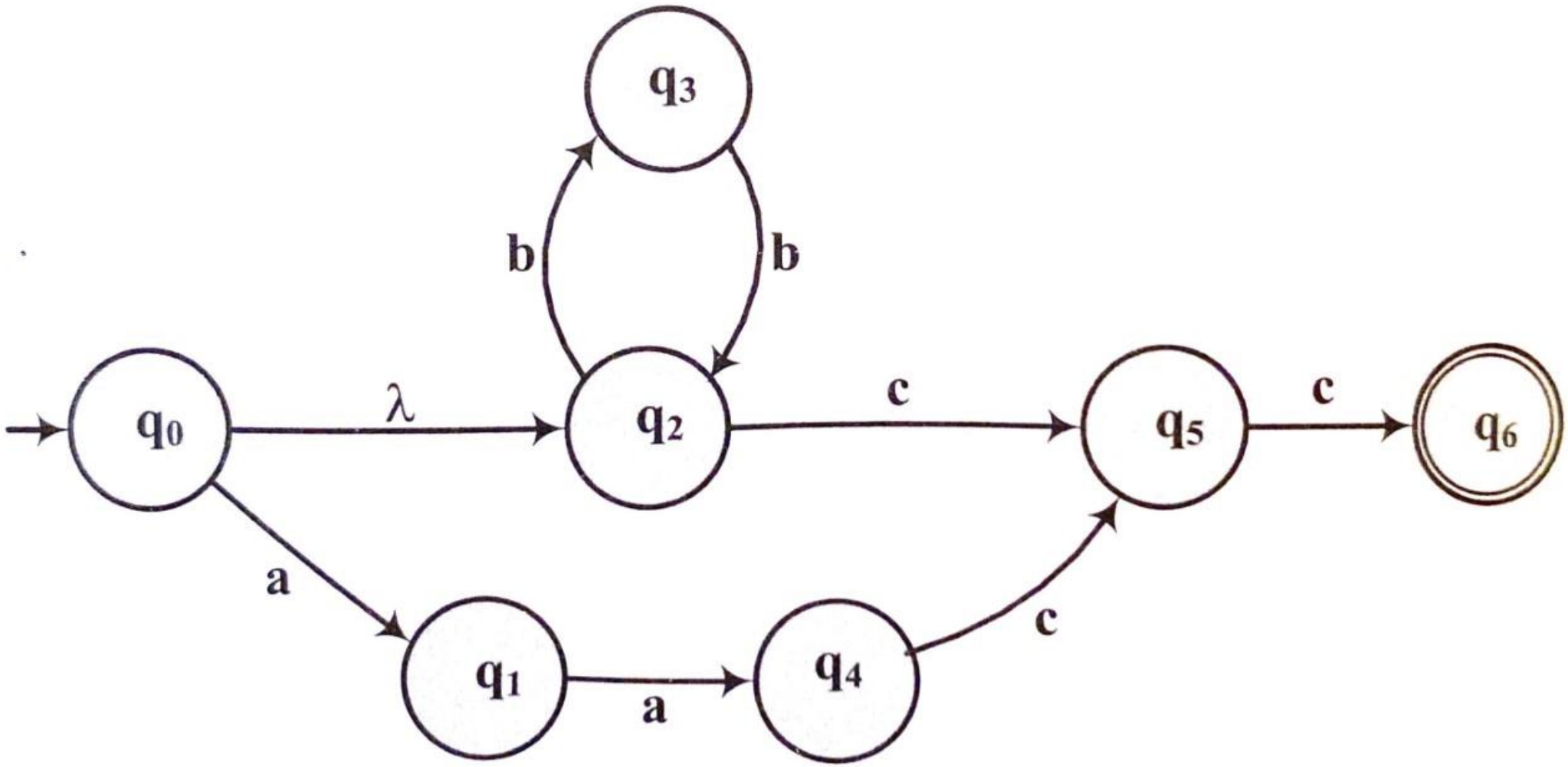


Figure 2.4b

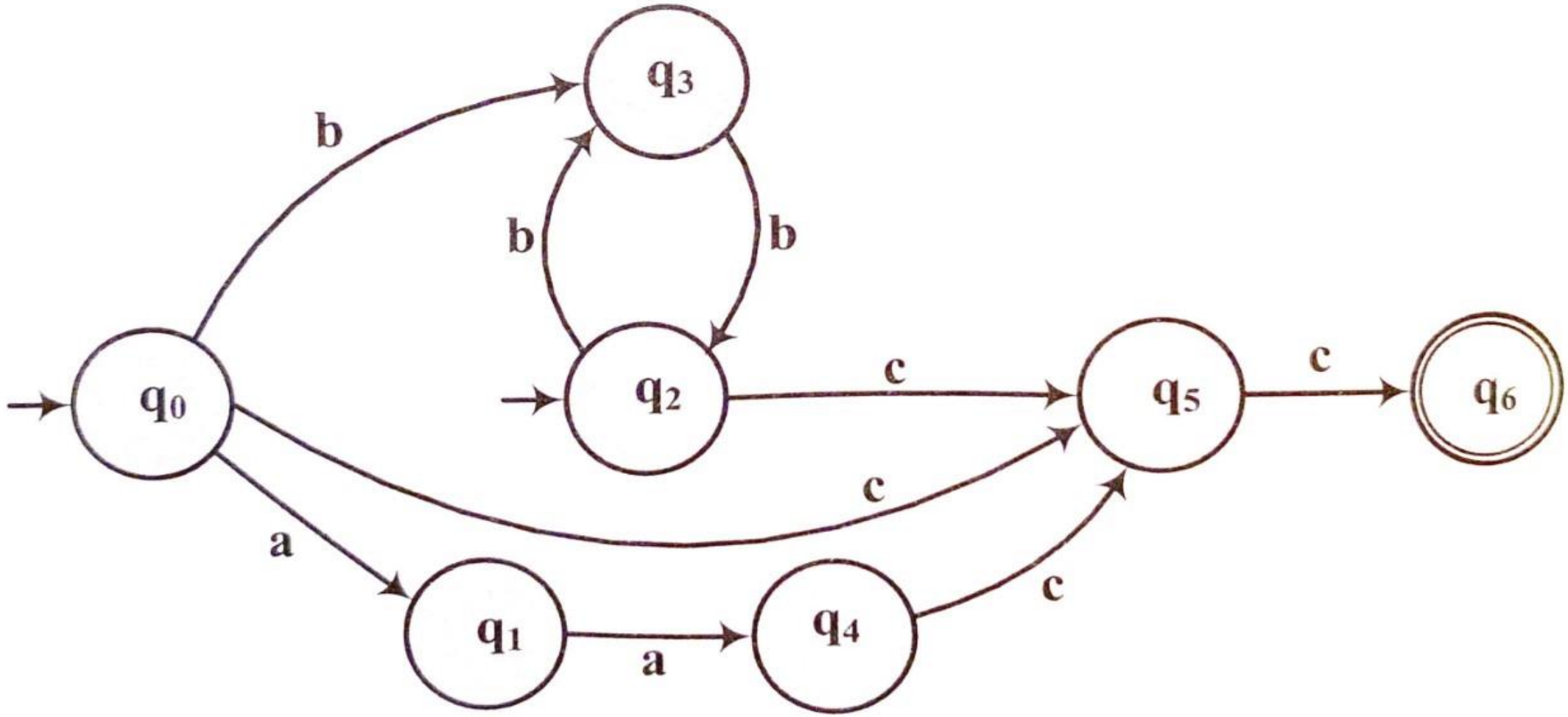


Figure 2.4c

- If there is more than one λ-transition in the chain structure in the transition graph whose λ-transitions will be destroyed, it is useful to start the elimination process from the end of the chain and continue it backwards.

2.4 Equivalence of Deterministic and Non-deterministic Finite Automata Models

- For deterministic and non-deterministic automata, if the class of sets recognized by both models is the same, they are said to be equivalent. Since, by definition, the non-deterministic model includes the deterministic model, in other words, since every DFA is also an NFA, a set accepted by a deterministic FA but not accepted by any NFA is unthinkable. But for every set accepted by the NFA, is it possible to find a DFA that recognizes that cluster? If the answer to this question is yes, it can be said that deterministic and non-deterministic models are equivalent.

- Let's try to find the answer to the above question by looking for an answer to another question. Given a string and a DFA, it is easy to find out if that string is recognized by this DFA using the transition graph. However, given a string and an NFA, it is not easy to find out if that string is recognized by this NFA using the transition graph.

Example 2.5 $M_6 =< Q, \Sigma, \delta, q_0, F >$

$Q = \{A, B, C\}$

$\Sigma = \{0,1\}$

$q_0 = A$

$F = \{C\}$

$\delta(A, 0) = \{A\}$

$\delta(A, 1) = \{B, C\}$

$\delta(B, 0) = \{B\}$

$\delta(B, 1) = \{A, C\}$

$\delta(C, 0) = \{A, B\}$

$\delta(C, 1) = \{C\}$

- Let's consider M_6 as an example. The transition graph of this machine, which is defined according to the non-deterministic model, is shown in figure 2.5.a. Since the initial state is A and the only final state is C, for a given string to be accepted by M_6, a corresponding path must be found between states A and C. Let's try to find a way to do this in a systematic way. For this purpose, we can make use of the transition table seen in figure 2.5.b.

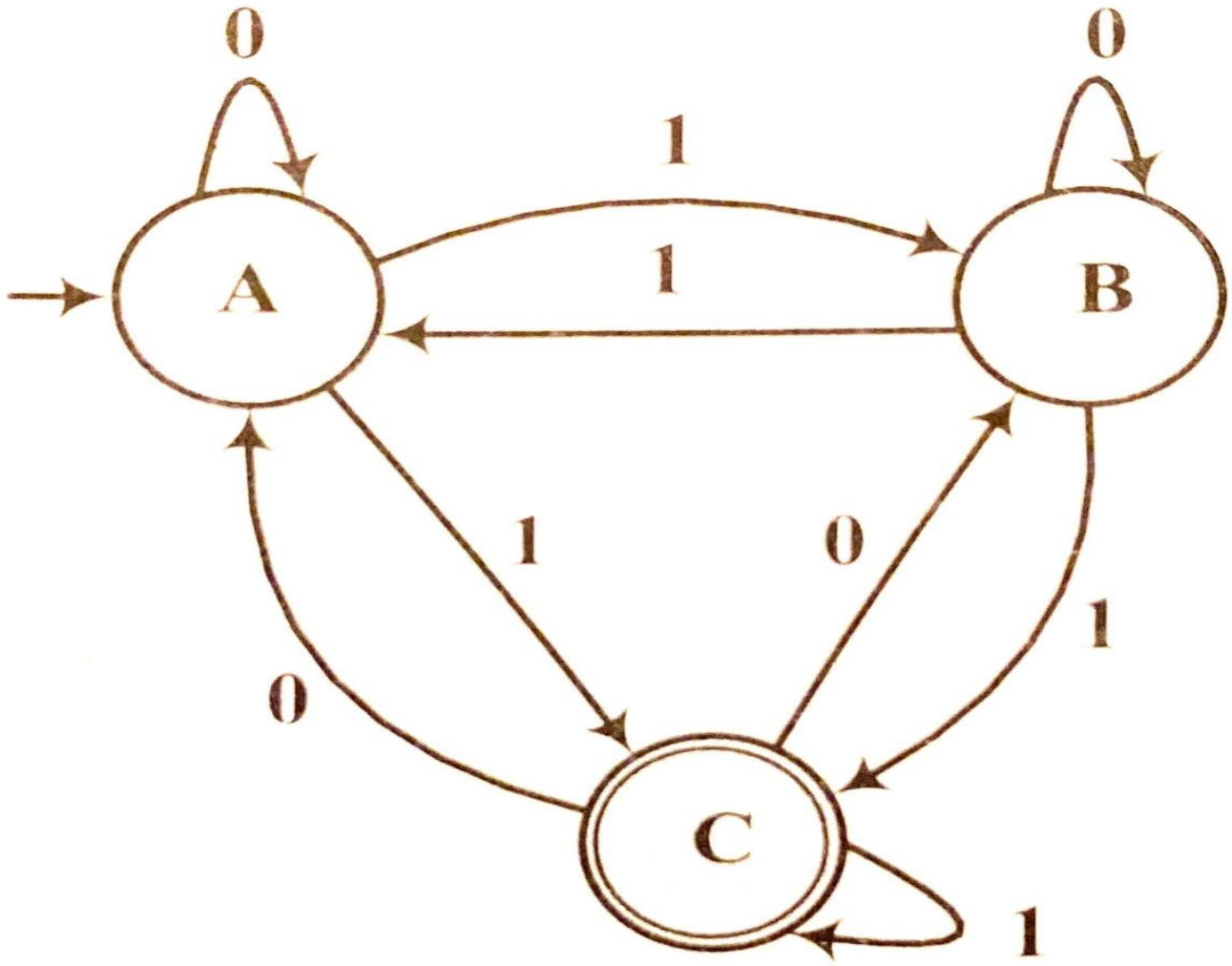


Figure 2.5a

	0	1
→ A	A	BC
B	B	AC
©	AB	C

Figure 2.5b

- The transition table is a chart that contains the same information as the transition diagram and shows which states can be reached from each state of the machine with each input symbol. In other words, the transition table makes it easy to find the successors of each state corresponding to each input symbol. However, for a given input string w to be recognized by M_6 , the only final state C must be present among the w-successors of the initial state A.

- For this purpose, we can prepare the chart shown in Figure 2.5.c that contains all the successors of the initial state. Given an input string w, it can be easily found with the help of this successor table whether w is accepted by M_6 . For example, if w=011, then A is 011-sequence AC and the string 011 is recognized by M_6 because AC contains an final case; on the other hand, if w = 110, it can easily be found that the 110-successor of A is AB and the string 110 is not recognized by M_6 , since there is no final state in AB.

	0	1
→ A	A	BC
(BC)	AB	AC
AB	AB	ABC
(AC)	AB	BC
(ABC)	AB	ABC

Figure 2.5c

- A deterministic automaton is defined by the chart in figure 2.5.c showing the successors of M_6. If we call it M_{D6}, it can be formally defined as follows:

$M_{D5}=< Q, \Sigma, \delta, q_0, F >$

$Q = \{A, BC, AB, AC, ABC\}$

$\Sigma = \{0,1\}$

$q_0 = A$

$F = \{BC, AC, ABC\}$

$\delta(A, 0) = A$

$\delta(A, 1) = BC$

$\delta(BC, 0) = AB$

$\delta(AB, 0) = AB$

$\delta(AB, 1) = ABC$

$\delta(AC, 0) = AB$

$\delta(AC, 1) = BC$

$\delta(ABC, 0) = AB$

$\delta(ABC, 1) = ABC$

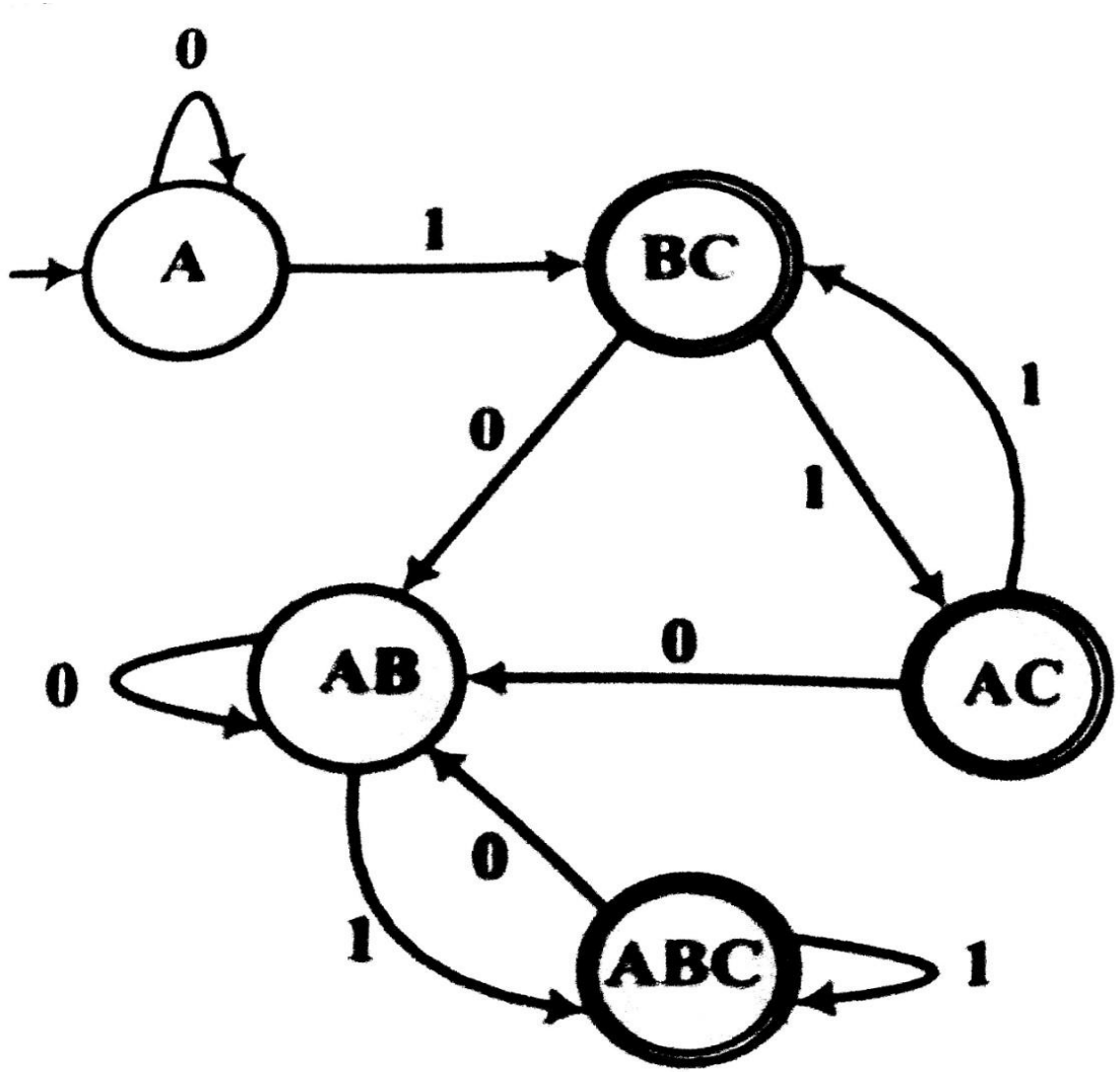


Figure 2.5d