Syllabus

CHHATTISGARH SWAMI VIVEKANAND TECHNICAL UNIVERSITY, BHILAI (C.G.)

Semester – B.E. V
Subject: **Theory of Computation**
Total theory periods-**40**
Total marks in end semester exam – **80**
Minimum number of class tests to be conducted – **02**

Branch-**Computer Science & Engineering.**
Code –**322514 (22)**
Total Tutorial Periods: **12**

UNIT-1. THE THEORY OF AUTOMATA :

Introduction to automata theory, Examples of automata machine, Finite automata as a language acceptor and translator. Deterministic finite automata. Non deterministic finite automata, finite automata with output (Mealy Machine. Moore machine). Finite automata with ? moves, Conversion of NFA to DFA by Arden's method, Minimizing number of states of a DFA. My hill Nerode theorem, Properties and limitation of FSM. Two way finite automata. Application of finite automata.

UNIT-2. REGULAR EXPRESSIONS :

Regular expression, Properties of Regular Expression. Finite automata and Regular expressions. Regular Expression to DFA conversion & vice versa. Pumping lemma for regular sets. Application of pumping lemma, Regular sets and Regular grammar. Closure properties of regular sets. Decision algorithm for regular sets and regular grammar.

Syllabus

UNIT-3. GRAMMARS.

Definition and types of grammar. Chomsky hierarchy of grammar. Relation between types of grammars. Role and application areas of grammars. Context free grammar. Left most linear & right most derivation trees. Ambiguity in grammar. Simplification of context free grammar. Chomsky normal from. Greibach normal form, properties of context free language. Pumping lemma from context free language. Decision algorithm for context tree language.

UNIT-4. PUSH DOWN AUTOMATA AND TURING MACHINE.

Basic definitions. Deterministic push down automata and non deterministic push down automata. Acceptance of push down automata. Push down automata and context free language. Turing machine model. Representation of Turing Machine Construction of Turing Machine for simple problem's. Universal Turing machine and other modifications. Church's Hypothesis. Post correspondence problem. Halting problem of Turing Machine

UNIT-5 COMPUTABILITY

Introduction and Basic concepts. Recursive function. Partial recursive function. Partial recursive function. Initial functions, computability, A Turing model for computation. Turing computable functions, Construction of Turing machine for computation. Space and time complexity. Recursive enumerable language and sets.

Text Books

(1) Theory of Computer Science (Automata Language & Computation), K.L.P. Mishra and N. Chandrasekran, PHI.

(2) Introduction to Automata theory. Language and Computation, John E. Hopcropt & Jeffery D. Ullman, Narosa Publishing House.

Reference Books

(1) Theory of Automata and Formal Language, R.B. Patel & P. Nath, Umesh Publication.

(2) An Indtroduction and finite automata theory, Adesh K. Pandey, TMH.

(3) Theory of Computation, AM Natrajan. Tamilarasi, Bilasubramani, New Age International Publishers.

Unit-I
Theory of Automata

Introduction to automata theory, Examples of automata machine, Finite automata as a language acceptor and translator. Deterministic finite automata. Non deterministic finite automata, finite automata with output (Mealy Machine. Moore machine). Finite automata with ? moves, Conversion of NFA to DFA by Arden's method, Minimizing number of states of a DFA. My hill Nerode theorem, Properties and limitation of FSM. Two way finite automata. Application of finite automata.

Introduction to Automata Theory

1 DEFINITION OF AN AUTOMATON

/e shall give the most general definition of an automaton and later modify it to mputer applications. An automaton is defined as a system where energy, materials d information are transformed, transmitted and used for performing some functions ithout direct participation of man. Examples are automatic machine tools, automatic acking machines, and automatic photo printing machines.

In Computer Science the term 'automaton' means "discrete automaton" and is efined in a more abstract way as shown in Fig. 2.1.

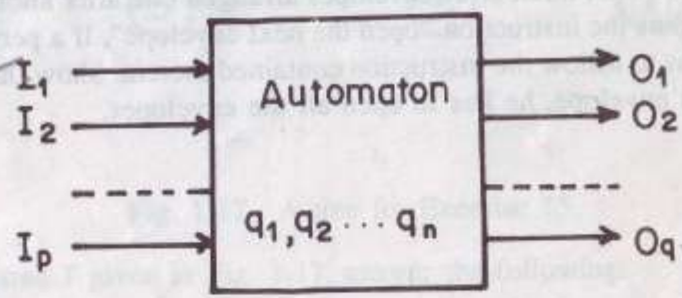


Fig. 2.1 Model of a discrete automaton.

Introduction to Automata Theory

Its characteristics are now described.

(i) *Input.* At each of the discrete instants of time $t_1, t_2, \ldots,$ input values $I_1, I_2, \ldots,$ each of which can take a finite number of fixed values from the input alphabet Σ, are applied to the input side of model shown in Fig. 2.1.

(ii) *Output.* $O_1, O_2, \ldots, O_q$ are the outputs of the model, each of which can the finite numbers of fixed values from an output O.

(iii) *States.* At any instant of time the automaton can be in one of the states $q_1, q_2, \ldots, q_n$.

(iv) *State relation.* The next state of an automaton at any instant of time is determined by the present state and the present input.

(v) *Output relation.* Output is related to either state only or to both the input and the state. It should be noted that at any instant of time the automaton is in some state. On 'reading' an input symbol, the automaton moves to a next state which is given by the state relation.

NOTE: An automaton in which the output depends only on the input is called an automaton without a memory. An automaton in which the output depends on the states also is called automaton with a finite memory. An automaton in which the output depends only on the states of the machine is called a *Moore machine*. An automaton in which the output depends on the state and the input at any instant of time is called a *Mealy machine*.

Introduction to Automata Theory

efinition 2.1 Analytically, a finite automaton can be represented by a 5-tuple $, \Sigma, \delta, q_0, F)$, where

(i) Q is a finite nonempty set of states;

(ii) Σ is a finite nonempty set of inputs called input alphabet;

(iii) δ is a function which maps $Q \times \Sigma$ into Q and is usually called direct nsition function. This is the function which describes the change of states ring the transition. This mapping is usually represented by a transition table a transition diagram.

(iv) $q_0 \in Q$ is the initial state; and

(v) $F \subseteq Q$ is the set of final states. It is assumed here that there may be ore than one final state.

TE: The transition function which maps $Q \times \Sigma^*$ into Q (i.e. maps a state and tring of input symbols including the empty string into a state) is called indirect nsition function. We shall use the same symbol δ to represent both types of nsition functions and the difference can be easily identified by nature of mapping ymbol or a string), i.e. by the argument. δ is also called the next state function. e above model can be represented graphically by Fig. 2.4.

Introduction to Automata Theory

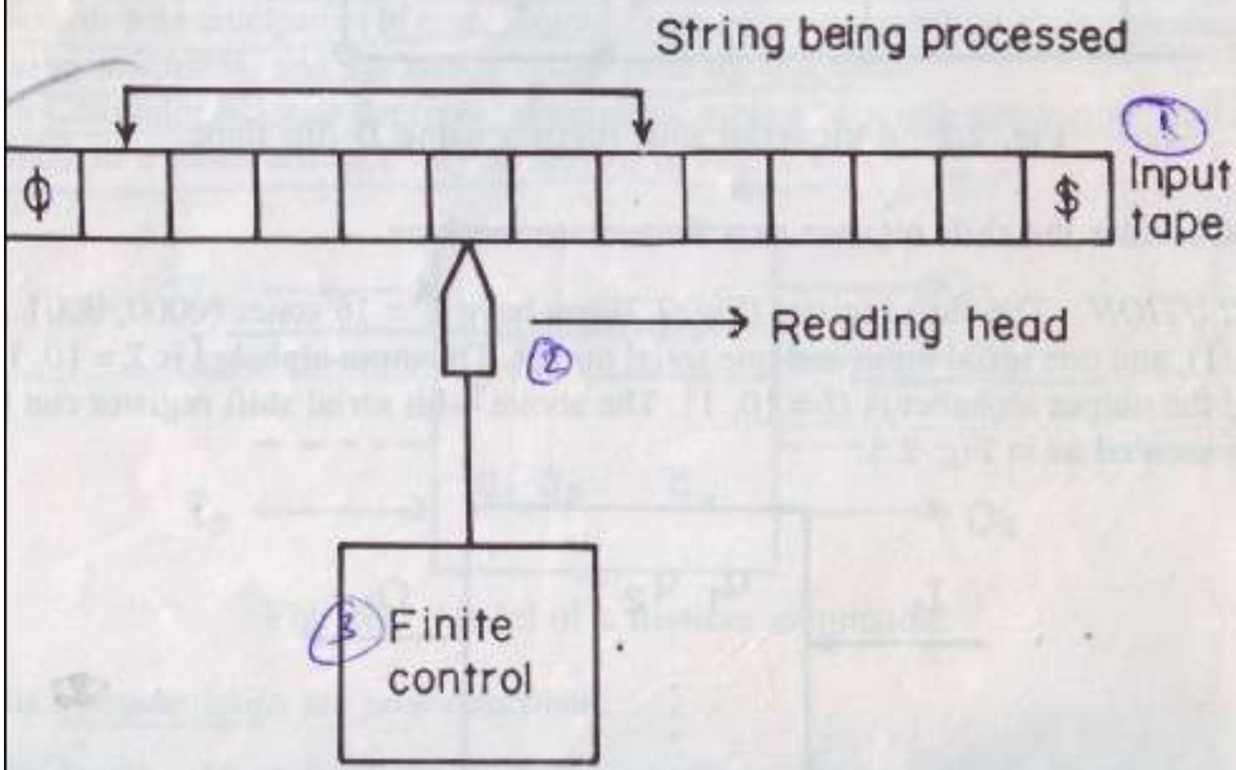


Fig. 2.4 Block diagram of a finite automaton.

Figure 2.4 is the block diagram for a finite automaton. The various components
e explained as follows:

Introduction to Automata Theory

(i) *Input tape.* The input tape is divided into squares, each square containing a single symbol from the input alphabet Σ. The end squares of the tape contain end-markers ¢ at the left end and $ at the right end. Absence of end-markers indicates that the tape is of infinite length. The left-to-right sequence of symbols between the end-markers is the input string to be processed.

(ii) *Reading head.* The head examines only one square at a time and can move one square either to the left or to the right. For further analysis, we restrict the movement of R-head only to the right side.

(iii) *Finite control.* The input to the finite control will be usually: symbol under the R-head, say a, or the present state of the machine, say q, to give the following outputs: (a) A motion of R-head along the tape to the next square (In some a null move, i.e. R-head remaining to the same square is permitted); (b) the next state of the finite state machine given by $\delta(q, a)$.

Transition System

A transition graph or a transition system is a finite directed labelled graph in which each vertex (or node) represents a state and the directed edges indicate the transition of a state and the edges are labelled with input/output.

A typical transition system is shown in Fig. 2.5. In the figure, the initial state is represented by a circle with an arrow pointing towards it, the final state by two

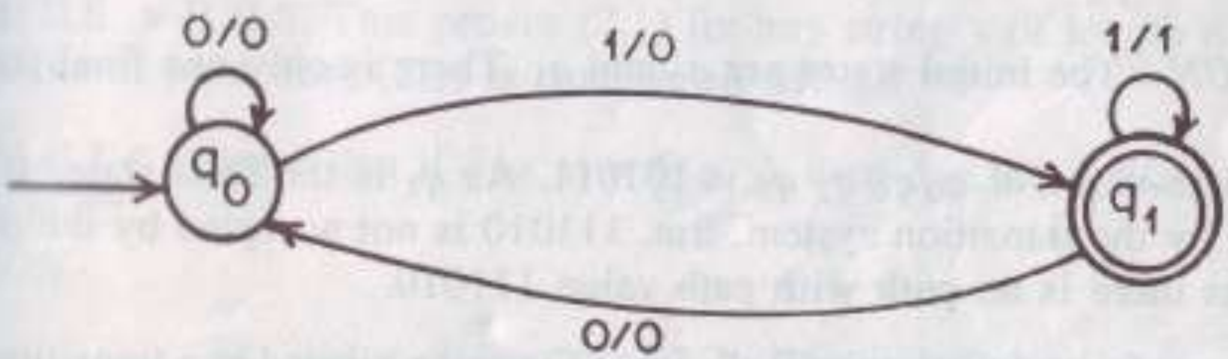


Fig. 2.5 A transition system.

concentric circles, and the other states are represented by just a circle. The edges are labelled by input/output (e.g. by 1/0 or 1/1). For example, if the system is in state q_0 and the input 1 is applied, the system moves to state q_1 as there is a directed edge from q_0 to q_1 with label 1/0. It outputs 0.

Property of Transition Function

perty 1 $\delta(q, \Lambda) = q$ in a finite automaton. This means the state of the tem can be changed only by an input symbol.

perty 2 For all strings w and input symbols a,

$$\delta(q, aw) = \delta(\delta(q, a), w)$$

$$\delta(q, wa) = \delta(\delta(q, w), a)$$

s property gives the state after the automaton consumes or reads the first ibol of a string aw and the state after the automaton consumes a prefix of the ng wa.

Acceptability of String by FA

Definition 2.4 A string x is accepted by a finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ if $\delta(q_0, x) = q$ for some $q \in F$. This is basically the acceptability of a string by the final state.

Example

States	Inputs 0	Inputs 1
$\rightarrow (q_0)$	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

OLUTION

$$\begin{aligned}\delta(q_0, \downarrow 110101) &= \delta(q_1, \downarrow 10101)\\ &= \delta(q_0, \downarrow 0101)\\ &= \delta(q_2, \downarrow 101)\\ &= \delta(q_3, \downarrow 01)\\ &= \delta(q_1, \downarrow 1)\\ &= \delta(q_0, \Lambda) = q_0\end{aligned}$$

ence,

$$q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_0 \xrightarrow{0} q_2 \xrightarrow{1} q_3 \xrightarrow{0} q_1 \xrightarrow{1} q_0$$

e symbol $\downarrow$ indicates the current input symbol being processed by the machine.

Types of Automata

Two Types

1. Automata without output
   - I. DFA (Deterministic Finite Automata)
   - II. NFA(Nondeterministic Finite Automata)
     - a. NFA without ϵ(or Λ)
     - b. NFA with ϵ(or Λ)
2. Automata with output
   - I. Mealy Machine
   - II. Moore Machine

DFA

efinition 2.1 Analytically, a finite automaton can be represented by a 5-tuple
, Σ, δ, q_0, F), where

(i) Q is a finite nonempty set of states;
(ii) Σ is a finite nonempty set of inputs called input alphabet;
(iii) δ is a function which maps $Q \times \Sigma$ into Q and is usually called direct
nsition function. This is the function which describes the change of states
ring the transition. This mapping is usually represented by a transition table
a transition diagram.
(iv) $q_0 \in Q$ is the initial state; and
(v) $F \subseteq Q$ is the set of final states. It is assumed here that there may be
ore than one final state.

NFA(NFA without ϵ)

Definition 2.5 A nondeterministic finite automaton (NDFA) is a 5-tuple (Q, Σ, δ, 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.

NFA(NFA without ϵ)
Example

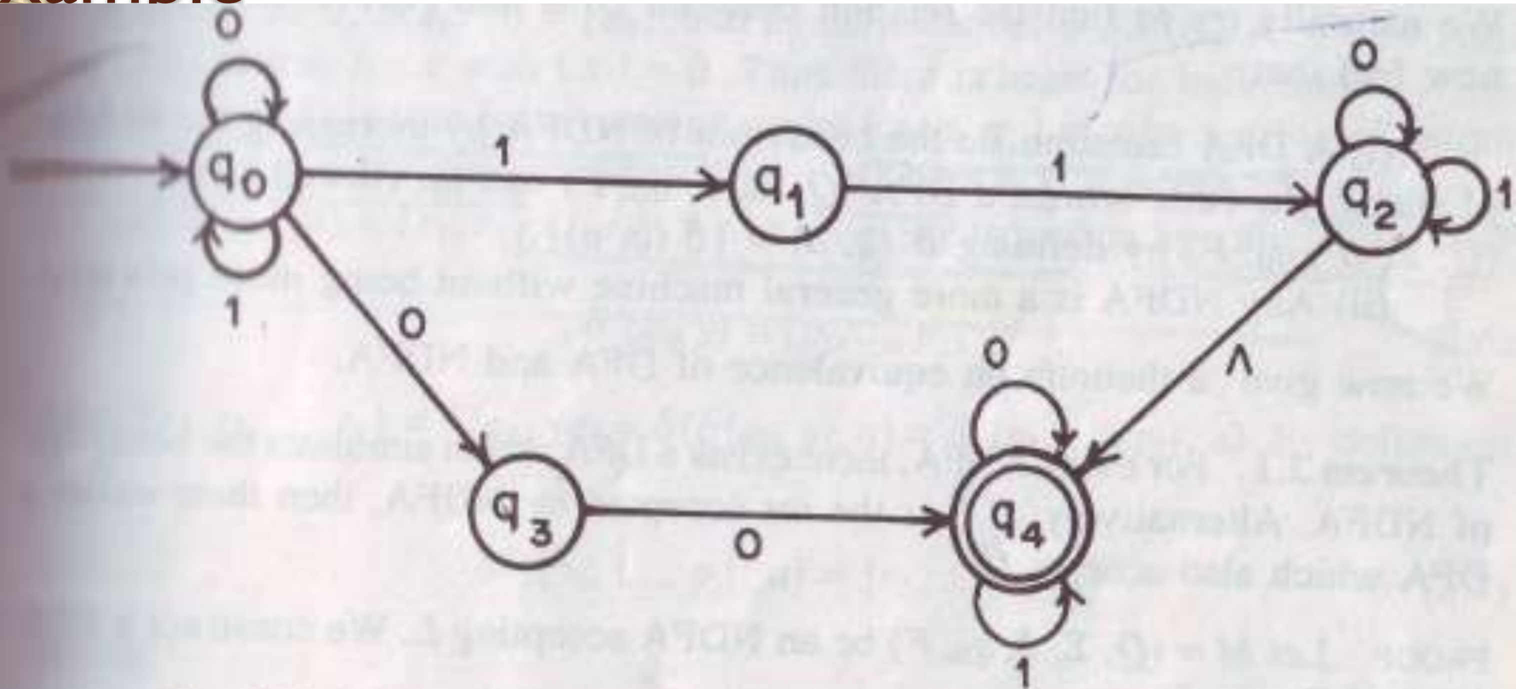


Fig. 2.8 Transition system for a nondeterministic automaton.

The sequence of states for the input string 0100 is given in Fig. 2.9. 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.

Acceptability in NFA

Definition 2.6 A string $w \in \Sigma^*$ is accepted by NDFA M if $\delta(q_0, w)$ contains some final state.

Acceptability in NFA

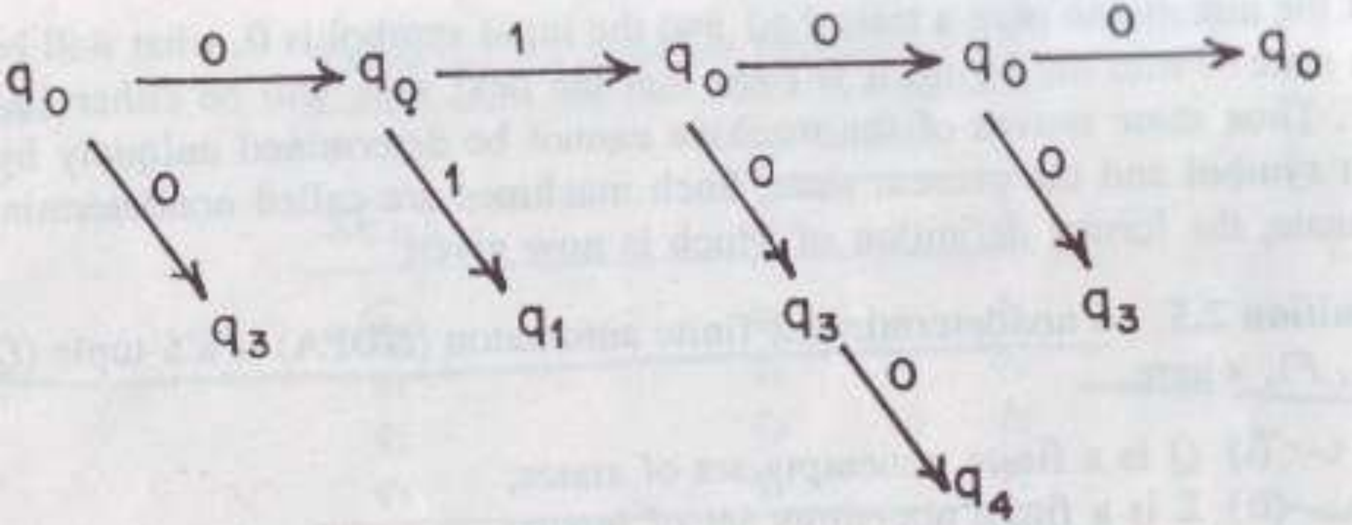


Fig. 2.9 States reached while processing 0100.

ccepted by M if a final state is *one* among the possible states M can reach
application of w.

finition 2.7 The set accepted by an automaton M (deterministic or nondeter-
nistic) is the set of all input strings accepted by M. It is denoted by $T(M)$.

Equivalence of DFA and NFA

THE EQUIVALENCE OF DFA AND NDFA

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

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

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

'e now give a theorem on equivalence of DFA and NDFA.

heorem 2.1 For every NDFA, there exists a DFA which simulates the behaviour NDFA. Alternatively, if L is the set accepted by NDFA, then there exists a FA which also accepts L.

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

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

here

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

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

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

Equivalence of DFA and NFA

(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

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

Example

Table 2.2 State Table for Example 2.6

State/Σ	0	1
$\to (q_0)$	q_0	q_1
q_1	q_1	q_0, q_1

XAMPLE 2.6 Construct a deterministic automaton equivalent to $M = (\{q_0, q_1\},$ $\{0, 1\}, \delta, q_0, \{q_0\})$. δ is given by its state table (Table 2.2).

OLUTION 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 2.3.

Table 2.3 State Table of M_1

States/Σ	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$[q_0]$	$[q_0]$	$[q_1]$
$[q_1]$	$[q_1]$	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$

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

Example

EXAMPLE 2.7 Find a deterministic acceptor equivalent to

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

δ is given in Table 2.4.

Table 2.4 State Table for Example 2.7

States/Σ	a	b
$\rightarrow q_0$	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 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 in Table 2.5.

Table 2.5 State Table of M_1

States/Σ	a	b
$[q_0]$	$[q_0, q_1]$	$[q_2]$
$[q_2]$	$\emptyset$	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_1, q_2]$
$[q_1, q_2]$	$[q_0]$	$[q_0, q_1]$

Example

EXAMPLE 2.8 Construct a deterministic finite automaton equivalent to $M = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, q_0, \{q_3\})$. δ is given in Table 2.6.

Table 2.6 State Table for Example 2.8

States/Σ	a	b
$\rightarrow q_0$	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

Solution

$q_3]$, $[q_0, q_3]$, $[q_1, q_3]$, $[q_2, q_3]$, $[q_0, q_1, q_3]$, $[q_0, q_2, q_3]$, $[q_1, q_2, q_3]$ and $q_1, q_2, q_3]$. δ is given in Table 2.7.

Table 2.7 State Table of M_1

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

Finite Automata with Output

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

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

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

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

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

Moore Machine

Definition 2.8 The 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.

Mealy Machine

Definition 2.9 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.

Example
Mealy Machine

Table 2.10 Mealy Machine of Example 2.9

Present state	Next state			
	input $a = 0$		input $a = 1$	
	state	output	state	output
$\rightarrow q_1$	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

Example
Moore Machine

Table 2.13 Moore Machine of Example 2.9

Present state	Next state		Output
	$a = 0$	$a = 1$	
$\rightarrow q_0$	q_3	q_{20}	0
q_1	q_3	q_{20}	1
q_{20}	q_1	q_{40}	0
q_{21}	q_1	q_{40}	1
q_3	q_{21}	q_1	0
q_{40}	q_{41}	q_3	0
q_{41}	q_{41}	q_3	1

Procedure for Transforming Moore machine to Mealy machine

We modify the acceptability of input string by a Moore machine by neglecting the response of the Moore machine to input Λ. We thus define that Mealy Machine M and Moore Machine M' are equivalent if for all input strings w, $bZ_M(w) = Z_{M'}(w)$, where b is the output of Moore machine for its initial state. We give the following result: Let $M_1 = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ be a Moore machine. Then the following procedure may be adopted to construct an equivalent Mealy machine M_2.

Construction

(a) We have to define the output function λ' for Mealy machine as a function of present state and input symbol. We define λ' by

$$\lambda'(q, a) = \lambda(\delta(q, a)) \quad \text{for all states } q \text{ and input symbols } a.$$

(b) the transition function is the same as that of the given Moore machine.

Example

Table 2.14 Moore Machine of Example 2.10

Present state	Next state		Output
	$a = 0$	$a = 1$	
$\rightarrow q_0$	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

Mealy Machine

Table 2.15 Mealy Machine of Example 2.10

Present state	Next state			
	$a = 0$		$a = 1$	
	state	output	state	output
$\rightarrow q_0$	q_3	0	q_1	1
q_1	q_1	1	q_2	0
q_2	q_2	0	q_3	0
q_3	q_3	0	q_0	0

Moore machine to Mealy Machine

Table 2.16 Moore Machine of Example 2.11

Present state	Next state		Output
	$a = 0$	$a = 1$	
$\rightarrow q_1$	q_1	q_2	0
q_2	q_1	q_3	0
q_3	q_1	q_3	1

Mealy Machine

Table 2.17 Transition Table of Example 2.11

Present state	Next state			
	$a = 0$		$a = 1$	
	state	output	state	output
$\rightarrow q_1$	q_1	0	q_2	0
q_2	q_1	0	q_3	1
q_3	q_1	0	q_3	1

Table 2.18 Mealy Machine of Example 2.11

Present state	Next state			
	$a = 0$		$a = 1$	
	state	output	state	output
$\rightarrow q_1$	q_1	0	q_2	0
q_2	q_1	0	q_2	1

Moore to Mealy machine conversion-Example

EXAMPLE 2.11 Consider the Moore machine described by the transition table given in Table 2.16. Construct the corresponding Mealy machine.

Given

Table 2.16 Moore Machine of Example 2.11

Present state	Next state		Output
	$a = 0$	$a = 1$	
$\rightarrow q_1$	q_1	q_2	0
q_2	q_1	q_3	0
q_3	q_1	q_3	1

SOLUTION We construct the transition table in Table 2.17 by associating the output with the transitions.

In Table 2.17 the rows corresponding to q_2 and q_3 are identical. So, we can delete one of the two states, i.e., q_2 or q_3. We delete q_3. Table 2.18 gives the reconstructed table.

Moore to Mealy machine conversion-Example

Table 2.17 Transition Table of Example 2.11

Present state	Next state			
	$a = 0$		$a = 1$	
	state	output	state	output
$\rightarrow q_1$	q_1	0	q_2	0
q_2	q_1	0	q_3	1
q_3	q_1	0	q_3	1

Table 2.18 Mealy Machine of Example 2.11

Present state	Next state			
	$a = 0$		$a = 1$	
	state	output	state	output
$\rightarrow q_1$	q_1	0	q_2	0
q_2	q_1	0	q_2	1

In Table 2.17, we have deleted q_3-row and replaced q_3 by q_2 in the other rows.

Mealy to Moore Example

EXAMPLE 2.12 Consider a Mealy machine represented by Fig. 2.10. Construct a Moore machine equivalent to this Mealy machine.

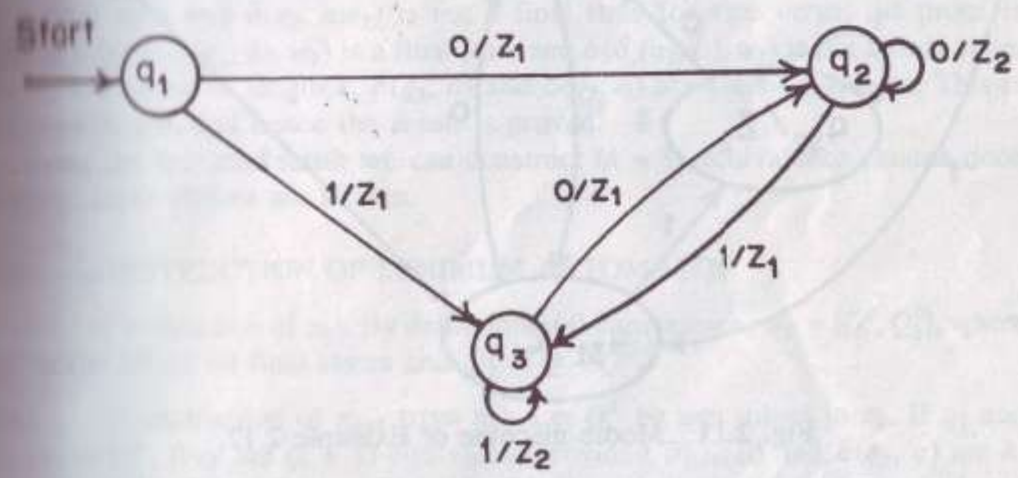


Fig. 2.10 Mealy machine of Example 2.12.

SOLUTION Let us convert the transition diagram into the transition Table 2.19. For the given problem: q_1 is not associated with any output. q_2 is associated with two different outputs Z_1 and Z_2; q_3 is associated with two different outputs Z_1 and Z_2. Thus we must split q_2 into q_{21} and q_{22} with outputs Z_1 and Z_2, respectively and q_3 into q_{31} and q_{32} with outputs Z_1 and Z_2, respectively. Table 2.19 may be reconstructed as Table 2.20.

Mealy to Moore Example

Table 2.19 Transition Table for Example 2.12

Present state	Next state: $a = 0$ state	Next state: $a = 0$ output	Next state: $a = 1$ state	Next state: $a = 1$ output
$\rightarrow q_1$	q_2	Z_1	q_3	Z_1
q_2	q_2	Z_2	q_3	Z_1
q_3	q_2	Z_1	q_3	Z_2

Table 2.20 Transition Table of Moore Machine

Present state	Next state: $a = 0$	Next state: $a = 1$	Output
$\rightarrow q_1$	q_{21}	q_{31}	
q_{21}	q_{22}	q_{31}	Z_1
q_{22}	q_{22}	q_{31}	Z_2
q_{31}	q_{21}	q_{32}	Z_1
q_{32}	q_{21}	q_{32}	Z_2

Figure 2.11 gives the transition diagram of the required Moore machine.

Mealy to Moore Example

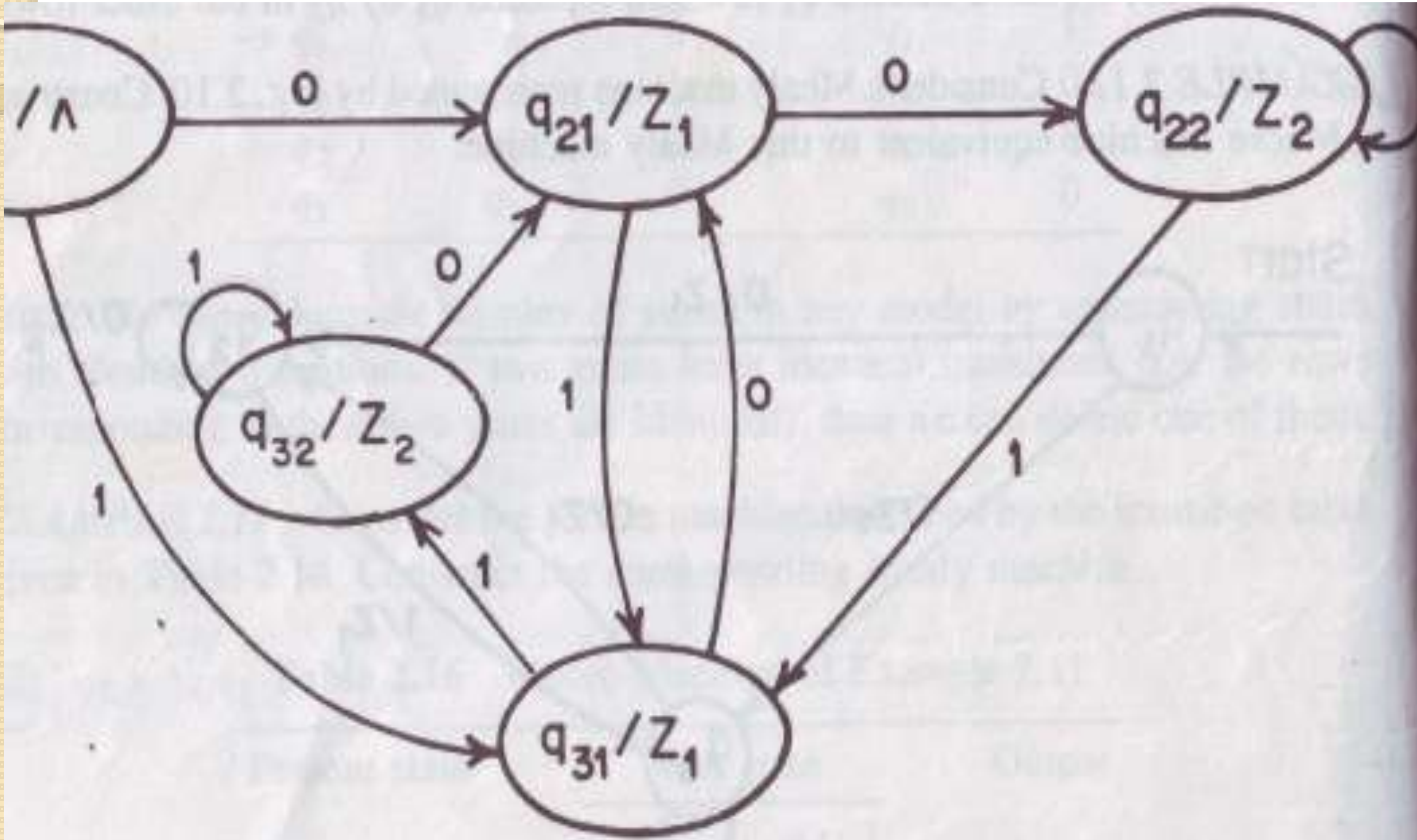


Fig. 2.11 Moore machine of Example 2.12.

Minimization of Automata

3.9.1 CONSTRUCTION OF MINIMUM AUTOMATON

Step 1 (Construction of π_0). By definition of 0-equivalence, $\pi_0 = \{Q_1^0, Q_2^0\}$, where Q_1^0 is the set of all final states and $Q_2^0 = Q - Q_1^0$.

Step 2 (Construction of π_{k+1} from π_k). Let Q_i^k be any subset in π_k. If q_1 and q_2 are in Q_i^k, they are $(k + 1)$-equivalent provided $\delta(q_1, a)$ and $\delta(q_2, a)$ are k-equivalent. Find out whether $\delta(q_1, a)$ and $\delta(q_2, a)$ are in the same equivalence class in π_k for every $a \in \Sigma$. If so, q_1 and q_2 are $(k + 1)$-equivalent. In this way, Q_i^k is further divided into $(k + 1)$-equivalence classes. Repeat this for every Q_i^k in π_k to get all the elements of π_{k+1}.

Step 3 Construct π_n for $n = 1, 2, \ldots$ until $\pi_n = \pi_{n+1}$.

Step 4 (Construction of minimum automaton). For the required minimum state automaton, the states are the equivalence classes obtained in step 3, i.e. the elements of π_n. The state table is obtained by replacing a state q by the corresponding equivalence class $[q]$.

Example – FA minimization

EXAMPLE 2.13 Construct a minimum state automaton equivalent to the finite automaton given in Fig. 2.12.

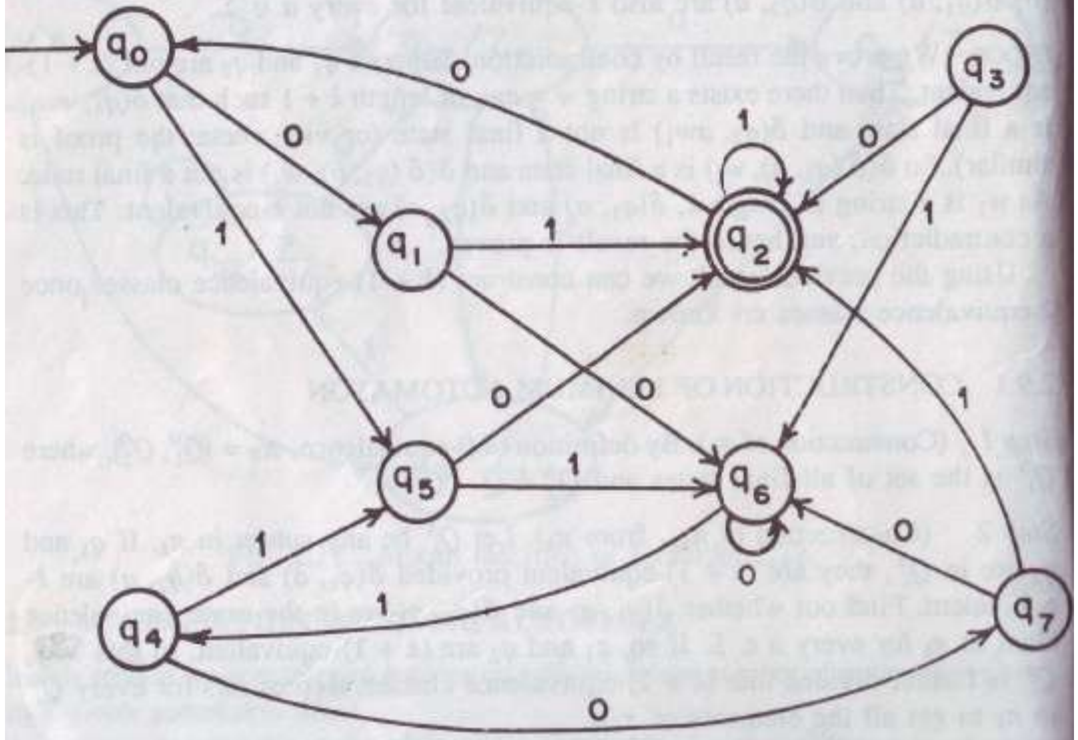


Fig. 2.12 FA of Example 2.13.

Example – FA minimization

Table 2.21 Transition Table for Example 2.13

State/Σ	0	1
$\to q_0$	q_1	q_5
q_1	q_6	q_2
(q_2)	q_0	q_2
q_3	q_2	q_6
q_4	q_7	q_5
q_5	q_2	q_6
q_6	q_6	q_4
q_7	q_6	q_2

By applying step 1, we get

$$Q_1^0 = F = \{q_2\}, \quad Q_2^0 = Q - Q_1^0$$

So,

$$\pi_0 = \{\{q_2\}, \{q_0, q_1, q_3, q_4, q_5, q_6, q_7\}\}$$

Example – FA minimization

$\{q_2\}$ in π_0 cannot be further partitioned. So, $Q'_1 = \{q_2\}$. Consider q_0 and $q_1 \in Q_2^0$. The entries under 0-column corresponding to q_0 and q_1 are q_1 and q_6; they lie in Q_2^0. The entries under 1-column are q_5 and q_2. $q_2 \in Q_1^0$ and $q_5 \in Q_2^0$. Therefore, q_0 and q_1 are not 1-equivalent. Similarly, q_0 is not 1-equivalent to q_3, q_5 and q_7.

Now, consider q_0 and q_4. The entries under 0-column are q_1 and q_7. Both are in Q_2^0. The entries under 1-column are q_5, q_5. So q_4 and q_0 are 1-equivalent. Similarly, q_0 is 1-equivalent to q_6. $\{q_0, q_4, q_6\}$ is a subset in π_1. So, $Q'_2 = \{q_0, q_4, q_6\}$. Repeat the construction by considering q_1 and any one of the states q_3, q_5, q_7. q_1 is not 1-equivalent to q_3 or q_5 but 1-equivalent to q_7. Hence, $Q'_3 = \{q_1, q_7\}$. The elements left over in Q_2^0 are q_3 and q_5. By considering the entries under 0-column and 1-column, we see that q_3 and q_5 are 1-equivalent. So $Q'_4 = \{q_3, q_5\}$. Therefore,

$$\pi_1 = \{\{q_2\}, \{q_0, q_4, q_6\}, \{q_1, q_7\}, \{q_3, q_5\}\}$$

$\{q_2\}$ is also in π_2 as it cannot be partitioned further. Now the entries under 0-column corresponding to q_0 and q_4 are q_1 and q_7, and these lie in the same equivalence class in π_1. The entries under 1-column are q_5, q_5. So q_0 and q_4 are 2-equivalent. But q_0 and q_6 are not 2-equivalent. Hence, $\{q_0, q_4, q_6\}$ is partitioned into $\{q_0, q_4\}$ and $\{q_6\}$. q_1 and q_7 are 2-equivalent. q_3 and q_5 are also 2-equivalent. Thus, $\pi_2 = \{\{q_2\}, \{q_0, q_4\}, \{q_6\}, \{q_1, q_7\}, \{q_3, q_5\}\}$ q_0 and q_4 are 3-equivalent. q_1 and q_7 are 3-equivalent. Also, q_3 and q_5 are 3-equivalent. Therefore,

$$\pi_3 = \{\{q_2\}, \{q_0, q_4\}, \{q_6\}, \{q_1, q_7\}, \{q_3, q_5\}\}$$

As $\pi_2 = \pi_3$, π_2 gives the equivalence classes, the minimum state automaton is

$$M' = (Q', \{0, 1\}, \delta', q'_0, F')$$

Example – FA minimization

re

$$Q' = \{[q_2], [q_0, q_4], [q_6], [q_1, q_7], [q_3, q_5]\}$$

$$q'_0 = [q_0, q_4], \qquad F' = [q_2]$$

δ' is given by Table 2.22.

Table 2.22 Transition Table of Minimum State Automaton

State/Σ	0	1
$[q_0, q_4]$	$[q_1, q_7]$	$[q_3, q_5]$
$[q_1, q_7]$	$[q_6]$	$[q_2]$
$[q_2]$	$[q_0, q_4]$	$[q_2]$
$[q_3, q_5]$	$[q_2]$	$[q_6]$
$[q_6]$	$[q_6]$	$[q_0, q_4]$

TE: The transition diagram for the minimum state automaton is given in . 2.13. The states q_0 and q_4 are identified and treated as one state. (So also q_1, q_7 and q_3, q_5.) But the transitions in both the diagrams (i.e. Figs. 2.12 and

Example – FA minimization

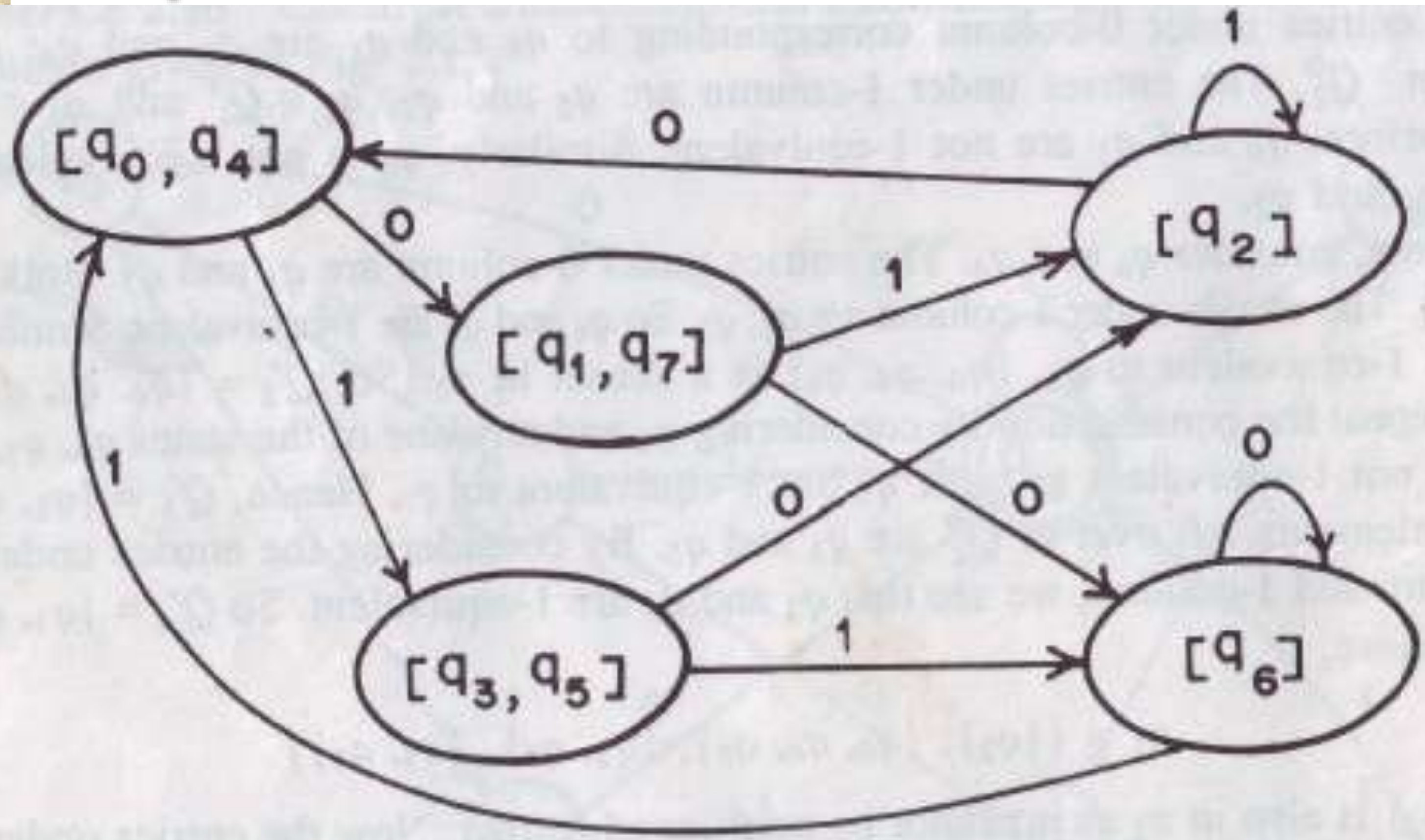
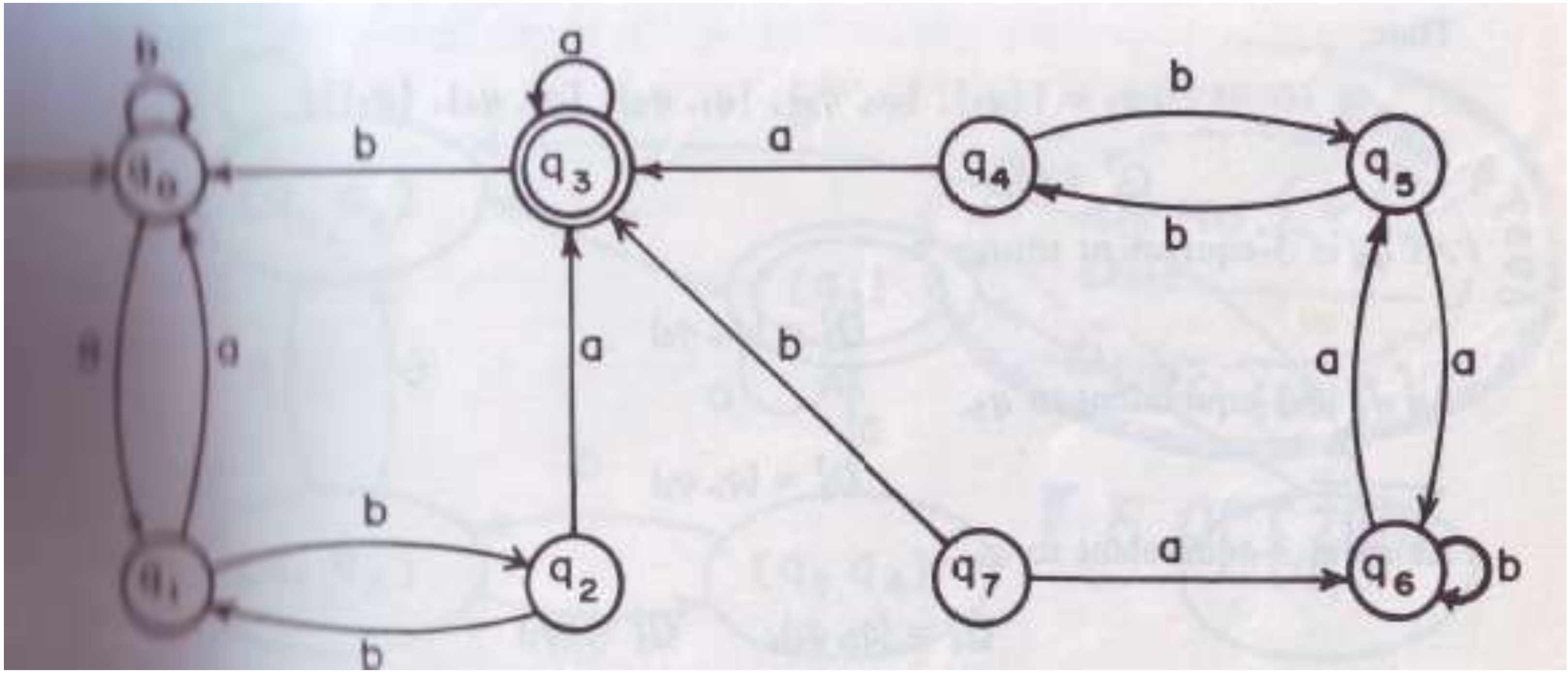


Fig. 2.13 Minimum state automaton of Example 2.13.

13) are the same. If there is an arrow from q_i to q_j with label a, then there is arrow from $[q_i]$ to $[q_j]$ with the same label in the diagram for minimum state utomaton. Symbolically, if $\delta(q_i, a) = q_j$, then $\delta'([q_i], a) = [q_j]$.

Question



Assignment

7. The transition table of a nondeterministic finite automaton M is given in Table 2.25. Construct a deterministic finite automaton equivalent to M.

Table 2.25 Transition Table for Exercise 2.7

State	0	1	2
$\rightarrow q_0$	$q_1 q_4$	q_4	$q_2 q_3$
q_1		q_4	
q_2			$q_2 q_3$
(q_3)		q_4	
q_4			

8. Construct a DFA equivalent to the NDFA given in Fig. 2.8.

9. $M = (\{q_1, q_2, q_3\}, \{0, 1\}, \delta, q_1, \{q_3\})$ is a nondeterministic finite automaton, where δ is given by

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

$$\delta(q_2, 0) = \{q_1, q_2\} \quad \delta(q_2, 1) = \emptyset$$

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

Construct an equivalent DFA.

Assignment

..., containing ...

Construct a Mealy machine which is equivalent to the Moore machine given Table 2.26.

Table 2.26 Moore Machine of Exercise 2.11

Present state	Next state		Output
	$a = 0$	$a = 1$	
$\rightarrow q_0$	q_1	q_2	1
q_1	q_3	q_2	0
q_2	q_2	q_1	1
q_3	q_0	q_3	1

2. Construct a Moore machine equivalent to the Mealy machine M given in Table 2.27.

Table 2.27 Mealy Machine of Exercise 2.12

Present state	Next state			
	$a = 0$		$a = 1$	
	state	output	state	output
$\rightarrow q_1$	q_1	1	q_2	0
q_2	q_4	1	q_4	1
q_3	q_2	1	q_3	1
q_4	q_3	0	q_1	1

13. Construct a Mealy machine which can output EVEN, ODD according as the total number of 1's encountered is even or odd. The input symbols are 0 and 1.

14. Construct a minimum state automaton equivalent to a given automaton M whose transition table is given in Table 2.28.

Table 2.28 FA of Exercise 2.14

States	Input	
	a	b
$\rightarrow q_0$	q_0	q_3
q_1	q_2	q_5
q_2	q_3	q_4
q_3	q_0	q_5
q_4	q_0	q_6
q_5	q_1	q_4
(q_6)	q_1	q_3

Regular Set and Regular Grammer

3.1 REGULAR EXPRESSIONS

Regular expressions are useful for representing certain sets of strings in an algebraic fashion. Actually these describe the languages accepted by finite state automata.

We give a formal recursive definition of regular expressions over Σ as follows:

1. Any terminal symbol (i.e. an element of Σ), Λ and Ø are regular expressions. When we view *a* in Σ as a regular expression, we denote it by **a**.
2. The union of two regular expressions $\mathbf{R}_1$ and $\mathbf{R}_2$, written as $\mathbf{R}_1 + \mathbf{R}_2$, is also a regular expression.
3. The concatenation of two regular expressions $\mathbf{R}_1$ and $\mathbf{R}_2$, written as $\mathbf{R}_1\mathbf{R}_2$, is also a regular expression.
4. The iteration (or closure) of a regular expression **R**, written as $\mathbf{R}^*$, is also a regular expression.
5. If **R** is a regular expression, then (**R**) is also a regular expression.
6. The regular expressions over Σ are precisely those obtained recursively by the application of the rules 1–5 once or several times.

Regular Set

Definition 4.1 Any set represented by a regular expression is called a regular set.

If, for example, $a, b \in \Sigma$, then (a) **a** denotes the set $\{a\}$, (b) **a + b** denotes $\{a, b\}$, (c) **ab** denotes $\{ab\}$, (d) **a*** denotes the set $\{\Lambda, a, aa, aaa, \ldots\}$ and (e) **(a + b)*** denotes $\{a, b\}^*$.

Now we shall explain the evaluation procedure for the three basic operations. Let $\mathbf{R}_1$ and $\mathbf{R}_2$ denote any two regular expressions. Then (a) a string in $\mathbf{R}_1 + \mathbf{R}_2$ is a string from $\mathbf{R}_1$ or a string from $\mathbf{R}_2$; (b) a string in $\mathbf{R}_1\mathbf{R}_2$ is a string from $\mathbf{R}_1$ followed by a string from $\mathbf{R}_2$, and (c) a string in $\mathbf{R}^*$ is a string obtained by concatenating n elements for some $n \geq 0$. Consequently, (a) the set represented by $\mathbf{R}_1 + \mathbf{R}_2$ is the union of the sets represented by $\mathbf{R}_1$ and $\mathbf{R}_2$, (b) the set represented by $\mathbf{R}_1\mathbf{R}_2$ is the concatenation of the sets represented by $\mathbf{R}_1$ and $\mathbf{R}_2$ (Recall that the concatenation AB of sets A and B of strings over Σ is given by $AB = \{w_1w_2 | w_1 \in A, w_2 \in B\}$, and (c) the set represented by $\mathbf{R}^*$ is $\{w_1w_2 \ldots w_n | w_i$ is in the set represented by $\mathbf{R}$ and $n \geq 0\}$.

Reg. Set to Regular Expression

EXAMPLE 4.1 Describe the following sets by regular expressions: (a) {101}, (b) {*abba*}, (c) {01, 10}, (d) {Λ, *ab*}, (e) {*abb*, *a*, *b*, *bba*}, (f) {Λ, 0, 00, 000, ...}, and (g) {1, 11, 111, ...}.

SOLUTION (a) Now, {1}, {0} are represented by **1** and **0**, respectively. 101 is obtained by concatenating 1, 0 and 1. So, {101} is represented by **101**.

(b) **abba** represents {*abba*}.

(c) As {01, 10} is the union of {01} and {10}, {01, 10} is represented by **01 + 10**.

(d) The set {Λ, *ab*} is represented by Λ + **ab**.

(e) The set {*abb*, *a*, *b*, *bba*} is represented by **abb + a + b + bba**.

(f) As {Λ, 0, 00, 000, ...} is simply {0}*, it is represented by **0***.

(g) Any element in {1,11, 111, ...} can be obtained by concatenating 1 and any element of {1}*. Hence **1(1)*** represents {1, 11, 111, ...}.

Reg. Set to Regular Expression

EXAMPLE 4.2 Describe the following sets by regular expressions:

(a) L_1 = the set of all strings of 0's and 1's ending in 00.

(b) L_2 = the set of all strings of 0's and 1's beginning with 0 and ending with 1.

(c) $L_3 = \{\Lambda, 11, 1111, 111111, \ldots\}$.

SOLUTION (a) Any string in L_1 is obtained by concatenating any string over {0, 1} and the string 00. {0, 1} is represented by **0 + 1**. Hence L_1 is represented by **(0 + 1)* 00**.

(b) As any element of L_2 is obtained by concatenating 0, any string over {0, 1} and 1, L_2 can be represented by **0(0 + 1)* 1**.

(c) Any element of L_3 is either Λ or a string of even number of 1's, i.e. a string of the form $(11)^n$, $n \geq 0$. So L_3 can be represented by **(11)***.

Identities of RE

4.1.1 IDENTITIES FOR REGULAR EXPRESSIONS

Two regular expressions **P** and **Q** are equivalent (we write **P** = **Q**) if **P** and **Q** represent the same set of strings.

We now give the identities for regular expressions; these are useful for simplifying regular expressions.

I_1 $\emptyset + \mathbf{R} = \mathbf{R}$

I_2 $\emptyset\mathbf{R} = \mathbf{R}\emptyset = \emptyset$

I_3 $\Lambda\mathbf{R} = \mathbf{R}\Lambda = \mathbf{R}$

I_4 $\Lambda^* = \Lambda$ and $\emptyset^* = \Lambda$

I_5 $\mathbf{R} + \mathbf{R} = \mathbf{R}$

I_6 $\mathbf{R}^*\mathbf{R}^* = \mathbf{R}^*$

I_7 $\mathbf{R}\mathbf{R}^* = \mathbf{R}^*\mathbf{R}$

I_8 $(\mathbf{R}^*)^* = \mathbf{R}^*$

I_9 $\Lambda + \mathbf{R}\mathbf{R}^* = \mathbf{R}^* = \Lambda + \mathbf{R}^*\mathbf{R}$

I_{10} $(\mathbf{PQ})^*\mathbf{P} = \mathbf{P}(\mathbf{QP})^*$

I_{11} $(\mathbf{P} + \mathbf{Q})^* = (\mathbf{P}^*\mathbf{Q}^*)^* = (\mathbf{P}^* + \mathbf{Q}^*)^*$

I_{12} $(\mathbf{P} + \mathbf{Q})\mathbf{R} = \mathbf{PR} + \mathbf{QR}$ and $\mathbf{R}(\mathbf{P} + \mathbf{Q}) = \mathbf{RP} + \mathbf{RQ}$

Arden's Theorem

Theorem 4.1 (Arden's theorem) Let **P** and **Q** be two regular expressions over Σ. If **P** does not contain Λ, then the following equation in **R**, viz.

$$R = Q + RP \tag{4.1}$$

has a unique solution (i.e. one and only one solution) given by $R = QP^*$.

PROOF

$$Q + (QP^*)P = Q(\Lambda + P^*P) = QP^* \text{ by } I_9$$

Hence (4.1) is satisfied when $R = QP^*$. This means $R = QP^*$ is a solution of (4.1).

To prove uniqueness, consider (4.1). Here, replacing **R** by **Q + RP** on the R.H.S., we get the equation

$$Q + RP = Q + (Q + RP)P$$

Arden’s Theorem

$$= Q + QP + RPP$$

$$= Q + QP + RP^2$$

$$= Q + QP + QP^2 + \dots + QP^i + RP^{i+1}$$

$$= Q(\Lambda + P + P^2 + \dots + P^i) + RP^{i+1}$$

From (4.1),

$$R = Q(\Lambda + P + P^2 + \dots + P^i) + RP^{i+1} \quad \text{for } i \geq 0 \qquad (4.2)$$

We now show that any solution of (4.1) is equivalent to **QP***. Suppose **R** satisfie (4.1), then it satisfies (4.2). Let w be a string of length i in the set **R**. Then w belongs to the set $Q(\Lambda + P + P^2 + \dots + P^i) + RP^{i+1}$. As **P** does not contain Λ, RP^{i+1} has no string of length less than $i + 1$ and so w is not in the set RP^{i+1}. This means w belongs to the set $Q(\Lambda + P + P^2 + \dots + P^i)$, and hence to **QP***.

Consider a string w in the set **QP***. Then w is in the set QP^k for some $k \geq 0$, and hence in $Q(\Lambda + P + P^2 + \dots + P^k)$. So w is on the R.H.S. of (4.2). Therefore, w is in **R** (L.H.S. of (4.2)). Thus **R** and **QP*** represent the same set. This proves the uniqueness of the solution of (4.1). ∎

RE

EXAMPLE 4.3 (a) Give an r.e. for representing the set L of strings in whic every 0 is immediately followed by at least two 1's.

(b) Prove that the regular expression $\mathbf{R} = \Lambda + 1^*(011)^*(1^*\ (011)^*)^*$ als describes the same set of strings.

SOLUTION (a) If w is in L, then either (i) w does not contain any 0, or (ii) contains a 0 preceded by 1 and followed by 11. So w can be written as $w_1 w_2$. w_n, where each w_i is either 1 or 011. So L is represented by the r.e. $(1 + 011)^*$

(b) $\mathbf{R} = \Lambda + \mathbf{P}_1\mathbf{P}_1^*$, where $\mathbf{P}_1 = 1^*(011)^*$

$= P_1^*$ using I_9

$= (1^*(011)^*)^*$

$= (\mathbf{P}_2^*\mathbf{P}_3^*)^*$ letting $\mathbf{P}_2 = 1$, $\mathbf{P}_3 = 011$

$= (\mathbf{P}_2 + \mathbf{P}_3)^*$ using I_{11}

$= (1 + 011)^*$

FA to RE Conversion

EXAMPLE 4.8 Consider the transition system given in Fig. 4.10. Prove th[e]
the strings recognised are (a + a(b + aa)*b)* a(b + aa)* a.

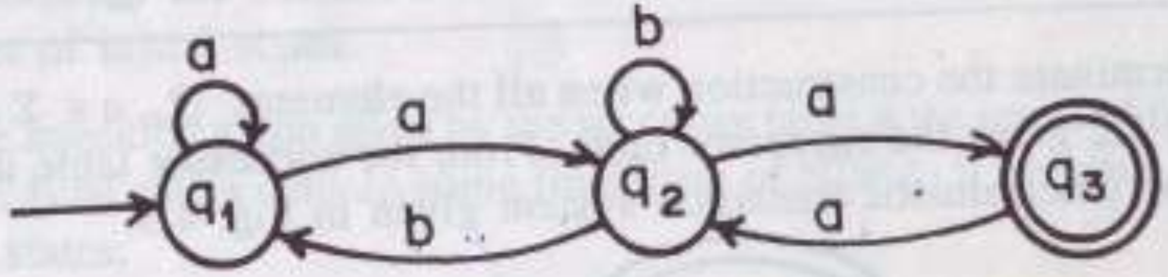


Fig. 4.10 Transition system of Example 4.8.

SOLUTION We can directly apply the above method since the graph does n[ot]
contain any Λ-move and there is only one initial state.

The three equations for q_1, q_2 and q_3 can be written as

$$q_1 = q_1a + q_2b + \Lambda, \qquad q_2 = q_1a + q_2b + q_3a, \qquad q_3 = q_2a$$

It is necessary to reduce the number of unknowns by repeated substitution.
substituting q_3 in q_2-equation, we get

$$q_2 = q_1a + q_2b + q_2aa$$

FA to RE Conversion

$$= q_1a + q_2(b + aa)$$

$$= q_1a\ (b + aa)^*$$

By applying Theorem 4.1. Substituting q_2 in q_1, we get

$$q_1 = q_1a + q_1a(b + aa)^*b + \Lambda$$

$$= q_1(a + a(b + aa)^*b) + \Lambda$$

Hence

$$q_1 = \Lambda(a + a(b + aa)^*b)^*$$

$$q_2 = (a + a(b + aa)^*b)^*\ a(b + aa)^*$$

$$q_3 = (a + a(b + aa)^*b)^*\ a(b + aa)^*a$$

Since q_3 is a final state, the set of strings recognised by the graph is given by

$$(a + a(b + aa)^*b)a(b + aa)^*a$$

FA to RE conversion

EXAMPLE 4.9 Prove that the FA whose transition diagram is given in Fig. 4.11 accepts the set of all strings over the alphabet $\{a, b\}$ with an equal number of

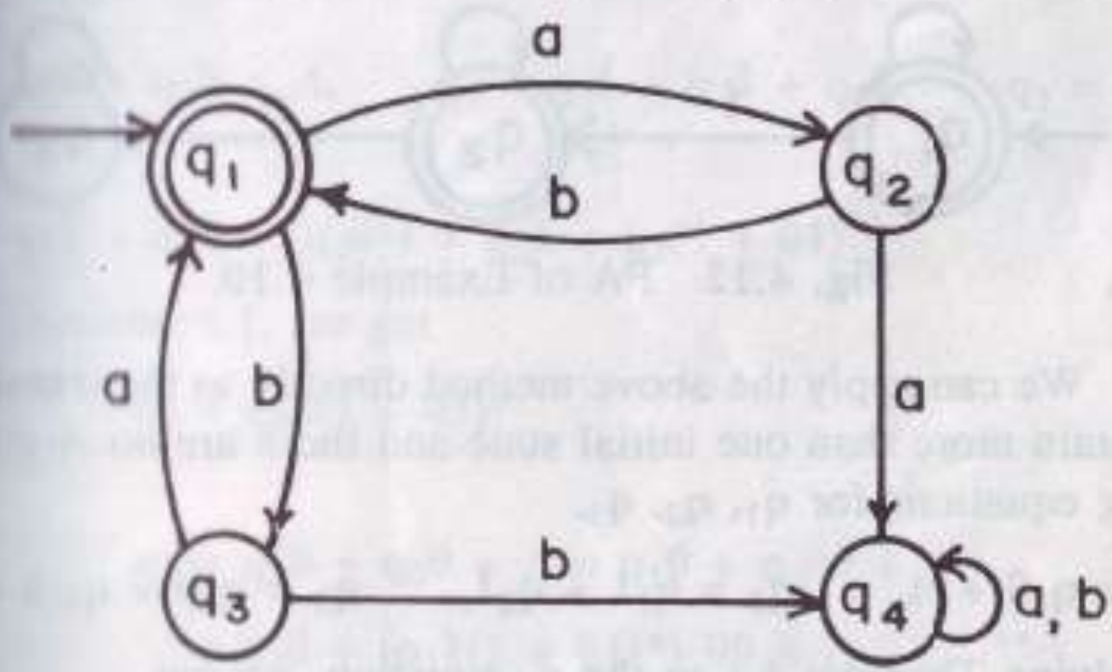


Fig. 4.11 FA of Example 4.9.

a's and b's, such that each prefix has atmost one more a than b's and atmost one more b than a's.

FA to RE conversion

SOLUTION We can apply the above method directly since the graph does not contain Λ-move and there is only one initial state. We get the following equations for q_1, q_2, q_3, q_4:

$$q_1 = q_2b + q_3a + \Lambda$$

$$q_2 = q_1a,$$

$$q_3 = q_1b$$

$$q_4 = q_2a + q_3b + q_4a + q_4b$$

As q_1 is the only final state and the q_1-equation involves only q_2 and q_3, we use

FA to RE conversion

only q_2- and q_3-equations (the q_4-equation is redundant for our purposes). Substituting for q_2 and q_3, we get

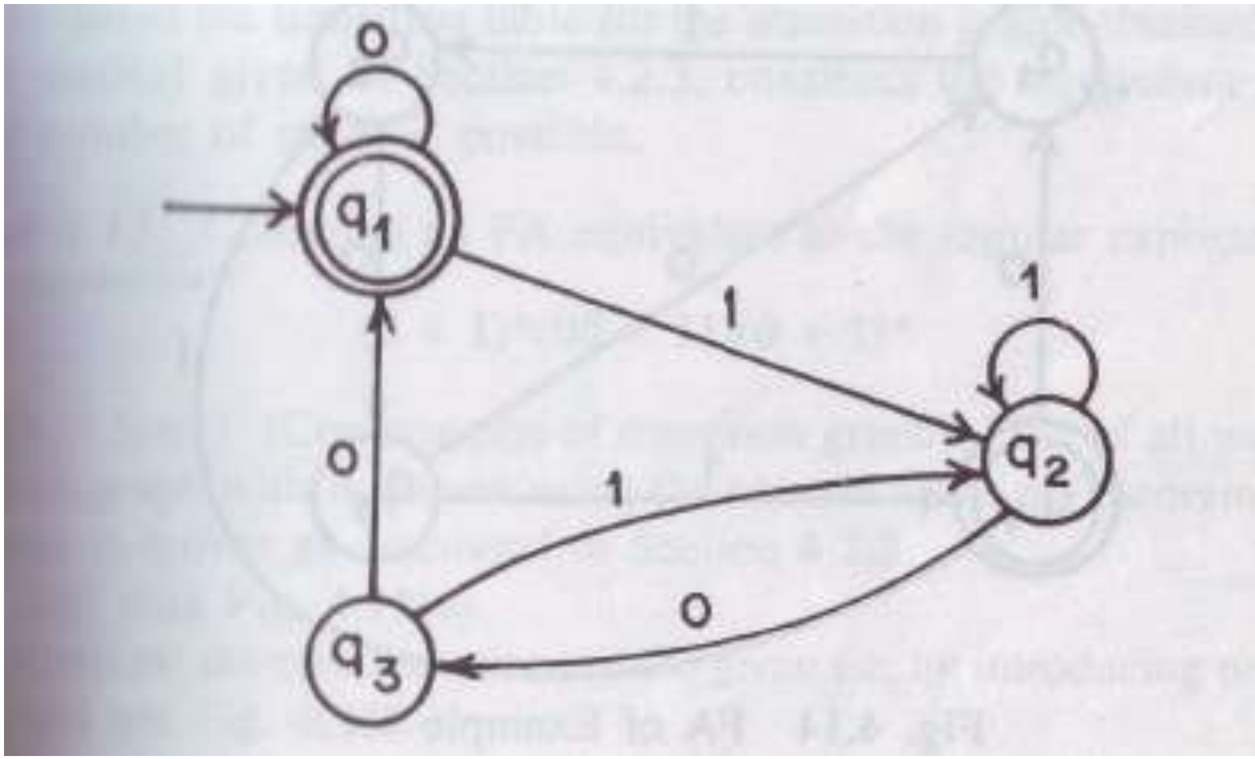
$$q_1 = q_1 ab + q_1 ba + \Lambda = q_1(ab + ba) + \Lambda$$

By applying Theorem 4.1, we get

$$q_1 = \Lambda(ab + ba)^* = (ab + ba)^*$$

As q_1 is the only final state, the strings accepted by the given FA are strings given by $(ab + ba)^*$. As any such string is a string of *ab*'s and *ba*'s we get equal number of *a*'s and *b*'s. If a prefix *x* of a sentence accepted by the FA has even number of symbols, then it should have equal number of *a*'s and *b*'s since *x* is a substring formed by *ab*'s and *ba*'s. If the prefix *x* has odd number of symbols, then we can write *x* as *ya* or *yb*. As *y* has even number of symbols, *y* has equal number of *a*'s and *b*'s. Thus *x* has one more *a* than *b* or vice versa.

FA to RE conversion



FA to RE conversion

EXAMPLE 4.10 Describe in English the set accepted by FA whose transition diagram is given in Fig. 4.12.

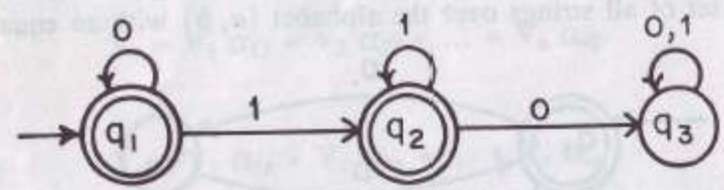


Fig. 4.12 FA of Example 4.10.

RE to DFA

EXAMPLE 4.13 Construct an FA equivalent to the regular expression.

$$(0 + 1)^*(00 + 11)(0 + 1)^*$$

SOLUTION *Step 1* (Construction of transition graph). First of all we construct the transition graph with Λ-moves using the constructions of Theorem 4.2. Then we eliminate Λ-moves as discussed in Section 4.2.2.

We start with Fig. 4.15(a).

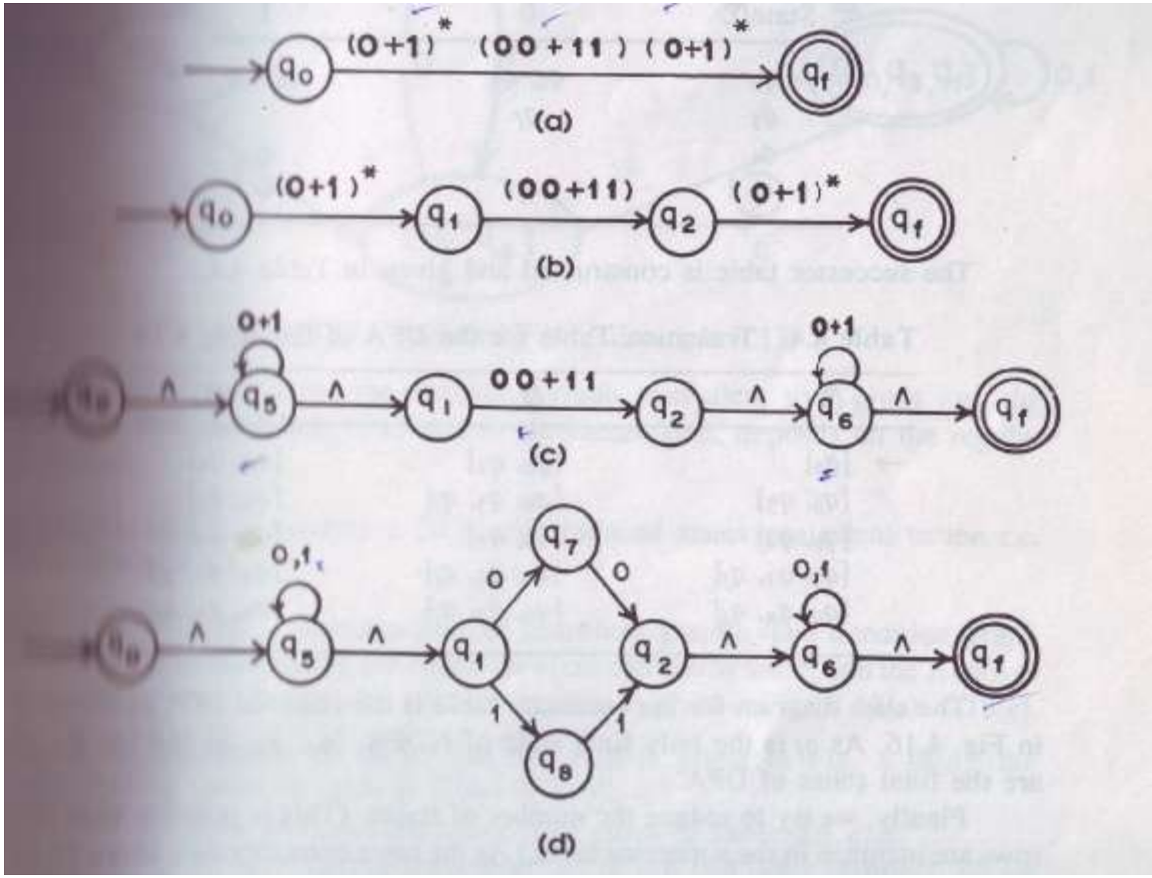
We eliminate the concatenations in the given r.e. by introducing new vertices q_1 and q_2 and get Fig. 4.15(b).

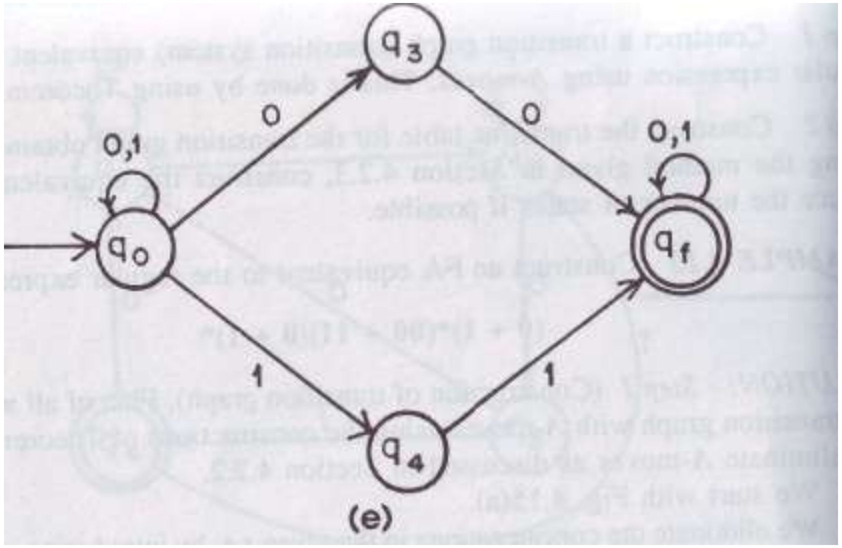
We eliminate * operations in Fig. 4.15(b) by introducing two new vertices q_5 and q_6 and Λ-moves as shown in Fig. 4.15(c).

We eliminate concatenations and + in Fig. 4.15(c) and get Fig. 4.15(d).

We eliminate Λ-moves in Fig. 4.15(d) and get Fig. 4.15(e) which gives the DFA equivalent to the given r.e.

RE to DFA




q3
0
0
0,1
0,1
q0
qf
1
1
q4
(e)