2.7 Equivalence between Two FSM's

The two finite automata are said to be equivalent if both the automata accept the same set of strings, over an input set Σ. When two FAS are equivalent then there is some string x over Σ, on acceptance of that string if one FA reaches to final state other FA also reaches to final state. We can compare whether two FAS are equivalent or not using following method.

Method for Comparing two FAS

Let M and M' be two FAS and Σ is a set of input strings.

1. We will construct a transition table have pair wise entries (q, q') where $q \in M$ and $q' \in M'$ for each input symbol.
2. If we get in a pair as one final state and other nonfinal state then we terminate construction of transition table declaring that two FAs are not equivalent.
3. The construction of transition table gets terminated when there is no new pair appearing in the transition table.

Let us take one example to understand the technique of comparing two FA.

Example 2.26 : *Consider the DFAs given below are they equivalent.*

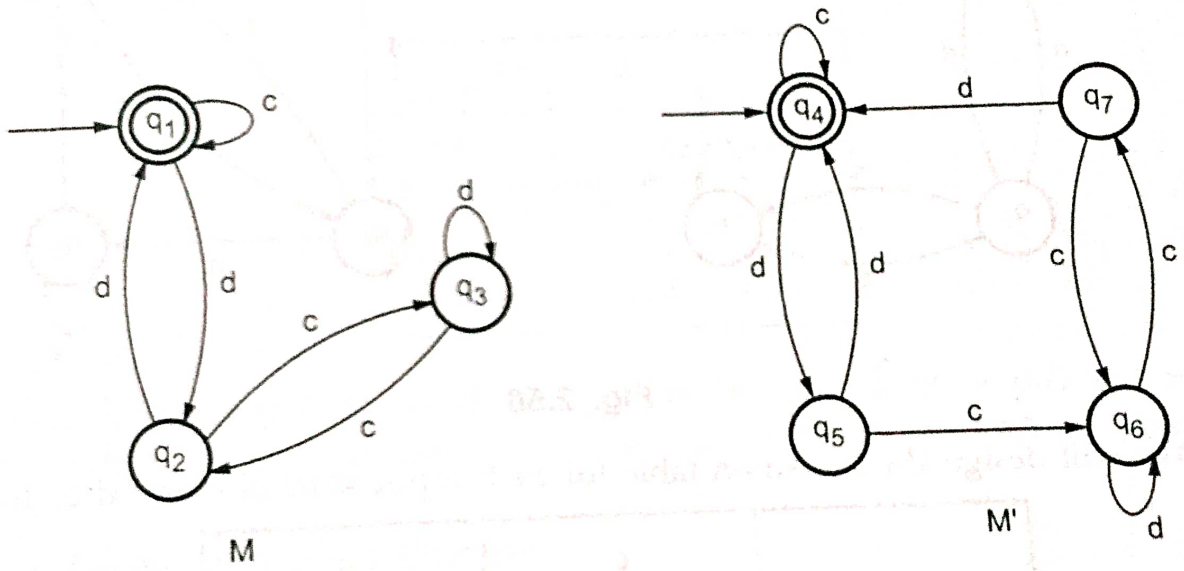


Fig. 2.49

Solution : We will first build the transition table for each input c and d. From first machine M on receiving input c in state q_1 we reach to state q_1 only. From second machine M′, for state q_4 on receiving c we reach to state q_4. Thus for state (q_1, q_4), for input c we get next state as (q_1, q_4). Similarly for input d in state (q_1, q_4) we get next state as (q_2, q_5).

Both q_1 and q_4 are final state obtained in pair (q_1, q_4). Both q_2 and q_5 are non final states obtained in pair (q_2, q_5) we will obtain transition for (q_2, q_5) for input c and d. The complete table is as given below -

	c	d
(q_1, q_4)	(q_1, q_4)	(q_2, q_5)
(q_2, q_5)	(q_3, q_6)	(q_1, q_4)
(q_3, q_6)	(q_2, q_7)	(q_3, q_6)
(q_2, q_7)	(q_3, q_6)	(q_1, q_4)

From the above table note that we do not get one final state and other non final state in a pair. Hence we declare that two DFAs are equivalent.

Example 2.27 : *Following are two FAs check whether they are equivalent or not.*

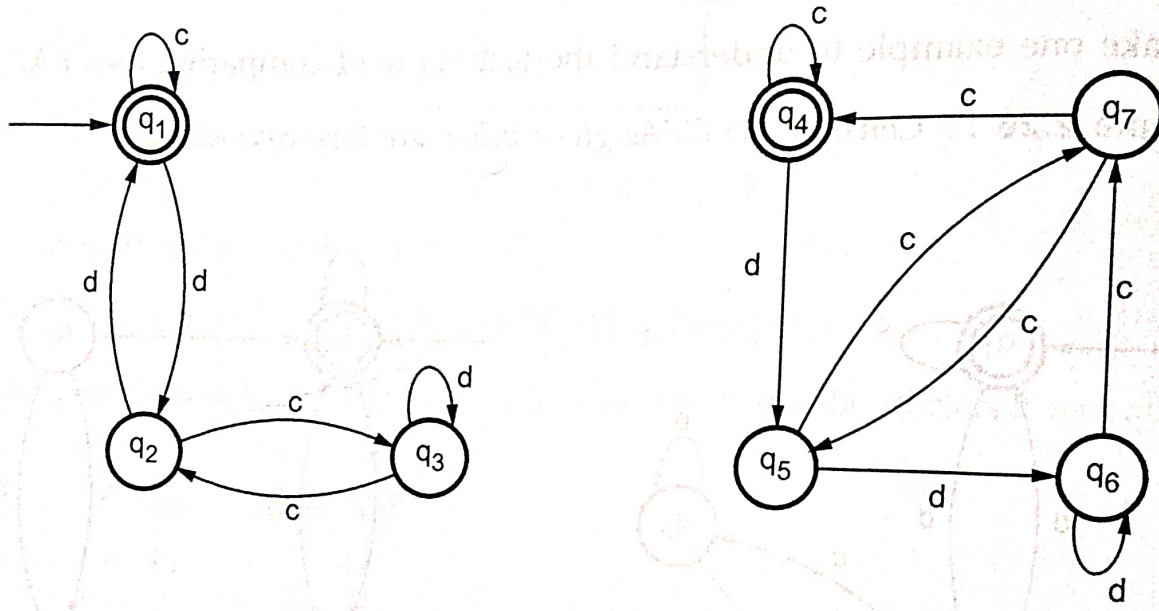


Fig. 2.50

Solution : We will design the transition table for each input symbol c and d as follows -

	c	d
(q_1, q_4)	(q_1, q_4)	(q_2, q_5)
(q_2, q_5)	(q_3, q_7)	(q_1, q_6)

We will terminate the construction of transition table because we get a pair (q_1, q_6) in which q_1 is a final state and q_6 is a non final state. As per equivalence rule final and non final state cannot form a pair. Hence the given FAs are not equivalent.

2.8 Finite Automata with Output

The finite automata is a collection of $(Q, \Sigma, \delta, q_0, F)$ where Q is a set of states including q_0 as a start state. In FA after reading the input string if we get final state then the string is said to be "acceptable". If we do not get final state then it is said that string is "rejected". That means there is no need of output for the finite Automata. The 'accept' or 'reject' acts like 'yes' or 'no' output for the machine. But if there is a need for specifying the output other than yes or no, then in such a case we require finite automata along with output. There are two types of FA with output and those are :

1) Moore machine

2) Mealy machine

1) Moore machine

Moore machine is a finite state machine in which the next state is decided by current state and current input symbol. The output symbol at a given time depends only on the present state of the machine. The formal definition of Moore machine is,

"Moore machine is a six tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$ where

Q is finite set of states.

Σ is finite set of input symbols.

Δ is an output alphabet.

δ is an transition function such that $Q \times \Sigma \rightarrow Q$. This is also known as state function.

λ is output function $Q \rightarrow \Delta$. This function is also known as machine function.

q_0 is the initial state of machine.

For example :

Consider the Moore machine given below -

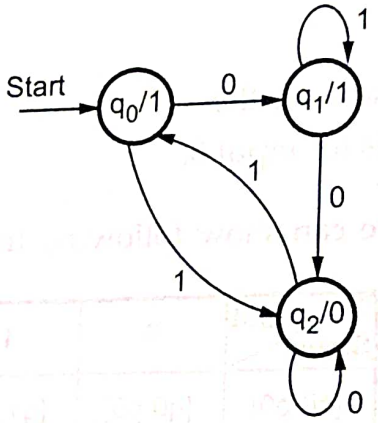


Fig. 2.52

The transition table will be -

Current state	Next state (δ)		Output (λ)
	0	1	
q_0	q_1	q_2	1
q_1	q_2	q_1	1
q_2	q_2	q_0	0

In Moore machine output is associated with every state. In the above given Moore machine when machine is in q_0 state the output will be 1. For the Moore machine if the length of input string is n then output string has length n+1.

For the string 0110 then the output will be 11110.

2) Mealy machine

Mealy machine is a machine in which output symbol depends upon the present input symbol and present state of the machine. The Mealy machine can be defined as -

Mealy machine is a six tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$ where

Q is finite set of states.

Σ is finite set of input symbols.

Δ is an output alphabet.

δ is state transition function such that $Q \times \Sigma \rightarrow Q$.

λ is machine function such that $Q \times \Sigma \rightarrow \Delta$.

q_0 is initial state of machine.

For example :

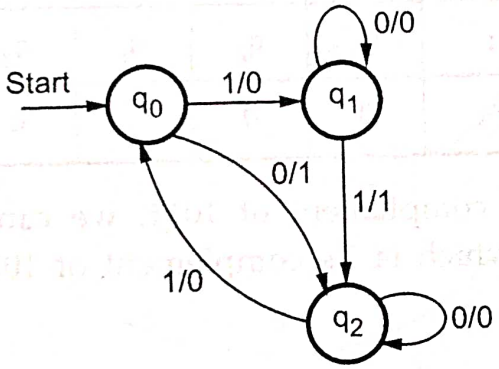


Fig. 2.53

For the input string 1001 the output will be 0001. In mealy machine the length of input string is equal to length of output string.

Example 2.29 : *Design a Moore machine to generate 1's complement of given binary number.*

Solution : To generate 1's complement of given binary number the simple logic which we will apply is that if input is 0 then output will be 1 and if input is 1 then output will be 0. That means there are three state one-start state, second state is for taking 0's as input and produces output as 1. Then third state is for taking 1's as input and producing output as 0.

Hence the Moore machine will be,

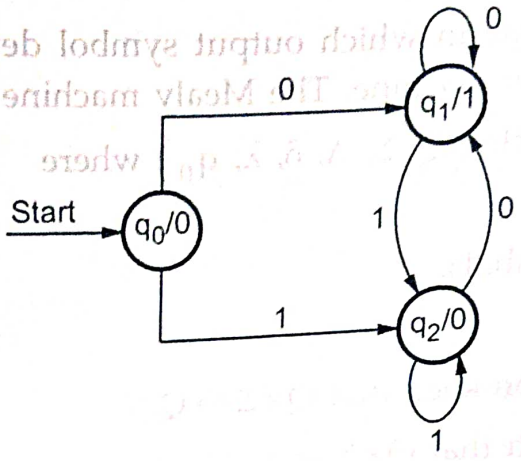


Fig. 2.54

For instance, take one binary number 1011 then

Input :		1	0	1	1
State :	q_0	q_2	q_1	q_2	q_2
Output :	0	0	1	0	0

Thus we get 00100 as 1's complement of 1011, we can neglect the initial 0 and the output which we get is 0100 which is 1's complement of 1011. The transition table can be drawn as below -

		δ		λ
	Current state	**Next state**		**Output**
		0	**1**	
→	q_0	q_1	q_2	0
	q_1	q_1	q_2	1
	q_2	q_1	q_2	0

Thus Moore machine $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$; where $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{0,1\}$ $\Delta = \{0,1\}$.

The transition table shows the δ and λ functions.

Example 2.30 : *Design a Moore and Mealy machine for a binary input sequence such that if it has a substring 101 the machine outputs A if input has substring 110 it outputs B otherwise it outputs C.*

Solution : For designing such a machine we need to take care of two conditions and those are checking 101 and checking 110. If we get 101 the output will be A. If we recognize 110 the output will be B. For other strings the output will be C. We can make a partial design of it as follows.

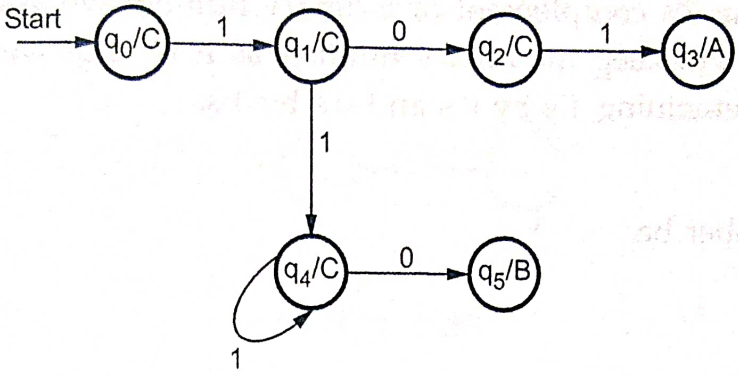


Fig. 2.55

Now we will insert the possibilities of 1's and 0's for each state. Then the Moore machine becomes

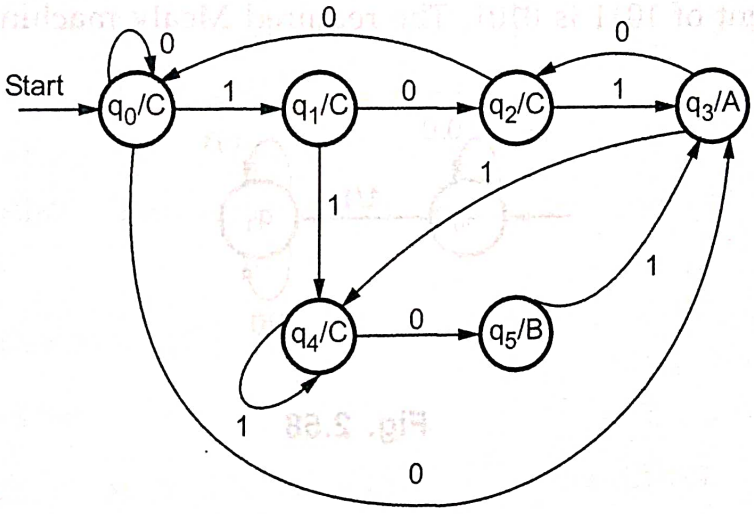


Fig. 2.56

Now the Mealy machine can be

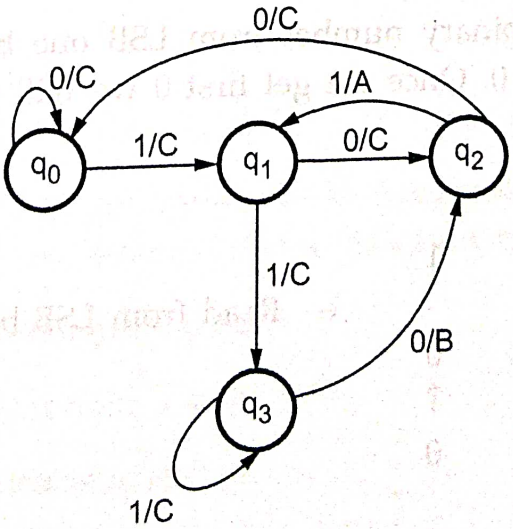


Fig. 2.57

2.9 Equivalence of Moore and Mealy Machines

The meaning of word equivalence is the two machines which accept the same language. Hence equivalence of Moore and Mealy machine means both the machines generate the same output string for same input string.

We cannot convert Moore machine to its equivalent Mealy machine directly because length of Moore machine is one longer than Mealy machine for the given input. We will make use of following method to convert Moore machine to Mealy machine.

2.9.1 Method for Conversion of Moore Machine to Mealy Machine

Let $M = (Q, \Sigma, \delta, \lambda, q_0)$ be a Moore machine. The equivalent Mealy machine can be represented by $M' = (Q, \Sigma, \delta, \lambda', q_0)$. The output function λ' can be obtained as -

$$\lambda'(q, a) = \lambda (\delta (q, a))$$

Let us understand this procedure with the help of some examples.

Example 2.37 : *The Moore machine to determine residue mod 3 for binary number is given below. Convert it to Mealy equivalent machine.*

Q \ Σ	0	1	Output (λ)
q_0	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_1	q_2	2

Solution : The transition diagram for the given problem can be drawn as -

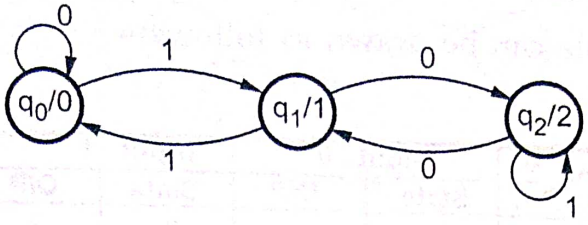


Fig. 2.66

The output function λ' can be obtained using following rule,

$$\lambda'(q, a) = \lambda(\delta(q, a))$$

Hence we will obtain output for every transition corresponding to input symbol.

$$\lambda'(q_0, 0) = \lambda(\delta(q_0, 0))$$

$$= \lambda(q_0) \quad \text{i.e. output of } q_0$$

$$\boxed{\lambda'(q_0, 0) = 0}$$

$$\lambda'(q_0, 1) = \lambda(\delta(q_0, 1))$$

$$= \lambda(q_1) \quad \text{i.e. output of } q_1$$

$$\boxed{\lambda'(q_0, 1) = 1}$$

$$\lambda'(q_1, 0) = \lambda(\delta(q_1, 0))$$

$$= \lambda(q_2)$$

$$\boxed{\lambda'(q_1, 0) = 2}$$

$$\lambda'(q_1, 1) = \lambda(\delta(q_1, 1))$$

$$= \lambda(q_0)$$

$$\boxed{\lambda'(q_1, 1) = 0}$$

$$\lambda'(q_2, 0) = \lambda(\delta(q_2, 0))$$

$$= \lambda(q_1)$$

$$\boxed{\lambda'(q_2, 0) = 1}$$

$$\lambda'(q_2, 1) = \lambda(\delta(q_2, 1))$$

$$= \lambda(q_2)$$

$\lambda'(q_2,1) = 2$

Hence the transition table can be drawn as follows -

Q \ Σ	Input 0		Input 1	
	State	O/P	State	O/P
q_0	q_0	0	q_1	1
q_1	q_2	2	q_0	0
q_2	q_1	1	q_2	2

The transition diagram of Mealy machine is,

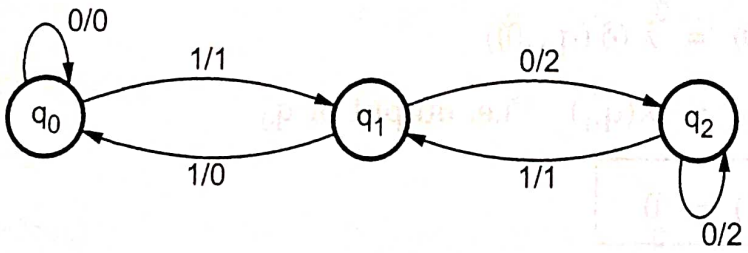


Fig. 2.67

The input string 10011 then the output for Mealy machine will be

Next state $q_1 \rightarrow q_2 \rightarrow q_2 \rightarrow q_1 \rightarrow q_0$

Output 1 2 2 1 0

The output sequence for Moore machine will be

Input 1 0 0 1 1

Next state $q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_2 \rightarrow q_1 \rightarrow q_0$

Output 0 1 2 2 1 0

The length of output sequence is n+1 in Moore machine and is n in Mealy machine which is desired.

Example 2.38 : *Convert the following Moore machine into equivalent Mealy machine* $M = (\{q_0, q_1\}, \{a,b\}, \{0,1\}\ \delta, \lambda, q_0)$

Solution :

δ	a	b	Output (λ)
q_0	q_0	q_1	0
q_1	q_0	q_1	1

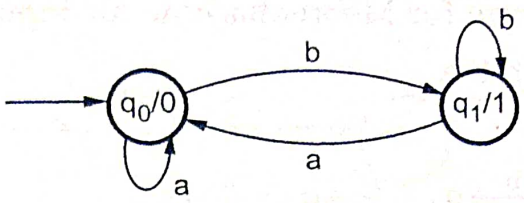


Fig. 2.68

The equivalent Mealy machine can be obtained as follows -

$$\lambda'(q_0, a) = \lambda(\delta(q_0, a))$$
$$= \lambda(q_0)$$
$$= 0$$
$$\lambda'(q_0, b) = \lambda(\delta(q_0, b))$$
$$= \lambda(q_1)$$
$$= 1$$
$$\lambda'(q_1, a) = \lambda(\delta(q_1, a))$$
$$= \lambda(q_0)$$
$$= 0$$
$$\lambda'(q_1, b) = \lambda(\delta(q_1, b))$$
$$= \lambda(q_1)$$
$$= 1$$

Hence the transition table can be drawn as follows -

Q \ Σ	Input a		Input b	
	State	O/P	State	O/P
q_0	q_0	0	q_1	1
q_1	q_0	0	q_1	1

The equivalent Mealy machine will be,

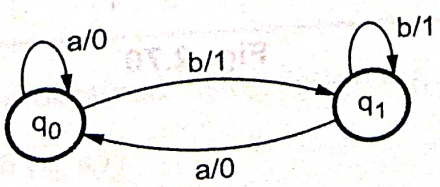


Fig. 2.69

Consider the output sequence for Moore machine for input sequence,

a b b a

Then

$$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{b} q_1 \xrightarrow{a} q_0$$

0 0 1 1 0

Similarly output sequence for Mealy machine will be

$$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_1 \xrightarrow{b} q_1 \longrightarrow q_0$$

0 1 1 1 0

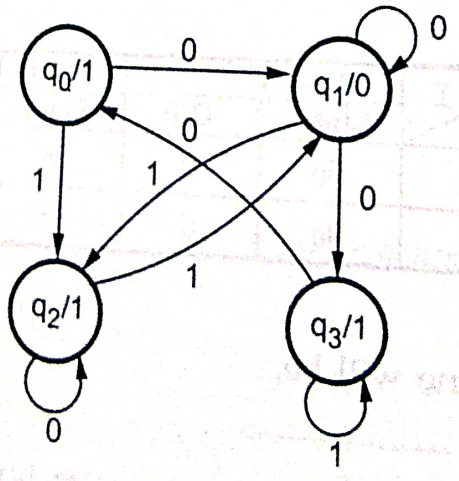
We can note that length of Moore machine is n+1 and that of Mealy machine is n.

Example 2.39 : *Construct a Mealy machine equivalent to the Moore machine given by the following table -*

March-2006, Set-2, 8 Marks

	a = 0 q_{i+1}	a = 1 q_{i+1}	Output
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

Solution : We will first draw the transition diagram from given transition diagram



Now we will construct equivalent Mealy machine as follows.

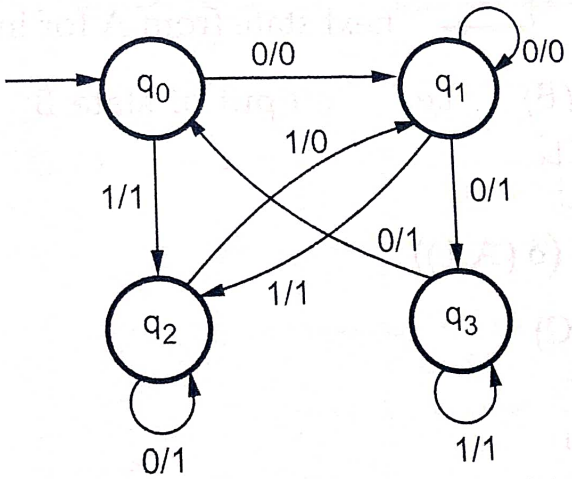


Fig. 2.70 (a)

2.9.2 Method for Conversion of Mealy Machine to Moore Machine

In Moore machine the output is associated with every state and in Mealy machine the output is given along the edge with input symbol. To convert Moore machine to Mealy machine state output symbols are distributed to input symbol paths. But while converting Mealy machine to Moore machine we will create a separate state for every new output symbol and according incoming and outgoing edges are distributed.

Let $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ be a Mealy machine then there exists a Moore machine M' equivalent to M.

The M' can be given as $M' = (Q \times \Delta, \Sigma, \Delta, \delta', \lambda', [q_0, b_0])$ where b_0 is an output symbol selected from Δ. We will calculate δ' and λ' as follows -

$$\delta'([q, b], a) = [\delta(q, a), \lambda(q, a)]$$

$$l([q, b]) = b$$

δ defines the move made by M′ on input a.

M′ on input a.

λ' defines the output made for the corresponding state q.

Thus we have to calculate δ' and λ' for $q_0, q_1, q_2, \ldots\ldots q_n$ on input $a_0, a_1, \ldots\ldots a_n$ and should emit the outputs $b_1, b_2, b_3, \ldots\ldots b_n$.

Let us understand this method of conversion with the help of some examples.

Example 2.41 : *Convert the following Mealy machine into equivalent Moore machine.*

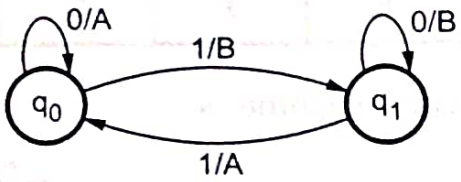


Fig. 2.73

Solution : The states for Moore machine will be $[q_0, A]$, $[q_0, B]$, $[q_1, A]$, $[q_1, B]$. Then we will calculate δ' and λ' as follows -

$$\delta'((q_0, A), 0) = [\delta(q_0, 0), \lambda(q_0, 0)]$$
$$= [q_0, A]$$
$$\lambda'([q_0, A]) = A$$
$$\delta'([q_1, A], 0) = [\delta(q_1, 0), \lambda(q_1, 0)]$$
$$= [q_1, B]$$
$$\lambda'([q_1, A]) = A$$
$$\delta'([q_1, A], 1) = [\delta(q_1, 1), \lambda(q_1, 1)]$$
$$= [q_0, A]$$
$$\lambda'([q_1, A]) = A$$

Hence,

	0	1	Output
$[q_0, A]$	$[q_0, A]$	$[q_1, B]$	A
$[q_0, B]$	$[q_0, A]$	$[q_1, B]$	B
$[q_1, A]$	$[q_1, B]$	$[q_0, A]$	A

$$\delta'([q_1, B], 0) = [\delta(q_1, 0), \lambda(q_1, 0)]$$
$$= [q_1, B]$$

$$\lambda'([q_1, B]) = B$$

$$\delta'((q_1, B), 1) = [\delta(q_1,1), \lambda(q_1,1)]$$

$$= [q_0, B]$$

$$\lambda'([q_1, B]) = B$$

Finally the transition table will be ,

$$\delta'((q_0, A), 1) = [\delta(q_0,1), \lambda(q_0,1)]$$

$$= [q_1, B]$$

$$\lambda'([q_0, A]) = A$$

Hence we can show a partial transition table as,

	0	1	Output
$[q_0, A]$	$[q_0, A]$	$[q_1, B]$	A

Continuing in this fashion we can calculate δ' and λ' as follows -

$$\delta'([q_0, B], 0) = [\delta(q_0,0), \lambda(q_0,0)]$$

$$= [q_0, A]$$

$$\lambda'[q_0, B]) = B$$

$$\delta'((q_1, B), 1) = [\delta(q_0,1), \lambda(q_0,1)]$$

$$= [q_1, B]$$

$$\lambda'([q_0, B]) = B$$

Hence

	0	1	Output
$[q_0, A]$	$[q_0, A]$	$[q_1, B]$	A
$[q_0, B]$	$[q_0, A]$	$[q_1, B]$	B

State \ I/P	0	1	Output
$[q_0, A]$	$[q_0, A]$	$[q_1, B]$	A
$[q_0, B]$	$[q_0, A]$	$[q_1, B]$	B
$[q_1, A]$	$[q_1, B]$	$[q_0, A]$	A
$[q_1, B]$	$[q_1, B]$	$[q_0, A]$	B

The transition diagram will be,

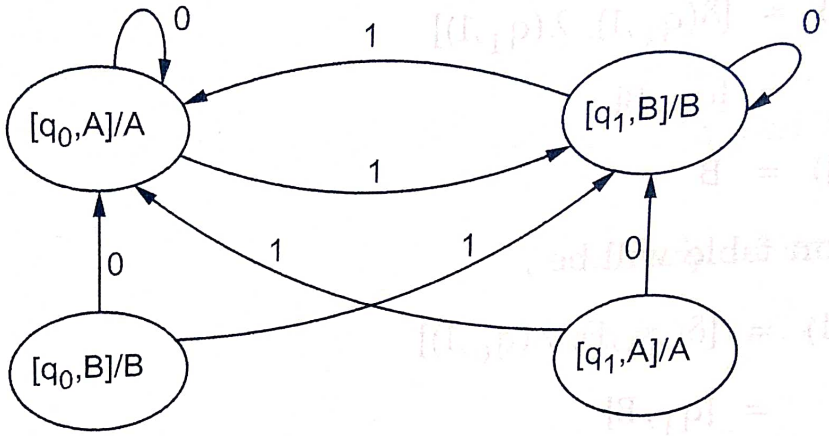


Fig. 2.74

3.2 Regular Set

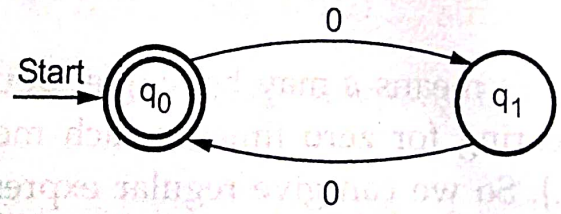
Regular sets are the sets which are accepted by finite automata.

For example : L = {ε, 00, 0000, 000000, ...}

i.e. the set of even number of zeros.

We can represent this set by a finite automata as shown in figure.

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



The given set L is said to be a regular set because it is represented by finite automata.

3.3 Regular Expressions

Let Σ be an alphabet which is used to denote the input set. The regular expression over Σ can be defined as follows.

1. ϕ is a regular expression which denotes the **empty set.**
2. ε is a regular expression and denotes the set $\{\varepsilon\}$ and it is a null string.
3. For each '*a*' in Σ '*a*' is a regular expression and denotes the set $\{a\}$.
4. If r and s are regular expressions denoting the languages L_1 and L_2 respectively, then

r+s is equivalent to $L_1 \cup L_2$ i.e. union.

rs is equivalent to $L_1 L_2$ i.e. concatenation

r^* is equivalent to L_1^* i.e. closure.

The r^* is known as **kleen closure** or closure which indicates occurrence of r for ∞ number of times.

For example if $\Sigma = \{a\}$ and we have regular expression $R = a^*$, then R is a set denoted by $R = \{\varepsilon, a, aa, aaa, aaaa, ...\}$

That is R includes any number of *a*'s as well as empty string which indicates zero number of *a*'s appearing, denoted by ε character.

Similarly there is a **positive closure** of L which can be shown as L^+. The L^+ denotes set of all the strings except the ε or null string. The null string can be denoted by ε or ^

If $\Sigma = \{a\}$ and if we have regular expression $R = a^+$ then R is a set denoted by

$$R = \{a, aa, aaa, aaaa, ...\}$$

We can construct L^* as

$$L^* = \varepsilon \cdot L^+$$

Let us try to use regular expressions with the help of some examples.

Example 3.2 : *Design the regular expression (r.e.) for the language accepting all combinations of a's except the null string over* $\Sigma = \{a\}$

Solution : The regular expression has to be built for the language

$$L = \{a, aa, aaa, ...\}$$

This set indicates that there is no null string. So we can denote r.e. as

$$R = a^+$$

As we know, positive closure indicates the set of strings without a null string.

Example 3.3 : *Design regular expression for the language containing all the strings containing any number of a's and b's.*

Solution : The regular expression will be

$$\text{r.e.} = (a + b)^*$$

This will give the set as L = {ε, *a*, *aa*, *ab*, b, b*a*, b*ab*, *abab*, ... any combination of *a* and b} .

The $(a + b)^*$ means any combination with *a* and b even a null string.

Example 3.4 : *Construct the regular expression for the language containing all strings having any number of a's and b's, except the null string.*

Solutoin : $\text{r.e.} = (a+b)^+$

This regular expression will give the set of strings of any combination of a's and b's except a null string.

Example 3.5 : *Construct the r.e. for the language accepting all the strings which are ending with 00 over the set* $\Sigma = \{0, 1\}$.

Solution : The r.e. has to be formed in which at the end, there should be 00. That means

$$\text{r.e.} = \text{(any combination of 0's and 1's) 00}$$

i.e. $$\text{r.e.} = (0+1)^* 00$$

Thus the valid string are 100, 0100, 1000, 10100 .... strings ending with 00.

Example 3.6 : *Write r.e. for the language accepting the strings which are starting with 1 and ending with 0, over the set* $\Sigma = \{0,1\}$.

Solution : The first symbol in r.e. should be 1 and the last symbol should be 0.

So, $$R = 1(0+1)^*0$$

Note that the condition is strictly followed by keeping starting and ending symbols correctly. In between them there can be any combination of 0 and 1 including a null string.

Example 3.7 : *If L = {The language starting and ending with a and having any combination of b's in between, then what is r ?*

Solution : The regular expression

$$r = a\,b^*\,a$$

Example 3.8 : *Describe in simple English the language represented by the following regular expression*

$$r = (a + ab)^*$$

Solution : We will first try to find out the set of strings, which can be possible by this r,

$$L(r) = \{a, aba, abab, aab, aaa, ...\}$$

The language is begining with *a* but not having consecutive (in *a* clump) b's.

Example 3.9 : *Write regular expression to denote the language L over Σ^*, where $\Sigma = \{a, b, c\}$ in which every string will be such that any number of a's is followed by any number of b's is followed by any number of c's.*

Solution : As we have seen any number of *a*'s means a^* any number of b's means b^* any number of c's means c^*. Since as given in problem statement, b's appear after *a*'s and c's appear after b's. So the regular expression could be -

$$r = a^*b^*c$$

Example 3.10 : *Write a regular expression to denote a language L over Σ^*, where $\Sigma = \{a, b, c\}$ such that every string will have atleast one a followed by atleast one b followed by atleast one c.*

Solution : Now, in this problem the condition is slightly changed to "atleast". That means the null string is not allowed at all. So, we can write

$$R = a^+b^+c^+$$

Example 3.11 : *Write r.e. to denote a language L which accepts all the strings which begin or end with either 00 or 11.*

Solution : The r.e. can be categorized into two subparts.

$$R = L_1 + L_2$$

L_1 = The strings which begin with 00 or 11.

L_2 = The strings which end with 00 or 11.