U18PCCS405 - FORMAL LANGUAGES & AUTOMATA THEORY

UNIT I FUNDAMENTALS

Strings, Alphabet, Language, Operations, Finite state machine, definitions, finite automaton model, acceptance of strings, and languages, deterministic finite automaton and non- deterministic finite automaton, transition diagrams and Language recognizers.

Theory of Automata

Theory of automata is a theoretical branch of computer science and mathematical. It is the study of abstract machines and the computation problems that can be solved using these machines. The abstract machine is called the automata. The main motivation behind developing the automata theory was to develop methods to describe and analyse the dynamic behaviour of discrete systems.

This automaton consists of states and transitions. The **State** is represented by **circles**, and the **Transitions** is represented by **arrows**.

Automata is the kind of machine which takes some string as input and this input goes through a finite number of states and may enter in the final state.

There are the basic terminologies that are important and frequently used in automata:

Symbols:

Symbols are an entity or individual objects, which can be any letter, alphabet or any picture.

Example:

1, a, b, #

Alphabets:

Alphabets are a finite set of symbols. It is denoted by ∑.

Examples:

∑ = {a, b}

∑ = {A, B, C, D}

∑ = {0, 1, 2}

∑ = {0, 1, ..., 5]

$\sum = \{\#, \beta, \Delta\}$

String:

It is a finite collection of symbols from the alphabet. The string is denoted by w.

Example 1:

If $\sum = \{a, b\}$, various string that can be generated from $\sum$ are {ab, aa, aaa, bb, bbb, ba, aba.....}.

- A string with zero occurrences of symbols is known as an empty string. It is represented by ε.
- The number of symbols in a string w is called the length of a string. It is denoted by |w|.

Example 2:

w = 010

Number of Sting |w| = 3

Language:

A language is a collection of appropriate string. A language which is formed over $\sum$ can be **Finite** or **Infinite**.

Example: 1

L1 = {Set of string of length 2}

= {aa, bb, ba, bb} **Finite Language**

Example: 2

L2 = {Set of all strings starts with 'a'}

= {a, aa, aaa, abb, abbb, ababb} **Infinite Language**

- ✓ **Power of an Alphabet:**
  - ✓ If $\sum$ is an alphabet, we can express the set of all strings of certain length from that alphabet by using an exponential notation. It is denoted by $\sum^k$ is the set of strings of length k, each of whose symbols is in $\sum$.

Example :

$\sum = \{0,1\}$ has 2 symbols

i) $\sum^1 = \{0,1\}$ $(\therefore 2^1 = 2)$

ii) $\sum^2 = \{00, 01, 10, 11\}$ $(\therefore 2^2 = 4)$

iii) $\sum^3 = \{000,001,010,011,100,101,110,111\}$ $(\therefore 2^3 = 8)$

- ✓ The set of strings over an alphabet $\sum$ is usually denoted by $\sum^*$.
  For instance, $\sum^* = \{0,1\}^* = \{\epsilon,0,1,00,01,10,11\}$
  $(\therefore \sum^* = \sum^0 \cup \sum^1 \cup \sum^2 \ldots\ldots)$ - with ϵ symbol.

- ✓ The set of strings over an alphabet $\sum$ excluding ϵ is usually denoted by $\sum^+$.
  For instance, $\sum^+ = \{0,1\}^+ = \{0,1,00,01,10,11\}$
  $(\therefore \sum^+ = \sum^* - \{\epsilon\}$ or $\sum^1 \cup \sum^2 \cup \sum^3 \ldots\ldots)$
  - without ϵ symbol.

Parts of the string:

Parts of a string

Prefix of S

A string obtained by removing zero or more trailing symbols of string S.

Example

ban is a prefix of banana.

Suffix of S

A string formed by deleting zero or more of the leading symbols of S.

Example

nana is a suffix of banana.

Substring of S

A string obtained by deleting a prefix and a suffix from S.

Example

nan is a substring of banana.

Every prefix and every suffix of S is a substring of S, but not every substring of S is a prefix or a suffix of S.

For every string S, both S and ϵ are prefixes, suffixes, and substrings of S.

Proper prefix

Any nonempty string X that is a prefix of S such that $S \neq X$.

Proper suffix

Any nonempty string X that is a suffix of S such that $S \neq X$.

Proper substring

Any nonempty string X that is a substring of S such that $S \neq X$.

Subsequence of S

Any string formed by deleting zero or more not necessarily contiguous symbols from S.

Example

baaa is a subsequence of banana.

- ✓ **Operations on Languages**
  - ✓ **Complementation**
    Let L be a language over an alphabet Σ. The complementation of L, denoted by $\bar{L}$, is $\Sigma^* - L$.
  - ✓ **Union**
    Let L_1 and L_2 be languages over an alphabet Σ. The union of L_1 and L_2, denoted by $L_1 \cup L_2$, is $\{x \mid x \text{ is in } L_1 \text{ or } L_2\}$.
  - ✓ **Intersection**
    Let L_1 and L_2 be languages over an alphabet Σ. The intersection of L_1 and L_2, denoted by $L_1 \cap L_2$, is $\{x \mid x \text{ is in } L_1 \text{ and } L_2\}$.

- ✓ **Concatenation**
  Let L_1 and L_2 be languages over an alphabet Σ. The concatenation of L_1 and L_2, denoted by $L_1 \cdot L_2$, is $\{w_1 \cdot w_2 \mid w_1 \text{ is in } L_1 \text{ and } w_2 \text{ is in } L_2\}$.
- ✓ **Reversal**
  Let L be a language over an alphabet Σ. The reversal of L, denoted by L^r, is $\{w^r \mid w \text{ is in } L\}$.
- ✓ **Kleene's closure**
  Let L be a language over an alphabet Σ. The Kleene's closure of L, denoted by L^*, is $\{x \mid \text{for an integer } n \geq 0\ x = x_1 x_2 \ldots x_n \text{ and } x_1, x_2, \ldots, x_n \text{ are in } L\}$.

$$L^* = \bigcup_{i=0}^{\infty} L^i \qquad (\text{e.g. } a^* = \{\varepsilon, a, aa, aaa, \ldots\})$$

- ✓ **Positive Closure**
  Let L be a language over an alphabet Σ. The closure of L, denoted by L+, is $\{x \mid \text{for an integer } n \geq 1,\ x = x_1 x_2 \ldots x_n \text{ and } x_1, x2, \ldots, x_n \text{ are in } L\}$

$$L^+ = \bigcup_{i=1}^{\infty} L^i \qquad (\text{e.g. } a^* = \{a, aa, aaa, \ldots\})$$

2.2 Finite State Machine

The finite state system represents **a mathematical model** of a system with certain input. The model finally gives certain output. The input when is given to the machine it is processed by various states, these states are called as **intermediate states.**

The very good example of finite state system is a control mechanism of elevator. This mechanism only remembers the current floor number pressed, it does not remember all the previously pressed numbers.

The finite state system is a very good design tool for the programs such as text editors and lexical analyzers (which is used in compilers). The lexical analyzer is a program which scans your program character by character and recognizes those words as tokens.

2.2.1 Definition

A finite automata is a collection of 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where,

Q is a finite **set of states**, which is non empty.

Σ is **input** alphabet, indicates input set.

q_0 is an **initial state** and q_0 is in Q i.e. $q_0 \in Q$.

F is a set of **final states.**

δ is a **transition function** or a mapping function. Using this function the next state can be determined.

2.2.2 Finite Automata Model

The finite automata can be represented as :

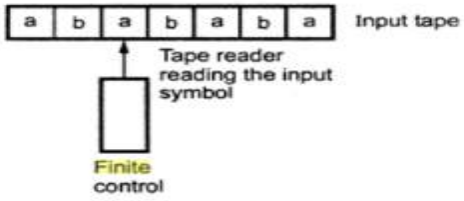


Fig. 2.1 Model for finite automata

The finite automata can be represented using

i) Input tape - It is a linear tape having some number of cells. Each input symbol is placed in each cell.

ii) Finite control - The finite control decides the next state on receiving particular input from input tape. The tape reader reads the cells one by one from left to right and at a time only one input symbol is read.

For example, suppose current state is q, and suppose reader is reading the symbol 'a' then it is finite control which decides what will be the next state at input 'a'. The transition from current state q with input w to next state q′ producing w′ will be represented as,

$$(q, w) \vdash (q', w')$$

If w is a string and M is a finite automata, then w is accepted by the FA.

iff $(w, s) \vdash^{*} (q, \varepsilon)$

with q as final state.

The set of strings accepted by a FA given by M then M is accepted by language L. The acceptance of M by some language L is denoted by L (M).

A machine M accepts a language L iff,

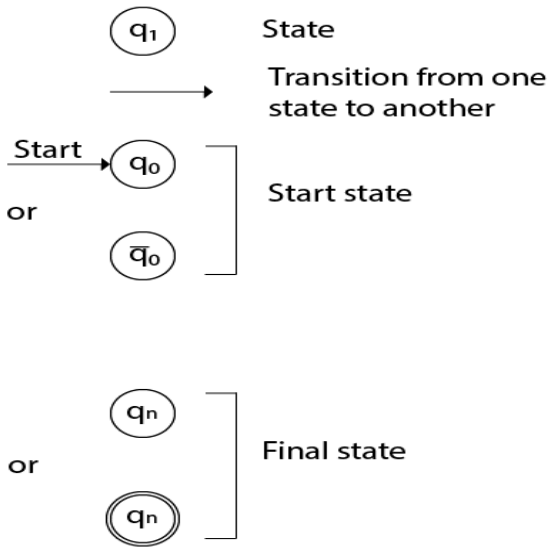
$$L = L(M).$$

Transition Diagram:

A transition diagram or state transition diagram is a directed graph which can be constructed as follows:

- There is a node for each state in Q, which is represented by the circle.
- There is a directed edge from node q to node p labeled a if $\delta(q, a) = p$.
- In the start state, there is an arrow with no source.
- Accepting states or final states are indicating by a double circle.

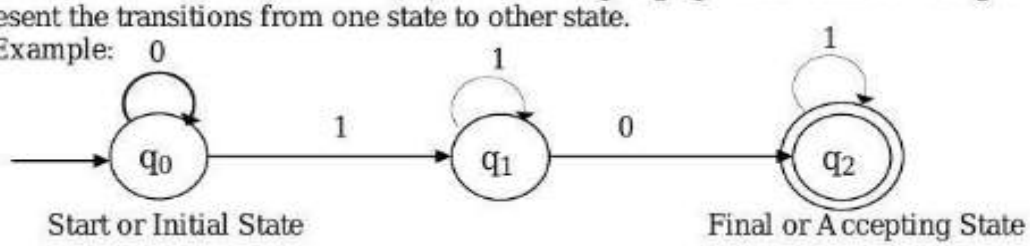
Some Notations that are used in the transition diagram:



✓ **Transition Diagram (Transition graph)**

It is a directed graph associated with the vertices of the graph corresponds to the states of the finite automata. (or) It is a 5-tuple graph used state and edges represent the transitions from one state to other state.

Example:



✓ **Transition Table.**

It is the tabular representation of the DFA. For a transition table the transition function is used.

Example:

States	Input	
	0	1
→{q0}	{q1}	{q0}
{q1}	-	{q2}
*{q2}	-	-

✓ **Transition Function.**

- The mapping function or transition function denoted by δ.
- Two parameters are passed to this transition function: (i) current state and (ii) input symbol.
- The transition function returns a state which can be called as next state.

δ (current_state, current_input_symbol) = next_state

Example:

δ (q0, a) = q1

✓ **Computation of a Finite Automaton**

- The automaton receives the input symbols one by one from left to right.
- After reading each symbol, M1 moves from one state to another along the transition that has that symbol as its label.
- When M1 reads the last symbol of the input it produces the output: accept if M1 is in an accept state, or reject if M1 is not in an accept state.

- ✓ **Applications**
  - It plays an important role in complier design.
  - In switching theory and design and analysis of digital circuits automata theory is applied.
  - Design and analysis of complex software and hardware systems.
  - To prove the correctness of the program automata theory is used.
  - To design finite state machines such as Moore and mealy machines.
  - It is base for the formal languages and these formal languages are useful of the programming languages.

- ✓ **Types of Finite Automata**
  - Finite Automata without output
    - Deterministic Finite Automata (DFA)
    - Non-Deterministic Finite Automata (NFA or NDFA)
    - Non-Deterministic Finite Automata with ε move (ε-NFA or ε-NDFA)
  - Finite Automata with output
    - Moore Machine
    - Mealy Machine

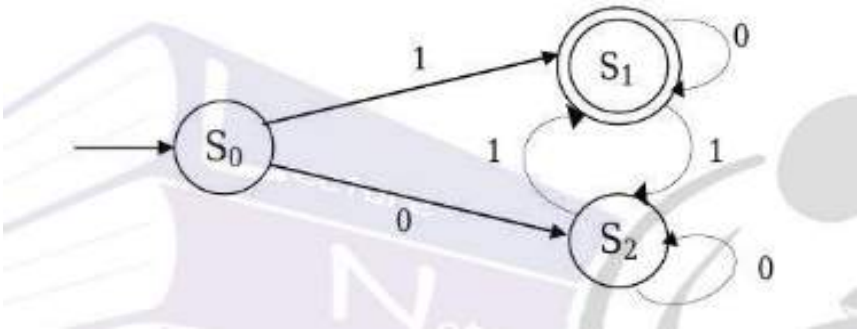
Deterministic Finite Automata (DFA)

Deterministic Finite Automaton is a FA in which there is **only one path for a specific input from current state to next state.** There is a unique transition on each input symbol.

- ✓ Formal Definition of Deterministic Finite Automata

A finite automaton is a 5-tuples; they are $M=(Q, \Sigma, \delta, q_0, F)$
where

Q is a finite set called the states
Σ is a finite set called the alphabet
$\delta : Q \times \Sigma \rightarrow Q$ is the transition function
$q_0 \in Q$ is the start state also called initial state
$F \subseteq Q$ is the set of accept states, also called the final states



Non-Deterministic Finite Automata (NDFA or NFA)

Non-Deterministic Finite Automaton is a FA in which there **many paths for a specific input from current state to next state.**

- ✓ Formal Definition of Non-Deterministic Finite Automata

  A finite automaton is a 5-tuples; they are $M=(Q, \Sigma, \delta, q_0, F)$

  where

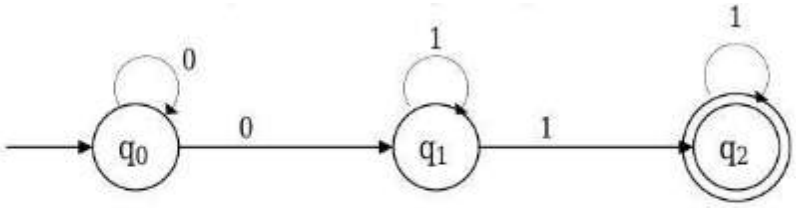
  Q is a finite set called the states

  Σ is a finite set called the alphabet

  $\delta : Q \times \Sigma \rightarrow 2^Q$ is the transition function

  $q_0 \in Q$ is the start state also called initial state

  $F \subseteq Q$ is the set of accept states, also called the final states



Finite Automaton with ε- moves

The finite automata is called NFA when there exists **many paths for a specific input or ε from current state to next state.** The **ε is a character** used to indicate null string.

- ✓ Formal Definition of Non-Deterministic Finite Automata

  A finite automaton is a 5-tuples; they are $M=(Q, \Sigma, \delta, q_0, F)$

  where

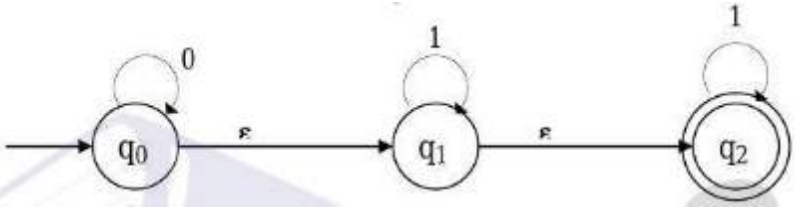
  Q is a finite set called the states

  Σ is a finite set called the alphabet

  $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$ is the transition function

  $q_0 \in Q$ is the start state also called initial state

  $F \subseteq Q$ is the set of accept states, also called the final states



Differentiate DFA and NFA

Sl. No	DFA	NFA
1.	DFA is Deterministic Finite Automata	NFA is Non-Deterministic Finite Automata
2.	For given state, on a given input we reach to deterministic and unique state.	For given state, on a given input we reach to more than one state.
3.	DFA is a subset of NFA	Need to convert NFA to DFA in the design of complier.
4.	$\delta : Q \times \Sigma \rightarrow Q$ Example: $\delta(q_0, a) = \{q_1\}$	$\delta : Q \times \Sigma \rightarrow 2^Q$ Example : $\delta(q_0, a) = \{q_1, q_2\}$

Some important points about DFA and NFA:

DFA

DFA refers to deterministic finite automata. Deterministic refers to the uniqueness of the computation. In the DFA, the machine goes to one state only for a particular input character. DFA does not accept the null move.

NFA

NFA stands for non-deterministic finite automata. It is used to transmit any number of states for a particular input. It can accept the null move.

1. Every DFA is NFA, but NFA is not DFA.
2. There can be multiple final states in both NFA and DFA.
3. DFA is used in Lexical Analysis in Compiler.
4. NFA is more of a theoretical concept.

Example 2.1 : *Design a FA which accepts the only input 101 over the input set* $Z = \{0, 1\}$.

Solution :

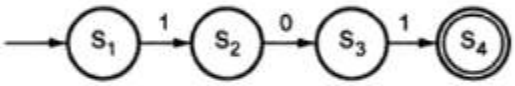


Fig. 2.5

Note that in the problem statement it is mentioned as only input 101 will be accepted. Hence in the solution we have simply shown the transitions, for input 101. There is no other path shown for other input.

Example 2.2 : *Design a FA which checks whether the given binary number is even.*

Solution : The binary number is made up of 0's and 1's when any binary number ends with 0 it is always even and when a binary number ends with 1 it is always odd. For example,

0100 is a even number, it is equal to 4.

0011 is a odd number, it is equal to 3.

And so while designing FA we will assume one start state, one state ending in 0 and other state for ending with 1. Since we want to check whether given binary number is even or not, we will make the state for 0, the final state.

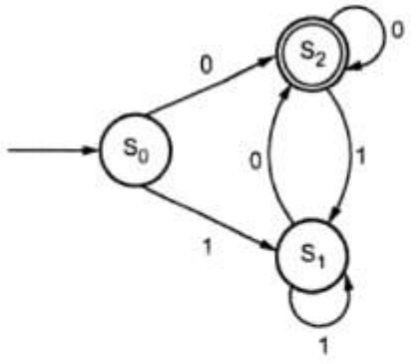


Fig. 2.6

The FA indicates clearly S_1 is a state which handles all the 1's and S_2 is a state which handles all the 0's. Let us take some input.

$$01000 \Rightarrow 0\underline{S_2}1000$$

$$01\underline{S_1}000$$

$$010\underline{S_2}00$$

$$0100\underline{S_2}0$$

$$01000S_2$$

Now at the end of input we are in final or in accept state so it is a even number. Similarly let us take another input.

$$1011 \Rightarrow 1\underline{S_1}011$$

$$10\underline{S_2}11$$

$$101\underline{S_1}1$$

$$1011S_1$$

Now we are at the state S_1 which is a non final state.

Another idea to represent FA with the help of transition table.

Input / States	0	1
→ S_0	S_2	S_1
S_1	S_2	S_1
(S_2)	S_2	S_1

Transition table

Thus the table indicates in the first column all the current states. And under the column 0 and 1 the next states are shown.

The first row of transition table can be read as : When current state is S_0, on input 0 the next state will be S_2 and on input 1 the next state will be S_1. The arrow marked to S_0 indicates that it is a start state and circle marked to S_2 indicates that it is a final state.

Example 2.3 : *Design FA which accepts only those strings which start with 1 and ends with 0.*

Solution : The FA will have a start state A from which only the edge with input 1 will go to next state.

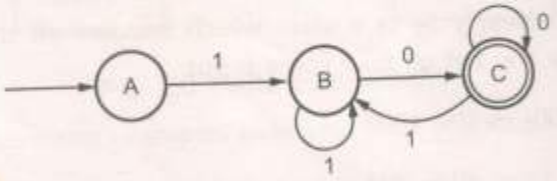


Fig. 2.6 For Ex. 2.3

In state B if we read 1 we will be in B state but if we read 0 at state B we will reach to state C which is a final state. In state C if we read either 0 or 1 we will go to state C or B respectively. Note that the special care is taken for 0, if the input ends with 0 it will be in final state.

Example 2.4 : *Design FA which accepts odd number of 1's and any number of 0's.*

Solution : In the problem statement, it is indicated that there will be a state which is meant for odd number of 1's and that will be the final state. There is no condition on number of 0's.

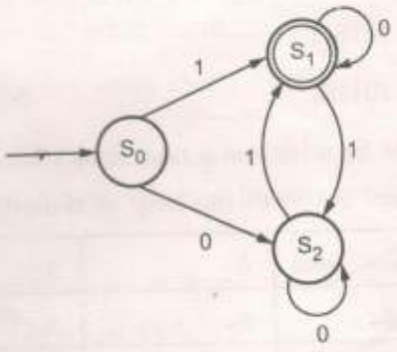


Fig. 2.7 For Ex. 2.4

At the start if we read input 1 then we will go to state S_1 which is a final state as we have read odd number of 1's. There can be any number of zeros at any state and therefore the self loop is applied to state S_2 as well as to state S_1. For example if the input is 10101101, in this string there are any number of zeros but odd number of ones. The machine will derive this input as follows -

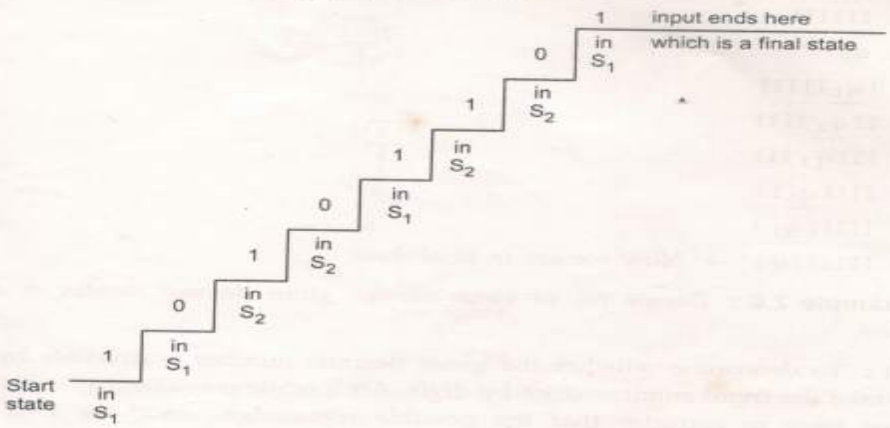


Fig. 2.8 Ladder diagram of processing the input

Example 2.5 : *Design FA which checks whether the given unary number is divisible by 3.*

Solution :

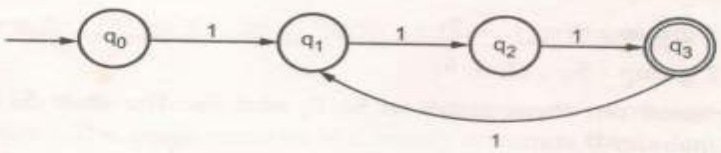


Fig. 2.9 For Ex. 2.5

The unary number is made up of ones. The number 3 can be written in unary form as 111, number 5 can be written as 11111 and so on. The unary number which is divisible by 3 can be 111 or 111111 or 111111111 and so on. The transition table is as follows

Input	1
→ q_0	q_1
q_1	q_2
q_2	q_3
(q_3)	q_1

Consider a number 111111 which is equal to 6 i.e. divisible by 3. So after complete scan of this number we reach to final state q_3.

Start 111111

State q_0

$1\underline{q_1}11111$

$11\underline{q_2}1111$

$111q_3111$

$1111\underline{q_1}11$

$11111\underline{q_2}1$

$111111q_3$ → Now we are in final state.

Example 2.6 : *Design FA to check whether given decimal number is divisible by three.*

Solution : To determine whether the given decimal number is divisible by three, we need to take the input number digit by digit. Also, while considering its divisibility by three, we have to consider that the possible remainders could be 1, 2 or 0. The remainder 0 means, it is divisible by 3. Hence from input set {0, 1, 2, ..., 9} (since decimal number is a input), we will get either remainder 0 or 1 or 2 while testing its divisibility by 3. So we need to group these digits according to their remainders. The groups are as given below -

remainder 0 group : S_0 : (0, 3, 6, 9)

remainder 1 group : S_1 : (1, 4, 7)

remainder 2 group : S_2 : (2, 5, 8)

We have named out these states as S_0, S_1 and S_2. The state S_0 will be the final state as it is remainder 0 state.

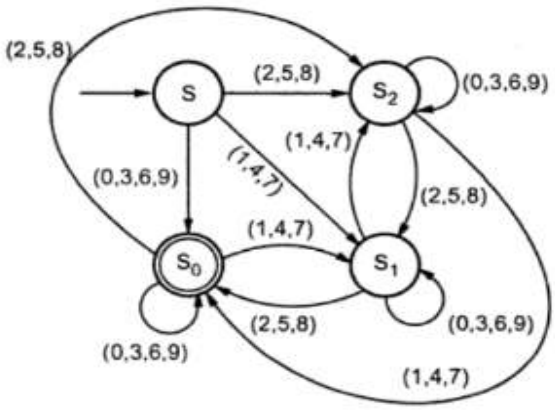


Fig. 2.11

Let us test the above FA, if the number is 36 then it will proceed by reading the number digit by digit.

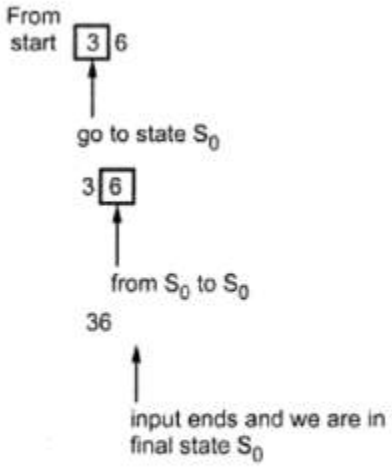


Fig. 2.12

Hence the number is divisible 3, isn't it ? Similarly if number is 121 which is not divisible by 3, it will be processed as

S 121

$1\underline{S_1}\,21$

$12\underline{S_0}\,1$

$121\,S_1$ which is remainder 1 state.

Example 2.7 : *Design FA which checks whether a given binary number is divisible by three.*

Solution : As we have discussed in previous example, the same logic is applicable to this problem even ! The input number is a binary number. Hence the input set will be $\Sigma = \{0, 1\}$. The start state is denoted by S, the remainder 0 is by S_0, remainder 1 by S_1 and remainder 2 by S_2.

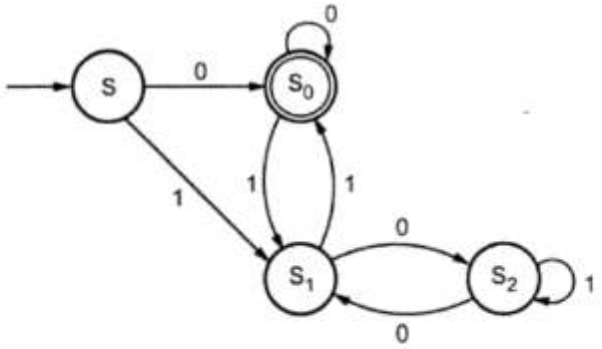


Fig. 2.13

In the above transition graph we have handled the case as if the input starts with 1 then it will be in q_3 state which is a dead state and never lead to final state. Thus this machine strictly handles the strings starting with 0 and ending with 1.

Example 2.11 : *Design FA to accept L, where L = {Strings in which a always appears trippled} over the set* $\Sigma = \{a, b\}$.

Solution : For this particular language the valid strings are *aaa*b, b*aaaaaa*, bb*aaa*b and so on. The *a* always appears in a clump of 3. The TG (transition graph) will look like this -

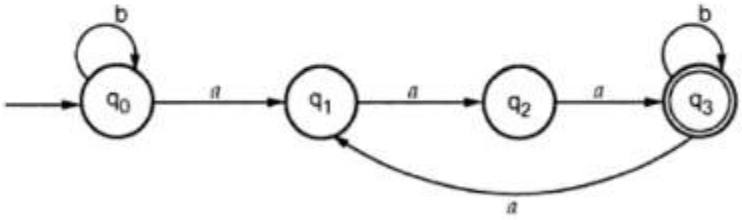


Fig. 2.18

Note that the sequence of tripple *a* is maintained to reach to the final state.

Note the difference between previous example and this, here there is no condition as *a* should be in clump but total number of *a*'s in a string are divisible by 3. Hence b's are allowed in between.

Example 2.12 : *Design FA to accept L where all the strings in L are such that total number of a's in them are divisible by 3.*

Solution : As we have seen earlier, while testing divisibility by 3, we group the input as remainder 0, remainder 1 and remainder 2.

Hence,

S_0 : State of remainder 0.

S_1 : State of remainder 1.

S_2 : State of remainder 2.

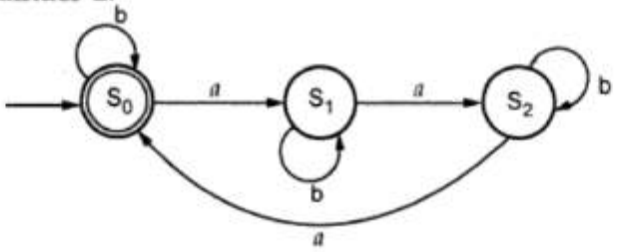


Fig. 2.19

Example 2.13 : *Design a FA that reads strings made up of letters in the word. CHARIOT and recognize those strings that contain the word 'CAT' as a substring.*

Solution : To design this FA we will first consider the input set which will be $\Sigma = \{C, H, A, R, I, O, T\}$. From this input set we have to recognize the word 'CAT'. We can do that as follows -

Step 1 :

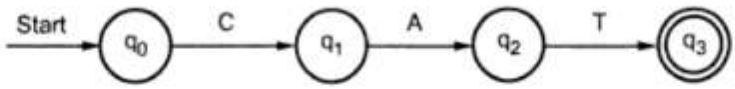


Fig. 2.20

Step 2 :

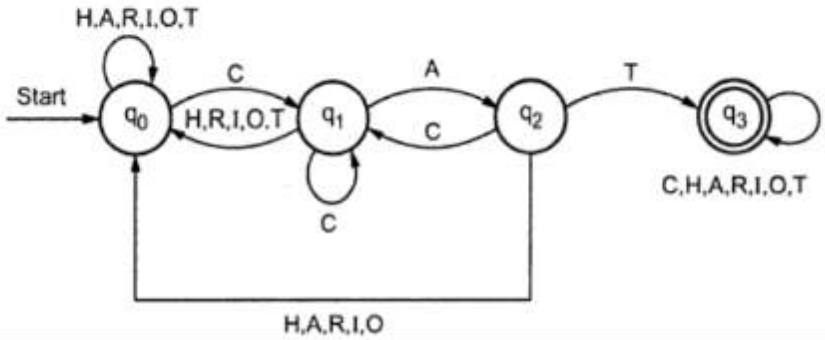


Fig. 2.21 Required finite automata

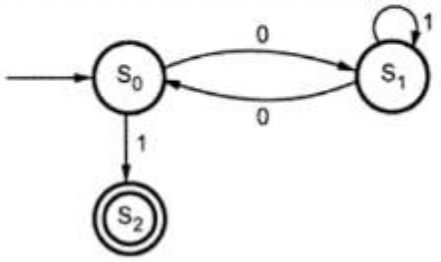
From Fig. 2.21 it is clear that on recognizing C initially from Σ = {C, H, A, R, I, O, T} we are moving to state q_1, other than C all the characters are remaining in state q_0 only. From q_1 the character A will be identified and from q_2 character T will be identifier. The transition table can be drawn as follows :

State \ Input	C	H	A	R	I	O	T
$\rightarrow q_0$	q_1	q_0	q_0	q_0	q_0	q_0	q_0
q_1	q_1	q_0	q_2	q_0	q_0	q_0	q_0
q_2	q_1	q_0	q_0	q_0	q_0	q_0	q_3
(q_3)	q_3	q_3	q_3	q_3	q_3	q_3	q_3

The substring may appear anywhere in the input string for any number of times. For example, 'HACCATHA'. From this 'CAT' can be recognized and we should reach to q_3 state finally.

Example 2.18 : *Design a DFA which accepts strings with even number of 0's followed by single 1 over* $\Sigma = \{0, 1\}$.

Solution : The DFA can be shown by a transition diagram as



Here state S_1 will accept all the odd number of 0's and S_0 will accept all even number of 0's. It should then be followed by single 1. Hence S_2 is a final state. Consider the input

	0	0	0	1	0	1
0	S_1	0	0	1	0	1
0	0	S_0	0	1	0	1
0	0	0	S_1	1	0	1
0	0	0	1	S_1	0	1
0	0	0	1	0	S_0	1
0	0	0	1	0	1	S_2

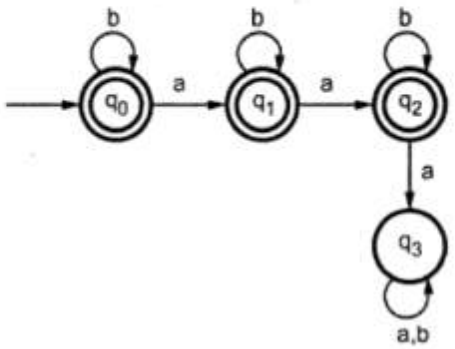
We now will reach to final state S_2, and the input ends here. Hence 000101 is a valid input string. The transition table for above drawn DFA can be represented as

Input / States	0	1
S_0	S_1	S_2
S_1	S_0	S_1
((S_2))	-	-

Transition table

Example 2.19 : *Design DFA which accepts all the strings not having more than two a's over* $\Sigma = \{a, b\}$.

Solution : In this DFA maximum two a's are accepted. If we try to accept third a then it should not lead us to final state. Such a DFA can be as shown below.



The transition table can be as shown -

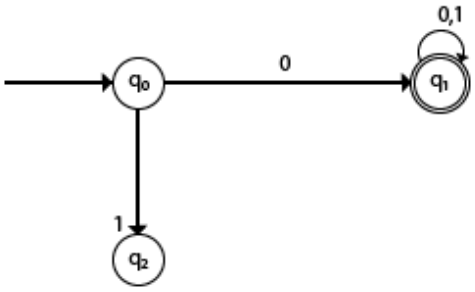
Input / States	a	b
q_0	q_1	q_0
q_1	q_2	q_1
q_2	q_3	q_2
q_3	q_3	q_3

Transition table

Example 2:

DFA with Σ = {0, 1} accepts all starting with 0.

Solution:



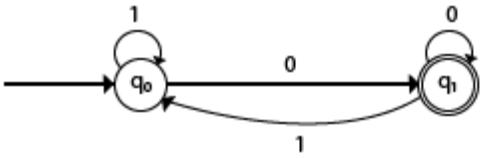
Explanation:

- In the above diagram, we can see that on given 0 as input to DFA in state q0 the DFA changes state to q1 and always go to final state q1 on starting input 0. It can accept 00, 01, 000, 001....etc. It can't accept any string which starts with 1, because it will never go to final state on a string starting with 1.

Example 3:

DFA with Σ = {0, 1} accepts all ending with 0.

Solution:



Explanation:

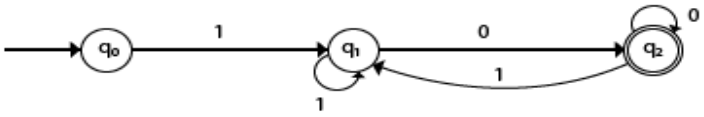
In the above diagram, we can see that on given 0 as input to DFA in state q0, the DFA changes state to q1. It can accept any string which ends with 0 like 00, 10, 110, 100....etc. It can't accept any string which ends with 1, because it will never go to the final state q1 on 1 input, so the string ending with 1, will not be accepted or will be rejected.

Example 1:

Design a FA with Σ = {0, 1} accepts those string which starts with 1 and ends with 0.

Solution:

The FA will have a start state q0 from which only the edge with input 1 will go to the next state.



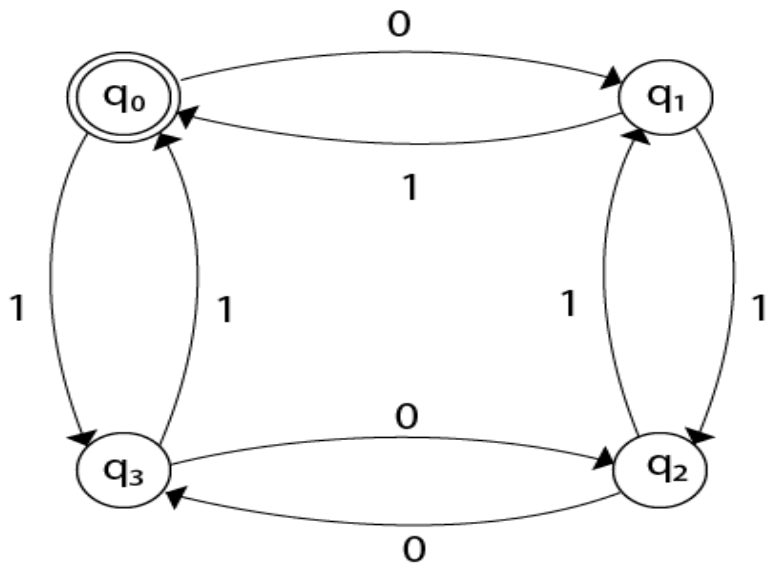
In state q1, if we read 1, we will be in state q1, but if we read 0 at state q1, we will reach to state q2 which is the final state. In state q2, if we read either 0 or 1, we will go to q2 state or q1 state respectively. Note that if the input ends with 0, it will be in the final state.

Example 3:

Design FA with Σ = {0, 1} accepts even number of 0's and even number of 1's.

Solution:

This FA will consider four different stages for input 0 and input 1. The stages could be:



Here q0 is a start state and the final state also. Note carefully that a symmetry of 0's and 1's is maintained. We can associate meanings to each state as:

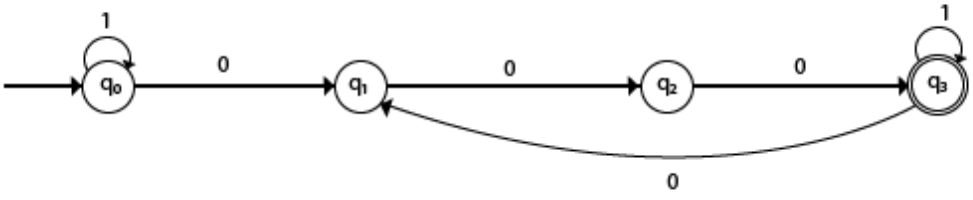
q0: state of even number of 0's and even number of 1's.
q1: state of odd number of 0's and even number of 1's.
q2: state of odd number of 0's and odd number of 1's.
q3: state of even number of 0's and odd number of 1's.

Example 4:

Design FA with Σ = {0, 1} accepts the set of all strings with three consecutive 0's.

Solution:

The strings that will be generated for this particular languages are 000, 0001, 1000, 10001, ... in which 0 always appears in a clump of 3. The transition graph is as follows:

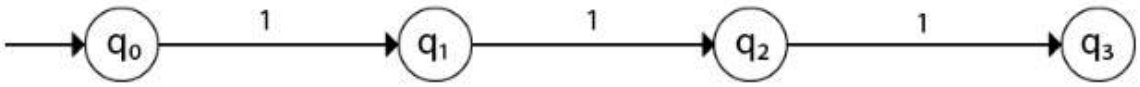


Example 5:

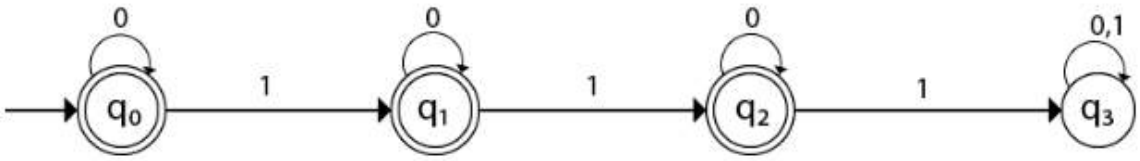
Design a DFA L(M) = {w | w ε {0, 1}*} and W is a string that does not contain consecutive 1's.

Solution:

When three consecutive 1's occur the DFA will be:



Here two consecutive 1's or single 1 is acceptable, hence



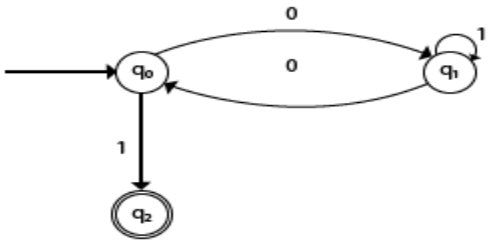
The stages q0, q1, q2 are the final states. The DFA will generate the strings that do not contain consecutive 1's like 10, 110, 101,..... etc.

Example 6:

Design a FA with Σ = {0, 1} accepts the strings with an even number of 0's followed by single 1.

Solution:

The DFA can be shown by a transition diagram as:



Acceptance of strings, and languages:

Example1:

Consider the transition diagram. In our formal notation this FA is denoted $M = (Q, \Sigma, \delta, q_0, F)$, where $Q = \{q_0, q_1, q_2, q_3\}$

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

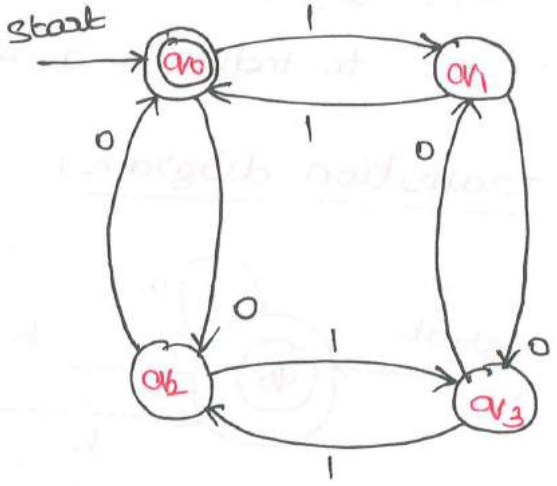
$F = \{q_0\}$

δ is

Transition Table:

	States	Inputs	
		0	1
→ *	q_0	q_2	q_1
	q_1	q_3	q_0
	q_2	q_0	q_3
	q_3	q_1	q_2

Transition diagram



Transition function

$\delta(q_0, 0) = q_2$

$\delta(q_0, 1) = q_1$

$\delta(q_1, 0) = q_3$

$\delta(q_1, 1) = q_0$

$\delta(q_2, 0) = q_0$

$\delta(q_2, 1) = q_3$

$\delta(q_3, 0) = q_1$

$\delta(q_3, 1) = q_2$

Suppose 11010l is input to M.

(i) to show the string 11010l is accepted or not by DFA.

Method 1: 11010l

$\delta(q_0, 1) = q_1$

$\delta(q_1, 1) = q_0$

$\delta(q_0, 0) = q_2$

$\delta(q_2, 1) = q_3$

$\delta(q_3, 0) = q_1$

$\delta(q_1, 1) = q_0$

Since the q_0 is the final state, the given input string is accepted.

Method 2:

$$\begin{aligned}
&\delta(q_0, 110101) \\
&= \delta(\delta(q_0,1), 10101) \\
&= \delta(q_1, 10101) \\
&= \delta(\delta(q_1,1), 0101) \\
&= \delta(q_0, 0101) \\
&= \delta(\delta(q_0,0), 101) \\
&= \delta(q_2, 101) \\
&= \delta(\delta(q_2,1), 01) = \delta(q_3, 01) \\
&= \delta(\delta(q_3,0), 1) = \delta(q_1, 1) \\
&= q_0
\end{aligned}$$

Here q_0 is the final state.

$\therefore$ It is accepted.

Method 3: 110101

① $\delta(q_0,1) = q_1$ and $\delta(q_1,1) = q_0$

$\delta(q_0,11) = \delta(\delta(q_0,1),1) = \delta(q_1,1) = q_0$

② $\delta(q_0,0) = q_2$

$\delta(q_2,110) = \delta(\delta(q_0,11),0)$

$= \delta(q_0,0)$

$\delta(q_0,110) = q_2$

③ $\delta(q_0, 1101) = \delta(\delta(q_0, 110), 1)$

$= \delta(q_2, 1)$

$= q_3$

④ $\delta(q_0, 11010) = \delta(\delta(q_0, 1101), 0)$

$= \delta(q_3, 0)$

$= q_1$

⑤ $\delta(q_0, 110101) = \delta(\delta(q_0, 11010), 1)$

$= \delta(q_1, 1)$

$= q_0$

the entire sequence of states is

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

∴ thus 110101 is in $L(M)$.

Language accepted by DFA:

> A string x is said to be accepted by a finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ if

$\delta(q_0, x) = p$ for some p in F.

> The language accepted by M, designated $L(M)$, is the set $\{x \mid \delta(q_0, x)$ is in $F\}$.

> A language is a regular set if it is the set accepted by some finite automaton.

① To show the string 100 is accepted or not by DFA:

100 : input string

$\delta(q_0, 1) = q_1$

$\delta(q_1, 0) = q_3$

$\delta(q_3, 0) = q_1$

∴ Since q_1 is not a final state, the string 100 is not accepted.

② To show the string 1110 is accepted or not by DFA

1110 : input string

$\delta(q_0, 1) = q_1$

$\delta(q_1, 1) = q_0$

$\delta(q_0, 1) = q_1$

$\delta(q_1, 0) = q_3$

Since q_3 is not a final state, the string 1110 is not accepted.

Example2:

Consider the DFA, $M = (Q, \Sigma, \delta, q_0, F)$, where δ is given by,

	States	Input symbol	
		0	1
→	q_0	q_0	q_1
	q_1	q_0	q_2
*	q_2	q_2	q_1

Show that the string 011010 is accepted or not.

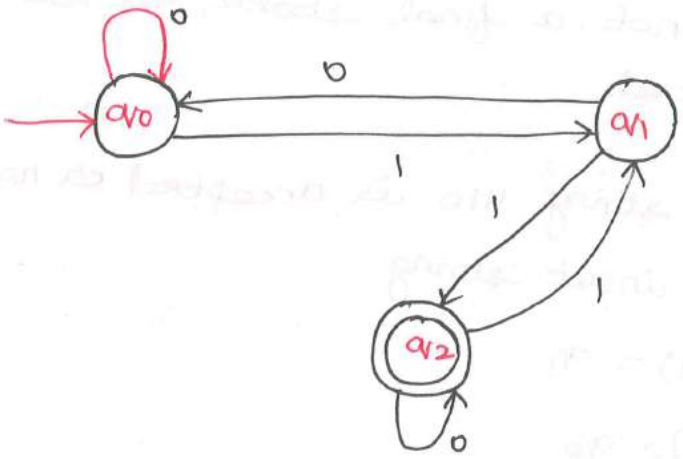
Let $Q = \{q_0, q_1, q_2\}$

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

$S = q_0$

$F = q_2$

Transition diagram:



the string accepted by this DFA = { 0110, 010110, 110, 11011, 0110 }

Transition function

$\delta(q_0, 0) = q_0$

$\delta(q_0, 1) = q_1$

$\delta(q_1, 0) = q_0$

$\delta(q_1, 1) = q_2$

$\delta(q_2, 0) = q_2$

$\delta(q_2, 1) = q_1$

To show that the string is accepted or not.

011010 : i/p string

$= \delta(q_0, 011010)$

$= \delta(\delta(q_0, 0), 11010)$

$= \delta(q_0, 11010)$

$= \delta(\delta(q_0, 1), 1010)$

$= \delta(q_1, 1010)$

$= \delta(\delta(q_1, 1), 010)$

$= \delta(q_2, 010)$

$= \delta(\delta(q_2, 0), 10)$

$= \delta(q_2, 10)$

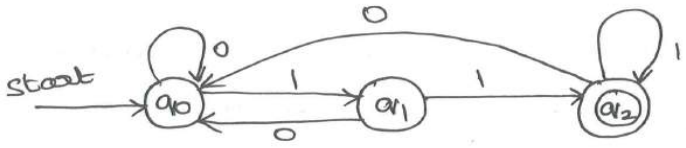
$= \delta(\delta(q_2, 1), 0)$

$= \delta(q_1, 0)$

$= q_0 \notin F$

the string is not accepted.

③ Show that the language is accepted by DFA is given as



the DFA is given by $M = (Q, \Sigma, \delta, S, F)$ here

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

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

$S = q_0 \in Q$

$F = \{q_2\} \subseteq Q$

Transition Table:

	States	Input symbol	
		0	1
→	q_0	q_0	q_1
	q_1	q_0	q_2
*	q_2	q_0	q_2

∴ the language accepted by DFA M is L(M).

$L(M) = \{0111, 0110111, 0101111, 111, 0110111\}$. set of strings accepted by DFA is L(M).

properties of Transition Function (δ)

(i) $\delta(q, \in) = q$.

this means that the state of the system can be changed only by an input symbol, remains in the original state.

Eg: $\delta(q_0, \in) = q_0$

$\delta(q_1, \in) = q_1$

No input supplied, it will be in the same state.

(ii) For all the strings 'w' and an input symbol 'a'.

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

eg: $\delta(q_0, aw) = \delta(\delta(q_0,a), w)$

$\delta(q_0, 1101) = \delta(\delta(q_0,1), 101)$

(iii) the transition function δ can be extended to $\bar{\delta}$ to operate on strings.

$$\bar{\delta}(q, \epsilon) = q$$
$$\bar{\delta}(q, xa) = \delta(\bar{\delta}(q,x), a)$$

NON-DETERMINISTIC FINITE AUTOMATA (NFA)

the finite automaton model to allow zero, one or more transitions from a state on the same input symbol. this new model is called a non-deterministic finite Automaton (NFA).

* Any set accepted by a NFA can also be accepted by a DFA.
* NFA is a useful concept in proving theorems.

* Non-determinism plays a central role in both the theory of languages and the theory of computation.

the NFA is defined by 5 tuples $(Q, \Sigma, \delta, S, F)$, where

Q - finite non-empty set of states

Σ - a finite set of input symbols

S - start states

F - set of final states.

the transition function δ is defined by

δ :- transition function that takes state and an input symbol as arguments and returns one (or) more states as output.

$\therefore \delta = Q \times \Sigma^* \text{ to } 2^Q = \hat{\delta}$

Extended transition function ($\bar{\delta}$)

(i) $\bar{\delta}(q, \in) = q$

(ii) $\bar{\delta}(q, xa) = \cup \delta(p, a)$

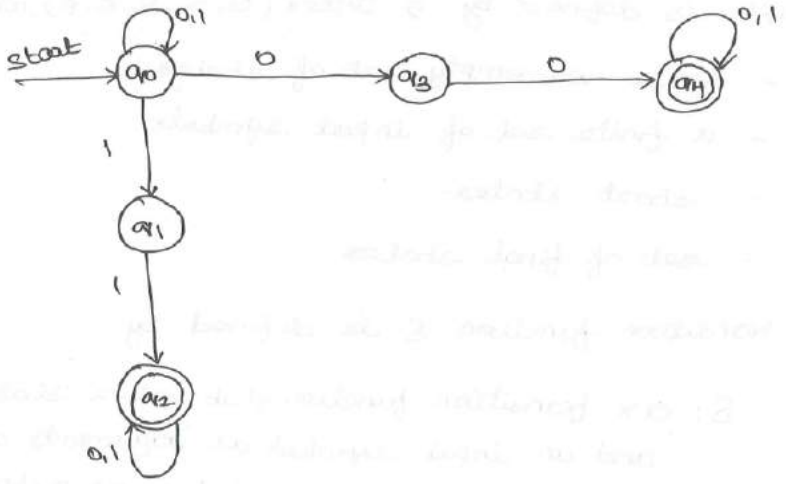
where $p - \bar{\delta}(q, x)$

Language acceptance by NFA:

the language accepted by NFA 'A' is $L(A)$.

$L(A) = \{ w \mid \delta(q, x) = p, \text{ whose } p \text{ is in } F \}$

① Consider the given NFA to check $w = 01001$ accepted or not.



the NFA, M is defined by $N(Q, \Sigma, \delta, S, F)$

$Q = \{q_0, q_1, q_2, q_3, q_4\}$

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

$S = \{q_0\}$

$F = \{q_2, q_4\}$

δ is defined by the transition table.

δ is defined by the transition table.

	States	Input symbol	
		0	1
→	q_0	$\{q_0, q_3\}$	$\{q_0, q_1\}$
	q_1	ϕ	$\{q_2\}$
*	q_2	$\{q_2\}$	$\{q_2\}$
	q_3	$\{q_4\}$	ϕ
*	q_4	$\{q_4\}$	$\{q_4\}$

To check whether the string $w = 01001$ is accepted by NFA.

$w = 01001$ be input

$\delta(q_0, 01001)$

(i) $\delta(q_0, 0) = \{q_0, q_3\}$ {from the transition table}

(ii) $\delta(q_0, 01) = \delta(\delta(q_0, 0), 1)$

$= \delta(\{q_0, q_3\}, 1)$

$= \delta(q_0, 1) \cup \delta(q_3, 1)$

$= \{q_0, q_1\} \cup \phi$

$\delta(q_0, 01) = \{q_0, q_1\}$

Similarly, we compute

(iii) $\delta(q_0, 010) = \delta(\delta(q_0, 01), 0)$

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

$= \delta(q_0, 0) \cup \delta(q_1, 0)$

$= \{q_0, q_3\} \cup \phi$

$\delta(q_0, 010) = \{q_0, q_3\}$

(iv) $\delta(q_0, 0100) = \delta(\delta(q_0, 010), 0)$

$= \delta(\{q_0, q_3\}, 0)$

$= \delta(q_0, 0) \cup \delta(q_3, 0)$

$= \{q_0, q_3\} \cup \{q_4\}$

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

(v) $\delta(q_0, 01001) = \delta(\delta(q_0, 0100), 1)$

$= \delta(\{q_0, q_3, q_4\}, 1)$

$= \delta(q_0, 1) \cup \delta(q_3, 1) \cup \delta(q_4, 1)$

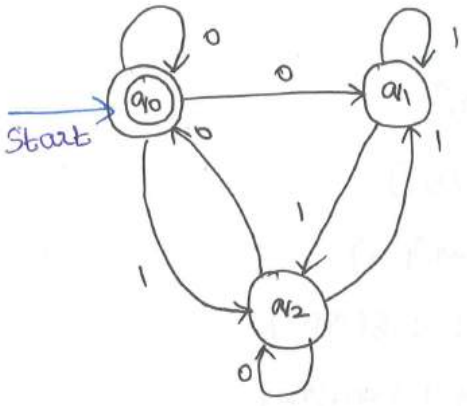
$= \{q_0, q_1\} \cup \phi \cup \{q_4\}$

$\delta(q_0, 01001) = \{q_0, q_1, q_4\}$

Since q_4 is a final state, the string is accepted. (ie) the final state q_4 is present in the solution.

$\therefore$ the string 01001 is accepted by NFA.

(2) For the NFA, shown check whether the input string 0100 is accepted (or) not.



the NFA, M is defined by $(Q, \Sigma, \delta, S, F)$

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

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

$S = \{q_0\} \subseteq Q$

$F = \{q_0\} \in Q$

the transition table is given by

States	Input symbol	
	0	1
→ q_0	$\{q_0, q_1\}$	$\{q_2\}$
q_1	ϕ	$\{q_1, q_2\}$
q_2	$\{q_0, q_2\}$	$\{q_1\}$

To check whether the string $w = 0100$ is accepted by NFA we use

$$\delta(q_0, 0100)$$

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

(ii) $\delta(q_0, 01) = \delta(\delta(q_0, 0), 1)$

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

$= \delta(q_0, 1) \cup \delta(q_1, 1)$

$= \{q_2\} \cup \{q_1, q_2\}$

$\therefore \delta(q_0, 01) = \{q_1, q_2\}$

(iii) $\delta(q_0, 010) = \delta(\delta(q_0, 01), 0)$

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

$= \delta(q_1, 0) \cup \delta(q_2, 0)$

$= \phi \cup \{q_0, q_2\}$

$\therefore \delta(q_0, 010) = \{q_0, q_2\}$

(iv) $\delta(q_0, 0100) = \delta(\delta(q_0, 010), 0)$

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

$= \delta(q_0, 0) \cup \delta(q_2, 0)$

$= \{q_0, q_1\} \cup \{q_0, q_2\}$

$\therefore \delta(q_0, 0100) = \{q_0, q_1, q_2\}$

the final state q_0 is present in the solutions.

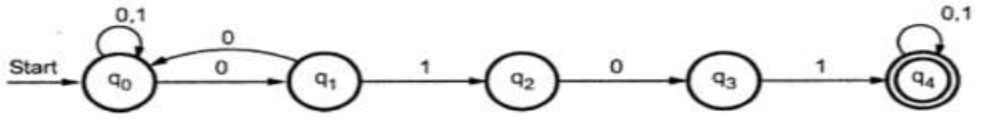
thus the string is accepted by NFA.

Example 2.24 : *Construct a NFA for the language*

L_1 = *{consisting a substring 0 1 0 1}*

$L_2 = \{a^n \cup b^n\}$.

Solution : We will consider L_1 first to design NFA. There can be any combination of 0 and 1 in the language but a substring 0101 must be present. We will get such a substring then it leads to a final state or accept state.



This is a NFA as for 0 input we have two different paths one going to q_0 and other is going to q_1.

The string 00010101 is acceptable by above given NFA and it is as shown below.

$00010101 \quad 0q_0 \vdash 010101$

$00q_0 \vdash 010101$

$000q_1 \vdash 10101$

$0001\,q_2 \vdash 0101$

$00010q_3 \vdash 101$

$000101q_4 \vdash 01$

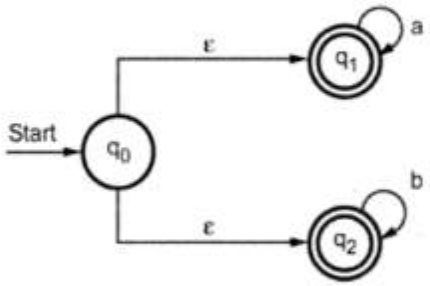
$0001010\ q_4 \vdash 1$

$00010101q_4 \vdash$

00010101 is accepted as q_4 is final state.

Now we will build a NFA for L_2. The language L_2 is a language in which there be any number of a's or any number of b's. It accepts {a, b, aa, bb, aaa, bbb, ...}.

Hence the NFA will be



The NFA shows two different states q_1 and q_2 for the input ε from q_0 state.

Here ε (pronounced as epsilon) is basically a null move i.e. a move carrying no symbol from input set Σ. But a state change occurs from one state to other.

Example 2.25 : *Design the NFA transition diagram for the transition table as given below -*

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

Here the NFA is $M = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, q_0, \{q_3\})$.

Solution : The transition diagram can be drawn by using the mapping function as given in table.

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

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

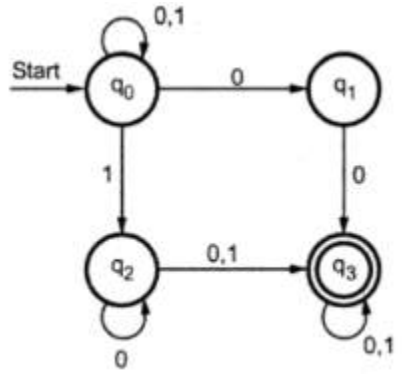
Then, $\delta(q_1, 0) = \{q_3\}$

Then, $\delta(q_2, 0) = \{q_2, q_3\}$

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

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

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

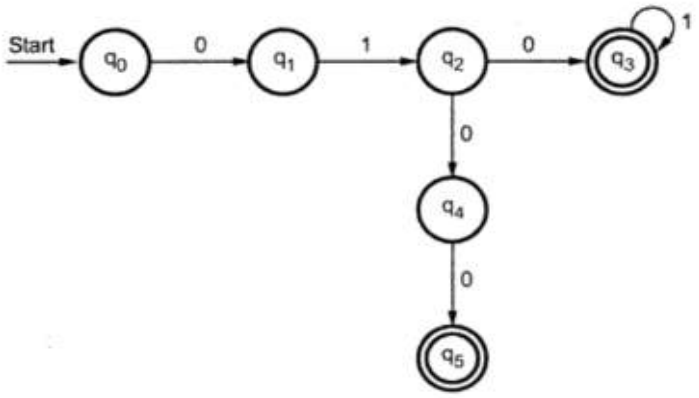


Example 2.26 : *Construct NFA for the language*

$$L = \{0101^n \cup 0100 \mid n \geq 0\}$$

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

Solution : Here in language L first three symbols are common i.e. 010. Hence the NFA can be drawn as



The states q_3 and q_5 are final states accepting 0101^n and 0100 respectively. The NFA then can be denoted by,

$$M = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \delta, q_0, \{q_3, q_5\})$$

Example 2.27 : *Construct a transition diagram for the NDFA* $M = (\{q_1, q_2, q_3\}, \delta, q_1, \{q_3\})$ *where* δ *is given by,*

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

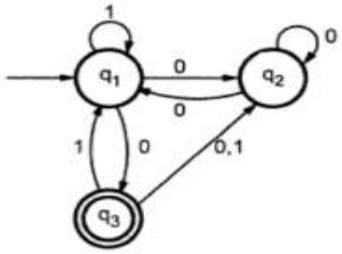
$\delta(q_2, 0) = \{q_1, q_2\}$ $\quad \delta(q_2, 1) = \phi$

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

Solution : Firstly we will design a transition table using the given mapping function δ.

States \ Input	0	1
→ q_1	$\{q_2, q_3\}$	q_1
q_2	$\{q_1, q_2\}$	ϕ
(q_3)	q_2	$\{q_1, q_2\}$

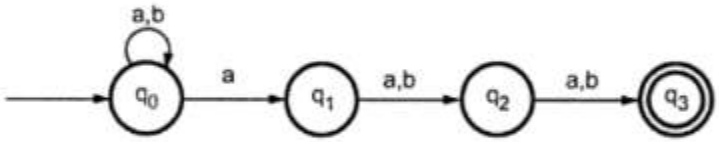
The NDFA will be



Example 2.28 : *Construct a NFA for a language L which accepts all the strings in which the third symbol from right end is always a. Over* $\Sigma = \{a, b\}$.

Solution : The strings in such a language are of the form

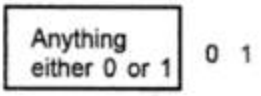
Thus we get third symbol from right end as 'a' always. The NFA then can be



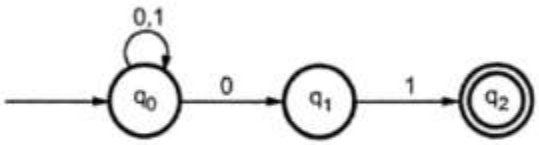
The above figure is a NFA because in state q_0 with input 'a' we can go to either state q_0 or state q_1.

Example 2.29 : *Design NFA accepting all strings ending with 01. Over* $\Sigma = \{0, 1\}$

Solution :

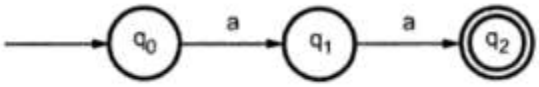


Hence, NFA would be

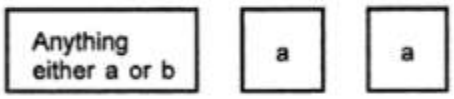


Example 2.30 : *Design NFA to accept strings with a's and b's such that the string end with 'aa'.*

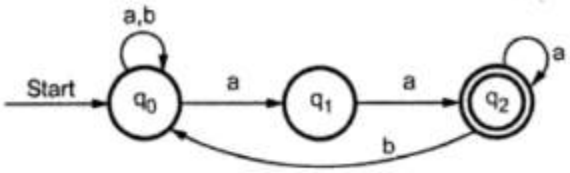
Solution : The simple FA which accepts a string with 'aa' is -



Now there can be a situation where in



Hence we can design a required NFA as



It can be denoted by,

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

We can test some strings for above drawn NFA.

Consider

$$\delta(q_0, a\,a\,a) \vdash \delta(q_0, a\,a)$$

$$\vdash \delta(q_1, a)$$

$$\vdash \delta(q_2, \varepsilon)$$

i.e. we reach to accept state.

$$\delta(q_0, a\,a\,a) \vdash \delta(q_0, a\,a)$$

$$\vdash \delta(q_0, a)$$

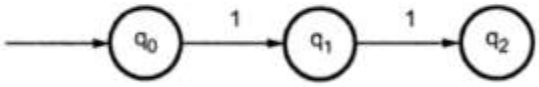
$$\vdash \delta(q_1, \varepsilon)$$

i.e. we are not in final state.

Thus there are two possibilities by which we move with string 'aaa' in above given NFA.

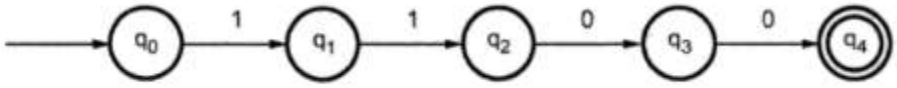
Example 2.31 : *Construct a NFA in which double '1' is followed by double '0'. Over* $\Sigma = \{0, 1\}$.

Solution : The FA with double 1 is as drawn below.



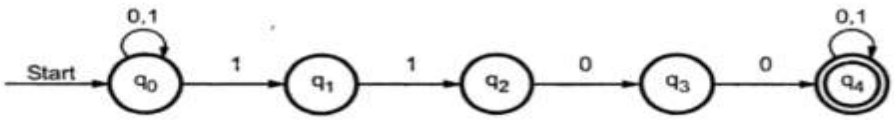
It should be immediately followed by double 0.

Then,



Now before double 1 there can be any string of 0 and 1. Similarly after double 0 there can be any string of 0 and 1.

Hence, the NFA becomes



The transition table for above transition diagram can be as given below -

States \ Input	1	0
q_0	$\{q_0, q_1\}$	q_0
q_1	q_2	-
q_2	-	q_3
q_3	-	q_4
q_4	q_4	q_4

Consider string 11100,

$\delta(q_0, 111000) \vdash \delta(q_0, 1100)$

$\vdash \delta(q_0, 100)$

$\vdash \delta(q_1, 00)$

Got stuck! As there is no path from q_1 for input symbol 0. We can process string 11100 in another way.

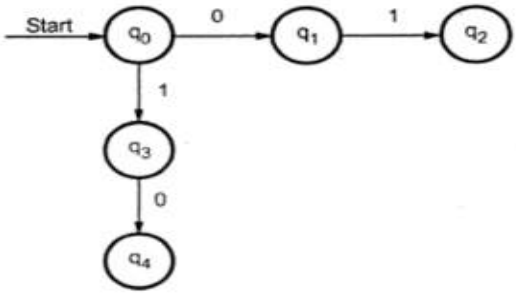
$$\begin{aligned}\delta(q_0, 11100) &\vdash \delta(q_0, 1100)\\ &\vdash \delta(q_1, 100)\\ &\vdash \delta(q_2, 00)\\ &\vdash \delta(q_3, 0)\\ &\vdash \delta(q_4, \varepsilon)\end{aligned}$$

As q_4 is accept state we get, the complete string scanned and reaching to final state.

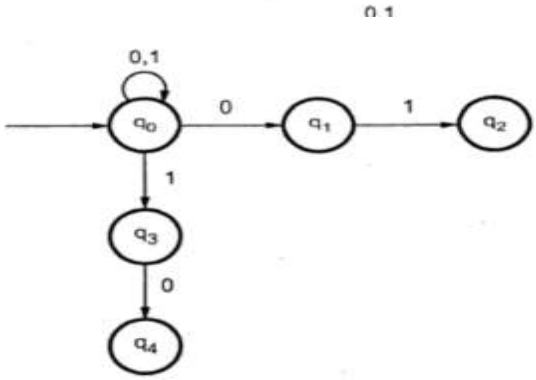
Example 2.32 : *Design NFA which accepts the string containing either '01' or '10'. Over* $\Sigma = \{0, 1\}$.

Solution : The NFA can be drawn in stages as follows.

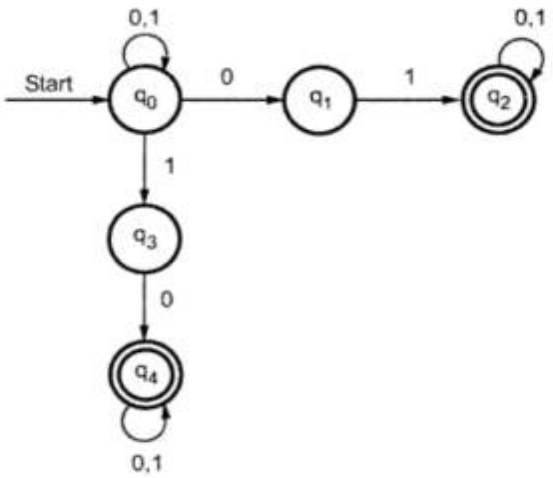
Consider a FA with '01' or '10'.



Now before '01' or '10' there can be any number or 0 and 1. Hence in next stage the FA can be



Now after '01' or '10' there can be any combination of 0 and 1. Hence the NFA in its final stage will be -



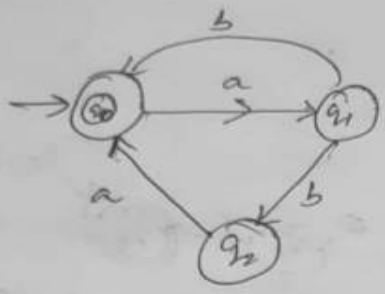
We can design a transition table as

States \ Input	0	1
q_0	$\{q_0, q_1\}$	$\{q_0, q_3\}$
→ q_1	–	q_2
(*) q_2	q_2	q_2
q_3	q_4	-
(*) q_4	q_4	q_4

δ { } Input Σ

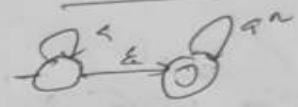
This section focuses on the fundamental aspects of automata theory. We have learnt basic concepts of DFA and NFA. Now let us discuss interconversion between them.

(13) $L = (ab \cup aba)^*$ $\Sigma = \{a, b\}$

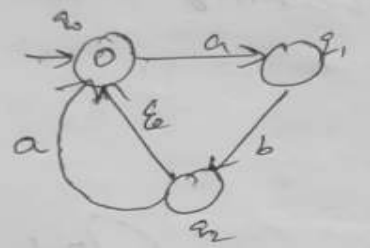


Note

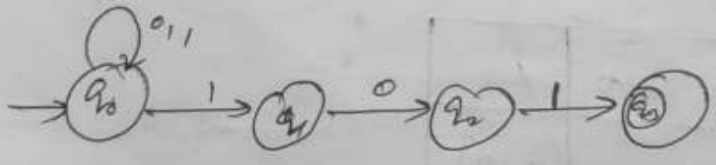
ε-transition



$a^n b^m \quad n>0 \quad b>0$



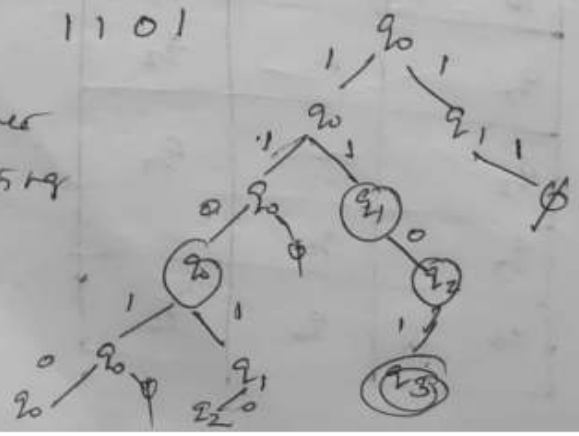
(14) L = NFA that accepts all strings ending in '101' $\Sigma = \{0, 1\}$



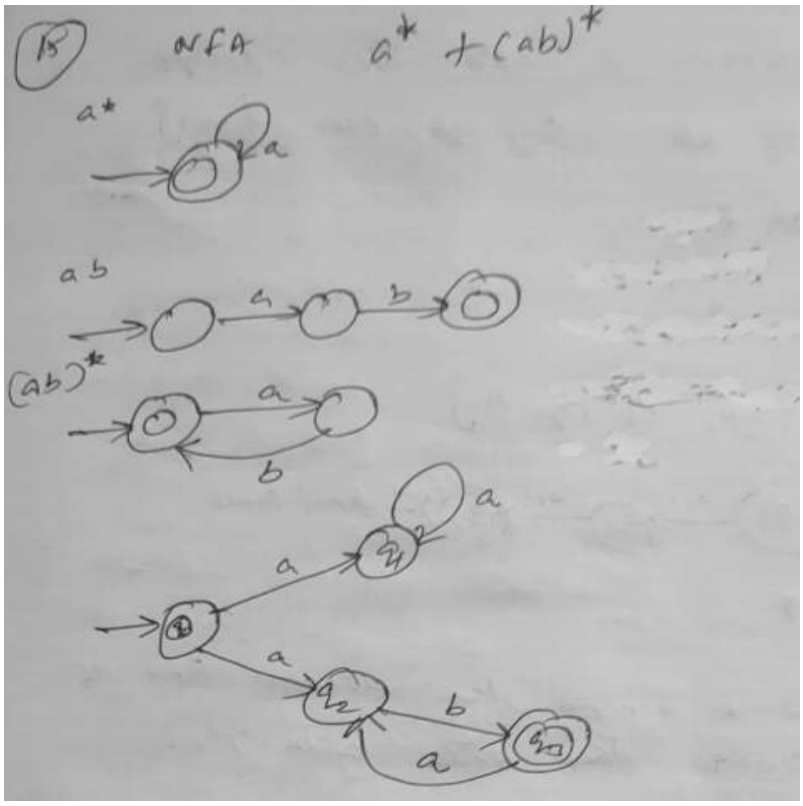
I/P string 1101

q_3 - final state

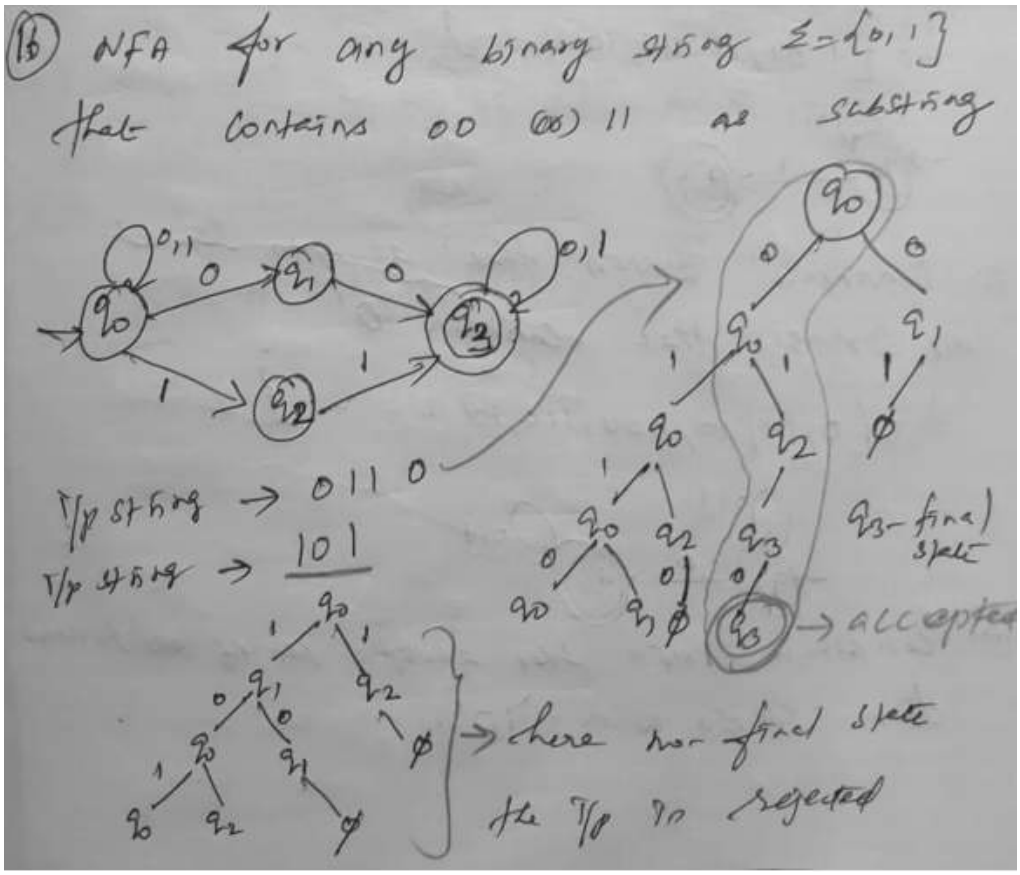
- the I/P string accepted



(a) NFA $a^* + (ab)^*$



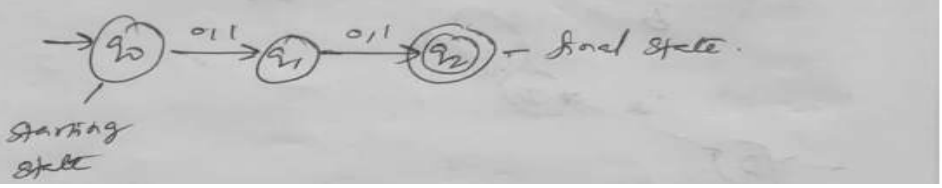
(b) NFA for any binary string $\Sigma = \{0, 1\}$ that contains 00 (or) 11 as substring



(17) Construct a NFA that accepts set of all strings over {0,1} of length 2.

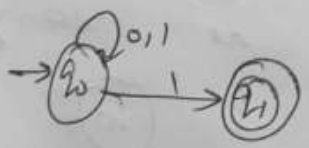
$\Sigma = \{0,1\}$

$L = \{00, 01, 10, 11\}$



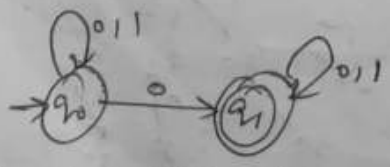
(18) Construct a NFA that accepts set of all strings that ends with '1'.

$L = \{1, 01, 11, 011, 101 - - -\}$



(19) Construct a NFA that accepts set of all strings that contain '0'

$L = \{0, 01, 10, 000, 110, 101 - - - -\}$

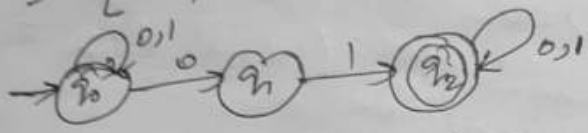


(20) Construct a NFA that accepts set of all strings that starts with '10'.

$L = \{10, 100, 101, 1011 \ldots\}$

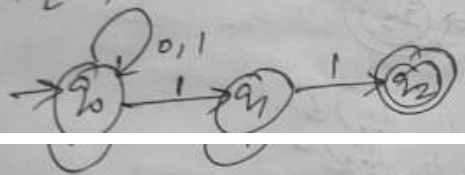
(21) Construct a NFA that accepts sets of all strings that contain '01'

$L = \{01, 101, 001, 011, 010 \ldots\}$



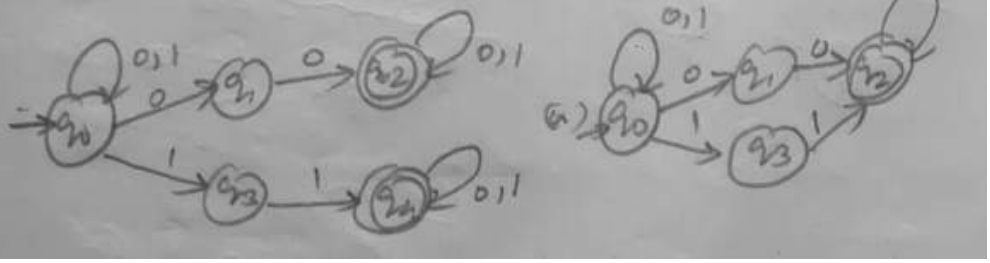
(22) Construct a NFA that accepts sets of all strings that ends with '11'.

$L = \{11, 011, 111, \ldots\}$



(23) Construct a NFA that accepts sets of all strings over $\{0,1\}$ that have atleast two consecutive 0's (or) 1's.

$L = \{00, 11, 001, 000, 111, 110 \ldots\}$



DFA vs NFA

NFA refers to Nondeterministic Finite Automaton. DFA refers to Deterministic Finite Automaton. DFA can be best described and understood as one machine. NFA is like multiple small machines that are performing computational activities at the same time. All DFAs are derived from NFAs. The main difference between **DFA and NFA**, the two classes handling the transition functions of finite automata/ finite automaton theory, impact their behaviour in many ways.

Basis of Difference	DFA	NFA
Reaction to symbols	For each symbolic representation of the alphabet, only a singular state transition can be attained in DFA.	No specifications are needed from the user with respect to how certain symbols impact the NFA.
Empty string transition	DFA is not capable of using an Empty String transition.	NFA can use an empty String transition.
Structure	DFA can be best described and understood as one machine.	NFA is like multiple small machines that are performing computational activities at the same time.
Rejection of string	DFA rejects the string in case it terminates in a state that is different from the accepting state.	NFA rejects the string in the event of all branches dying or refusing the string.
Backtracking	It is possible to use backtracking in DFA.	It is not possible to use backtracking at all times in the case of NFA.
Ease of construction	Given its complex nature, it is tougher to construct DFA.	NFA is more easily constructed in comparison to DFA.
Supremacy	All DFAs are derived from NFAs.	All NFAs are not DFAs.
Transition functions	The number related to the next state is one.	The number related to the next state/ states is either zero or one.
Complexities of time	The total time required for running any input string in DFA is less than what it is in NFA.	The total time required for running any input string in NFA is larger than that in comparison to DFA.
Full form	The full form of DFA is Deterministic Finite Automata.	The full form of NFA is Nondeterministic Finite Automata (NFA).

Basis of Difference	DFA	NFA
Space requirement	More space allocation needed.	Less space needed.
The setting of the next possible set	The next possible state is clearly set in DFA.	In NFA, every single pair of input symbols and states may contain many possible next states.

TWO MARKS

1.

What is a formal language?

Language is a set of valid strings from some alphabet. The set may be empty, finite or infinite. L(M) is the language defined by machine M and L(G) is the language defined by context free grammar. The two notations for specifying formal languages are:

- Grammar or regular expression (Generative approach)
- Automaton (Recognition approach)

2.

What is Automata Theory?

- Automata theory is the study of abstract machines (or abstract 'mathematical' machines or systems) and the computational problems that can be solved using these machines. These abstract machines are called automata.
- Automata which means that something is doing something by itself. Automata theory is also closely related to formal language theory, as the automata are often classified by the class of formal languages they are able to recognize. An automaton can be a finite representation of a formal language that may be an infinite set.

3.

Why are switching circuits called as finite state systems?

A switching circuit consists of a finite number of gates, each of which can be in any one of the two conditions 0 or 1. Although the voltages assume infinite set of values, the electronic circuitry is designed so that the voltages corresponding to 0 or 1 are stable and all others adjust to these values. Thus control unit of a computer is a finite state system.

4.

What are the Operations on strings?

- Length of string
- Empty or null string
- Concatenation of string
- Reverse of a string
- Powers of an alphabet
- Substring
- Reversal
- Kleene Closure

5.

Define alphabets in automata?

An alphabet is a finite, non-empty set no of symbols. It is denoted by Σ.

Example:

- ✓ Σ = {*a, b*}, an alphabet of 2 symbols *a* and *b*.
- ✓ Σ = {*0, 1, 2*} an alphabet of 3 symbols *0, 1*, and *2*.

6.

Define strings with examples?

A String or Word is a finite sequence of symbols from Σ. The group of characters also referred as a String.

Example:

- ✓ *a, b,* and *c* are symbols and *abcb* is a string.
- ✓ If Σ = {*a,b*} then *abab,aabbb,aaabb,*..... are all strings over the alphabet Σ ={*a,b*}.
- ✓ *0, 1, 11, 00,* and *01101* are strings over {0, 1}.

7.

What is substring with example?

Let *x* and *y* be strings over an alphabet Σ.

The string *x* is a substring of *y* if there exist strings *w* and *z* over Σ such that *y* = *w x z*.

- ε is a substring of every string.
- For every string *x*, *x* is a substring of *x* itself.

Example:

- ε, comput and computation are substrings of computation.

8.

Explain the following a. Length of string b. Empty or null string?

a. Length of string

Let w is a string. Length of the string is |w|=> number of symbols composing the string.

Example:

1) If w=abcd then |w|=4

2) If x=01010110 the |x|=8

3) If y= 0101 the |y|=4

4) If |Є| = 0

b. Empty or null string

The string consisting of zero symbols. It is denoted by Є.

Example:

{Є}=0.

9.

What is meant by Powers of an alphabet?

Let Σ be the alphabet

Σ * is the set of all strings over alphabet Σ.

Σ^m denotes the set of all strings over alphabet Σ of length m.

Example:

Σ = {0, 1}

Σ^0= {Є} empty string of length 0.

Σ^1= {0, 1} is the set of all strings of length one over Σ = {0, 1}

Σ^2= {00, 01, 10, 11} is the set of all strings of length two over Σ = {0, 1}.

10.

Define Kleen Closure?

Let Σ be the alphabet

" Kleen Closure"-> Σ* denotes the set of all strings (Σ) over the alphabet

Example:

If Σ ={a} the Σ*={ Є,a,aa,aaa,.....}

If Σ ={0,1} the Σ*={ Є,0,1,00,10,10,.....}

Σ^0={ Є}

Σ^1={a}

Σ^2={aa}

Therefore $\Sigma^*= \Sigma^0 U \Sigma^1 U \Sigma^2$,.....

11.

Define a language.

An alphabet is a finite set of symbols. A language is a set of strings of symbols from someone alphabet.

e.g. If Σ = {0,1},then Σ*={ Є,0,1,01,11,10,111,.....}

12.

List out the applications of automata theory?

- String Processing
- Analysis of algorithms
- Complexity Theory
- Cryptography
- Compiler Design
- Interpreters
- Parser
- Circuit design
- Robotics
- Artificial Intelligence
- Knowledge Engineering
- Natural language Processing

13.

What are Finite Automata? Give an example.

- A finite automaton (FA) consists of a finite set of states and a set of transitions from state to state that occur on input symbols chosen from an alphabet Σ.
- Each input symbol there is exactly one transition out of each state (possibly back to the state itself).
- One State, usually q_0, is the initial state, in which the automaton starts.
- Some states are designated as final or accepting states.

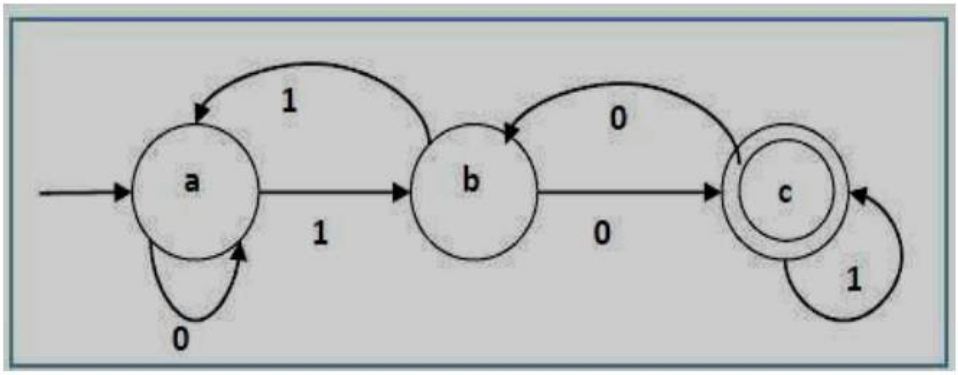
Example:

Let a deterministic finite automaton be

1. Q = {a, b, c},
2. ∑ = {0, 1},
3. q_0={a},
4. F={c}, and
5. Transition function δ as shown by the following table

Present State	Next State for Input 0	Next State for Input 1
a	a	b
b	c	a
c	b	c

Its graphical representation would be as follows –



14.

How does an FA work?

- At the beginning,
  - ✓ an FA is in the start state (initial state)
  - ✓ its tape head points at the first cell
- For each move, FA
  - ✓ reads the symbol under its tape head
  - ✓ changes its state (according to the transition function) to the next state determined by the symbol read from the tape and its current state
  - ✓ move its tape head to the right one cell

15.

. Give the purpose of transition diagram?

A directed graph, called a transition diagram, is associated with an FA as follows:

- The vertices of the graph correspond to the states of the FA.
- If there is a transition from state q to state p on input a, then there is an arc labeled a from state q to state p in the transition diagram.
- The FA accepts a string x if the sequence of transitions corresponding to the symbols of x leads from the start state to an accepting state.

16.

What is Transition Table?

The transition table is the tabular representation of the transition function "delta (∂)" with the rows denoting states and columns denoting input symbol.

17.

What are the applications of Finite Automata (FA)?

- Software for designing and checking the behavior of digital circuits.
- Lexical analyzer of a typical compiler.
- Software for scanning large bodies of text (e.g., web pages) for pattern finding.
- Software for verifying systems of all types that have a finite number of states (e.g., stock market transaction, communication/network protocol).

18.

Write the DFA Specifications?

Deterministic Finite Automata (DFA) is generally specified by a 5 tuples **$(Q, \Sigma, \delta, q_0, F)$,**

where

1. **Q** is a finite set of states
2. **Σ** is a finite input alphabet
3. $\mathbf{q_0}$ in Q is the start state or initial state
4. **F⊆Q** is the set of final states
5. **δ** is the transition function mapping **δ: QX Σ→Q**.(i.e.) δ(q,a) is a state for each state *q* and input symbol *a.*

19.

Explain working of DFA?

1. **Initialisation**
   - Reader (read head) should be over the leftmost symbol.
   - Finite Control is in start state.
2. **Single Step**
   - Reader moves to the next symbol to the right.
   - Control enters a new state that (deterministically) depends on current state and current symbol. There may be no desired next state, Machine will stop.
   - The machine repeats this action.
3. **No current symbol**
   - All Symbols have been read.
   - If control is in final state, the input string is accepted.
   - Otherwise, the input string is not accepted.

20.

What are the Applications of DFA?

- Compiler Design
- Design of Lexical Analyser
- Text Editor
- Pattern Matching
- File searching Program
- Text Processing

21.

. Define NFA.

Non-Deterministic Finite Automata (NFA) is generally specified by a 5 tuples **(Q, Σ, δ, q_0, F)**

Where,

1. **Q** is a finite set of states
2. **Σ** is a finite input alphabet
3. **q_0** in Q is the start state or initial state
4. **F⊆Q** is the set of final states
5. **δ** is the transition function mapping δ → Q x Σ* to 2^Q.

On receiving same inputs it goes to many states

- the machine can move without consuming any symbols and sometimes there are possible moves and sometimes there are move then one possible moves.
- the state is only partially determined by the current state and i/p symbol.

22.

What are the difference between NFA and DFA?

DFA	NFA
i. For each and every state and for each and every input symbol there exists at most one transition. **ii.** The transition mapping for DFA is **Q×Σ→Q.** **iii.** The language accepted by DFA is denoted as L(M). **iv.** Epsilon transition is not possible.	**i.** For each and every state and for each and every input symbol there exists more than one transition. **ii.** The transition mapping for NFA is **$Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$.** **iii.** The language accepted by NFA is denoted by L(M'). **iv.** Epsilon transition is possible.

23.

What are Transition Function and give their properties?

Changing from one state to another state on applying the inputs

Properties of Transition function:

1. $\delta(q, \varepsilon)=q$

   This means the state of the system can be changed only by an input symbol else remains in original state.

2. For all strings w and input symbol a $\delta(q,aw)= \delta(\delta(q,a),w)$

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

3. The transition function δ can be extended to δ or δ that operates on states and strings.

   Basis: $\delta(q, \varepsilon)=q$

   Induction: $\delta(q,xa)= \delta(\delta(q,x),a)$

UNIT-II

FINITE AUTOMATA

NFA with ε-transitions - Significance, acceptance of languages. Conversions and Equivalence : Equivalence between NFA with and without ε transitions, NFA to DFA conversion, minimization of FSM, equivalence between two FSM's, Finite Automata with output-Moore and Melay machines.

2.1 Introduction

In earlier chapter we have learnt the concept of finite automata. We have also discussed the two types of finite automata: DFA and NFA. These two types of automata are equivalent. That means we can convert any NFA to DFA. These automata are designed to accept some language. Basically there are two types of languages: **regular languages** and **non regular languages**. The finite automata is a mathematical model which always accepts regular languages.

In this chapter, we will discuss the equivalence between NFA and DFA. And then we will discuss the logical unit of representing regular language i.e. **Regular expressions**. We will also discuss the relationship between regular expression and finite state machines.

2.2 NFA with ε - Transitions

The ε-transitions in NFA are given in order to move from one state to another without having any symbol from input set Σ.

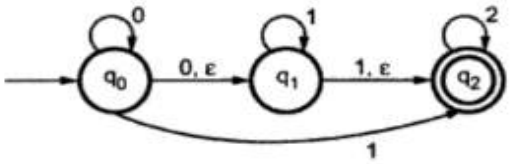


Fig. 2.1 NFA with ε - transitions

Consider the NFA with ε as :

In this NFA with ε, q_0 is a start state with input 0 one can be either in state q_0 or in state q_1. If we get at the start a symbol 1 then with ε - move we can change state from q_0 to q_1 and then with input we can be in state q_1. On the other hand, from start state q_0, with input 1 we can reach to state q_2. Thus it is not definite that on input 1 whether we will be in state q_1 or q_2. Hence it is called nondeterministic finite automata (NFA) and since there are some ε moves by which we can simply change the states from one state to other. Hence it is called NFA with ε.

2.2.1 Significance of NFA with ε

As construction of DFA is very difficult for certain input set, we construct NFA. This NFA then can be converted to DFA. ε is an empty string. The ε - transitions are used simply to change one state to other. Sometimes to reach to final state we do not get proper state from start state. In such a case we simply want to reach to certain state which leads to final state. Such a transition to that specific state should be without any input symbol. Hence we require some ε - moves by which a proper state can be obtained for reaching to final state. Thus ε - moves play an important role in NFA.

2.2.2 Acceptance of Language

The language L accepted by NFA with ε, denoted by $M = (Q, \Sigma, \delta, q_0, F)$ can be defined as :

Let, $M = (Q, \Sigma, \delta, q_0, F)$ be a NFA with ε

Where

Q is a finite set of states.

Σ is input set.

δ is a transition or a mapping function for transitions from $Q \times \{\Sigma \cup \varepsilon\}$ to 2^Q.

q_0 is a start state.

F is a set of final states such that $F \in Q$.

The string w in L accepted by NFA can be represented as

$L(M) = \{w \mid w \in \Sigma^*$ and δ transition for w from q_0 reaches to F$\}$

➠ **Example 2.1 :** *Construct NFA with ε which accepts a language consisting the strings of any number of a's followed by any number of b's. Followed by any number of c's.*

Solution : Here any number of a's or b's or c's means zero or more in number. That means there can be zero or more a's followed by zero or more b's followed by zero or more c's. Hence NFA with ε can be -

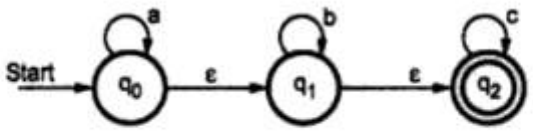


Fig. 2.2

Normally ε's are not shown in the input string. The transition table can be -

δ	Σ			
Input / State	a	b	c	ε
q_0	q_0	ϕ	ϕ	q_1
q_1	ϕ	q_1	ϕ	q_2
q_2	ϕ	ϕ	q_2	ϕ

We can parse the string aabbcc as follows -

$\delta(q_0, aabbcc) \vdash \delta(q_0, abbcc)$
$\vdash \delta(q_0, bbcc)$
$\vdash \delta(q_0, \varepsilon bbcc)$
$\vdash \delta(q_1, bbcc)$
$\vdash \delta(q_1, bcc)$
$\vdash \delta(q_1, cc)$
$\vdash \delta(q_1, \varepsilon cc)$
$\vdash \delta(q_2, cc)$
$\vdash \delta(q_2, c)$
$\vdash \delta(q_2, \varepsilon)$

Thus we reach to accept state, after scanning the complete input string.

Definition of ε - closure

The ε - closure (p) is a set of all states which are reachable from state p on ε - transitions such that :

i) ε - closure (p) = p where $p \in Q$.

ii) If there exists ε - closure (p) = {q} and $\delta(q, \varepsilon) = r$ then
ε - closure (p) = {q, r}

➡ **Example 2.2 :** *Find ε - closure for the following NFA with ε*

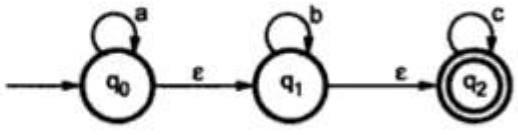


Fig. 2.3

Solution :

ε - closure $(q_0) = \{q_0, q_1, q_2\}$ means self state + ε - reachable states.

ε - closure $(q_1) = \{q_1, q_2\}$ means q_1 is a self state and q_2 is a state obtained from q_1 with ε input.

ε - closure $(q_2) = \{q_2\}$

2.3 Conversions and Equivalence

The NFA with ε can be converted to NFA without ε. And this NFA without ε can be converted to DFA. In this section we will discuss these conversions and will find them equivalent to each other.

Fig. 2.4 Moves to DFA

2.3.1 Conversion from NFA with ε to NFA without ε

In this method we try to remove all the ε transitions from given NFA. The method will be

1. Find out all the ε transitions from each state from Q. That will be called as ε-closure $\{q_i\}$ where $q_i \in Q$.
2. Then δ' transitions can be obtained. The δ' transitions means an ε-closure on δ moves.
3. Step-2 is repeated for each input symbol and for each state of given NFA.
4. Using the resultant states the transition table for equivalent NFA without ε can be built.

Theorem : If L is accepted by NFA with ε transitions, then there exist L which is accepted by NFA without ε transitions.

Theorem : If L is accepted by NFA with ε transitions, then there exist L which is accepted by NFA without ε transitions.

Proof :

Let, $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA with ε transitions.

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

$$F' = \begin{cases} F \cup \{q_0\} & \text{if } \varepsilon \text{ - closure contains state off} \\ F & \text{otherwise} \end{cases}$$

M' is a NFA without ε moves. The δ' function can be denoted by δ^* with some input. For example, $\delta'(q, a) = \delta^*(q, a)$ for some q in Q and a from Σ. We will apply the method of induction with input x. The x will not be ε because

$\delta'(q_0, \varepsilon) = \{q_0\}$

$\delta^*(q_0, \varepsilon) = \varepsilon$-closure (q_0). Therefore we will assume length of string to be 1.

Basis : $|x| = 1$. Then x is a symbol a.

$$\delta'(q_0, a) = \delta^*(q_0, a)$$

Induction : $|x| > 1$ Let $x = wa$

$$\delta'(q_0, wa) = \delta'(\delta'(q_0, w), a)$$

By inductive hypothesis,

$$\delta'(q_0, w) = \delta^*(q_0, w) = p$$

Now we will show that $\delta'(p, a) = \delta(q_0, wa)$

But $$\delta'(p, a) = \bigcup_{q \text{ in } p} \delta'(q, a) = \bigcup_{q \text{ in } p} \delta^*(q, a)$$

As $$p = \delta^*(q_0, w)$$

We have, $$\bigcup_{q \text{ in } p} \delta^*(q, a) = \delta^*(q_0, wa)$$

Thus by definition δ^*

$$\delta'(q_0, wa) = \delta^*(q_0, wa)$$

Rule for conversion

$$\delta(q, a) = \varepsilon \text{ - closure}\left(\delta\left(\hat{\delta}(q, \varepsilon), a\right)\right)$$

where $\hat{\delta}(q, \varepsilon) = \varepsilon$ - closure(q)

Before solving some examples based on conversion of NFA with ε to NFA without ε above rule should be remembered.

Example 2.3 :*Convert the given NFA with ε to NFA without ε.*

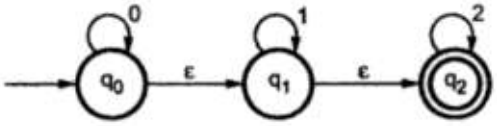


Fig. 2.5

Solution : We will first obtain ε - closure of each state i.e. we will find out ε - reachable states from current state.

Hence

$$\varepsilon \text{ - closure } (q_0) = \{q_0, q_1, q_2\}$$

ε - closure $(q_1) = \{q_1, q_2\}$

ε - closure $(q_2) = \{q_2\}$

As ε - closure (q_0) means with null input (no input symbol) we can reach to q_0 , q_1 or q_2. In a similar manner for q_1 and q_2 ε - closures are obtained. Now we will obtain δ transitions for each state on each input symbol.

$$\delta(q_0, 0) = \varepsilon\text{ - closure}\left(\delta\left(\hat{\delta}(q_0, \varepsilon), 0\right)\right)$$

$$= \varepsilon\text{ - closure}\left(\delta\left(\varepsilon\text{ - closure}(q_0), 0\right)\right)$$

$$= \varepsilon\text{ - closure}\left(\delta(q_0, q_1, q_2), 0\right)$$

$$= \varepsilon\text{ - closure}\left(\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0)\right)$$

$$= \varepsilon\text{ - closure}\left(q_0 \cup \phi \cup \phi\right)$$

$$= \varepsilon\text{ - closure}(q_0) = \{q_0, q_1, q_2\}$$

$$\delta(q_0, 1) = \varepsilon\text{ - closure}\left(\delta\left(\hat{\delta}(q_0, \varepsilon), 1\right)\right)$$

$$= \varepsilon\text{ - closure}\left(\delta(q_0, q_1, q_2), 1\right)$$

$$= \varepsilon\text{ - closure}\left(\delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_2, 1)\right)$$

$$= \varepsilon\text{ - closure}\left(\phi \cup q_1 \cup \phi\right)$$

$$= \varepsilon\text{ - closure}(q_1)$$

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

$$\delta(q_1, 0) = \varepsilon\text{ - closure}\left(\delta\left(\hat{\delta}(q_1, \varepsilon), 0\right)\right)$$

$$= \varepsilon\text{ - closure}\left(\delta\left(\varepsilon\text{ - closure}(q_1), 0\right)\right)$$

$$= \varepsilon\text{ - closure}\left(\delta(q_1, q_2), 0\right)$$

$$= \varepsilon\text{ - closure}\left(\delta(q_1, 0) \cup \delta(q_2, 0)\right)$$

$$= \varepsilon\text{ - closure}\left(\phi \cup \phi\right)$$

$$= \varepsilon\text{ - closure}(\phi)$$

$$= \phi$$

$$\delta(q_1, 1) = \varepsilon\text{ - closure}\left(\delta\left(\hat{\delta}(q_1, \varepsilon), 1\right)\right)$$

$$= \varepsilon\text{ - closure}\left(\delta\left(\varepsilon\text{ - closure}(q_1), 1\right)\right)$$

$$= \varepsilon\text{ - closure}\left(\delta(q_1, q_2), 1\right)$$

$= \varepsilon\text{ - closure } (\delta(q_1, 1) \cup \delta(q_2, 1))$

$= \varepsilon\text{ - closure } (q_1 \cup \phi)$

$= \varepsilon\text{ - closure } (q_1)$

$= \{q_1, q_2\}$

$\delta'(q_2, 0) = \varepsilon\text{ - closure } \left(\delta\left(\hat{\delta}(q_2, \varepsilon), 0\right)\right)$

$= \varepsilon\text{ - closure } \left(\delta\left(\varepsilon\text{ - closure}(q_2), 0\right)\right)$

$= \varepsilon\text{ - closure } (\delta(q_2, 0))$

$= \varepsilon\text{ - closure } (\phi)$

$= \phi$

$\delta'(q_2, 1) = \varepsilon\text{ - closure } \left(\delta\left(\hat{\delta}(q_2, \varepsilon), 1\right)\right)$

$= \varepsilon\text{ - closure } \left(\delta\ \left(\varepsilon\text{ - closure}(q_2), 1\right)\right)$

$= \varepsilon\text{ - closure } (\delta(q_2, 1))$

$= \varepsilon\text{ - closure } (\phi)$

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

$\delta'(q_0, 2) = \varepsilon\text{ - closure } \left(\delta\left(\hat{\delta}(q_0, \varepsilon), 2\right)\right)$

$= \varepsilon\text{ - closure } \left(\delta\left(\varepsilon\text{ - closure}(q_0), 2\right)\right)$

$= \varepsilon\text{ - closure } (\delta(q_0, q_1, q_2), 2)$

$= \varepsilon\text{ - closure } (\delta(q_0, 2) \cup \delta(q_1, 2) \cup \delta(q_2, 2))$

$= \varepsilon\text{ - closure } (\phi \cup \phi \cup q_2)$

$= \varepsilon\text{ - closure } (q_2)$

$= \{q_2\}$

$\delta'(q_1, 2) = \varepsilon\text{ - closure } \left(\delta\left(\hat{\delta}(q_1, \varepsilon), 2\right)\right)$

$= \varepsilon\text{ - closure } \left(\delta\left(\varepsilon\text{ - closure } (q_1), 2\right)\right)$

$= \varepsilon\text{ - closure } (\delta(q_1, q_2), 2)$

$= \varepsilon\text{ - closure } (\delta(q_1, 2) \cup \delta(q_2, 2))$

$$= \varepsilon\text{ - closure } (\phi \cup q_2)$$

$$= \{q_2\}$$

$$\delta(q_2, 2) = \varepsilon\text{ - closure } \left(\delta\left(\hat{\delta}(q_2, \varepsilon), 2\right)\right)$$

$$= \varepsilon\text{ - closure } \left(\delta(\varepsilon\text{ - closure}(q_2), 2)\right)$$

$$= \varepsilon\text{ - closure } (\delta(q_2, 2))$$

$$= \varepsilon\text{ - closure } (q_2)$$

$$= \{q_2\}$$

Now we will summarize all the computed δ transitions -

$\delta(q_0, 0) = \{q_0, q_1, q_2\}$ $\delta(q_0, 1) = \{q_1, q_2\}$ $\delta(q_0, 2) = \{q_2\}$

$\delta(q_1, 0) = \phi$ $\delta(q_1, 1) = \{q_1, q_2\}$ $\delta(q_1, 2) = \{q_2\}$

$\delta(q_2, 0) = \phi$ $\delta(q_2, 1) = \phi$ $\delta(q_2, 2) = \{q_2\}$

From this we can write the transition table as -

State \ Input	0	1	2
q_0	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
q_1	ϕ	$\{q_1, q_2\}$	$\{q_2\}$
q_2	ϕ	ϕ	$\{q_2\}$

The NFA will be -

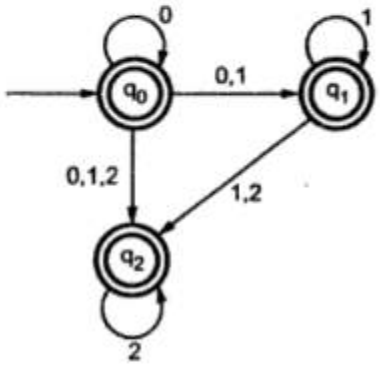


Fig. 2.6

Here q_0, q_1 and q_2 is a final state because ε - closure (q_0), ε - closure (q_1) and ε - closure (q_2) contains final state q_2.

Example 2.4 : *Convert the following NFA with ε to NFA without ε.*

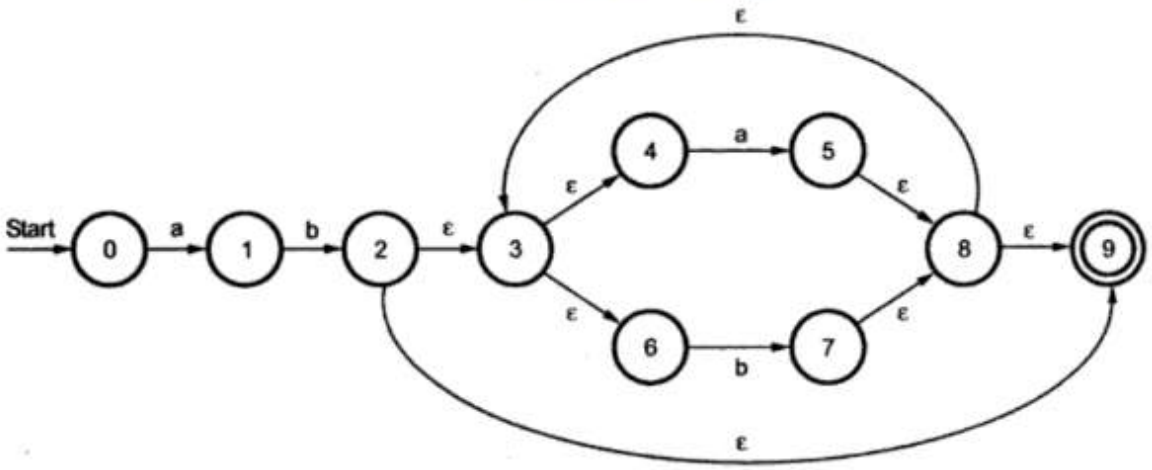


Fig. 2.7

Solution : We will first obtain ε - closure of every state. The ε - closure is basically an ε - transition from one state to other. Hence

ε - closure (0) = {0}

ε - closure (1) = {1}

ε - closure (2) = {2, 3, 4, 6, 9}

ε - closure (3) = {3, 4, 6}

ε - closure (4) = {4}

ε - closure (5) = {5, 8, 3, 4, 6, 9}

= {3, 4, 5, 6, 8, 9} sorted it!

ε - closure (6) = {6}

ε - closure (7) = {7, 8, 3, 4, 6, 9}

= {3, 4, 6, 7, 8, 9}

ε - closure (8) = {8, 3, 4, 6, 9}

= {3, 4, 6, 8, 9}

ε - closure (9) = {9}

Now we will obtain δ' transitions for each state and for each input symbol.

$$\delta'(0, a) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(0, \varepsilon), a\right)\right)$$

$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(0), a)\right)$

$= \varepsilon\text{-closure}\,(\delta(0, a))$

$= \varepsilon\text{-closure}(1)$

$= \{1\}$

$\delta'(0, b) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(0, \varepsilon), b\right)\right)$

$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(0), b)\right)$

$= \varepsilon\text{-closure}\,(\delta(0, b))$

$= \varepsilon\text{-closure}(\phi)$

$= \phi$

$\delta'(1, a) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(1, \varepsilon), a\right)\right)$

$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(1), a)\right)$

$= \varepsilon\text{-closure}\,(\delta(1, a))$

$= \varepsilon\text{-closure}(\phi)$

$= \phi$

$\delta'(1, b) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(1, \varepsilon), b\right)\right)$

$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(1), b)\right)$

$= \varepsilon\text{-closure}\,(\delta(1, b))$

$= \varepsilon\text{-closure}(2)$

$= \{2, 3, 4, 6, 9\}$

$\delta'(2, a) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(2, \varepsilon), a\right)\right)$

$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(2), a)\right)$

$= \varepsilon\text{-closure}\,(\delta(2, 3, 4, 6, 9), a)$

$= \varepsilon\text{-closure}(\delta(2, a) \cup \delta(3, a) \cup \delta(4, a) \cup \delta(6, a) \cup \delta(9, a))$

$= \varepsilon\text{-closure}(\phi \cup \phi \cup 5 \cup \phi \cup \phi)$

$= \varepsilon\text{-closure}(5)$

$= \{3, 4, 5, 6, 8, 9\}$

$$\delta'(2, b) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(2, \varepsilon), b\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(2), b)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(2, 3, 4, 6, 9), b\right)$$

$$= \varepsilon\text{-closure}\left(\delta(2, b) \cup \delta(3, b) \cup \delta(4, b) \cup \delta(6, b) \cup \delta(9, b)\right)$$

$$= \varepsilon\text{-closure}(\phi \cup \phi \cup \phi \cup 7 \cup \phi)$$

$$= \varepsilon\text{-closure}(7)$$

$$= \{3, 4, 6, 7, 8, 9\}$$

$$\delta'(3, a) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(3, \varepsilon), a\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(3), a)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(3, 4, 6), a\right)$$

$$= \varepsilon\text{-closure}\left(\delta(3, a) \cup \delta(4, a) \cup \delta(6, a)\right)$$

$$= \varepsilon\text{-closure}(\phi \cup 5 \cup \phi)$$

$$= \varepsilon\text{-closure}(5)$$

$$= \{3, 4, 5, 6, 8, 9\}$$

$$\delta'(3, b) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(3, \varepsilon), b\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(3), b)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(3, 4, 6), b\right)$$

$$= \varepsilon\text{-closure}\left(\delta(3, b) \cup \delta(4, b) \cup \delta(6, b)\right)$$

$$= \varepsilon\text{-closure}(\phi \cup \phi \cup 7)$$

$$= \varepsilon\text{-closure}(7)$$

$$= \{3, 4, 6, 7, 8, 9\}$$

$$\delta'(4, a) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(4, \varepsilon), a\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(4), a)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(4, a)\right)$$

$$= \varepsilon\text{-closure}(5)$$

$$= \{3, 4, 5, 6, 8, 9\}$$

$$\delta'(2, b) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(2, \varepsilon), b\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(2), b)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(2, 3, 4, 6, 9), b\right)$$

$$= \varepsilon\text{-closure}(\delta(2, b) \cup \delta(3, b) \cup \delta(4, b) \cup \delta(6, b) \cup \delta(9, b))$$

$$= \varepsilon\text{-closure}(\phi \cup \phi \cup \phi \cup 7 \cup \phi)$$

$$= \varepsilon\text{-closure}(7)$$

$$= \{3, 4, 6, 7, 8, 9\}$$

$$\delta'(3, a) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(3, \varepsilon), a\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(3), a)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(3, 4, 6), a\right)$$

$$= \varepsilon\text{-closure}(\delta(3, a) \cup \delta(4, a) \cup \delta(6, a))$$

$$= \varepsilon\text{-closure}(\phi \cup 5 \cup \phi)$$

$$= \varepsilon\text{-closure}(5)$$

$$= \{3, 4, 5, 6, 8, 9\}$$

$$\delta'(3, b) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(3, \varepsilon), b\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(3), b)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(3, 4, 6), b\right)$$

$$= \varepsilon\text{-closure}(\delta(3, b) \cup \delta(4, b) \cup \delta(6, b))$$

$$= \varepsilon\text{-closure}(\phi \cup \phi \cup 7)$$

$$= \varepsilon\text{-closure}(7)$$

$$= \{3, 4, 6, 7, 8, 9\}$$

$$\delta'(4, a) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(4, \varepsilon), a\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(4), a)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(4, a)\right)$$

$$= \varepsilon\text{-closure}(5)$$

$$= \{3, 4, 5, 6, 8, 9\}$$

$$\delta'(4, b) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(4, \varepsilon), b\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(4), b)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(4, b)\right)$$

$$= \varepsilon\text{-closure}\left(\phi\right)$$

$$= \phi$$

$$\delta'(5, a) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(5, \varepsilon), a\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(5), a)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(3, 4, 5, 6, 8, 9), a\right)$$

$$= \varepsilon\text{-closure}\left(\delta(3, a) \cup \delta(4, a) \cup \delta(5, a) \cup \delta(6, a) \cup \delta(8, a) \cup \delta(9, a)\right)$$

$$= \varepsilon\text{-closure}(\phi \cup 5 \cup \phi \cup \phi \cup \phi \cup \phi)$$

$$= \varepsilon\text{-closure}(5)$$

$$= \{3, 4, 5, 6, 8, 9\}$$

$$\delta'(5, b) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(5, \varepsilon), b\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(5), b)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(3, 4, 5, 6, 8, 9), b\right)$$

$$= \varepsilon\text{-closure}\left(\delta(3, b) \cup \delta(4, b) \cup \delta(5, b) \cup \delta(6, b) \cup \delta(8, b) \cup \delta(9, b)\right)$$

$$= \varepsilon\text{-closure}(\phi \cup \phi \cup \phi \cup 7 \cup \phi \cup \phi)$$

$$= \varepsilon\text{-closure}(7)$$

$$= \{3, 4, 6, 7, 8, 9\}$$

$$\delta'(6, a) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(6, \varepsilon), a\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(6), a)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(6, a)\right)$$

$$= \varepsilon\text{-closure}(\phi)$$

$$= \phi$$

$$\delta'(6, b) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(6, \varepsilon), b\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(6), b)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(6, b)\right)$$

$$= \varepsilon\text{-closure}\,(7)$$

$$= \{3, 4, 6, 7, 8, 9\}$$

$$\delta'(7, a) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(7, \varepsilon), a\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(7), a)\right)$$

$$= \varepsilon\text{-closure}\,(\delta(3, 4, 6, 7, 8, 9), a)$$

$$= \varepsilon\text{-closure}\,\left(\delta(3, a) \cup \delta(4, a) \cup \delta(6, a) \cup \delta(7, a)\right.$$

$$\left. \cup \delta(8, a) \cup \delta(9, a)\right)$$

$$= \varepsilon\text{-closure}(\phi \cup 5 \cup \phi \cup \phi \cup \phi)$$

$$= \varepsilon\text{-closure}(5)$$

$$= \{3, 4, 5, 6, 8, 9\}$$

$$\delta'(7, b) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(7, \varepsilon), b\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(7), b)\right)$$

$$= \varepsilon\text{-closure}\,(\delta(3, 4, 6, 7, 8, 9), b)$$

$$= \varepsilon\text{-closure}\,\left(\delta(3, b) \cup \delta(4, b) \cup \delta(6, b) \cup \delta(7, b)\right.$$

$$\left. \cup \delta(8, b) \cup \delta(9, b)\right)$$

$$= \varepsilon\text{-closure}(\phi \cup \phi \cup 7 \cup \phi \cup \phi \cup \phi)$$

$$= \varepsilon\text{-closure}(7)$$

$$= \{3, 4, 6, 7, 8, 9\}$$

$$\delta'(8, a) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(8, \varepsilon), a\right)\right)$$

$$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(8), a)\right)$$

$$= \varepsilon\text{-closure}\,(\delta(3, 4, 6, 8, 9), a)$$

$$= \varepsilon\text{-closure}\left(\delta(3, a) \cup \delta(4, a) \cup \delta(6, a) \cup \delta(8, a) \cup \delta(9, a)\right)$$

$$= \varepsilon\text{-closure}(\phi \cup 5 \cup \phi \cup \phi \cup \phi)$$

$= \varepsilon\text{-closure}(5)$

$= \{3, 4, 5, 6, 8, 9\}$

$\delta(8, b) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(8, \varepsilon), b\right)\right)$

$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(8), b)\right)$

$= \varepsilon\text{-closure}\left(\delta(3, 4, 6, 8, 9), b\right)$

$= \varepsilon\text{-closure}\left(\delta(3, b) \cup \delta(4, b) \cup \delta(6, b) \cup \delta(8, b) \cup \delta(9, b)\right)$

$= \varepsilon\text{-closure}(\phi \cup \phi \cup 7 \cup \phi \cup \phi)$

$= \varepsilon\text{-closure}(7)$

$= \{3, 4, 6, 7, 8, 9\}$

$\delta(9, a) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(9, \varepsilon), a\right)\right)$

$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(9), a)\right)$

$= \varepsilon\text{-closure}\left(\delta(9, a)\right)$

$= \varepsilon\text{-closure}(\phi)$

$= \phi$

$\delta(9, b) = \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(9, \varepsilon), b\right)\right)$

$= \varepsilon\text{-closure}\left(\delta(\varepsilon\text{-closure}(9), b)\right)$

$= \varepsilon\text{-closure}\left(\delta(9, b)\right)$

$= \varepsilon\text{-closure}(\phi)$

$= \phi$

Now we will build the transition table using above calculated δ transitions.

State \ Input	a	b
{0}	{1}	ϕ
{1}	ϕ	{2, 3, 4, 6, 9}
({2})	{3, 4, 5, 6, 8, 9}	{3, 4, 6, 7, 8, 9}
{3}	{3, 4, 5, 6, 8, 9}	{3, 4, 6, 7, 8, 9}

\

{4}	{3, 4, 5, 6, 8, 9}	ϕ
(5)	{3, 4, 5, 6, 8, 9}	{3, 4, 6, 7, 8, 9}
{6}	ϕ	{3, 4, 6, 7, 8, 9}
(7)	{3, 4, 5, 6, 8, 9}	{3, 4, 6, 7, 8, 9}
(8)	{3, 4, 5, 6, 8, 9}	{3, 4, 6, 7, 8, 9}
(9)	ϕ	ϕ

State {2} is a final state since ε - closure (2) contains 9 which is actually a final state in given NFA. Similarly state 5, 7, 8 and 9 are final states.

Example 2.5 : *Convert the following NFA with ε to NFA without ε.*

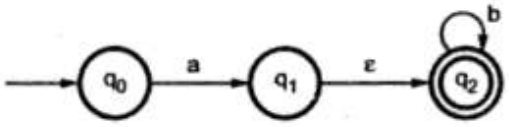


Fig. 2.8

Solution : We will first obtain ε - closures of q_0, q_1 and q_2 as follows.

$$\varepsilon\text{ - closure }(q_0) = \{q_0\}$$

$$\varepsilon\text{ - closure }(q_1) = \{q_1, q_2\}$$

$$\varepsilon\text{ - closure }(q_2) = \{q_2\}$$

Now the δ' transitions on each input symbol is obtained as

$$\delta'(q_0, a) = \varepsilon\text{ - closure}\left(\delta\left(\hat{\delta}(q_0, \varepsilon), a\right)\right)$$

$$= \varepsilon\text{ - closure}\left(\delta\left(\varepsilon\text{ - closure}(q_0), a\right)\right)$$

$$= \varepsilon\text{ - closure}\left(\delta(q_0, a)\right)$$

$$= \varepsilon\text{ - closure}(q_1)$$

$$= \{q_1, q_2\}$$

$$\delta'(q_0, b) = \varepsilon\text{ - closure}\left(\delta\left(\hat{\delta}(q_0, \varepsilon), b\right)\right)$$

$$= \varepsilon\text{ - closure}\left(\delta\left(\varepsilon\text{ - closure}(q_0), b\right)\right)$$

$$= \varepsilon\text{ - closure}\left(\delta(q_0, b)\right)$$

$$= \phi$$

$$
\begin{aligned}
\delta'(q_1, a) &= \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(q_1, \varepsilon), a\right)\right) \\
&= \varepsilon\text{-closure}\left(\delta\left(\varepsilon\text{-closure}(q_1), a\right)\right) \\
&= \varepsilon\text{-closure}\left(\delta(q_1, q_2), a\right) \\
&= \varepsilon\text{-closure}\left(\delta(q_1, a) \cup \delta(q_2, a)\right) \\
&= \varepsilon\text{-closure}(\phi \cup \phi) \\
&= \phi \\
\delta'(q_1, b) &= \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(q_1, \varepsilon), b\right)\right) \\
&= \varepsilon\text{-closure}\left(\delta\left(\varepsilon\text{-closure}(q_1), b\right)\right) \\
&= \varepsilon\text{-closure}\left(\delta(q_1, q_2), b\right) \\
&= \varepsilon\text{-closure}\left(\delta(q_1, b) \cup \delta(q_2, b)\right) \\
&= \varepsilon\text{-closure}(\phi \cup q_2) \\
&= \varepsilon\text{-closure}(q_2) \\
&= \{q_2\} \\
\delta'(q_2, a) &= \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(q_2, \varepsilon), a\right)\right) \\
&= \varepsilon\text{-closure}\left(\delta\left(\varepsilon\text{-closure}(q_2), a\right)\right) \\
&= \varepsilon\text{-closure}\left(\delta(q_2, a)\right) \\
&= \varepsilon\text{-closure}(\phi) \\
&= \phi \\
\delta'(q_2, b) &= \varepsilon\text{-closure}\left(\delta\left(\hat{\delta}(q_2, \varepsilon), b\right)\right) \\
&= \varepsilon\text{-closure}\left(\delta\left(\varepsilon\text{-closure}(q_2), b\right)\right) \\
&= \varepsilon\text{-closure}\left(\delta(q_2, b)\right) \\
&= \varepsilon\text{-closure}(q_2) \\
&= \{q_2\}
\end{aligned}
$$

The transition table can be -

	State \ Input	a	b
→	q_0	$\{q_1, q_2\}$	ϕ
	(q_1)	ϕ	$\{q_2\}$
	(q_2)	ϕ	$\{q_2\}$

States q_1 and q_2 becomes the final as ε - closures of q_1 and q_2 contains the final state q_2. The NFA can be shown by following transition diagram -

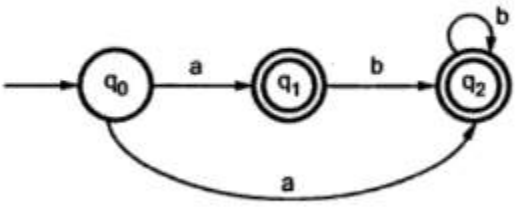


Fig. 2.9

Example 2.6 : *Convert the given NFA with ε to ordinary NFA.*

Solution :

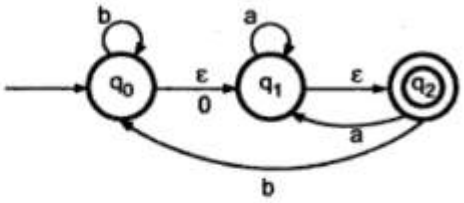


Fig. 2.10

We will first obtain ε - closure of each state.

ε - closure $(q_0) = \{q_0, q_1, q_2\}$

ε - closure $(q_1) = \{q_1, q_2\}$

ε - closure $(q_2) = \{q_2\}$

Now δ transitions for each state on each input.

$$\delta(q_0, a) = \varepsilon \text{ - closure } \left(\delta(\hat{\delta}(q_0, \varepsilon), a)\right)$$

$$= \varepsilon \text{ - closure } (\delta(\varepsilon \text{ - closure } (q_0), a))$$

$= \varepsilon$ - closure $(\delta((q_0, q_1, q_2), a))$

$= \varepsilon$ - closure $(\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a))$

$= \varepsilon$ - closure $(\phi \cup q_1 \cup q_1)$

$= \varepsilon$ - closure (q_1)

$= \{q_1, q_2\}$

$\delta'(q_0, b) = \varepsilon$ - closure $\left(\delta(\hat{\delta}(q_0, \varepsilon), b)\right)$

$= \varepsilon$ - closure $(\delta(\varepsilon$ - closure $(q_0), b))$

$= \varepsilon$ - closure $(\delta((q_0, q_1, q_2), b))$

$= \varepsilon$ - closure $(\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b))$

$= \varepsilon$ - closure $(q_0 \cup \phi \cup q_0)$

$= \varepsilon$ - closure (q_0)

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

$\delta'(q_1, a) = \varepsilon$ - closure $\left(\delta(\hat{\delta}(q_1, \varepsilon), a)\right)$

$= \varepsilon$ - closure $(\delta(\varepsilon$ - closure $(q_1), a))$

$= \varepsilon$ - closure $(\delta(\{q_1, q_2\}, a))$

$= \varepsilon$ - closure $(\delta(q_1, a) \cup \delta(q_2, a))$

$= \varepsilon$ - closure $(q_1 \cup q_1)$

$= \varepsilon$ - closure (q_1)

$= \{q_1, q_2\}$

$\delta'(q_1, b) = \varepsilon$ - closure $\left(\delta(\hat{\delta}(q_1, \varepsilon), b)\right)$

$= \varepsilon$ - closure $(\delta(\varepsilon$ - closure $(q_1), b))$

$= \varepsilon$ - closure $(\delta((q_1, q_2), b))$

$= \varepsilon$ - closure $(\delta(q_1, b) \cup \delta(q_2, b))$

$= \varepsilon$ - closure $(\phi \cup q_0)$

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

$\delta'(q_2, a) = \varepsilon$ - closure $\left(\delta(\hat{\delta}(q_2, \varepsilon), a)\right)$

$= \varepsilon$ - closure $(\delta(\varepsilon$ - closure $(q_2), a))$

$$= \varepsilon \text{ - closure } (\delta(q_2, a))$$

$$= \varepsilon \text{ - closure } (q_1)$$

$$= \{q_1, q_2\}$$

$$\delta'(q_2, b) = \varepsilon \text{ - closure } \left(\delta(\hat{\delta}(q_2, \varepsilon), b)\right)$$

$$= \varepsilon \text{ - closure } (\delta(\varepsilon \text{ - closure } (q_2), b))$$

$$= \varepsilon \text{ - closure } (\delta(q_2, b))$$

$$= \varepsilon \text{ - closure } (q_0)$$

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

The transition table can be -

		a	b
→	q_0	$\{q_1, q_2\}$	$\{q_0, q_1, q_2\}$
	q_1	$\{q_1, q_2\}$	$\{q_0, q_1, q_2\}$
	(*) q_2	$\{q_1, q_2\}$	$\{q_0, q_1, q_2\}$

2.4 NFA to DFA Conversion

As we have discussed, the finite automata can either be DFA or NFA. You might be thinking now, who is better NFA or DFA. Which has more power ? Here is a theorem which tell you that any NFA can be converted to its equivalent DFA. That is any language L acceptable by NFA can be acceptable by its equivalent DFA. The basic idea in this theorem is that, DFA keeps track of all the states, that NFA could be in reading the same input as the DFA has read.

Let us see the theorem.

Theorem : Let L be a set accepted by non-deterministic finite automation. Then there exists a deterministic finite automation that accepts L.

Proof : Let

$M = (Q, \Sigma, \delta, q_0, F)$ be an NFA for language L. Then define DFA M' such that

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

The states of M' are all the subset of M'. The $Q' = 2^Q$.

F' be the set of all the final states in M.

The elements in Q′ will be denoted by $[q_1, q_2, q_3, \ldots q_i]$ and the elements in Q are denoted by $\{q_0, q_1, q_2, \ldots\}$. The $[q_1, q_2, \ldots q_i]$ will be assumed as one state in Q′ if in the NFA q_0 is a initial state it is denoted in DFA as $q_0' = [q_0]$. We define,

$$\delta'([q_1, q_2, q_3, \ldots q_i], a) = [p_1, p_2, p_3, \ldots p_j]$$

if and only if,

$$\delta(\{q_1, q_2, q_3, \ldots q_i\}, a) = \{p_1, p_2, p_3, \ldots p_j\}$$

This means that whenever in NFA, at the current states $\{q_1, q_2, q_3, \ldots q_i\}$ if we get input a and it goes to the next states $\{p_1, p_2, \ldots p_j\}$ then while constructing DFA for it the current state is assumed to be $[q_1, q_2, q_3, \ldots q_i]$. At this state, the input is a and the next is assumed to be $[p_1, p_2, \ldots p_j]$. On applying δ function on each of the states $q_1, q_2, q_3, \ldots q_i$ the new states may be any of the states from $[p_1, p_2, \ldots p_j]$. The theorem can be proved with the induction method by assuming length of input string x

$$\delta'(q_0', x) = [q_1, q_2, \ldots q_i]$$

if and only if,

$$\delta(q_0, x) = \{q_1, q_2, q_3, \ldots q_i\}$$

Basis : If length of input string is 0

i.e. $|x| = 0$, that means x is ε then $q_0' = [q_0]$

Induction : If we assume that the hypothesis is true for the input string of length m or less than m. Then if x a is a string of length m+1. Then the function δ' could be written as

$$\delta'(q_0', xa) = \delta'(\delta'(q_0, x), a)$$

By the induction hypothesis,

$$\delta'(q_0', x) = [p_1, p_2, \ldots p_j]$$

if and only if,

$$\delta(q_0, x) = \{p_1, p_2, p_3, \ldots p_j\}$$

By definition of δ'

$$\delta'([p_1, p_2, \ldots p_j], a) = [r_1, r_2, \ldots r_k]$$

if and only if,

$$\delta(\{p_1, p_2, \ldots p_j\}, a) = \{r_1, r_2, \ldots r_k\}$$

Thus

$$\delta'(q_0', xa) = [r_1, r_2, \ldots r_k]$$

if and only if

$$\delta(q_0, x\ a) = \{r_1, r_2, \ldots r_k\}$$

is shown by inductive hypothesis.

Thus $L(M) = L(M')$

With the help of this theorem, let us try to solve few examples.

Conversion from NFA to DFA

We will discuss the method of converting NFA to its equivalent DFA.

Let, $M = (Q, \Sigma, \delta, q_0, F)$ is a NFA which accepts the language L(M). There should be equivalent DFA denoted by $M' = (Q', \Sigma', \delta', q_0', F')$ such that L(M) = L(M').

The conversion method will follow following steps -

1) The start state of NFA M will be the start for DFA M'. Hence add q_0 of NFA (start state) to Q'. Then find the transitions from this start state.
2) For each state $[q_1, q_2, q_3, \ldots\ldots q_i]$ in Q' the transitions for each input symbol Σ can be obtained as,
   i) $\delta'([q_1, q_2, \ldots\ldots q_i], a) = \delta(q_1, a) \cup \delta(q_2, a) \cup \ldots\ldots \delta(q_i, a)$
   $= [q_1, q_2, \ldots\ldots q_k]$ may be some state.
   ii) Add the state $[q_1, q_2, \ldots\ldots q_k]$ to DFA if it is not already added in Q'.
   iii) Then find the transitions for every input symbol from Σ for state $[q_1, q_2, \ldots\ldots q_k]$. If we get some state $[q_1, q_2, \ldots\ldots q_n]$ which is not in Q' of DFA then add this state to Q'.
   iv) If there is no new state generating then stop the process after finding all the transitions.
3) For the state $[q_1, q_2, \ldots\ldots q_n] \in Q'$ of DFA if any one state q_i is a final state of NFA then $[q_1, q_2, \ldots\ldots q_n]$ becomes a final state. Thus the set of all the final states $\in F'$ of DFA.

Example 2.7 : *Let* $M = (\{q_0, q_1\}, \{0,1\}, \delta, q_0, \{q_1\})$

be NFA where $\delta(q_0, 0) = \{q_0, q_1\}, \delta(q_0, 1) = \{q_1\}$

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

Construct its equivalent DFA.

Solution : Let the DFA $M' = (Q', \Sigma, \delta', q_0', F')$

Now, the δ' function will be computed as follows -

As $\delta(q_0, 0) = \{q_0, q_1\}$ $\delta'([q_0], 0) = [q_0, q_1]$

As in NFA the initial state is q_0, the DFA will also contain the initial state $[q_0]$.

Let us draw the transition table for δ function for a given NFA.

		State \ Input	0	1		
$\delta(q_0, 0)$	$\Rightarrow$	$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_1\}$	$\Leftarrow$	$\delta(q_0, 1)$
$\delta(q_1, 0)$	$\Rightarrow$	(q_1)	ϕ	$\{q_0, q_1\}$	$\Leftarrow$	$\delta(q_1, 1)$

δ Function for NFA

From the transition table we can compute that there are $[q_0]$, $[q_1]$, $[q_0, q_1]$ states for its equivalent DFA. We need to compute the transition from state $[q_0, q_1]$.

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

$$= \{q_0, q_1\} \cup \phi$$

$$= \{q_0, q_1\}$$

So, $\delta'([q_0, q_1], 0) = [q_0, q_1]$

Similarly,

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

$$= \{q_1\} \cup \{q_0, q_1\}$$

$$= \{q_0, q_1\}$$

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

As in the given NFA q_1 is a final state, then in DFA wherever q_1 exists that state becomes a final state. Hence in the DFA final states are $[q_1]$ and $[q_0, q_1]$. Therefore set of final states $F = \{[q_1], [q_0, q_1]\}$.

The equivalent DFA is -

State \ Input	0	1
$\rightarrow q_0$	$[q_0, q_1]$	$[q_1]$
$([q_1])$	ϕ	$[q_0, q_1]$
$([q_0, q_1])$	$[q_0, q_1]$	$[q_0, q_1]$

Transition table for equivalent DFA

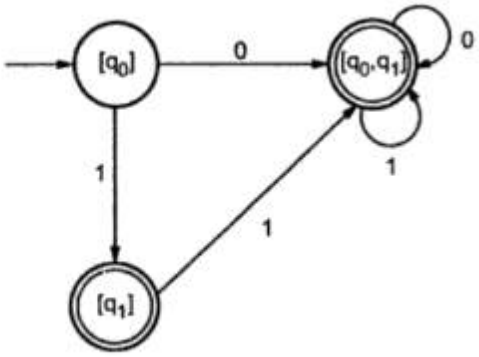


Fig. 2.11

Even we can change the names of the states of DFA.

$A = [q_0]$

$B = [q_1]$

$C = [q_0, q_1]$

With these new names the DFA will be as follows -

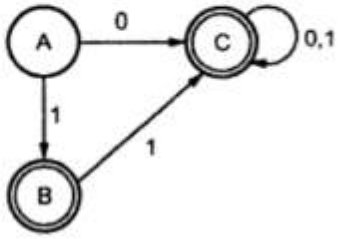


Fig. 2.12 An equivalent DFA

➠ **Example 2.8 :** *Convert the given NFA to DFA.*

State \ Input	0	1
→ q_0	$\{q_0, q_1\}$	q_0
q_1	q_2	q_1
q_2	q_3	q_3
((q_3))	ϕ	q_2

Solution : As we know, first we will compute δ' function.

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

Hence $\delta'([q_0], 0) = [q_0, q_1]$

Similarly,

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

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

Thus we have got a new state $[q_0, q_1]$.

Let us check how it behaves on input 0 and 1.

So,

$$\begin{aligned}\delta'([q_0, q_1], 0) &= \delta([q_0], 0) \cup \delta([q_1], 0) \\ &= \{q_0, q_1\} \cup \{q_2\} \\ &= \{q_0, q_1, q_2\}\end{aligned}$$

Hence a new state is generated i.e. $[q_0, q_1, q_2]$

Similarly

$$\begin{aligned}\delta'([q_0, q_1], 1) &= \delta([q_0], 1) \cup \delta([q_1], 1) \\ &= \{q_0\} \cup \{q_1\} \\ &= \{q_0, q_1\}\end{aligned}$$

No new state is generated here.

Again δ' function will be computed for $[q_0, q_1, q_2]$, the new state being generated.

State	0	1
$[q_0]$	$[q_0, q_1]$	$[q_0]$
$[q_0, q_1]$	$[q_0, q_1, q_2]$	$[q_0, q_1]$
$[q_0, q_1, q_2]$	$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_3]$

As, you have observed in the above table for a new state $[q_0, q_1, q_2]$ the input 0 will give a new state $[q_0, q_1, q_2, q_3]$ and input 1 will give a new state $[q_0, q_1, q_3]$, because

$$\begin{aligned}\delta([q_0, q_1, q_2], 0) &= \delta([q_0], 0) \cup \delta([q_1], 0) \cup \delta([q_2], 0) \\ &= \{q_0, q_1\} \cup \{q_2\} \cup \{q_3\} \\ &= \{q_0, q_1, q_2, q_3\} \\ &= [q_0, q_1, q_2, q_3]\end{aligned}$$

Same procedure for input 1. Thus the final DFA is as given below.

State	0	1
$\rightarrow [q_0]$	$[q_0, q_1]$	$[q_0]$
$(*)[q_1]$	$[q_2]$	$[q_1]$
$[q_2]$	$[q_3]$	$[q_3]$
$[q_3]$	ϕ	$[q_2]$
$[q_0, q_1]$	$[q_0, q_1, q_2]$	$[q_0, q_1]$
$(*)[q_0, q_1, q_2]$	$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_3]$
$(*)[q_0, q_1, q_3]$	$[q_0, q_1, q_2]$	$[q_0, q_1, q_2]$
$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_2, q_3]$

DFA for example 2.8

Example 2.9 :*Construct DFA equivalent to the given NFA.*

State \ Input	0	1
→ p	{p, q}	p
q	r	r
r	s	–
s	s	s

Solution : The NFA M = {{p, q, r, s}, {0, 1}, δ {p}, {s}}

The equivalent DFA will be constructed.

State \ Input	0	1
→ [p]	[p, q]	[p]
[q]	[r]	[r]
[r]	[s]	–
[s]	[s]	[s]
[p, q]	[p, q, r]	[p, r]

Continuing with the generated new states.

State \ Input	0	1
→ [p]	[p, q]	[p]
[q]	[r]	[r]
[r]	[s]	–
([s])	[s]	[s]
[p, q]	[p, q, r]	[p, r]
[p, q, r]	[p, q, r, s]	[p, r]
[p, r]	[p, q, s]	[p]
([p, q, r, s])	[p, q, r, s]	[p, r, s]
([p, q, s])	[p, q, r, s]	[p, r, s]
([p, r, s])	[p, q, s]	[p, s]
([p, s])	[p, q, s]	[p, s]

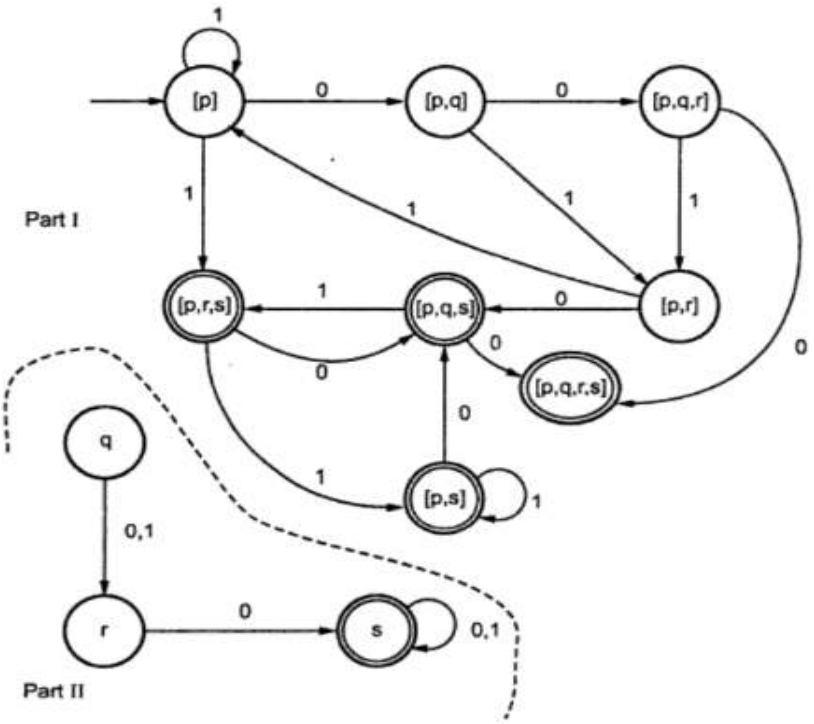


Fig. 2.14

Example 2.11 : *Convert the following NFA into DFA.*

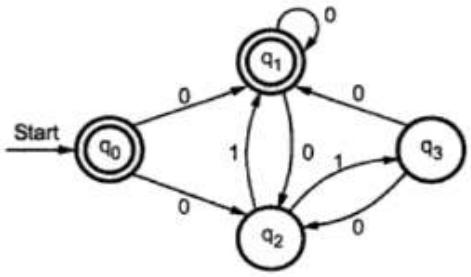


Fig. 2.17

Solution : We will first design a transition table from given transition diagram.

State \ Input	0	1
q_0	$\{q_1, q_2\}$	ϕ
q_1	$\{q_1, q_2\}$	ϕ
q_2	ϕ	$\{q_1, q_3\}$
q_3	$\{q_1, q_2\}$	ϕ

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

We can write this as

$$\delta'([q_0], 0) = [q_1, q_2]$$

Here $[q_1, q_2]$ becomes one newly generated state when 0 input is received in state $[q_0]$

Similarly

$$\delta'([q_0], 1) = \phi \quad \text{No state getting generated.}$$

$$\delta(q_1, 0) = \{q_1, q_2\} \text{ we can write it as}$$

$$\delta'([q_1], 0) = [q_1, q_2]$$

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

$$\delta'([q_2], 0) = \phi$$

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

i.e. $\delta'([q_2], 1) = [q_1, q_3]$ again a new state $[q_1, q_3]$ gets generated.

Similarly,

$$\delta'([q_3], 0) = [q_1, q_2]$$

$$\delta'([q_3], 1) = \phi$$

Now we will obtain δ' transitions on newly generated states $[q_1, q_2]$ and $[q_1, q_3]$.

$$\begin{aligned}\delta'([q_1, q_2], 0) &= \delta([q_1], 0) \cup \delta([q_2], 0) \\ &= \{q_1, q_2\} \cup \phi \\ &= \{q_1, q_2\}\end{aligned}$$

$$\therefore \quad \delta'([q_1, q_2], 0) = [q_1, q_2]$$

Similarly,

$$\begin{aligned}\delta'([q_1, q_2], 1) &= \delta([q_1], 1) \cup \delta([q_2], 1) \\ &= \phi \cup \{q_1, q_3\} \\ &= \{q_1, q_3\}\end{aligned}$$

$$\therefore \quad \delta'([q_1, q_2], 1) = [q_1, q_3]$$

$$\begin{aligned}\text{Now, } \delta'([q_1, q_3], 0) &= \delta([q_1], 0) \cup \delta([q_3], 0) \\ &= \{q_1, q_2\} \cup \{q_1, q_2\} \\ &= \{q_1, q_2\}\end{aligned}$$

$$\therefore \quad \delta'([q_1, q_3], 0) = [q_1, q_2]$$

Similarly,

$$\delta([q_1, q_3], 1) = \delta([q_1], 1) \cup \delta([q_3], 1)$$
$$= \phi \cup \phi$$
$$= \phi$$

$$\therefore \quad \delta([q_1, q_3], 1) = \phi$$

As now no new states are getting generated.

The transition table for DFA using above δ transitions i

State \ Input	0	1
→ $[q_0]$	$[q_1, q_2]$	ϕ
$[q_1]$	$[q_1, q_2]$	ϕ
$[q_2]$	ϕ	$[q_1, q_3]$
$[q_3]$	$[q_1, q_2]$	ϕ
$[q_1, q_2]$	$[q_1, q_2]$	$[q_1, q_3]$
$[q_1, q_3]$	$[q_1, q_2]$	ϕ

The transition diagram can be -

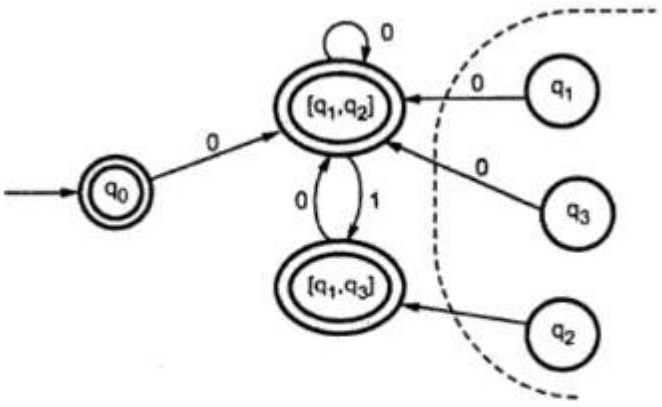


Fig. 2.18

As we can see the marked part of DFA consisting of q_1, q_2, q_3 is of no use since there is no path from start state to these states. Hence we will never reach to these states. So we can eliminate these states. Hence new DFA becomes.

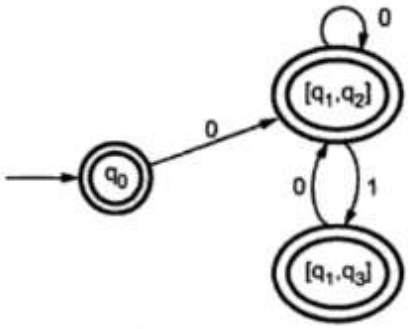


Fig. 2.19

As states q_0 and q_1 are final states original NFA states containing q_1 i.e. $[q_1, q_3]$ and $[q_1, q_2]$ are also final states.

➠ **Example 2.12 :** *Convert the following NFA into its equivalent DFA. Also recognize the language generated by this DFA.*

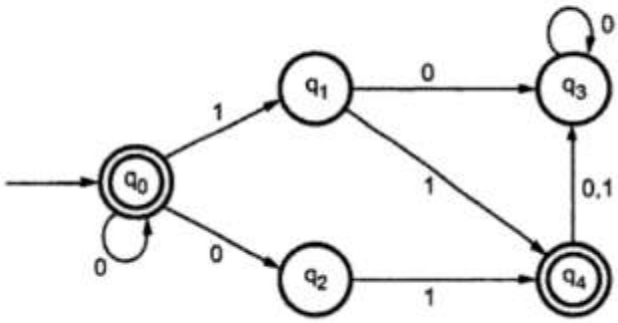


Fig. 2.20

Solution : We will first desing the transition table from given transition diagram.

State \ Input	0	1
→ q_0	$\{q_0, q_2\}$	q_1
q_1	q_3	q_4
q_2	ϕ	q_4
q_3	q_3	ϕ
q_4	q_3	q_3

$\delta'([q_0], 0) = [q_0, q_2]$ new state generated

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

$\delta'([q_1], 0) = [q_3]$

$\delta'([q_1], 1) = [q_4]$

$\delta'([q_2], 0) = \phi$

$\delta'([q_2], 1) = [q_4]$

$\delta'([q_3], 0) = [q_3]$

$\delta'([q_3], 1) = \phi$

$\delta'([q_4], 0) = [q_3]$

$\delta'([q_4], 1) = [q_3]$

Now we will obtain input transitions on $[q_0, q_2]$.

$$\begin{aligned}\delta'([q_0, q_2], 0) &= \delta([q_0], 0) \cup \delta([q_2], 0) \\ &= \{q_0, q_2\} \cup \phi \\ &= \{q_0, q_2\}\end{aligned}$$

i.e. $\delta'([q_0, q_2], 0) = [q_0, q_2]$

Similarly,

$$\begin{aligned}\delta'([q_0, q_2], 1) &= \delta([q_0], 1) \cup \delta([q_2], 1) \\ &= [q_1, q_4] \text{ new state generated.}\end{aligned}$$

We will obtain δ' transitions for new state $[q_1, q_4]$.

$$\begin{aligned}\delta'([q_1, q_4], 0) &= \delta([q_1], 0) \cup \delta([q_4], 0) \\ &= \{q_3\} \cup \{q_3\} \\ &= \{q_3\}\end{aligned}$$

$\therefore \quad \delta'([q_1, q_4], 0) = [q_3]$

$$\begin{aligned}\text{Now } \delta'([q_1, q_4], 1) &= \delta([q_1], 1) \cup \delta([q_4], 1) \\ &= \{q_4\} \cup \{q_3\} \\ &= \{q_4, q_3\}\end{aligned}$$

$\therefore \quad \delta'([q_1, q_4], 1) = [q_4, q_3]$ new state generated.

We will obtain δ' transitions for new state $[q_4, q_3]$

$$\delta([q_4, q_3], 0) = \delta([q_4], 0) \cup \delta([q_3], 0)$$
$$= \{q_3\} \cup \{q_3\}$$
$$= \{q_3\}$$
$$\therefore \quad \delta([q_4, q_3], 0) = [q_3]$$

Similarly,

$$\delta([q_4, q_3], 1) = \delta([q_4], 1) \cup \delta([q_3], 1)$$
$$= \{q_3\} \cup \phi$$
$$= \{q_3\}.$$
$$\therefore \quad \delta([q_4, q_3], 1) = [q_3]$$

Now since no new state is being generated.

We will now design the transition diagrams from the above δ transitions.

State \ Input	0	1
→ $[q_0]$	$[q_0, q_2]$	$[q_1]$
$[q_1]$	$[q_3]$	$[q_4]$
$[q_2]$	ϕ	$[q_4]$
$[q_3]$	$[q_3]$	ϕ
$[q_4]$	$[q_3]$	$[q_3]$
$[q_0, q_2]$	$[q_0, q_2]$	$[q_1, q_4]$
$[q_1, q_4]$	$[q_3]$	$[q_4, q_3]$
$[q_4, q_3]$	$[q_3]$	$[q_3]$

The states q_0 and q_4 are basically final states. Hence newly generated states containing q_4 are final states. Therefore $[q_1, q_4]$ and $[q_4, q_3]$ are final states. Now the transition diagram for DFA are -

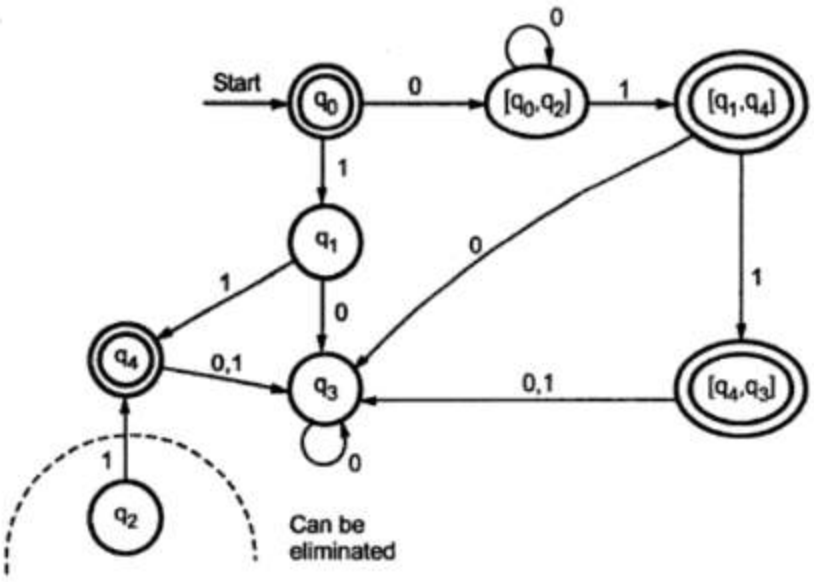


Fig. 2.21

The DFA will then be -

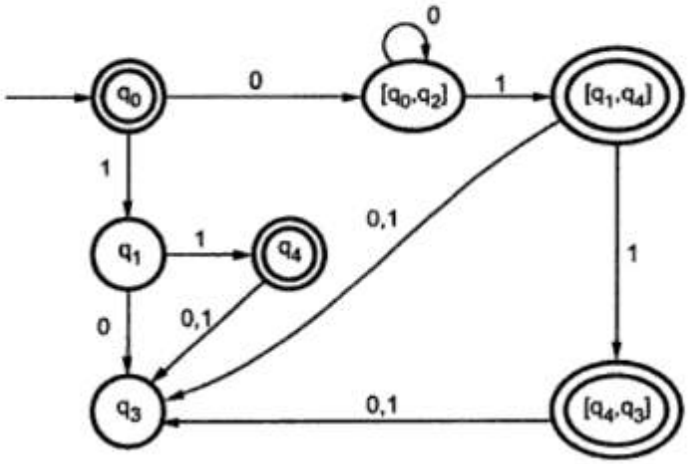


Fig. 2.22

This is a language accepting '11' or 0^n 11 where $n \geq 0$. That means it is a language of any number of zeros followed by 11 or 1.

Example 2.13 : *For the following state transition table draw the state transition diagram. Find its equivalent machine. For the string abbaaab test whether both give same result or not. q_0 is initial state and q_3 is a final state.*

q/Σ	0	1
q_0	q_1	q_2
q_1	q_1	q_1, q_3
q_2	ϕ	ϕ
q_3	q_0, q_3	q_3

Solution : The transition diagram for the given transition table can be -

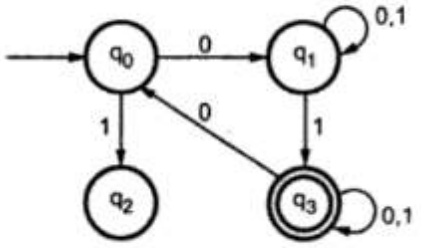


Fig. 2.23

This is clearly a non deterministic Finite Automata.

Now we will convert it to an equivalent DFA.

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

$$\therefore \quad \delta([q_0], 0) = [q_1]$$

Similarly, $\delta([q_0], 1) = [q_2]$

$$\delta([q_1], 0) = \{q_1\}$$

$$= [q_1]$$

$$\delta([q_1], 1) = \{q_1, q_3\}$$

$$= [q_1, q_3] \leftarrow \text{new state generated.}$$

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

$$\delta([q_2], 1) = \phi$$

$$\delta([q_3], 0) = [q_0, q_3] \leftarrow \text{new state generated.}$$

$$\delta([q_3], 1) = [q_3]$$

Now we will obtain δ' transitions on the new states generated.

$$\begin{aligned}
\delta'([q_1, q_3], 0) &= \delta([q_1], 0) \cup \delta([q_3], 0) \\
&= \{q_1\} \cup \{q_0, q_3\} \\
&= \{q_0, q_1, q_3\} \\
&= [q_0, q_1, q_3] \leftarrow \text{new state generated.} \\
\delta'([q_1, q_3], 1) &= \delta([q_1], 1) \cup \delta([q_3], 1) \\
&= \{q_1, q_3\} \cup \{q_3\} \\
&= \{q_1, q_3\} \\
&= [q_1, q_3] \\
\delta'([q_0, q_3], 0) &= \delta([q_0], 0) \cup \delta([q_3], 0) \\
&= \{q_1\} \cup \{q_0, q_3\} \\
&= \{q_0, q_1, q_3\} \\
&= [q_0, q_1, q_3] \\
\delta'([q_0, q_3], 1) &= \delta([q_0], 1) \cup \delta([q_3], 1) \\
&= \{q_2\} \cup \{q_3\} \quad \leftarrow \text{new state generated.} \\
&= [q_2, q_3]
\end{aligned}$$

Again we get two more new states $[q_0, q_1, q_3]$ and $[q_2, q_3]$. Hence we will obtain δ' transitions on them

$$\begin{aligned}
\delta'([q_0, q_1, q_3], 0) &= \delta([q_0], 0) \cup \delta([q_1], 0) \cup \delta([q_3], 0) \\
&= \{q_1\} \cup \{q_1\} \cup \{q_0, q_3\} \\
&= \{q_0, q_1, q_3\} \\
&= [q_0, q_1, q_3] \\
\delta'([q_0, q_1, q_3], 1) &= \delta([q_0], 1) \cup \delta([q_1], 1) \cup \delta([q_3], 1) \\
&= \{q_2\} \cup \{q_1\} \cup \{q_3\} \\
&= [q_1, q_2, q_3] \leftarrow \text{new state generated} \\
\delta'([q_2, q_3], 0) &= \delta([q_2], 0) \cup \delta([q_3], 0) \\
&= \phi \cup \{q_0, q_3\} \\
&= [q_0, q_3] \\
\delta'([q_2, q_3], 1) &= \delta([q_2], 1) \cup \delta([q_3], 1)
\end{aligned}$$

$= \phi \cup \{q_3\}$

$= \{q_3\}$

$= [q_3]$

The δ transition on new state $[q_1, q_2, q_3]$ is -

$\delta([q_1, q_2, q_3], 0) = \delta([q_1], 0) \cup \delta([q_2], 0) \cup \delta([q_3], 0)$

$= \{q_1\} \cup \phi \cup \{q_0, q_3\}$

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

$= [q_0, q_1, q_3]$

$\delta([q_1, q_2, q_3], 1) = \delta([q_1], 1) \cup \delta([q_2], 1) \cup \delta([q_3], 1)$

$= \{q_1, q_3\} \cup \phi \cup \{q_3\}$

$= \{q_1, q_3\}$

$= [q_1, q_3]$

From the above computed δ transitions we can build the transition table as -

	q/Σ	0	1
$\rightarrow$	$[q_0]$	$[q_1]$	$[q_2]$
	$[q_1]$	$[q_1]$	$[q_1, q_3]$
	$[q_2]$	ϕ	ϕ
*	$[q_3]$	$[q_0, q_3]$	$[q_3]$
*	$[q_1, q_3]$	$[q_0, q_1, q_3]$	$[q_1, q_3]$
*	$[q_0, q_3]$	$[q_0, q_1, q_3]$	$[q_2, q_3]$
*	$[q_2, q_3]$	$[q_0, q_3]$	$[q_3]$
*	$[q_0, q_1, q_3]$	$[q_0, q_1, q_3]$	$[q_1, q_2, q_3]$
*	$[q_1, q_2, q_3]$	$[q_0, q_1, q_3]$	$[q_1, q_3]$

The DFA can be represented using following transition diagram.

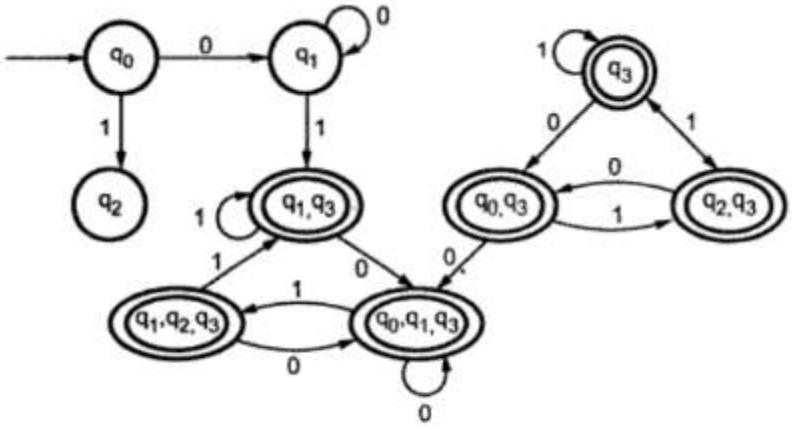


Fig. 2.24

To simulate string abbaaab we use above DFA. Assume input 0 = a and input 1 = b. Then

$\delta(q_0, abbaaab) \vdash a\, q_1\, bbaaab$

$\vdash ab\,(q_1, q_3)\, bbaaab$

$\vdash abb(q_1, q_3)\, baaab$

$\vdash abbb(q_1, q_3)aaab$

$\vdash abbba(q_0, q_1, q_3)\, aab$

$\vdash abbbaa\,(q_0, q_1, q_3)\, ab$

$\vdash abbbaaa\,(q_0, q_1, q_3)\, b$

$\vdash abbbaaab(q_1, q_2, q_3)$

is in final state.

Thus the given input can be accepted by a DFA.

➠ **Example 2.14 :** *Convert the given NFA to DFA.*

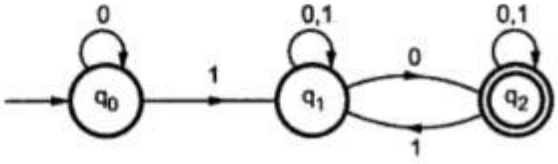


Fig. 2.25

Solution : Conversion of NFA to DFA -

For the given transition diagram in Fig. 2.25 we will first construct the transition table.

	State / Σ	0	1
→	q_0	q_0	q_1
	q_1	$\{q_1, q_2\}$	q_1
	(*) q_2	$\{q_2\}$	$\{q_1, q_2\}$

Now we will obtain δ transitions for each state.

$$\delta([q_0], 0) = \{q_0\}$$
$$= [q_0]$$
$$\delta([q_0], 1) = \{q_1\}$$
$$= [q_1]$$
$$\delta([q_1], 0) = \{q_1, q_2\}$$
$$= [q_1, q_2] \rightarrow \text{new state generated.}$$
$$\delta([q_1], 1) = \{q_1\} = [q_1]$$

$$\delta([q_2], 0) = \{q_2\}$$
$$= [q_2]$$
$$\delta([q_2], 1) = \{q_1, q_2\}$$
$$= [q_1, q_2]$$

Now we will obtain δ transitions on $[q_1, q_2]$

$$\delta([q_1, q_2], 0) = \delta([q_1], 0) \cup \delta([q_2], 0)$$
$$= \{q_1, q_2\} \cup \{q_2\}$$
$$= \{q_1, q_2\}$$
$$= [q_1, q_2]$$
$$\delta([q_1, q_2], 1) = \delta([q_1], 1) \cup \delta([q_2], 1)$$
$$= \{q_1\} \cup \{q_1, q_2\}$$
$$= [q_1, q_2]$$

The state $[q_1, q_2]$ is final state as well because it contains a final state q_2.The transition table for the constructed DFA will be -

	State / Σ	0	1
→	$[q_0]$	$[q_0]$	$[q_1]$
	$[q_1]$	$[q_1, q_2]$	$[q_1]$
	$[q_2]$	$[q_2]$	$[q_1, q_2]$
	$[q_1, q_2]$	$[q_1, q_2]$	$[q_1, q_2]$

The transition diagram will be -

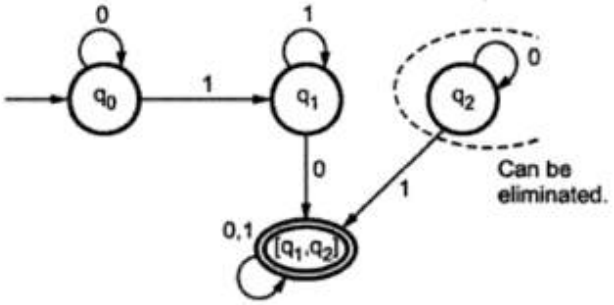


Fig. 2.26

Consider a string 0010 we will get -

Using NFA

q_0 0010 $\vdash$ $0q_0$0110

$\vdash$ $00q_0110$

$\vdash$ $001q_110$

$\vdash$ $0011q_10$

$\vdash$ $00110q_2$ or $\vdash$ $00110q_1$

Accept non-accept.

Using DFA

$q_0$00110 $\vdash$ $0q_0$0110

$\vdash$ $00q_0110$

$\vdash$ $001q_110$

$\vdash$ $0011q_10$

$\vdash$ 00110 $[q_1, q_2]$

Accept state.

Thus both the NFA and DFA accept the same string 00110.

2.4.1 Conversion of NFA with ε to DFA

We have already seen the method of converting NFA with ε to NFA without ε. Now we will learn how to convert NFA with ε to its equivalent DFA. In this method we first convert NFA with ε to NFA without ε. Then the NFA without ε can be converted to its equivalent DFA.

Method for converting NFA with ε to DFA

Step 1 : Consider $M = (Q, \Sigma, \delta, q_0, F)$ is a NFA with ε. We have to convert this NFA with ε to equivalent DFA denoted by

$$M_D = (Q_D, \Sigma, \delta_D, q_0, F_D)$$

Then obtain,

ε - closure$(q_0) = \{p_1, p_2, p_3, \ldots p_n\}$ then $[p_1, p_2, p_3, \ldots p_n]$ becomes a start state of DFA.

Now $[p_1, p_2, p_3, \ldots p_n] \in Q_D$

Step 2 : We will obtain δ transitions on $[p_1, p_2, p_3, \ldots p_n]$ for each input.

$$\delta_D\ ([P_1, P_2, \ldots P_n], a) = \varepsilon \text{ - closure}\big(\delta(P_1, a) \cup \delta(P_2, a) \cup \ldots \delta(P_n, a)\big)$$

$$= \bigcup_{i=1}^{n} \varepsilon \text{ - closure}\, \delta(p_i, a)$$

where a is input $\in \Sigma$.

Step 3 : The states obtained $[p_1, p_2, p_3, \ldots p_n] \in Q_D$. The states containing final state in p_i is a final state in DFA.

Now let us see some examples of conversion based on this procedure.

Example 2.15 : *Convert the following NFA with ε to equivalent DFA.*

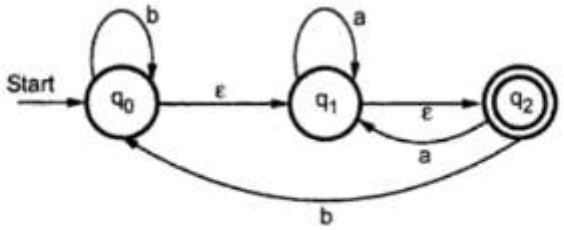


Fig. 2.27

Solution : To convert this NFA we will first find ε-closures.

$$\varepsilon\text{-closures } \{q_0\} = \{q_0, q_1, q_2\}$$

$$\varepsilon\text{-closures } \{q_1\} = \{q_1, q_2\}$$

$$\varepsilon\text{-closures } \{q_2\} = \{q_2\}$$

Let us start from ε-closure of start state

$$\varepsilon\text{-closure } \{q_0\} = \{q_0, q_1, q_2\}$$ we will call this state as A.

Now let us find transitions on A with every input symbol.

$$\delta'(A, a) = \varepsilon\text{-closure } (\delta(A, a))$$

$$= \varepsilon\text{-closure } (\delta(q_0, q_1, q_2), a)$$

$$= \varepsilon\text{-closure } \{\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a)\}$$

$$= \varepsilon\text{-closure } \{q_1\}$$

$$= \{q_1, q_2\}.$$ Let us call it as state B.

$$\delta'(A, b) = \varepsilon\text{-closure } (\delta(A, b))$$

$$= \varepsilon\text{-closure } (\delta(q_0, q_1, q_2), b)$$

$$= \varepsilon\text{-closure } \{\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b)\}$$

$$= \varepsilon\text{-closure } \{q_0\}$$

$$= \{q_0, q_1, q_2\}$$ i.e. A.

Hence we can state that

$$\delta'(A, a) = B$$

$$\delta'(A, b) = A$$

Now let us find transitions for state $B = \{q_1, q_2\}$

$$\delta'(B, a) = \varepsilon\text{-closure } (\delta(q_1, q_2), a)$$

$$= \varepsilon\text{-closure } \{q_1\}$$

$$= \{q_1, q_2\}$$ i.e. B

$$\delta'(B, b) = \varepsilon\text{-closure } (\delta(q_1, q_2), b)$$

$$= \varepsilon\text{-closure } \{\delta(q_1, b) \cup \delta(q_2, b)\}$$

$$= \varepsilon\text{-closure } \{q_0\}$$

$$= \{q_0, q_1, q_2\}$$ i.e. A.

Hence the generated DFA is

	a	b
→ (A)	B	A
(B)	B	A

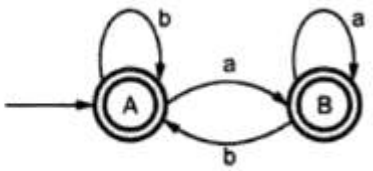


Fig. 2.28

➠ **Example 2.16** : *Convert the given NFA into its equivalent DFA -*

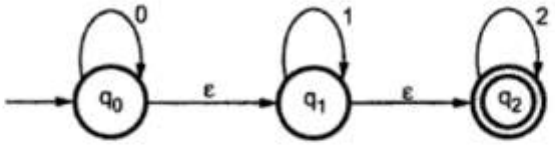


Fig. 2.29

Solution : Let us obtain ε-closure of each state.

$$\varepsilon\text{-closure}(q_0) = \{q_0, q_1, q_2\}$$

$$\varepsilon\text{-closure}(q_1) = \{q_1, q_2\}$$

$$\varepsilon\text{-closure}(q_2) = \{q_2\}$$

Now we will obtain δ' transition. Let ε-closure $(q_0) = \{q_0, q_1, q_2\}$ call it as state A.

$$\delta'(A,0) = \varepsilon\text{-closure}\{\delta((q_0, q_1, q_2), 0)\}$$

$$= \varepsilon\text{-closure}\{\delta(q_0,0) \cup \delta(q_1,0) \cup \delta(q_2,0)\}$$

$$= \varepsilon\text{-closure}\{q_0\}$$

$$= \{q_0, q_1, q_2\} \quad \text{i.e. state A}$$

$$\delta'(A,1) = \varepsilon\text{-closure}\{\delta((q_0, q_1, q_2), 1)\}$$

$$= \varepsilon\text{-closure}\{\delta(q_0,1) \cup \delta(q_1,1) \cup \delta(q_2,1)\}$$

$$= \varepsilon\text{-closure}\{q_1\}$$

$$= \{q_1, q_2\} \quad \text{Call it as state B}$$

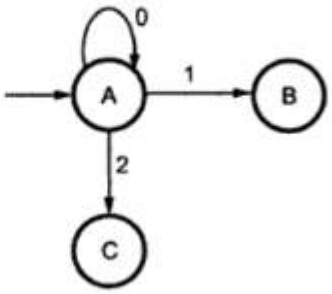
$$\delta'(A,2) = \varepsilon\text{-closure}\{\delta((q_0, q_1, q_2), 2)\}$$

$$= \varepsilon\text{-closure}\{\delta(q_0,2) \cup \delta(q_1,2) \cup \delta(q_2,2)\}$$
$$= \varepsilon\text{-closure}\{q_2\}$$
$$= \{q_2\} \qquad \text{Call it as state C.}$$

Thus we have obtained

$$\delta'(A,0) = A$$
$$\delta'(A,1) = B$$
$$\delta'(A,2) = C$$

i.e.



Now we will find transitions on states B and C for each input.

Hence

$$\delta'(B,0) = \varepsilon\text{-closure}\{\delta(q_1,q_2),0\}$$
$$= \varepsilon\text{-closure}\{\delta(q_1,0) \cup \delta(q_2,0)\}$$
$$= \varepsilon\text{-closure}\{\phi\}$$
$$= \phi$$
$$\delta'(B,1) = \varepsilon\text{-closure}\{\delta(q_1,q_2),1\}$$
$$= \varepsilon\text{-closure}\{\delta(q_1,1) \cup \delta(q_2,1)\}$$
$$= \varepsilon\text{-closure}\{q_1\}$$
$$= \{q_1,q_2\} \text{ i.e state B itself.}$$
$$\delta'(B,2) = \varepsilon\text{-closure}\{\delta(q_1,q_2),2\}$$
$$= \varepsilon\text{-closure}\{\delta(q_1,2) \cup \delta(q_2,2)\}$$
$$= \varepsilon\text{-closure}\{q_2\}$$
$$= \{q_2\} \text{ i.e state C.}$$

Hence

$$\delta'(B, 0) = \phi$$

$$\delta'(B, 1) = B$$

$$\delta'(B, 2) = C$$

The partial transition diagram will be

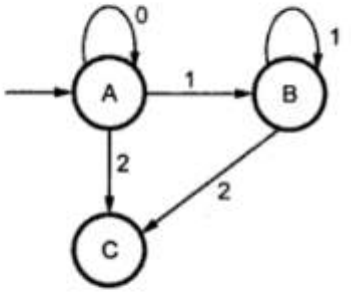


Fig. 2.31

Now we will obtain transitions for C :

$$\delta'(C, 0) = \varepsilon\text{-closure}\{\delta(q_2, 0)\}$$

$$= \varepsilon\text{-closure}\{\phi\}$$

$$= \phi$$

$$\delta'(C, 1) = \varepsilon\text{-closure}\{\delta(q_2, 1)\}$$

$$= \varepsilon\text{-closure}\{\phi\}$$

$$= \phi$$

$$\delta'(C, 2) = \varepsilon\text{-closure}\{\delta(q_2, 2)\}$$

$$= q_2$$

Hence the DFA is

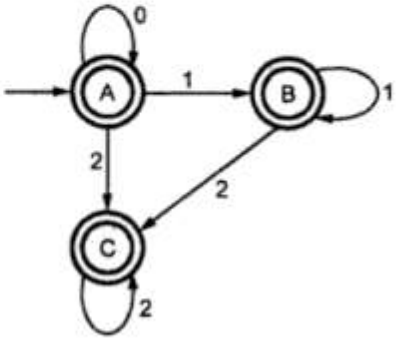


Fig. 2.32

As A = $\{q_0, q_1, q_2\}$ in which final state q_2 lies hence A is final state in B = $\{q_1, q_2\}$ the state q_2 lies hence B is also final state in C = $\{q_2\}$, the state q_2 lies hence C is also a final state.

Example 2.17 : *Convert the given NFA with ε to its equivalent DFA.*

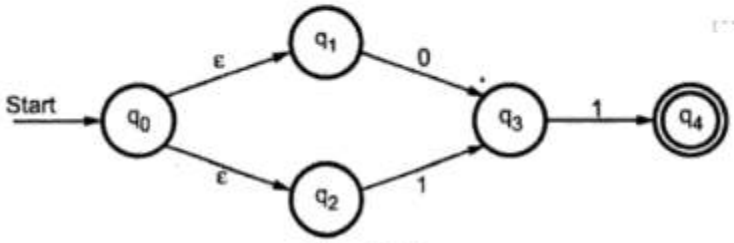


Fig. 2.33

Solution :

ε-closure $\{q_0\}$ = $\{q_0, q_1, q_2\}$

ε-closure $\{q_1\}$ = $\{q_1\}$

ε-closure $\{q_2\}$ = $\{q_2\}$

ε-closure $\{q_3\}$ = $\{q_3\}$

ε-closure $\{q_4\}$ = $\{q_4\}$

Now, Let ε-closure $\{q_0\}$ = $\{q_0, q_1, q_2\}$ be state A.

Hence $\delta'(A, 0)$ = ε-closure $\{\delta((q_0, q_1, q_2), 0)\}$

= ε-closure $\{\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0)\}$

= ε-closure $\{q_3\}$

= $\{q_3\}$ call it as state B.

$\delta'(A, 1)$ = ε-closure $\{\delta((q_0, q_1, q_2), 1)\}$

= ε-closure $\{\delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_2, 1)\}$

= ε-closure $\{q_3\}$

= q_3 = B.

The partial DFA will be

Now,

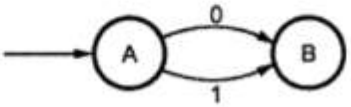


Fig. 2.34

$$\delta'(B,0) = \varepsilon\text{-closure}\{\delta(q_3,0)\}$$
$$= \phi$$
$$\delta'(B,1) = \varepsilon\text{-closure}\{\delta(q_3,1)\}$$
$$= \varepsilon\text{-closure}\{q_4\}$$
$$= \{q_4\} \text{ i.e. state C}$$
$$\delta'(C,0) = \varepsilon\text{-closure}\{\delta(q_4,0)\}$$
$$= \phi$$
$$\delta'(C,1) = \varepsilon\text{-closure}\{\delta(q_4,1)\}$$
$$= \phi$$

The DFA will be,

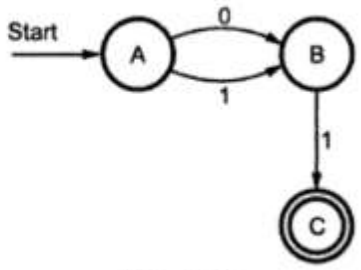


Fig. 2.35

Example 2.18 : *Consider following NFA with ε.*

State \ Input	a	b	c	ε
→ p	{p}	{q}	{r}	ϕ
q	{q}	{r}	ϕ	{p}
(r)	{r}	ϕ	{p}	{q}

Convert it to its equivalent DFA.

Solution : We will first compute ε - closure for start state p.

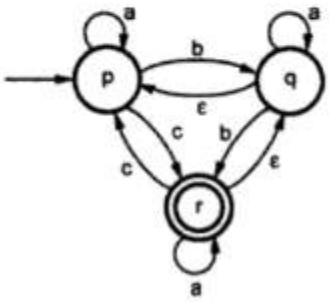


Fig. 2.36

ε - closure (p) = {p} call it as state A.

Now we will obtain δ transitions on state A.

State A

$$\delta(A, a) = \varepsilon\text{ - closure } (\delta(A, a))$$
$$= \varepsilon\text{ - closure}(\delta(p, a))$$
$$= \varepsilon\text{ - closure } (p)$$
$$= \{p\} \text{ i.e. state A only.}$$
$$\delta(A, b) = \varepsilon\text{ - closure } (\delta(A, b))$$
$$= \varepsilon\text{ - closure}(\delta(p, b))$$
$$= \varepsilon\text{ - closure } (q)$$
$$= \{q, p\} \text{ i.e. } \{p, q\} \text{ Let us call it state B.}$$
$$\delta(A, b) = B$$
$$\delta(A, c) = \varepsilon\text{ - closure } (\delta(A, c))$$
$$= \varepsilon\text{ - closure } (\delta(p, c))$$
$$= \varepsilon\text{ - closure } (r) = \{q, r\}. \text{ Call it as state C.}$$

State B = {p, q}

$$\delta(B, a) = \varepsilon\text{ - closure } (\delta(B, a))$$
$$= \varepsilon\text{ - closure}(\delta(p, q), a)$$
$$= \varepsilon\text{ - closure } (\delta(p, a) \cup \delta(q, a))$$
$$= \varepsilon\text{ - closure } (p \cup q)$$
$$= \varepsilon\text{ - closure } (p, q)$$
$$= \varepsilon\text{ - closure}(p) \cup \varepsilon\text{ - closure}(q)$$
$$= \{p\} \cup \{q\} = \{p, q\} \text{ i.e. state B only.}$$
$$\delta(B, a) = B.$$
$$\delta(B, b) = \varepsilon\text{ - closure } (\delta(B, b))$$
$$= \varepsilon\text{ - closure}(\delta(p, q), b)$$
$$= \varepsilon\text{ - closure } (\delta(p, b) \cup \delta(q, b))$$
$$= \varepsilon\text{ - closure } (q \cup r)$$
$$= \varepsilon\text{ - closure } (q, r)$$
$$= \varepsilon\text{ - closure}(q) \cup \varepsilon\text{ - closure}(r)$$
$$= \{p, q\} \cup \{q, r\} = \{p, q, r\} \text{ i.e. state D.}$$

$\therefore \quad \delta(B, b) = D.$

$\delta(B, c) = \varepsilon$ - closure $(\delta(B, c))$

$= \varepsilon$ - closure$(\delta(p, q), c)$

$= \varepsilon$ - closure $(\delta(p, c) \cup \delta(q, c))$

$= \varepsilon$ - closure $(r \cup \phi)$

$= \varepsilon$ - closure (r)

$= \{q, r\}$

$\delta(B, c) = C$

State C = {q, r}

$\delta(C, a) = \varepsilon$ - closure $(\delta(C, a))$

$= \varepsilon$ - closure$(\delta(q, r), a)$

$= \varepsilon$ - closure $(\delta(q, a) \cup \delta(r, a))$

$= \varepsilon$ - closure $(q \cup r)$

$= \varepsilon$ - closure $(q) \cup \varepsilon$ - closure(r)

$= \{p, q\} \cup \{q, r\}$

$= \{p, q, r\}$ i.e. state D.

$\therefore \quad \delta(C, a) = D$

$\delta(C, b) = \varepsilon$ - closure $(\delta(C, b))$

$= \varepsilon$ - closure$(\delta(q, r), b)$

$= \varepsilon$ - closure $(\delta(q, b) \cup \delta(r, b))$

$= \varepsilon$ - closure $(r \cup \phi)$

$= \varepsilon$ - closure (r)

$= \{q, r\}$ i.e. state C.

$\therefore \quad \delta(C, b) = C$

$\delta(C, c) = \varepsilon$ - closure $(\delta(C, c))$

$= \varepsilon$ - closure$(\delta(q, r), c)$

$= \varepsilon$ - closure $(\delta(q, c) \cup \delta(r, c))$

$= \varepsilon$ - closure $(\phi \cup p)$

$= \varepsilon$ - closure (p)

$= \{p\}$ i.e. state A.

$\therefore \quad \delta(C, c) = A$

State D = {p, q, r}

$\delta(D, a) = \varepsilon$ - closure $(\delta(D, a))$

$= \varepsilon$ - closure$\big(\delta(p, q, r), a\big)$

$= \varepsilon$ - closure $\big(\delta(p, a) \cup \delta(q, a) \cup \delta(r, a)\big)$

$= \varepsilon$ - closure $(p \cup q \cup r)$

$= \varepsilon$ - closure $(p) \cup \varepsilon$ - closure$(q) \cup \varepsilon$ - closure(r)

$= \{p, q, r\}$ i.e. state D.

$\therefore \quad \delta(D, a) = D$

$\delta(D, b) = \varepsilon$ - closure $(\delta(D, b))$

$= \varepsilon$ - closure$\big(\delta(p, q, r), b\big)$

$= \varepsilon$ - closure $\big(\delta(p, b) \cup \delta(q, b) \cup \delta(r, b)\big)$

$= \varepsilon$ - closure $(q \cup r \cup \phi)$

$= \varepsilon$ - closure(q, r)

$= \varepsilon$ - closure $(q) \cup \varepsilon$ - closure(r)

$= \{p, q, r\}$ i.e. state D.

$\therefore \quad \delta(D, b) = D$

$\delta(D, c) = \varepsilon$ - closure $(\delta(D, c))$

$= \varepsilon$ - closure$\big(\delta(p, q, r), c\big)$

$= \varepsilon$ - closure $\big(\delta(p, c) \cup \delta(q, c) \cup \delta(r, c)\big)$

$= \varepsilon$ - closure $(r \cup \phi \cup p)$

$= \varepsilon$ - closure $(r) \cup \varepsilon$ - closure(p)

$= \{q, r\} \cup \{p\}$

$= \{p, q, r\}$ i.e. state D.

$\therefore \quad \delta(D, c) = D$

The transition table from above calculations can be obtained as

State \ Input	a	b	c
→ A	A	B	C
B	B	D	C
(C)	D	C	A
(D)	D	D	D

As state A = {p} it is a start state and states C and D contain final state r, hence these are final states. The transition diagram for the DFA is

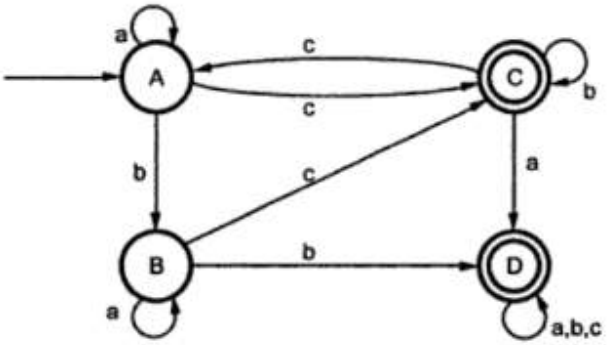


Fig. 2.37

➠ **Example 2.19 :** *Construct DFA for the NFA - ε Fig. 2.38 shown below.*

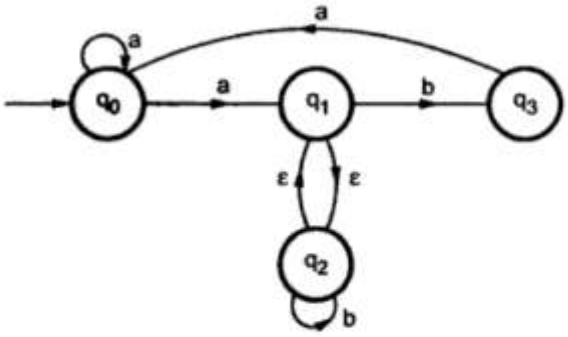


Fig. 2.38

Solution : We will first write ε - closure of start state q_0.

ε - closure (q_0) = $\{q_0\}$ call it as state A.

Now obtain δ transitions for input a and b for state A.

$$\delta(A, a) = \varepsilon \text{ - closure } (\delta(A, a))$$
$$= \varepsilon \text{ - closure } (\delta(q_0, a))$$
$$= \varepsilon \text{ - closure } (q_0, q_1)$$
$$= \varepsilon \text{ - closure } (q_0) \cup \varepsilon \text{ - closure } (q_1)$$
$$= \{q_0\} \cup \{q_1, q_2\}$$
$$= \{q_0, q_1, q_2\} \rightarrow \text{call it as state B.}$$
$$\delta(A, b) = \varepsilon \text{ - closure } (\delta(A, b))$$
$$= \varepsilon \text{ - closure } (\delta(q_0, b))$$
$$= \varepsilon \text{ - closure } (\phi)$$
$$= \phi$$

Now consider state B = $\{q_0, q_1, q_2\}$

$$\delta(B, a) = \varepsilon \text{ - closure } \{\delta(B, a)\}$$
$$= \varepsilon \text{ - closure } \{\delta\{q_0, q_1, q_2\}, a\}$$
$$= \varepsilon \text{ - closure } \{\delta\{q_0, a\} \cup \delta\{q_1, a\} \cup \delta\{q_2, a\}\}$$
$$= \varepsilon \text{ - closure } \{\{q_0, q_1\} \cup \phi \cup \phi\}$$
$$= \{q_0, q_1, q_2\} \text{ i.e. state B.}$$
$$= B$$
$$\delta(B, b) = \varepsilon \text{ - closure } \{\delta(B, b)\}$$
$$= \varepsilon \text{ - closure } \{\delta(\{q_0, q_1, q_2\}, b)\}$$
$$= \varepsilon \text{ - closure } \{\delta\{q_0, b\} \cup \delta\{q_1, b\} \cup \delta\{q_2, b\}\}$$
$$= \varepsilon \text{ - closure } \{\phi \cup \{q_3\} \cup \{q_2\}\}$$
$$= \varepsilon \text{ - closure } \{\{q_2, q_3\}\}$$
$$= \varepsilon \text{ - closure } \{q_2\} \cup \varepsilon \text{ - closure } \{q_3\}$$
$$= \{q_2\} \cup \{q_3\}$$
$$= \{q_2, q_3\} \rightarrow \text{call it as state C.}$$

The δ transitions on C are obtained as -

$$\delta(C, a) = \varepsilon \text{ - closure } \{\delta(C, a)\}$$

$= \varepsilon$ - closure $\{\delta((q_2, q_3), a)\}$

$= \varepsilon$ - closure $\{\delta(q_2, a) \cup \delta(q_3, a)\}$

$= \varepsilon$ - closure $\{\phi \cup \{q_0\}\}$

$= \varepsilon$ - closure $\{q_0\}$

$= \{q_0\}$ i.e. A

$\delta(C, b) = \varepsilon$ - closure $\{\delta(C, b)\}$

$= \varepsilon$ - closure $\{\delta\{q_2, q_3\}, b\}$

$= \varepsilon$ - closure $\{\delta(q_2, b) \cup \delta(q_3, b)\}$

$= \varepsilon$ - closure $\{q_2 \cup \phi\}$

$= \varepsilon$ - closure $\{q_2\}$

$= \{q_1, q_2\} \rightarrow$ call it as state D.

$\delta(D, a) = \varepsilon$ - closure $\{\delta(D, a)\}$

$= \varepsilon$ - closure $\{\delta(\{q_1, q_2\}, a)\}$

$= \varepsilon$ - closure $\{\delta(q_1, a) \cup \delta(q_2, a)\}$

$= \varepsilon$ - closure $\{\phi \cup \phi\}$

$= \phi$

$\delta(D, b) = \varepsilon$ - closure $\{\delta(\{q_1, q_2\}, b)\}$

$= \varepsilon$ - closure $\{\delta(q_1, b) \cup \delta(q_2, b)\}$

$= \varepsilon$ - closure $\{q_3 \cup q_2\}$

$= \varepsilon$ - closure $\{q_2, q_3\}$

$= \varepsilon$ - closure $\{q_2\} \cup \varepsilon$ - closure $\{q_3\}$

$= \{q_2, q_3\}$ i.e. C.

$= C$

Now no new state is generated. Hence we will write transition table for above computations.

State \ Input	a	b
→ A	B	ϕ
B	B	C
C	A	D
D	ϕ	C

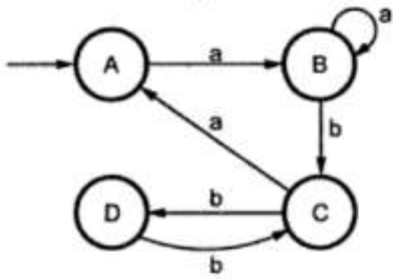
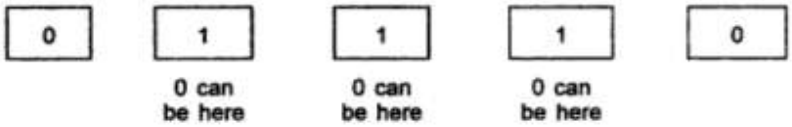


Fig. 2.39

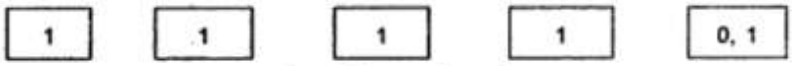
Example 2.20 : *Develop a deterministic Finite Automata accepting the language given over the alphabet {0, 1}. L = {The set of all strings such that every block of five consecutive contains at least two 0's}.*

Ans. : We consider the five consecutive symbols as



0 can be here | 0 can be here | 0 can be here

Similarly,

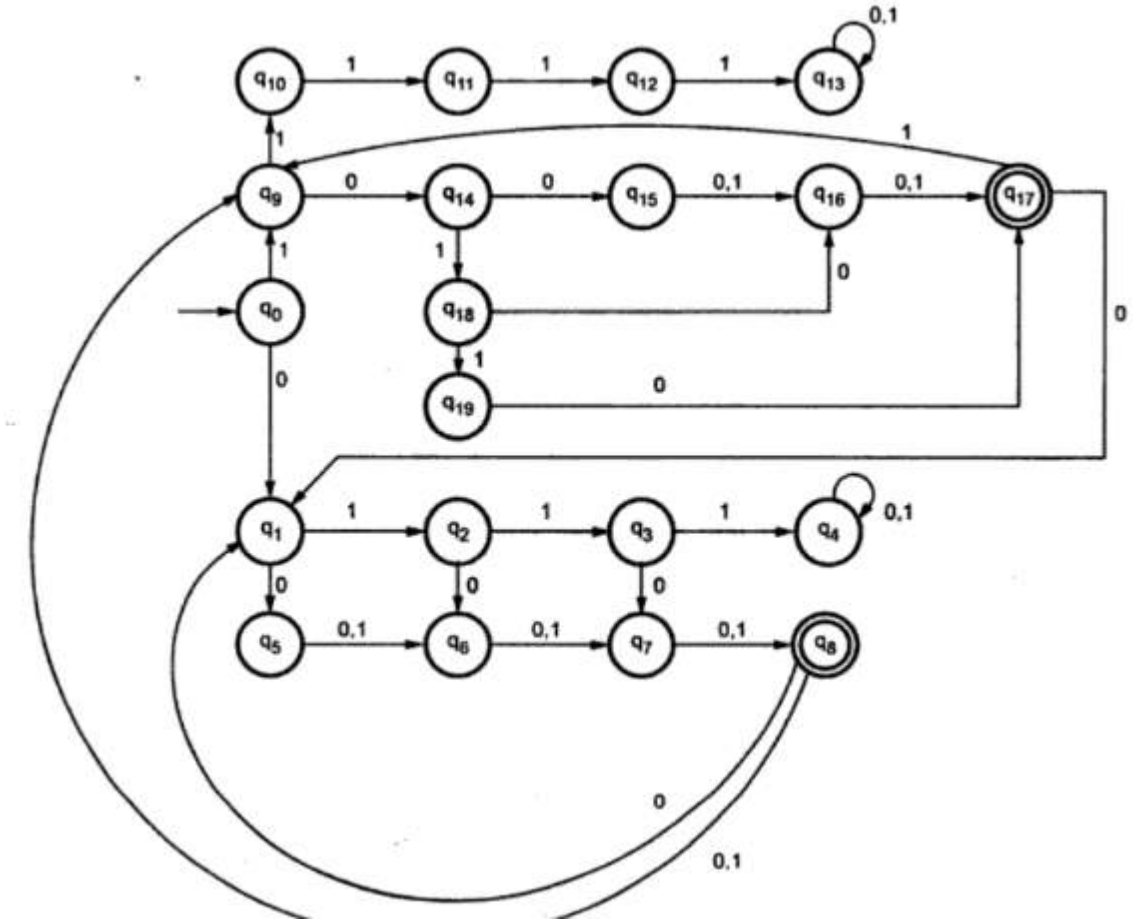


is totally not allowed

But,

0 can be here | 0 can be here | 0 can be here

Thus by considering various combinations of 0's and 1's over a block of 5 symbols we can draw DFA as follows -



Thus we avoid 4 or 5 1's in the strings. Not having 4 or 5 1's means allowing at least two 0's in the block of 5 symbols. With this simple logic the DFA can be constructed.

3.5 Minimization of FSM

The minimization of FSM means reducing the number of states from given FA. Thus we get the FSM with redundant states after minimizing the FSM.

While minimizing FSM we first find out which two states are equivalent we can represent those two states by one representative state.

The two states q_1 and q_2 are equivalent if both $\delta(q_1, x)$ and $\delta(q_2, x)$ are final states or both of them are nonfinal states for all $x \in \Sigma^*$ (Σ^* indicate any string of any length) we can minimize the given FSM by finding equivalent states.

Method for Construction of Minimum State Automata :

Step 1 : We will create a set Π_0 as $\Pi_0 = \{ Q_1^0, Q_2^0 \}$ where Q_1^0 is set of all final states and $Q_2^0 = Q - Q_1^0$ where Q is a set of all the states in DFA.

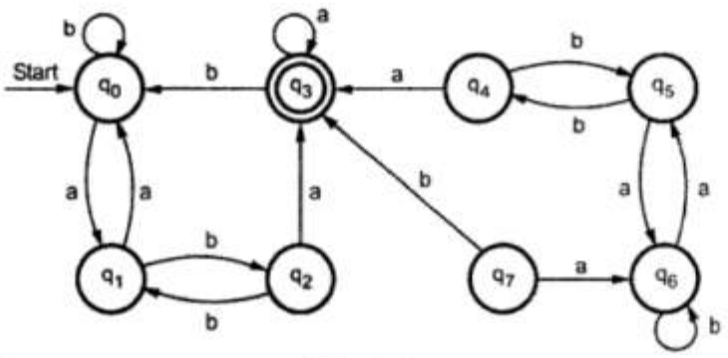
Step 2 : Now we will construct Π_{K+1} from Π_K. Let Q_i^K be any subset in Π_K. If q_1 and q_2 are in Q_i^K they are (K + 1) equivalent provided $\delta(q_1, a)$ and $\delta(q_2, a)$ are K equivalent. Find out whether $\delta(q_1, a)$ and $\delta(q_2, a)$ are residing in the same equivalence class Π_K. Then it is said that q_1 and q_2 are (K + 1) equivalent. Thus Q_i^K is further divided into (K + 1) equivalence classes. Repeat step 2 for every Q_i^K in Π_K and obtain all the elements of Π_{K+1}.

Step 3 : Construct Π_n for n = 1, 2, ... until $\Pi_n = \Pi_{n+1}$.

Step 4 : Then replace all the equivalent states in one equivalence class by representative state. This helps in minimizing the given DFA.

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

Example 3.7 : *Construct the minimum state automaton for the following transition diagram.*



Solution : We will first construct a transition table for the given DFA.

We will start constructing equivalence classes.

State \ Input	a	b
→ q_0	q_1	q_0
q_1	q_0	q_2
q_2	q_3	q_1
(*) q_3	q_3	q_0
q_4	q_3	q_5
q_5	q_6	q_4
q_6	q_5	q_6
q_7	q_6	q_3

There is only one final state i.e. q_3. Hence we can partition Q as Q_1^0 and Q_2^0.

$$Q_1^0 = F = \{q_3\} \text{ As } \{q_3\} \text{ cannot be partitioned further.}$$

$$Q_2^0 = Q - Q_1^0$$

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

Now, we will compare q_0 with q_2.

	a	b
q_0	q_1	q_0
q_2	q_3	q_1

Under a - column of q_0 and q_2 we get q_1 and q_3 respectively. But q_1 and q_3 lie in different states. Hence they are not 1 - equivalent.

Similarly consider q_0 and q_4.

	a	b
q_0	q_1	q_0
q_4	q_3	q_5

Under a - column of q_0 and q_4 we get q_1 and q_3 which lie in different sets. Hence q_0 is not 1 - equivalent to q_4. Similarly q_0 is not 1 - equivalent to q_7 (observe b-column of q_0 and q_7). But q_0 is 1 - equivalent to q_1, q_5 and q_6.

$$Q_1' = \{q_3\}$$

$$Q_2' = \{q_0, q_1, q_5, q_6\}$$

q_2 is 1 - equivalent to q_4. Hence

$$Q_3' = \{q_2, q_4\}$$

Since under a-column we get q_3 only for q_2 and q_4. And under b - column of q_2 and q_4 we get q_1 and q_5. But q_1 and q_5 lie in one set. Thus q_2 and q_4 are 1 - equivalent.

The only element left over in Q_2^0 is q_7.

$\therefore$ $Q_4' = \{q_7\}$ The equivalence class is -

$$\Pi_1 = \{\{q_3\}, \{q_0, q_1, q_5, q_6\}, \{q_2, q_4\}, \{q_7\}\}$$

$$Q_1^2 = \{q_3\}$$

Now q_0 is 2 - equivalent to q_6 because

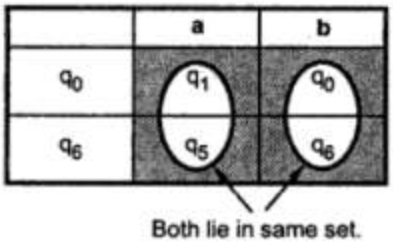


Fig. 3.5 (a)

But q_0 is not 2 - equivalent to q_1 and q_5.

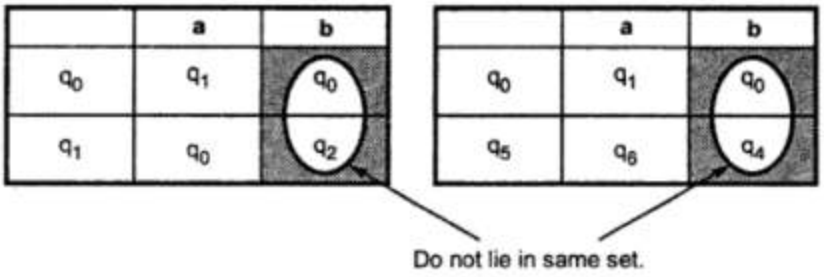


Fig. 3.5 (b)

Hence $Q_2^2 = \{q_0, q_6\}$

But q_1 is 2 - equivalent to q_5.

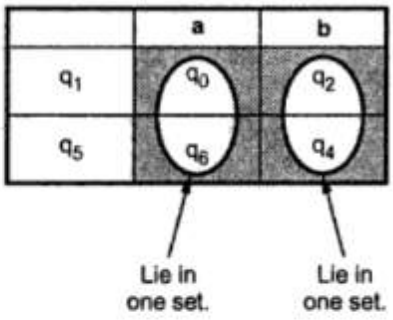


Fig. 3.5 (c)

Hence $Q_3^2 = \{q_1, q_5\}$

Similarly q_2 is 2 - equivalent to q_4.

$Q_4^2 = \{q_2, q_4\}$

$Q_5^2 = \{q_7\}$

Thus, $\boxed{\Pi_2 = \{\{q_3\}, \{q_0, q_6\}, \{q_1, q_5\}, \{q_2, q_4\}, \{q_7\}\}}$

$$Q_1^3 = \{q_3\}$$

As q_0 is 3 - equivalent to q_6,

$$Q_2^3 = \{q_0, q_6\}$$

As q_1 is 3 - equivalent to q_5,

$$Q_3^3 = \{q_1, q_5\}$$

As q_2 is 3 - equivalent to q_4,

$$Q_4^3 = \{q_2, q_4\}$$

and $$Q_5^3 = \{q_7\}$$

$\therefore$ $\boxed{\Pi_3 = \{\{q_3\}, \{q_0, q_6\}, \{q_1, q_5\}, \{q_2, q_4\}, \{q_7\}\}}$

As $\Pi_3 = \Pi_2$ we have got equivalence class from Π_2. Hence we can write a transition table as

	State \ Input	a	b
→	$[q_0, q_6]$	$[q_1, q_5]$	$[q_0, q_6]$
	$[q_1, q_5]$	$[q_0, q_6]$	$[q_2, q_4]$
	$[q_2, q_4]$	$[q_3]$	$[q_1, q_5]$
*	$[q_3]$	$[q_3]$	$[q_0, q_6]$
	$[q_7]$	$[q_0, q_6]$	$[q_3]$

The transition diagram with minimized states is -

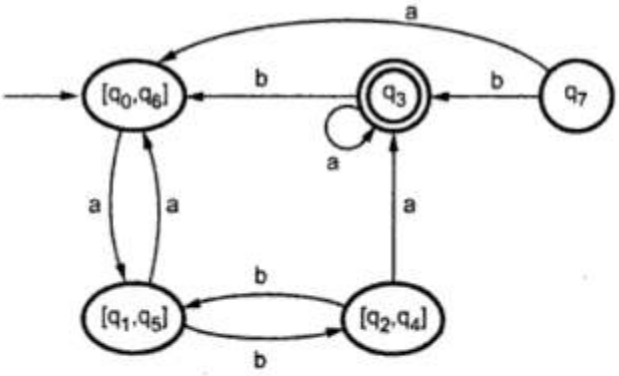


Fig. 3.6

Example 3.8 : *Construct a minimum state DFA for the following given automata -*

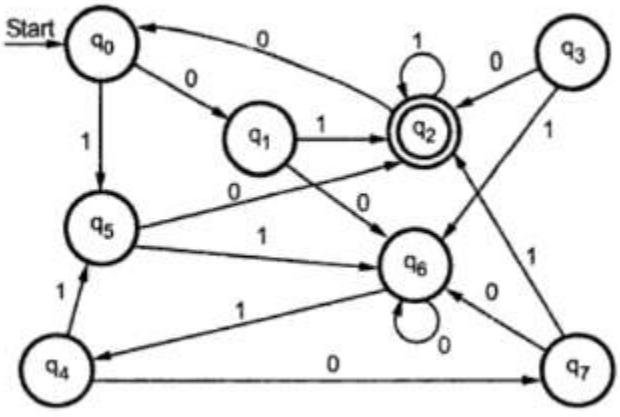


Fig. 3.7

Solution : We will first construct a transition table from given FA.

	State \ Input	0	1
→	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

Now we will first obtain Q_1^0.

$$Q_1^0 = F = \{q_2\} \rightarrow \text{Final state}$$

$$Q_2^0 = Q - Q_1^0$$

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

Hence $\Pi_0 = \{\{q_2\}\ \{q_0, q_1, q_3, q_4, q_5, q_6, q_7\}\}$

We cannot partition $\{q_2\}$ further. Hence $Q_1' = \{q_2\}$.

Now consider another set from Π_0 i.e. $\{q_0, q_1, q_3, q_4, q_5, q_6, q_7\}$. We will now compare input entries of q_0 for all remaining states in this set. Consider

State \ Input	0	1
q_0	q_1	q_5
q_1	q_6	q_2

Here $q_5 \in Q_2^0$ and $q_2 \in Q_1^0$.
Hence they are not 1-equivalent.

Fig. 3.8 (a)

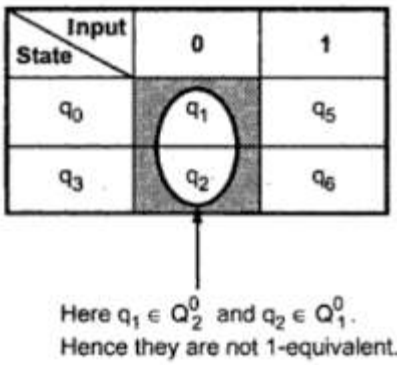


Fig. 3.8 (b)

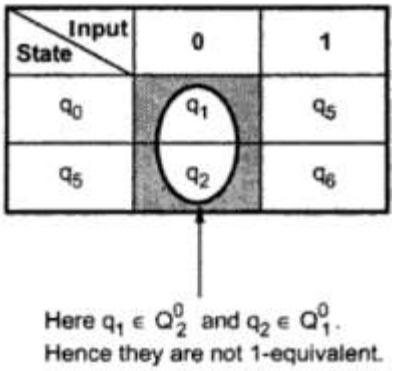


Fig. 3.8 (c)

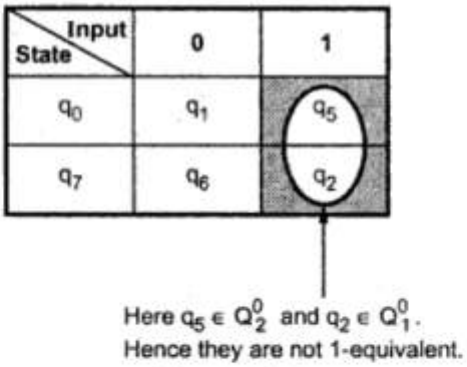


Fig. 3.8 (d)

Thus q_0 is not ρ - equivalent to q_3, q_5 and q_7. Now we will obtain the set of equivalent sets. Consider q_0 with q_4, q_6.

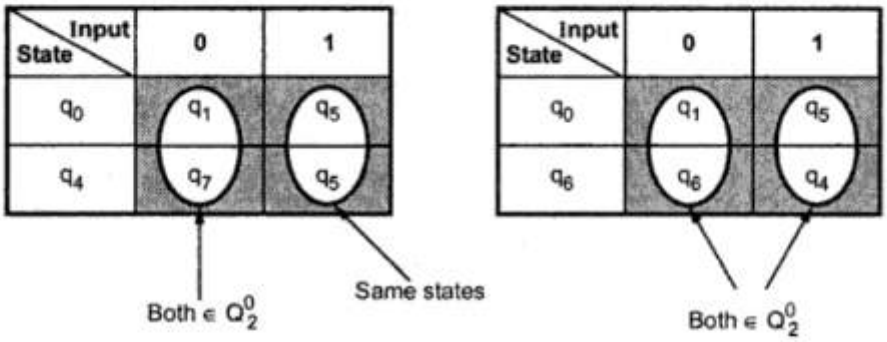


Fig. 3.8 (e)

Hence $\{q_0, q_4, q_6\}$ is one subset in Π_1.

Therefore $Q'_2 = \{q_0, q_4, q_6\}$.

We will consider a subset $\{q_1, q_3, q_5, q_7\}$. Now we will find the 1 - equivalent subset from this subset. Hence we need to compare q_1 with q_3, q_5 and q_7 respectively.

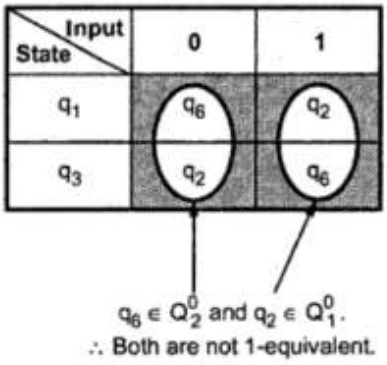


Fig. 3.8 (f)

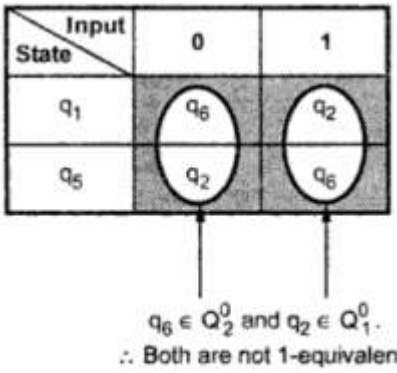


Fig. 3.8 (g)

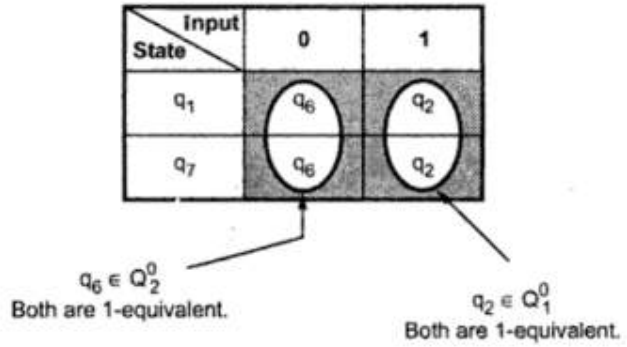


Fig. 3.8 (h)

i.e. q_1 is 1 - equivalent to q_7. Hence $\{q_1, q_7\}$ will be one subset. We cannot partition this subset further. Hence $Q'_3 = \{q_1, q_7\}$.

Now we will compare q_3 with q_5.

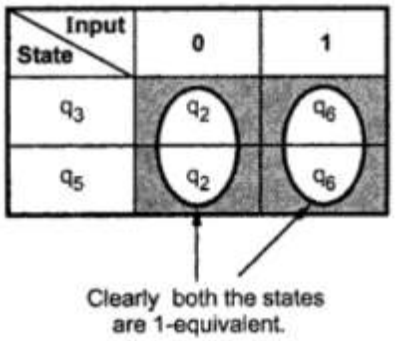


Fig. 3.8 (i)

$\therefore \quad Q'_4 = \{q_3, q_5\}$

Now,

$$\Pi_2 = \{\{q_2\}, \{q_0, q_4, q_6\}, \{q_1, q_7\}, \{q_3, q_5\}\}$$

From Π_2 we can consider a subset $\{q_0, q_4, q_6\}$. We will again compare q_0 with q_4 and q_6.

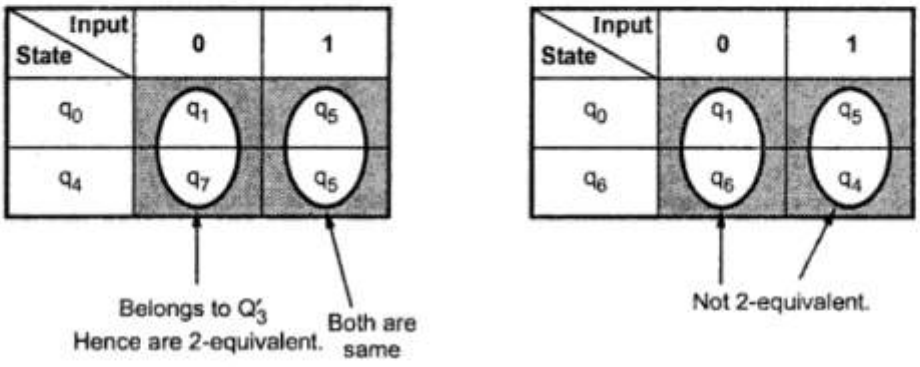


Fig. 3.8 (j)

Thus q_0 and q_4 and 2 - equivalent and q_0, q_6 are not 2 - equivalent.

Similarly q_3 and q_5 are 2 - equivalent.

Also q_1 and q_7 are 2 - equivalent.

Thus $\Pi_2 = \{\{q_2\}, \{q_0, q_4\}, \{q_6\}, \{q_1, q_7\}, \{q_3, q_5\}\}$

Now, q_0 and q_4 are 3 - equivalent.

q_1 and q_7 are 3 - equivalent.

q_3 and q_5 are 3 - equivalent.

As $\Pi_2 = \Pi_3$. The Π_2 gives equivalence classes. Therefore we can construct a finite automata with minimized state as

$$M' = (Q', \{0, 1\}, \delta, q_0', F)$$

where Q' is a set of states in FA.

$$\text{i.e. } = \{[q_2], [q_0, q_4], [q_6], [q_1, q_7], [q_3, q_5]\}$$

q_0' is initial state i.e. $[q_0, q_4]$

F' is a set of final states i.e. $[q_2]$

The δ can be shown by following transition table.

State \ Input	0	1
$[q_0, q_4]$	$[q_1, q_7]$	$[q_3, q_5]$
$[q_1, q_7]$	$[q_6]$	$[q_2]$
$[q_2]$	$[q_0, q_4]$	$[q_2]$
$[q_3, q_5]$	$[q_2]$	$[q_6]$
$[q_6]$	$[q_6]$	$[q_0, q_4]$

Here q_0 and q_4, q_1 and q_7, q_3 and q_5 are treated as one state. Now we can draw the transition diagram for minimized machine as -

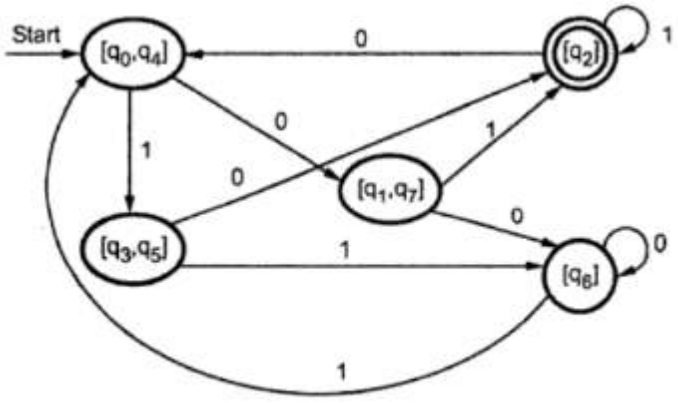


Fig. 3.9

Example 3.9 : *Construct a minimum state automaton for following DFA.*

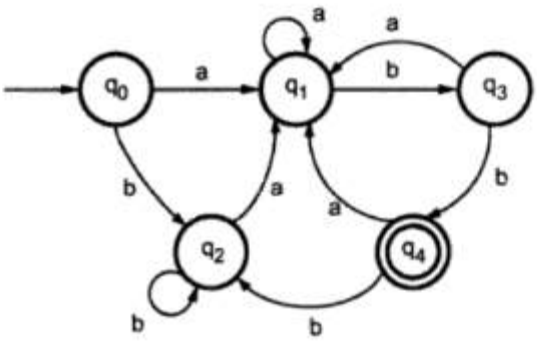


Fig. 3.10

Solution : It will be easier if we construct a transition table for given finite automata.

State \ Input	a	b
→ q_0	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	q_4
(q_4)	q_1	q_2

Now we will first obtain Q_1^0.

$\therefore$ $Q_1^0 = F = \{q_4\} \rightarrow$ Final state

$Q_2^0 = Q - Q_1^0$.

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

Hence $\Pi_0 = \{\{q_4\}, \{q_0, q_1, q_2, q_3\}\}$

As we cannot partition $Q_1^0 = \{q_4\}$ further we say $Q_1' = \{q_4\}$.

Now consider subset $\{q_0, q_1, q_2, q_3\}$. We will compare q_0 with q_1, q_2 and q_3.

State \ Input	a	b
q_0	q_1	q_2
q_1	q_1	q_3

Both are 1-equivalent.

(a)

State \ Input	a	b
q_0	q_1	q_2
q_2	q_1	q_2

Clearly q_0 and q_2 are equivalent.

(b)

Input / State	a	b
q_0	q_1	q_2
q_3	q_1	q_4

$q_2 \in Q_2^0$ and

$q_4 \in Q_1^0$ Hence both are not 1-equivalent.

(c)

Fig. 3.11

$$Q'_2 = \{q_0, q_1, q_2\}$$

$$Q'_3 = \{q_3\}$$

Hence, $$\Pi_1 = \{\{q_4\}, \{q_0, q_1, q_2\}, \{q_3\}\}$$

Now we will compare q_0 with q_1 and q_2 respectively.

Input / State	a	b
q_0	q_1	q_2
q_1	q_1	q_3

As $q_2 \in Q'_2$ and $q_3 \in Q'_3$ Both are not 2-equivalent

Fig. 3.11 (d)

Input / State	a	b
q_0	q_1	q_2
q_2	q_1	q_2

Clearly both the states are equivalent.

Fig. 3.11 (e)

Hence we get, $\Pi_2 = \{\{q_4\}, \{q_0, q_2\}, \{q_1\}, \{q_3\}\}$

We can now state $\{q_0, q_2\}$ are 3 - equivalent. And we cannot partition $\{q_1\}$ and $\{q_3\}$ further. Hence

$$\Pi_3 = \{\{q_4\}, \{q_0, q_2\}, \{q_1\}, \{q_3\}\}$$

As $\Pi_3 = \Pi_2$ and Π_2 gives equivalence classes we can construct minimum state finite automata as -

State \ Input	a	b
→ $[q_0, q_2]$	q_1	$[q_0, q_2]$
q_1	$[q_1]$	$[q_3]$
q_3	$[q_1]$	$[q_4]$
(*q_4*)	$[q_1]$	$[q_0, q_2]$

The transition diagram can be -

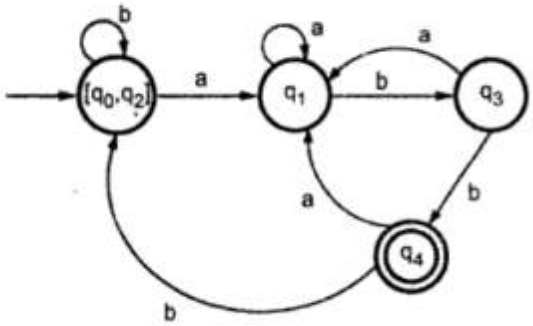


Fig. 3.12

Minimization of DFA (Alternate Method)

This method helps in finding equivalent states in one stroke. Hence it is an efficient method of minimization of given DFA. Let us understand this method with the help of some example.

Example 3.10 : *Minimize the DFA as given below.*

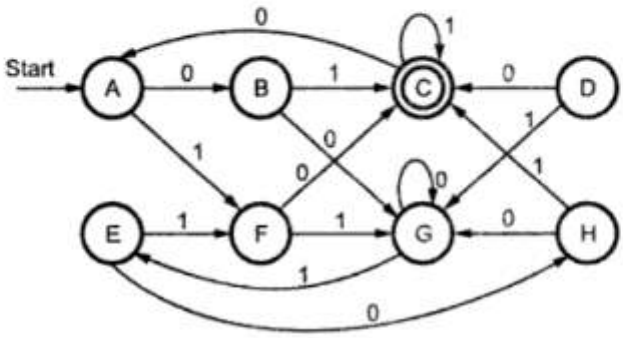
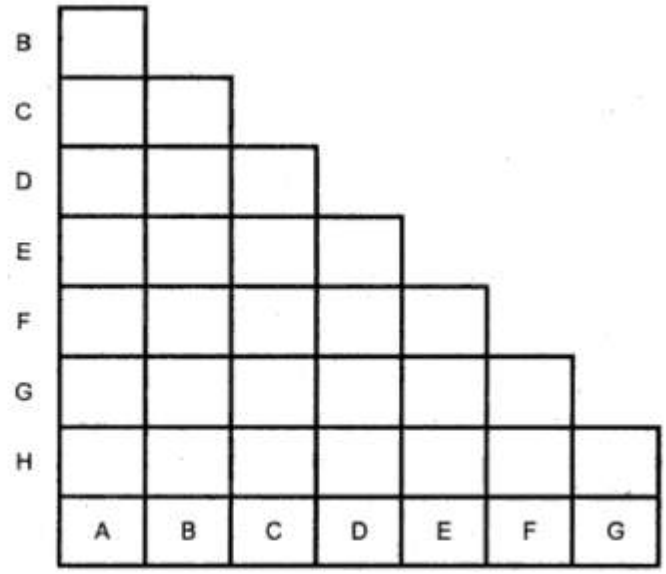


Fig. 3.13

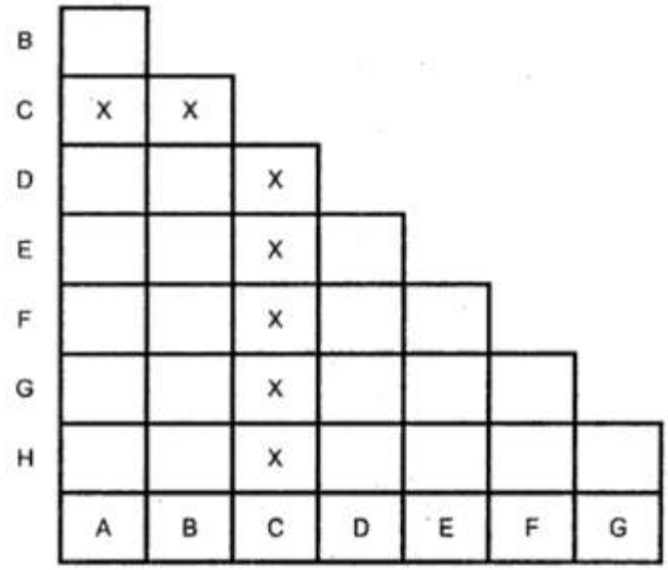
Solution : We will build the transitions table for given DFA as follows.

	0	1
→ A	B	F
B	G	C
C	A	C
D	C	G
E	H	F
F	C	G
G	G	E
H	G	C

We will now construct a table for each pair of states.



We will mark X for all the final and nonfinal states. Hence



Thus we have marked X for (A, C), (B, C), (C, D), (C, E), (C, F), (C, G), (C, H).

Now we will consider every pair form above table. Consider pair (G, H).

δ (G, 0) = G, δ (G, 1) = E

δ (H, 0) = G, δ (H, 1) = C

As for δ (G, 1) = E and δ (H, 1) = C. We can see X in (C, E) pair. Hence pair (G, H) is not equivalent.

Hence we will mark X in pair (G, H). Thus we will find the equivalent pairs.

Consider pair (B, H).

δ (B, 0) = G, δ (B, 1) = C

δ (H, 0) = G, δ (H, 1) = C

Thus the pair (B, H) is equivalent. Thus we obtain,

B	X						
C	X	X					
D	X	X	X				
E		X	X	X			
F	X	X	X		X		
G	X	X	X	X	X	X	
H	X		X	X	X	X	X
	A	B	C	D	E	F	G

Thus we get equivalent pairs as (A, E), (B, H), (D, F). Hence the minimized DFA will be,

	0	1
→ A	B	D
B	G	C
(C)	A	C
D	C	G
G	G	A

Example 3.11 : *Minimize the finite automata in Fig. 3.14 below and show both the given and the reduced one are equivalent.*

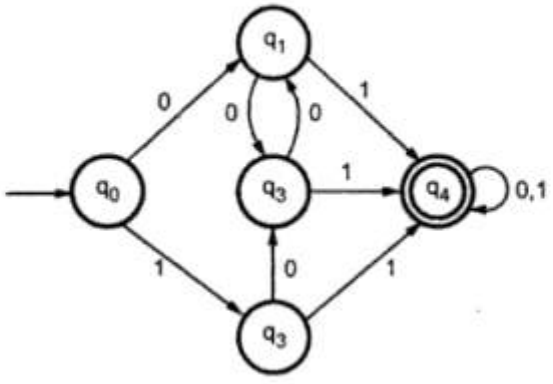


Fig. 3.14

Solution : We will first construct the transition table for given DFA -

State \ Input	0	1
→ q_0	q_1	q_3
q_1	q_2	q_4
q_2	q_1	q_4
q_3	q_2	q_4
(q_4)	q_4	q_4

Now we will partition the states Q in Q_1^0 and Q_2^0. The state q_4 is a final state. Hence

$$Q_1^0 = F = \{q_4\}$$
$$Q_2^0 = Q - Q_1^0 = \{q_0, q_1, q_2, q_3\}$$
$$\Pi_0 = \{\{q_4\}, \{q_0, q_1, q_2, q_3\}\}$$

Now we will compare q_0 with q_1. We find that under 0 column we get q_1 and q_2. And q_1 and q_2 lie in the same set i.e. Q_2^0. But under 1-column of q_0 and q_1 we get q_3, q_4 respectively. But $q_3 \in Q_2^0$ and $q_4 \in Q_1^0$. Hence they are not 1-equivalent. Similarly compare q_0 and q_2. We will find that under 0 column of q_0 and q_2 we get q_1 only. But under 1-column of q_0 and q_2 we get q_3 and q_4. The $q_3 \in Q_2^0$ and $q_4 \in Q_1^0$. Hence q_0 and q_2 are not 1-equivalent. Similarly q_0 is not 1-equivalent to q_3 because of 1-column entries.

$$Q_1' = \{q_4\}$$
$$Q_2' = \{q_0\}$$
$$Q_3' = \{q_1, q_2, q_3\}$$

Hence,

$$\Pi_1 = \{\{q_4\}, \{q_0\}, \{q_1, q_2, q_3\}\}$$

Now in set $\{q_1, q_2, q_3\}$ we compare q_1 with q_2 under 0-column we get q_2 and q_1 which are in the same set, and under 1-column we get q_4. And for state q_1 and q_3 we get same entries under 0 - column and under 1-column. Hence we can group them $\{q_2, q_3\}$.

The equivalence class becomes

$$\Pi_2 = \{\{q_4\}, \{q_0\}, \{q_1\}, \{q_2, q_3\}\}$$

Now further we cannot partition any of the set. We get

$$\Pi_3 = \{\{q_0\}, \{q_2\}, \{q_4\}, \{q_1, q_3\}\}$$

Hence minimized DFA is -

State \ Input	0	1
→ q_0	q_1	q_1
q_1	q_2	q_4
q_2	q_1	q_4
q_4	q_4	q_4

The transition diagram will be -

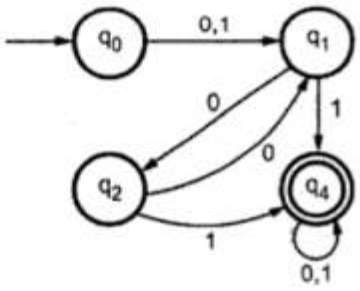


Fig. 3.15

3.4 Equivalence between Two FSMs

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 pairwise 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 FAs.

Example 3.5 :*Consider the DFAs given below. Are they equivalent ?*

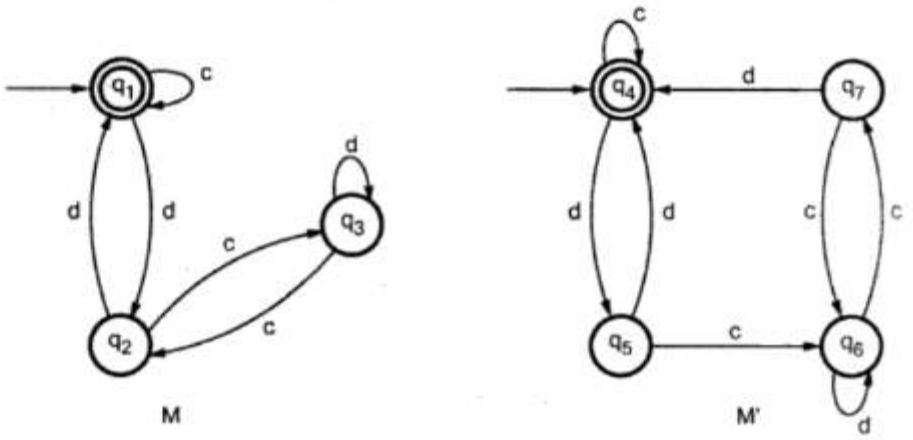


Fig. 3.2

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 states obtained in pair (q_1, q_4). Both q_2 and q_5 are nonfinal 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 nonfinal state in a pair. Hence we declare that two DFAs are equivalent.

Example 3.6 : *Following are two FAs check whether they are equivalent or not.*

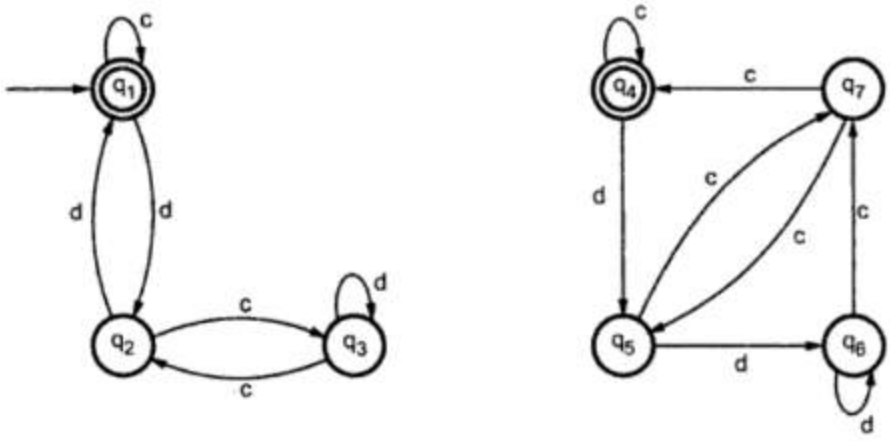


Fig. 3.3

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 nonfinal state. As per equivalence rule final and nonfinal 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 -

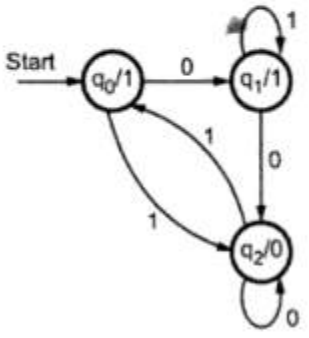


Fig. 2.45

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 :

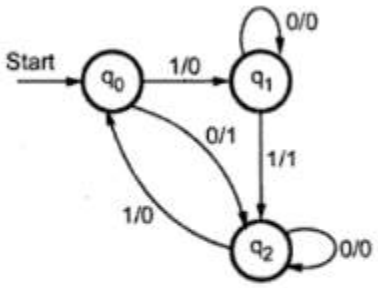


Fig. 2.46

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 2.22 : *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,

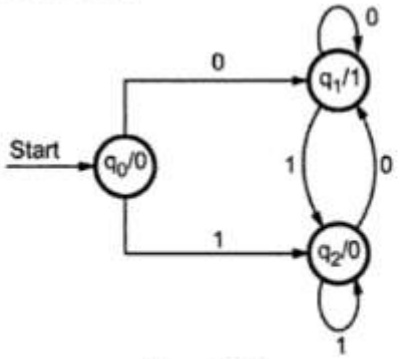


Fig. 2.47

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 -

	Current state	Next state (δ) 0	Next state (δ) 1	Output (λ)
→	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 2.23 : *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.

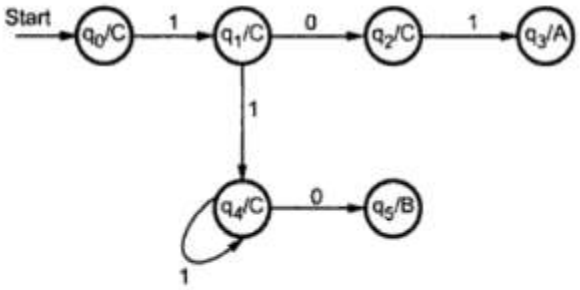


Fig. 2.48

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

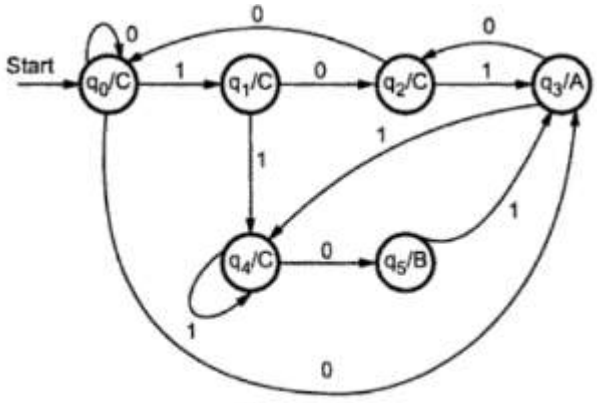


Fig. 2.49

Now the Mealy machine can be

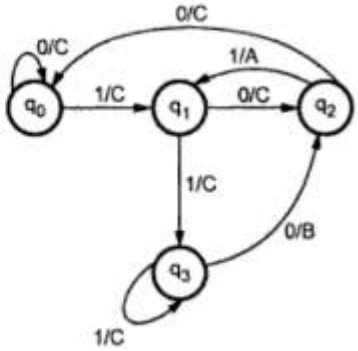


Fig. 2.50

Example 2.24 : *Design a Mealy machine to find 2's complement of a given binary number.*

Solution: For designing 2's complement of a binary number we assume that input is read from LSB to MSB. We will keep the binary number as it is until we read first 1. Keep that 1 as it is then change remaining 1's by 0's and 0's by 1's.

For example :

Let the binary number be

1011

←——

read from LSB

Keep the first 1 from LSB as it is and toggle the remaining bits we will get

0101

Thus 2's complement of 1011 is 0101. The required Mealy machine will be -

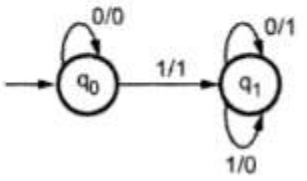


Fig. 2.51

Example 2.25 : *Design a Moore machine which will increment the given binary number by 1.*

Solution : We will read the binary number from LSB one bit at a time. We will replace each 1 by 0 until we get first 0. Once we get first 0 we will replace it by 1 and then keep remaining bits as it is.

For example :

1	0	1	1	
				← Read from LSB bit by bit
1	0	1	0	
			↑	
1	0	0	0	
		↑		
1	1	0	0	
	↑			
1	1	0	0	
↑				

Using this logic we can built a Mealy machine as follows -

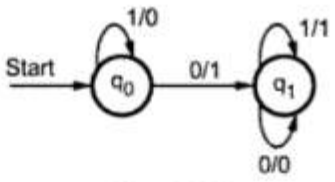


Fig. 2.52

Example 2.26 : *Give Moore and Mealy machine for* $\Sigma=\{0, 1, 2\}$, *print the residue modulo 5 of input treated as a ternary number.*

Solution : The ternary number is made up of 0,1 and 2. The interpretation of ternary number n can be -

i) If we write 0 after n then number becomes 3n.

ii) If we write 1 after n then number becomes 3n+1.

iii) If we write 2 after n then number becomes 3n+2.

For example :

If n = 4 then its value is $4\times3^0 = 4$.

If we write 0 after 4 i.e.

$$40 = 4\times3^1+0\times3^0 = 12 \qquad \text{i.e. } (3\times4)$$

If we write 1 after 4 then

$$41 = 4\times3^1+1\times3^0 = 13 \qquad \text{i.e. } 3n+1$$

If we write 2 after 4 we get

$$42 = 4\times3^1+2\times3^0 = 14 \qquad \text{i.e. } 3n+2$$

For residue modulo 5 we will get remainder 0, remainder 1, remainder 2, remainder 3 and remainder 4 values. Then we assume various states for these remainders as -

q_0 - remainder 0 state

q_1 - remainder 1 state

q_2 - remainder 2 state

q_3 - remainder 3 state

q_4 - remainder 4 state

Now consider n = 4,

For 4.0 we get decimal value 12 that means 12 % 5= 2 it gives remainder 2.

For 4.1 we get decimal value 13 that means 13 % 5= 3 it gives remainder 3.

Similarly 4.2 gives remainder 4.

n = 4 itself is remainder 4. Hence we can design,

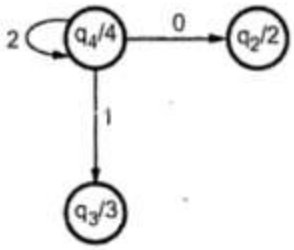


Fig. 2.53

Considering all possible cases we can design Moore machine as,

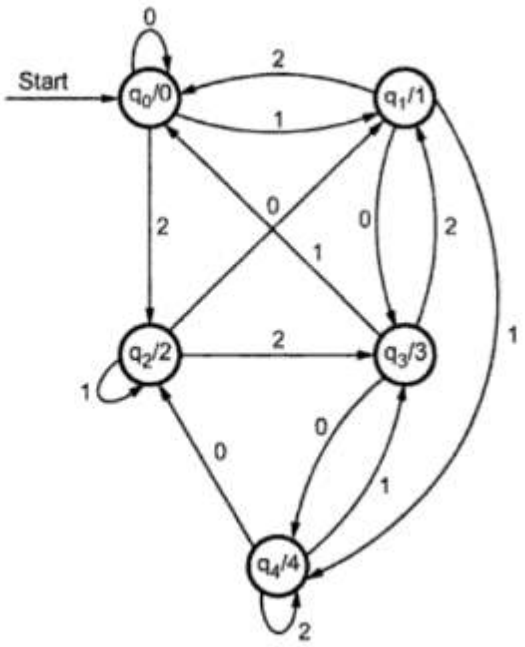


Fig. 2.54

Similarly the Mealy machine can be -

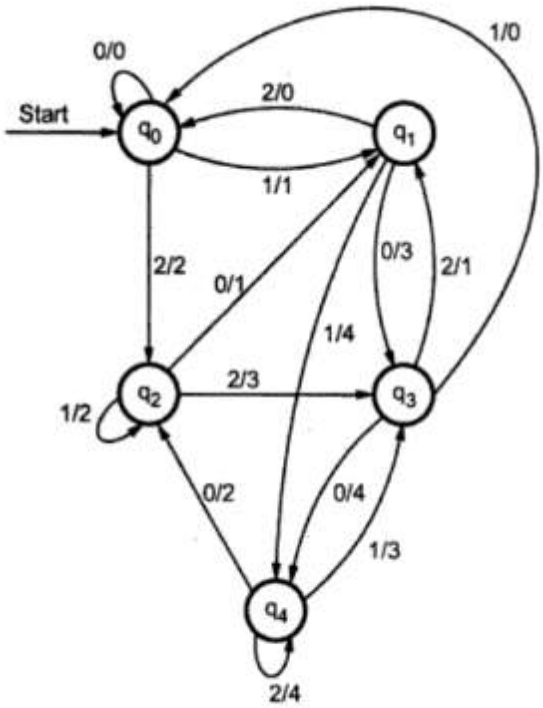


Fig. 2.55

Example 2.27 : *Design a Moore machine that will read sequences made up of letters A, E, I, O, U and will give as output the same sequences except that in case where an I directly follows an E, it will be changed to u. Design the Mealy machine for the same.*

Solution : We will assume a separate state for each alphabet. The output of that state will be corresponding letter.

For instance : For alphabet A the state will be q_0 and output of q_0 will be A. For alphabet E the state will be q_1 and output of q_1 will be E. Continuing in this fashion we will have q_0, q_1, q_2, q_3 and q_4 states. The start state will be q_0 we will take care of one thing and that is when I comes immediately after E it will lead to the state q_4 because output of state q_4 is u. With this logic the corresponding Moore machine will be -

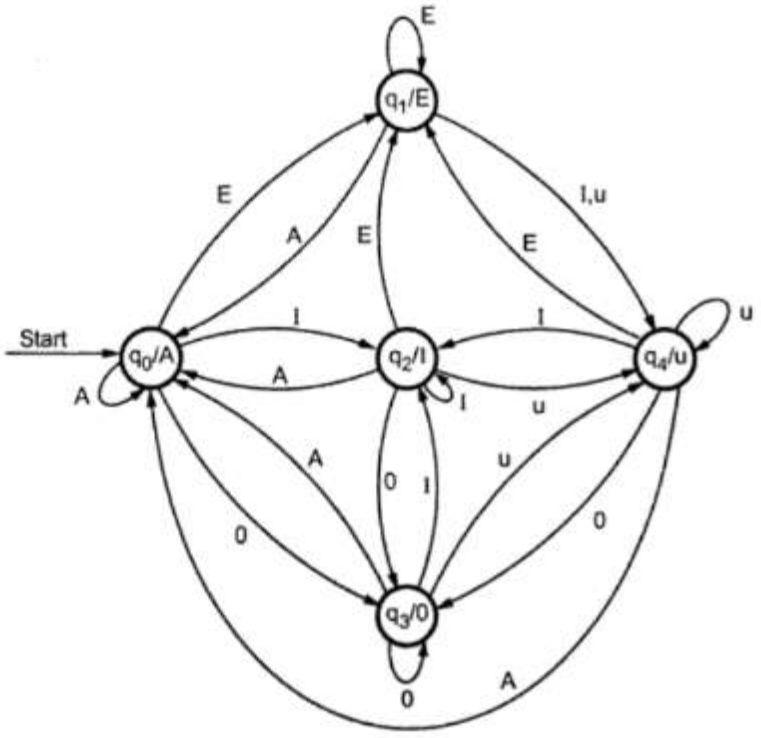


Fig. 2.56

Now we will draw the Mealy machine for the same problem.

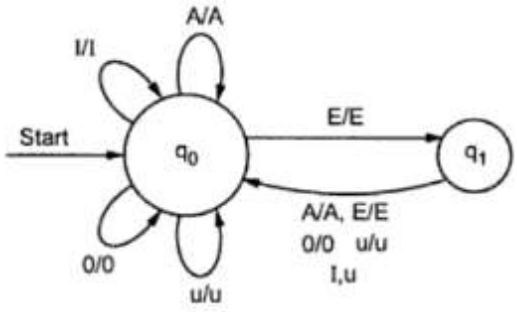


Fig. 2.57

Here the Mealy machine has the output along the edge itself we have got only two state. The transition changes from q_0 to q_1 when we get E. The I which is immediately following E will have the output u. Consider the string AIEOAEI will be given an output.

Example 2.28 : *Construct a Moore machine to determine residue mod 3 for binary number.*

Solution : This Moore machine is also called remainder 3 tester. In this machine we will get remainder 0, remainder 1 and remainder 2. To interpret the given binary number in its decimal value we consider n as a number if 0 is written after n then its value becomes 2n. If 1 is written after n then it's value becomes 2n + 1. For instance, if n = 0 then its decimal value is 0 then,

$$01 = 2n+1 = 1\times 0+1 = 1$$

$$011 = 2n+1 = (1\times 2)+1 = 3$$

Consider n = 1 its decimal value is 1. After 1 if 0 comes then its value will be

$$10 = 2n = 2\times 1 = 2$$

If 1 comes after 10 then its value becomes,

$$101 = 2n+1 = (2\times 2) + 1 = 5$$

With this logic we can construct a Moore machine with 3 states. q_0 is the start state and is considered as remainder 0 state. q_1 is considered to be remainder 1 state and q_2 is considered as remainder 2 state.

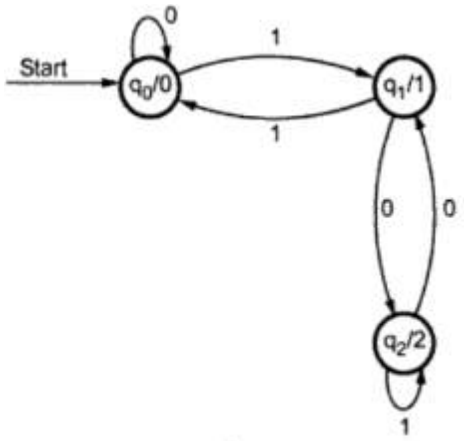


Fig. 2.58

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.

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 2.29 : *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 -

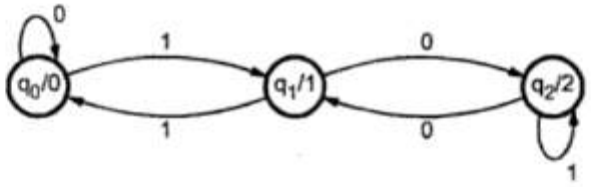


Fig. 2.59

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.

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

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

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

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

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

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

$$\lambda'(q_1,0) = \lambda(\delta(q_1,0))$$
$$= \lambda(q_2)$$
$$\boxed{\lambda'(q_1,0) = 2}$$
$$\lambda'(q_1,1) = \lambda(\delta(q_1,1))$$
$$= \lambda(q_0)$$
$$\boxed{\lambda'(q_1,1) = 0}$$
$$\lambda'(q_2,0) = \lambda(\delta(q_2,0))$$
$$= \lambda(q_1)$$
$$\boxed{\lambda'(q_2,0) = 1}$$
$$\lambda'(q_2,1) = \lambda(\delta(q_2,1))$$
$$= \lambda(q_2)$$
$$\boxed{\lambda'(q_2,1) = 2}$$

Hence the transition table can be drawn as follows –

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,

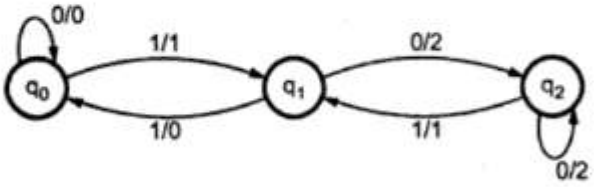


Fig. 2.60

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 2.30 : *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

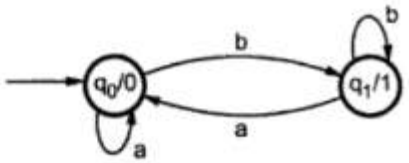


Fig. 2.61

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,

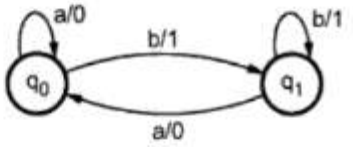


Fig. 2.62

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.

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 2.31 : *Convert the following Mealy machine into equivalent Moore machine.*

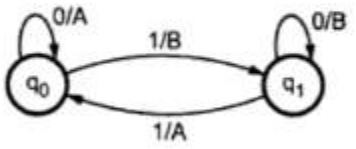


Fig. 2.63

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,

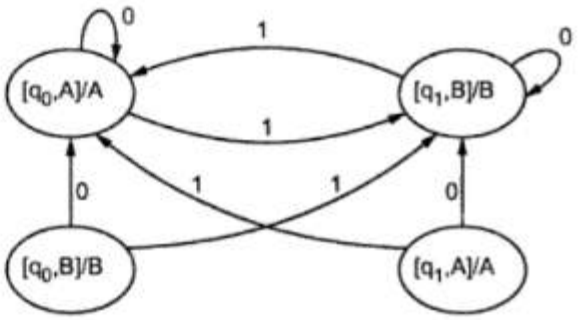


Fig. 2.64

Example 2.32 : *Convert the following Mealy machine into equivalent Moore machine.*

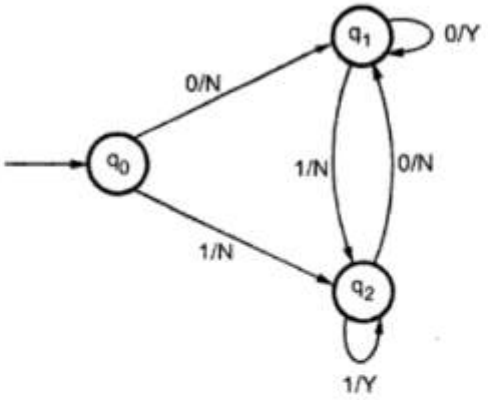


Fig. 2.65

Solution : We will write the transition table for given transition graph -

	Input 0	Output	Input 1	Output
q_0	q_1	N	q_2	N
q_1	q_1	Y	q_2	N
q_2	q_1	N	q_2	Y

Now we will find out the states and corresponding outputs for Moore machine. The states for Moore machine will be -

$[q_0, N]$, $[q_0, Y]$, $[q_1, N]$, $[q_1, Y]$, $[q_2, N]$, $[q_2, Y]$

Now let us calculate δ' and λ' for all the above given states -

$$\delta'([q_0, N], 0) = [\delta(q_0, 0), \lambda(q_0, 0)]$$
$$= [q_1, N]$$
$$\lambda'([q_0, N]) = N$$

Similarly,

$$\delta'([q_0, N), 1) = [\delta(q_0, 1), \lambda(q_0, 1)]$$
$$= [q_2, N]$$
$$\lambda'([q_0, N]) = N$$

The partial transition table is -

State \ I/P	0	1	Output
$[q_0, N]$	$[q_1, N]$	$[q_2, N]$	N

Now, for remaining states the corresponding transitions and outputs can be obtained as follows -

$$\delta'([q_0, Y], 0) = [\delta(q_0, 0), \lambda(q_0, 0)]$$
$$= [q_1, N]$$
$$\lambda'([q_0, Y]) = Y$$
$$\delta'([q_0, Y], 1) = [\delta(q_0, 1), \lambda(q_0, 1)]$$
$$= [q_2, N]$$
$$\lambda'([q_0, Y]) = Y$$
$$\delta'([q_1, N)], 0) = [\delta(q_1, 0), \lambda(q_1, 0)]$$
$$= [q_1, Y]$$
$$\lambda'([q_1, N]) = N$$
$$\delta'([q_1, N), 1) = [\delta(q_1, 1), \lambda(q_1, 1)]$$
$$= [q_2, N]$$
$$\lambda'([q_1, N]) = N$$

$$\delta'([q_1, Y], 0) = [\delta(q_1,0), \lambda(q_1,0)]$$
$$= [q_1, Y]$$
$$\lambda([q_1, Y]) = Y$$
$$\delta'([q_2, N], 0) = [\delta(q_2,0), \lambda(q_2,0)]$$
$$= [q_1, N]$$
$$\lambda'([q_2, N]) = N$$
$$\delta'([q_2, N], 1) = [\delta(q_2,1), \lambda(q_2,1)]$$
$$= [q_2, Y]$$
$$\lambda'([q_2, N]) = N$$
$$\delta'([q_1, Y], 1) = [\delta(q_1,1), \lambda(q_1, 1)]$$
$$= [q_2, N]$$
$$\lambda'([q_1, Y]) = Y$$
$$\delta'([q_2, Y], 0) = [\delta(q_2,0), \lambda(q_2, 0)]$$
$$= [q_1, N]$$
$$\lambda'([q_2, Y]) = Y$$
$$\delta'([q_2, Y], 1) = [\delta(q_2,1), \lambda(q_2, 1)]$$
$$= [q_2, Y]$$

$$\lambda'([q_2, Y]) = Y$$

The transition table can be draw as -

State \ Σ	0	1	Output
$[q_0, N]$	$[q_1, N]$	$[q_2, N]$	N
$[q_0, Y]$	$[q_1, N]$	$[q_2, N]$	Y
$[q_1, N]$	$[q_1, Y]$	$[q_2, N]$	N
$[q_1, Y]$	$[q_1, Y]$	$[q_2, N]$	Y
$[q_2, N]$	$[q_1, N]$	$[q_2, Y]$	N
$[q_2, Y]$	$[q_1, N]$	$[q_2, Y]$	Y

Unit - 3

Regular Set :-

Any Set that Represent the Value of the Regular Expression is called as Regular Set

$M = (S, T, F, S_0, F)$

where $S \Rightarrow$ Finite Set of State
$T \Rightarrow$ Input Symbol / Alphabet
$F \Rightarrow$ Mapping Function $S * T$ to S
$S_0 \Rightarrow$ Starting Symbol
$F \Rightarrow$ Final Symbol

Property of Regular Set :-

Property 1 : The union of two Regular Set is Regular

Proof

Let us take two Regular Expression

$RE = a(aa)^*$ & $RE = (aa)^*$

So $L_1 = \{ a, aaa, aaaaa \ldots \}$ (String of odd length excluding NULL)

$L_2 = \{ \varepsilon, aa, aaaa, aaaaaa \ldots \}$ (String even length excluding NULL)

$L_1 \cup L_2 = \{ \varepsilon, a, aa, aaa, aaaa, \ldots \}$ (String of all possible length including $\wedge$

$RE(L_1 \cup L_2) = a^*$

Hence its proved.

Property 4 : The difference of two Regular Set is Regular

Proof :-

Let us take two Reg Expression

$RE_1 = a(a^*)$ $RE_2 = (aa)^*$

So $L_1 = \{a, aa, aaa, aaaa \ldots\}$ {all possible length excluding NULL)

$L_2 = \{\varepsilon, aa, aaaa, aaaaaa \ldots\}$ (all possible ever length including NULL)

$L_1 - L_2 = \{a, aaa, aaaaa, aaaaaaa \ldots\}$ (Strings of all odd length excluding NULL)

$RE(L_1 - L_2) = a(aa)^*$

Hence, proved

Property 5 : The Reversal of a Regular Set is Regular.

Proof :- We have to prove L^R is also regular if L a regular Set

$L = \{01, 10, 11, 10\}$

$RE(L) = \{01 + 10 + 11 + 10\}$

$L^R = \{10, 01, 11, 01\}$

$RE(L^R) = 01 + 10 + 11 + 10$

Hence its proved

Property 6: The closure of a Regular Set is Regular.

Proof:-

if $L = \{a, aaa, aaaaa \ldots\}$ (String of odd length excluding NULL)

ie $RE(L) = a(aa)^*$

$L^* = \{a, aa, aaa, aaaa, aaaaa \ldots\}$ (String of all length excluding NULL)

$RE(L^*) = a(a^*)$

Hence Proved

Property 7: The concatenation of two Regular set is Regular

Proof

Let $RE_1 = (0+1)^*0$ & $RE_2 = 01(0+1)^*$

Here $L_1 = \{0, 00, 10, 000, 010 \ldots\}$ (Set of Strings ending in 0)

$L_2 = \{01, 010, 011 \ldots\}$ (Set of String begin with 01

$L_1 L_2 = \{001, 0010, 0011, 0001, 00010, \ldots\}$

$RE = (0+1)^*001(0+1)^*$

Hence proved.

eg Construct the R.E for the language accepting all the String combination which 'a's over the Set $\Sigma = \{a\}$

Soln $L = \{ \varepsilon, a, aa, aaa \ldots \}$

$R = a^*$

eg 2 Design the R.E for the language accepting all Combination of "a's" except null String over $\Sigma = \{a\}$

Soln $L = \{ a, aa, aaa \ldots \}$

$R = a^+$

eg 3 Design the R.E for language containing all the String containing any number of a's and b's

Soln $R.L = \{ \varepsilon, a, aa, ab, b \ldots \}$

$= (a+b)^*$

eg 4: Construct the R.E for the Lang accepting all the String which are ending with 00 over the Set $\Sigma = \{0, 1\}$

$L = \{ 01, 001, 011 \} 00$

$= (0+1)^* 00$

Regular Expression:-

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

$\varphi \Rightarrow$ Reg. Exp which denotes the empty set

$\varepsilon \Rightarrow$ Reg. Exp & denote the set $\{\varepsilon\}$ and its null String

For each 'a' in Σ is 'a' regular Expres and denote by set $\{a\}$

If r & s are Reg Exp denoting the lang L_1 and L_2 respectively

Union $r+s$ equvalent $L_1 \cup L_2$ is union

Concatenation

$r.s$ equlent to $L_1 L_2$

closure

r^* equivalent to L^*

Kleen closure

The r^* is known as Kleen closure which indicate occurrence of r for number of tiries

Eg 5, If L = { The Language Starting and ending with a & having any combination of b's between } that what is r?

$L = \{aba, abba, abbba \ldots\}$

$r = ab^* a$

Eg 6 Describe the Simple English the Language represented by the following Regular Expression

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

Soln

$$Lr = \{a, aba, abab, aab \ldots\}$$

Eg 7 Write R.E to denote the Language Lover Σ^*, when $\Sigma = \{a, b, c\}$ in which every String will be such that any number of a's followed by any number of b's is followed by any number of c's

$$L = \{abc, aabcc, aaabbbccc \ldots$$

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

Finite Automata to Regular expression conversion

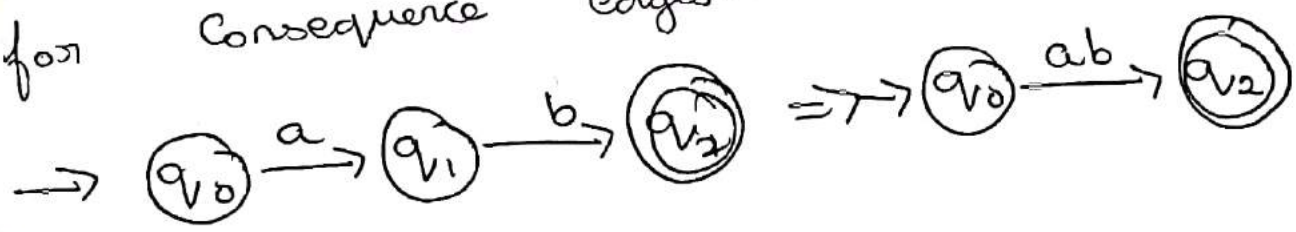
Constrain:

While converting from FA (DFA / NFA) we always write Regular Expression for Final State.

A Stage having more than one edge into a Single Direction

A Stage having Consequtive Edges:-

We can write RE by using multiplication for Consequence edges:-

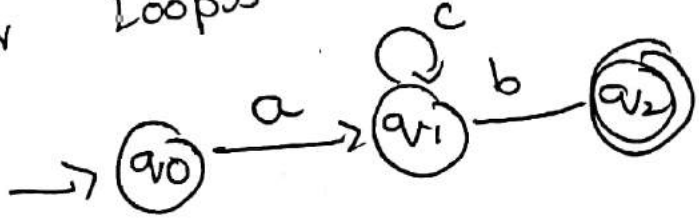


Loops

There are two types of loop

Self Loop

for Loops we use closure (*) Sign



If a State has more than one Self loop

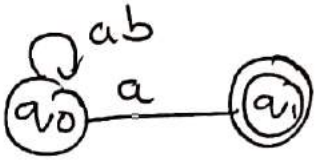
State ~~rep~~ q_1 can repeat any combination of a & b then:- $a\,[c+d]^*\,b$

2) Loop between the two States:

2 ways

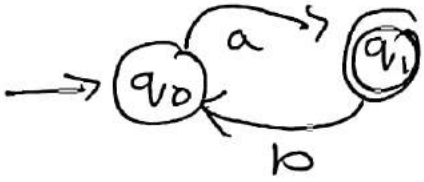
1, Resolving to Left Side

$\Rightarrow$



$\Rightarrow (ab)^* a$

2, Resolving to ~~right~~ Right Side

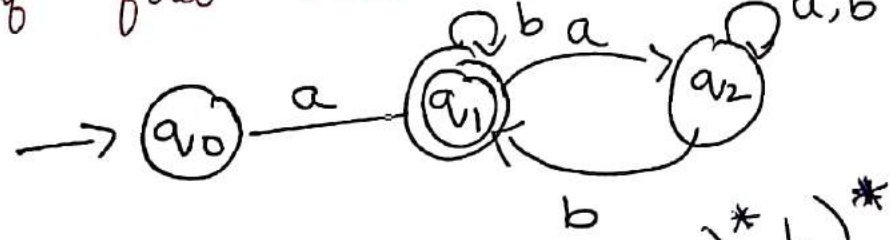


$\Rightarrow$

$\Rightarrow a(ba)^*$

$(ab^*)^*a = a(ba)^*$

If final State is middle State with the Loop

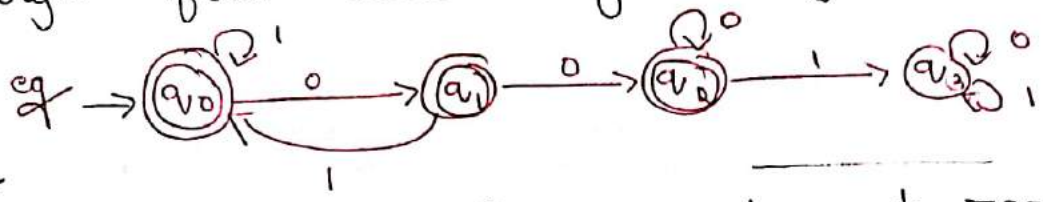


$$a(b + a(a+b)^* b)^*$$

4, If finite Automata has more than one final State

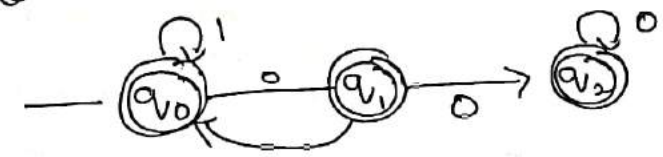
In this case we have to write Regular Expression for all the state and combine it the result

In other case we can convert them into Single final State using ε Symbol

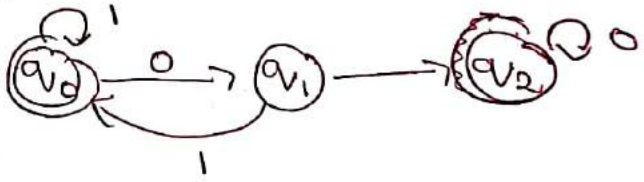
eg 

Sol

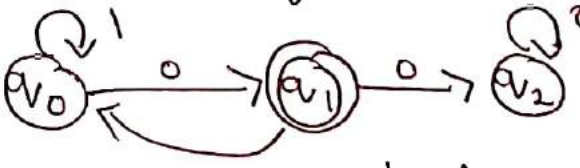
[∴ The State q_3 have no does not reach final State So we consider it as Trap / Dead State]



R.E for q_0 [Assume q_1 & q_2 are non final)

$$q_0 = [1 + 01]^*$$ 

R.E for q_1 [assume q_0 & q_2 are Non final]

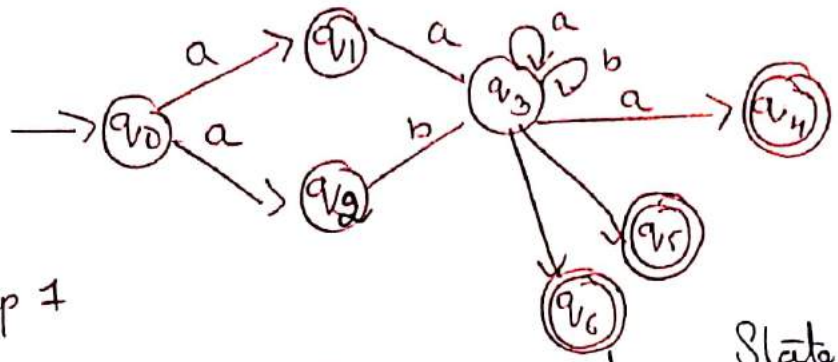
$$q_1 = [1 + 01]^* 0$$ 

RE for q_2 [assume q_0 & q_1 are non final]

$$q_2 = [1 + 01]^* 00^*$$ 

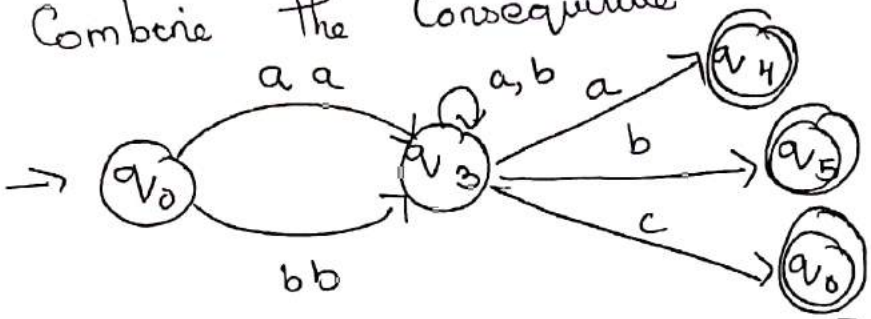
$$RE = [1 + 01]^* + [1 + 01]^* 0 + [1 + 01]^* 00^*$$
$$= [1 + 01]^* [\varepsilon + 0 + 00^*]$$

(2) Convert FA to Regular Expression :-



Step 1

Combine the Consequtive State by multiplication



RE for q_4 [Consume Remaing NF]

$$q_4 \Rightarrow (aa+bb)(a+b)^* a$$

RE for q_5

$$q_5 = (aa+bb)(a+b)^* b$$

RE for q_6 $q_6 = (aa+bb)(a+b)^* c$

Combain of all the Result

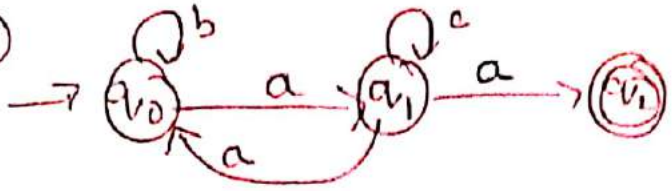
$$R.E = [aa+bb][a+b]^* a + [aa+bb] [a+b]^* b + [aa+bb] + [a+b]^* c$$

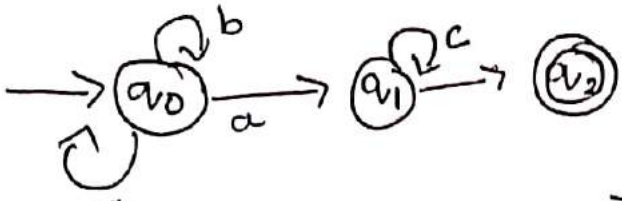
$$= [aa+bb][a+b]^* [a+b+c] \Rightarrow$$

[Take Common Term]

Problem for Reverse Loop -

Problem (1)

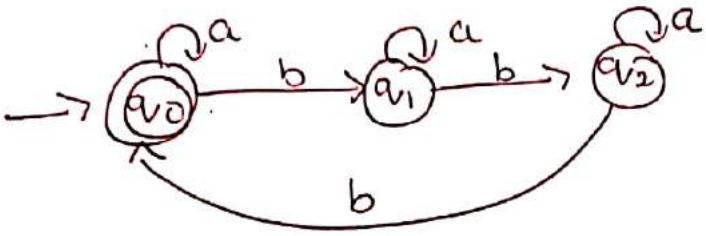




ac^*a

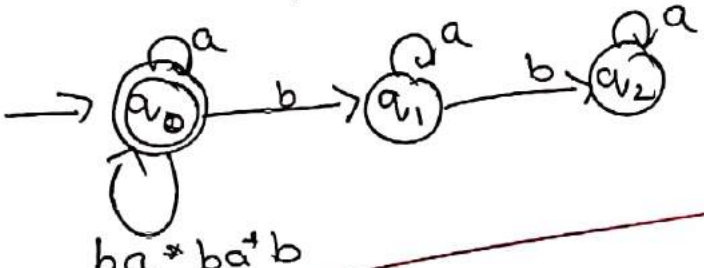
$[b + ac^*a]^* ac^*a$

(2)



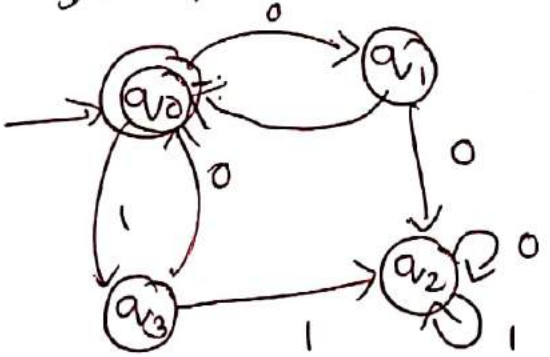
[Initial & final State Same]

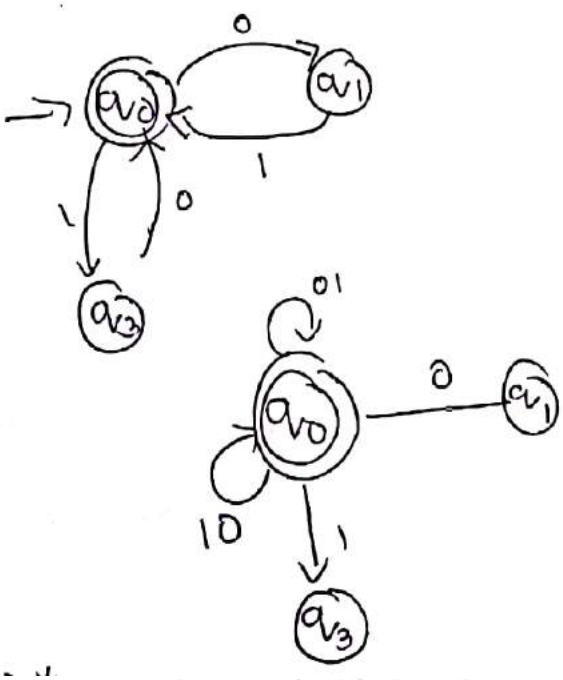
ba^*ba^*b



$\Rightarrow [a + ba^*ba^*b]^*$

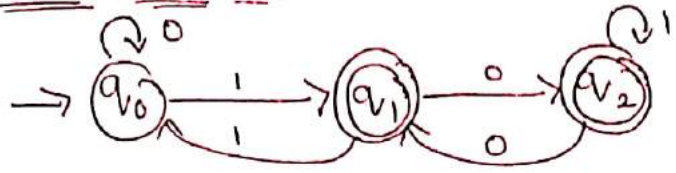
With Dead / Trap State





~~TOC~~

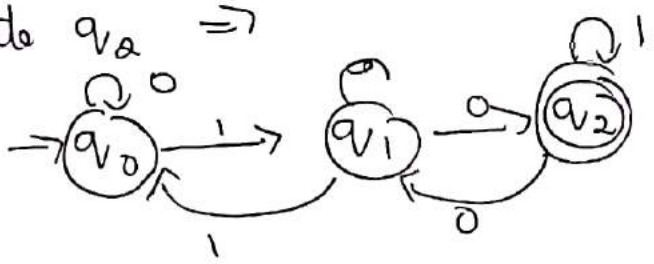
Problem on Loop



For State q_1 →

$$(0+11)^* 1 (01^* 0) = \alpha$$

For State q_2 ⇒



$$\therefore R = (0+11)^* 1(01^* 0) * 01^*$$

Combine

$$(0+11)^* 1(01^* 0) + (0+11)^* 1(01^* 0) * 01^*)^*$$

ARDEN'S Theorem:-

If P & Q are two Regular Expression over Σ and if P does not contain ε then the following equation in R given by $R = Q + RP$ has a unique Solution That is $R = QP^*$

$$R = Q + RP \longrightarrow (1)$$

Sub $R = QP^*$ in ①

$= Q + QP^* P$

$\Rightarrow Q(\varepsilon + P^* P)$ $[\varepsilon + R^* R = R^*]$

$= QP^*$

$\Rightarrow$ Hence its proved

② prove $R = QP^*$ is Unique Solution

$R = Q + RP$ ⟶ ②

Given $R = QP^*$ ⟶ ①

Sub ② in ①

$R = Q + [Q + RP] P$

$= Q + QP + RP^2$. Sub $R = [Q + RP]$

$= Q + QP + [Q + RP] P^2$

$= Q + QP + QP^2 + RP^3$

$\vdots$

$= Q + QP + QP^2 + QP^3 \ldots QP^n + QP^* P^{n+1}$

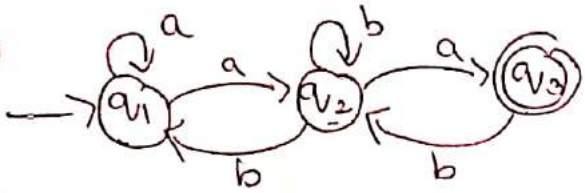
[Take Q as common]

$= Q[\varepsilon + P + P^2 \ldots\ldots P^n + P^* P^{n-1}]$

$= QP^*$

NFA to Regular Expression Conversion

(1)



Soln [check incoming transitions]

$$q_3 \Rightarrow q_2 a \longrightarrow (1)$$

$$q_2 = q_1 a + q_2 b + q_3 b \longrightarrow (2)$$

$$q_1 \Rightarrow q_1 a \; a_2 b + \varepsilon \longrightarrow (3)$$

(1) $\Rightarrow q_3 = q_2 a$ [Sub q_2 value]

$$= (q_1 a + q_2 b + q_3 b) a$$

$$= q_1 aa + q_2 ba + q_3 ba \longrightarrow (4)$$

(2) $\Rightarrow q_2 = q_1 a + q_2 b + q_3 b$ [Sub value of q_3 from (1)]

$$= q_1 a + q_2 b + (q_2 a) b$$

$$\Rightarrow q_1 a + q_2 b + q_2 ab$$

$$q_2 \Rightarrow q_1 a + q_2 (b + ab) \quad [R = Q + RP]$$

$$q_2 \Rightarrow q_1 a \cdot (b + ab)^* \longrightarrow (5) \quad [R = QP^* \text{ by Arden's Theorem}]$$

(3) $\Rightarrow q_1 = \varepsilon + q_1 a + q_2 b$ [Sub q_2 value in q_1]

$$q_1 = \underbrace{\varepsilon}_{Q} + \underbrace{q_1 a}_{R} + \underbrace{(q_1 a (b + ab)^*) b}_{P} \qquad R = Q + RP$$

$$= \varepsilon (a + a(b + ab^*)b)^* \qquad R = QP^*$$

$q_1 = \epsilon + a(b+ab)^* b)^*$ → ⑥ $(\epsilon . R = R)$

Sub all Value for final State

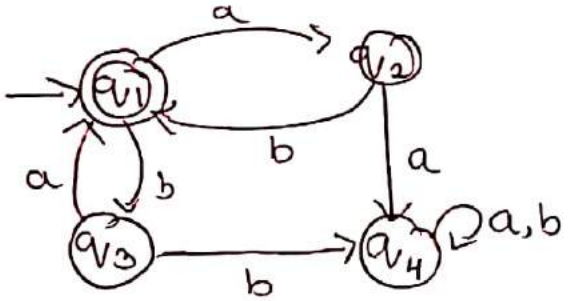
$q_3 = q_2 a$ [Sub eq 5]

$= q_1 a(b+ab)^* a$

$q_3 = (\epsilon + a(b+ab)^* b)^* a(b+ab)^* a$ {Sub Value q_1 from ⑥}

= Regular Exp for NFA

DFA to Regular Expression Conversion



Sol: $q_1 = \epsilon + q_2 b + q_3 a$ → ①

$q_2 = q_1 a$ → ②

$q_2 \Rightarrow q_1 b$ → ③

$q_4 \Rightarrow q_2 a + q_3 b + q_4 ab$

Take final State

① $\Rightarrow q_1 = \epsilon + q_2 b + q_3 a$ [Sub q_2 q_3 Value

$= \epsilon + q_1 ab + q_1 ba$

$= \epsilon + q_1 [ab + ba]$ [by adern Theorem]

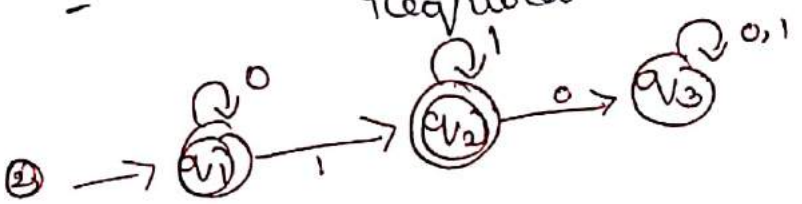
$q_1 = \epsilon [ab+ba]^*$

$\epsilon R = R$

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

$q_1 = (ab + ba)^*$

Required Regular Expression



[incoming δ]

Soln

$\Rightarrow q_1 = \epsilon + q_1 0 \longrightarrow (1)$

$q_2 \Rightarrow q_1 1 + q_2 1 \longrightarrow (2)$

$q_3 = q_2 0 + q_3 0 1 \longrightarrow (3)$

final State [By adern Theoren]

$\frac{q_1}{P} = \frac{\epsilon}{Q} + \frac{q_1}{R}\frac{0}{P}$

$= \epsilon 0^*$ [$\epsilon R = R$]

$q_1 = 0^* \longrightarrow (4)$

Nxt final State

$q_2 = q_1 1 + q_2 1$ [Sub q_1 value from eq (4)]

$\frac{q_2}{R} = \frac{0^* 1}{Q} + \frac{q_2 1}{R P}$ [By adens Theorem]

$= (6)$

$q_2 = 0^* 1 (1)^* \longrightarrow (5)$

Combain the Reullt (4) & (5)

$= 0^* + 0^* 1 (1)^*$

$= 0^* (\epsilon + 1 1^*) = 0^* (1)^* = 0^* 1^*$

DFA to RE: by Raji Theorem:-

$$R_{ij}^{k} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} \left[R_{kk}^{(k-1)} \right]^{*} R_{kj}$$

$i \Rightarrow$ Starting State $\quad j \Rightarrow$ final State $\quad K \Rightarrow$ No of nodes

$R \Rightarrow$ Regular Expression

$$R_{12}^{2} = R_{12}^{1} + R_{12}^{1} \left[R_{22}^{1} \right]^{*} R_{22}$$

Possibility

$$R_{11}^{0} = 1 + \varepsilon$$

$$R_{12}^{0} = 0$$

$$R_{21}^{0} = \phi$$

$$R_{22}^{0} = 0 + 1 + \varepsilon$$

$\phi \Rightarrow$ No Path

$K = 1$

$$R_{12}^{1} = R_{12}^{0} + R_{11}^{0} \left[R_{11}^{0} \right]^{*} R_{12}$$

$$= 0 + (1+\varepsilon)(1+\varepsilon)^{*}(0)$$

$$= 0 + (1+\varepsilon)\, 1^{*} 0$$

$$= 0 + 1^{*} 0$$

$$= 0 + 1^{*} 0$$

$$\boxed{R_{12}^{1} = 1^{*} 0}$$

$$R_{22}^{1} = R_{22}^{0} + R_{21}^{0} [R_{11}^{0}]^* R_{12}^{0}$$

$$= (0+1+\epsilon) + \phi [1+\epsilon]^* 0$$

$$\cdot (0+1+\epsilon)$$

[ϕ × anything = ϕ]

$$R_{22}^{1} = (0+1+\epsilon$$

$$R_{12}^{2} = R_{12}^{1} + R_{12}^{1} [R_{22}^{1}]^* R_{22}^{1}$$

$$= 1^*0 + [1^*0 [\epsilon+0+1]^* [\epsilon+0+1]]$$

$$= 1^*0 + [1^*0 [\epsilon+0+1]^* [\epsilon+0+1]]$$

$$= 1^*0 + [1^*0 [0+1]^* [\epsilon+0+1]] \quad \epsilon + R = R$$

$$= 1^*0 + [1^*0 (0+1)^*]$$

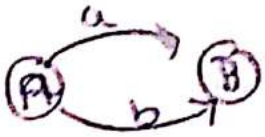
$$= 1^*0 + 1^* (0+1)^*$$

$$\boxed{R_{12}^{2} = 1^*0 (0+1)^*}$$

Regular Expression To Finite Automata

Regular Expression to FA

① $(a+b)$



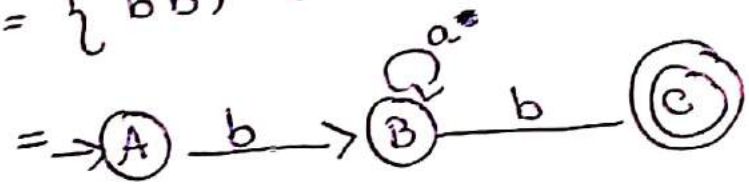
② $a.b$

③ a^*

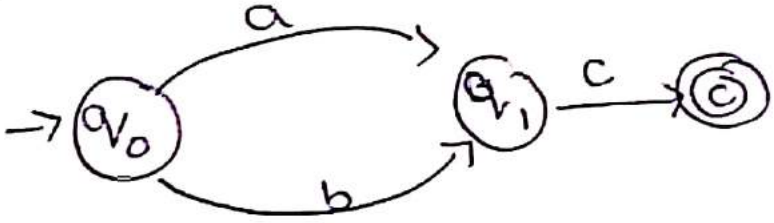
Convert the following Reg Expression to their Equivalent Finite Automata

1, ba^*b

$L = \{ bb, bab, baab \ldots \}$ [Relevant possible input]



④ $(a+b)c$



$L = \{ ac, bc \}$

③ $a(bc)^*$

$L = \{ a, abc, abcbc, abcbc, abcbc \ldots \}$

An Example Proof using Identities of Regular Expression

Prove That $(1+00^*1)+(1+00^*1)(0+10^*1)^*(0+10^*1)$ is equal to $0^*1(0+10^*1)^*$

Proof

$$LHS = (1+00^*1)+(1+00^*1)(0+10^*1)^*(0+10^*1)$$

(Take the Common Term)

$$= (1+00^*1)\left[\varepsilon + (0+10^*1)^*(0+10^*+1)\right]$$

$\varepsilon + R^*R = R^*$

$$= (1+00^*1)(0+10^*1)^*$$

$$= (\varepsilon \cdot 1 + 00^*1)(0+10^*1)^* \qquad \varepsilon \cdot R = R$$

$$= (\varepsilon \cdot 00^*)(1(0+10^*1)^*$$

$$= \cancel{(\varepsilon + 00^*)} \qquad \varepsilon + R^*R = R^*$$

$$= 0^*1(0+10^*1)^*$$

$LHS = RHS$

Hence its proved.

Regular Expression To Finite Automata
[Thompson's Construction]

① $R = \varphi$

{φ - { φ ⇒ No path }}

{ε ⇒ without input moving to next state}

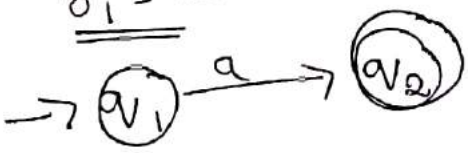
② $R = \varepsilon$

③ $R = a$

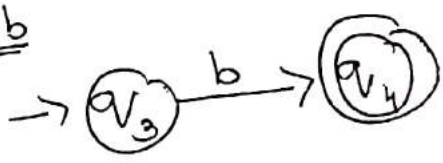
④ $R = a + b$

$r_1 = a \qquad r_2 = b$

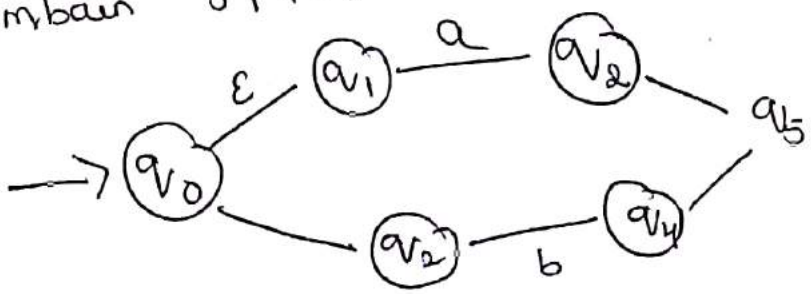
$r_1 = a$



$r_2 = b$



Combain $r_1 + r_2$



⑤ $R = a^*$

$L = \{ \varepsilon, a, aa, aaa \ldots \}$

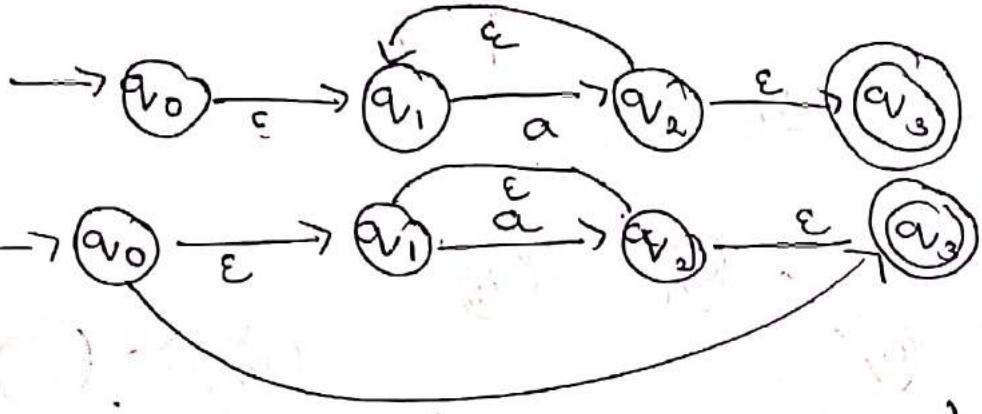
$r_1 = a$

For any Combination of 'a'

Start State $= q_1$ Final State $= q_2$

q_2 to q_1

$L = \{ a, aa, aaa \ldots$



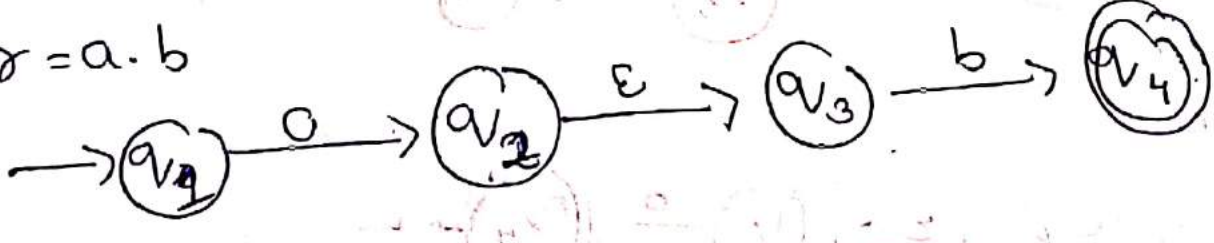
$L = \{ \varepsilon, a, aa, aaa \ldots \}$

) $R = a.b$

$r_1 = a$

$r_2 = b$

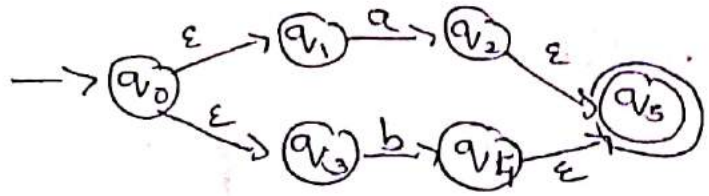
$r = a.b$



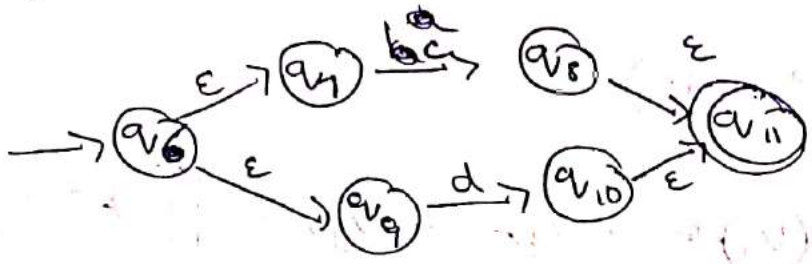
a

$R = (a+b)(c+d)$

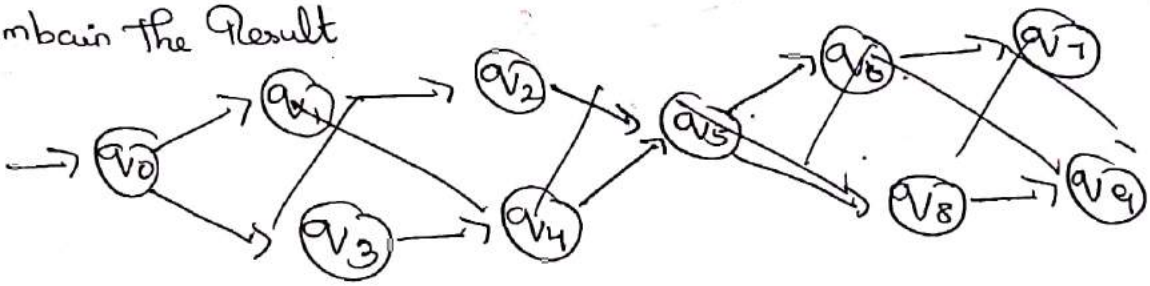
$r = (a+b)$

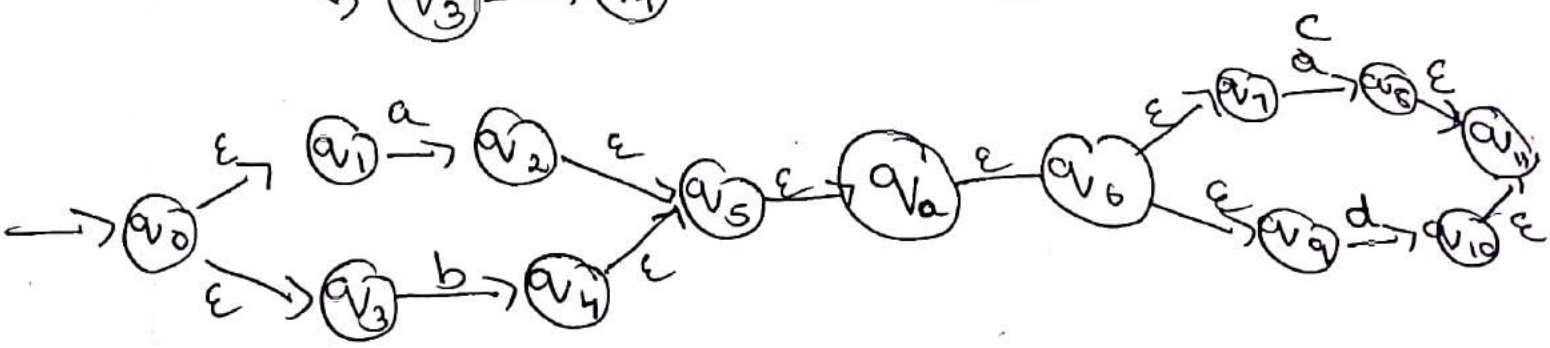


$r = (c+b)$



Combain the Result





Pumping Lemma for Regular Grammar

Theorem: Let L be a RL Then there exist a Constant 'p' such that for every String w in L

& - $|w| \geq p$

we can break 'w' into three String $w = xyz$ such that

[w - String
|w| - length of String]

$|y| > 0$

$|xy| \leq p$

For all $k \geq 0$ the String $xy^k z$ is also in L

Application

Pumping Lemma is to be applied to show that certain Languages are not regular. It should never be used to show a lang is regular.

If L is regular, it satisfies pumping Lemma

If L does not satisfy pumping Lemma its not Regular

To Prove the Lang Not Regular

→ At first we have to assume that L is regular

→ So, the pumping Lemma should hold function L

→ It has to have a pumping Length (Say P)
⇒ All Strings longer than P can be pumped
→ Now find a String w in L such that $|w| \geq P$
→ Divide w into xyz
→ Show the $xy^iz \in L$ for some i
⇒ Then consider all ways that w can be divided into xyz
⇒ Show that none of these can satisfy all 3 pumping condition at the same time
⇒ W cannot be pumped ⇒ CONTradiction

$L = \{a^n b^n / n \geq 1\}$ is Not Regular using pumping Lemma.

Sol

Assume that L is Regular

Pumping length = P

assume $P = 7$

$S = a^P b^P$

$\therefore S = \{a\,a\,a\,a\,a\,a\,a\,b\,b\,b\,b\,b\,b\,b\}$

Case 1 - The Y is in the 'a' part [we are going to split into 3 there was no constrain for splitting but for position x - 1st y - 2 parts z - last part]

aa aaaaa bbbbbbb
x y z

Case 2: The Y is in 'b' part

aaaaaaa bbbbbbb
x y z

Case 3: The Y is in the 'a' & 'b' part

aaaa aaa bbbbbbb
x y z

$x = aa \quad y = aaaaa \quad z = bbbbbbb$

i=2 case 1

$xy^i z = xy^2 z$

aaaaaaa aaaabbbbbbb

12(a's) ≠ 7(b's)

i=2

$xy^i z$ in case 2 $\quad x = aaaaaaaa$

$y = bbbbb$

$z = bb$

$xyz = xy^2z$

aaaaaaabb bbbb bbbb b

7'(a's) ≠ 12'(b's)

i=2 case 3

$xyz = xy^2z$

aa aaaaaa aabb aabb bbbbb

a's ≠ gt doesnot follow pattern

$|xy| \leq p$

$\therefore$ Hence its proved this Lang Not Regular

Show that $L = \{0^n 1^{n+1} \mid n > 0\}$ is not Regular ?

Sol:

assume L is Regular Language

Pumping length $P = 3$

$S = 0^P 1^{P+1}$

$P = 3$

$S = 0^3 1^4$

$S = \{0001111\}$

Case:

$x\ y^2\ z$

$\underbrace{000}_{x}\ \underbrace{1111}_{z}$

$\underbrace{0}_{x}\ \underbrace{00}_{y}\ \underbrace{1111}_{z}$

$\Rightarrow x = 0 \quad y = 00 \quad z = 1111$

$\Rightarrow 0\ 00\ 00\ 1111$

$\Rightarrow 5 \neq 4$

$|xy| > 0$ ✓

$|xy| \leq P$

$5 \not\leq 3$ ✗

L is Not Regular

Kleene Closure:

If L_1 is regular Lang its Kleen Closure L_1^* will also be regular.

eg $L_1 = (a \cup b)$

$L_2^* = (a \cup b)^*$

Complement:-

If $L(G_1)$ is regular Lang its complement $L'(G_1)$ will also be regular.

eg $L(G_1) = \{a^n \mid n > 3\}$

$L'(G_1) = \{a^n \mid n <= 3\}$

Difference:

If L and M are regular Languages then so

L - M = String is L but not M

Let A & B be a DFA whose Lang are L &

respectively

Construct C, the product automaton of A &

B make the final State of C be pair whe

A - State is final but B - State is not

Positive Closure:

R3 is regular Expression whose lan

is L, M. R^+ is regular expression who

Lang L^+

Reverse Operator

Given Lang L, L^R is the set of strings whose reversal is in L

eg $L = \{0, 01, 100\}$

$L^R = \{0, 10, 001\}$

Homomorphism:-

A homomorphism on an alphabet is a function that gives string for each symbol in that alphabet $h(L) = \{h(w) : w \in L\}$

eg $h(0) = ab$, $h(1) = ab$, $h(01) = abab$

eg $\Sigma = \{a, b\}$, $\Delta = \{0, 1, 2\}$

$h(a) = 01$ $h(b) = 112$

$L = a^*b$

$h(L) = h(a^*b) = h(a^*)h(b)$

$= h(a)^* h(b) = (01)^* 112$

Inverse homomorphism:

Let h be an homomorphism and L a language whose alphabet is the output Langu

of h $h^{-1}(L) = \{x \mid h(x) \text{ is in } L\}$

eg Let $\Sigma = \{0, 1, 2\}$, $\Delta = (a, b)$

$h(0) = a$ $h(1) = ab$ $h = ba$

$L = a(ba)^*$ $h^{-1}(L) = ?$

$L = a\{\epsilon, ba, baba, bababa \ldots\}$

$= \{a, aba, ababa \ldots\}$

$= \{a, 1002, 102, 1102, 0202, 1202, \ldots\}$

$= \{0, 10, 110, 022, 102, 02, \ldots\}$

$\} \Rightarrow 1^*0$

$0 2^*$

$1^* 02^*$

$= a \cdot b \mid m \geq 0$ of $m \geq 0$

Closure Properties of Regular Language

Union:- If L_1 and L_2 are two regular lang and their union $L_1 \cup L_2$ will also be regular.

eg $L_1 = \{a^n \mid n \geq 0\}$ & $L_2 = \{b^n \mid n \geq 0\}$

$L_3 = L_1 \cup L_2$

$= \{a^n \cup b^n \mid n \geq 0\}$

$\Rightarrow$ Its a regular.

Concatenation

If L_1 and L_2 are two regular lang their concatenation $L_1 . L_2$ will also be regular.

eg $L_1 = \{a^n \mid n \geq 0\}$ & $L_2 = \{b^m \mid m \geq 0$

$L_3 = L_1 L_2$

$= \{a^n b^m \mid m \geq 0$ & $n \geq 0\}$

Its regular.

Intersection

If L_1 and L_2 are two regular lang their intersection $L_1 \cap L_2$ will be regular

$L_1 = \{a^m b^n \mid n \geq$ & $m \geq 0\}$ &

$L_2 = \{a^m b^n \cup b^n a^m \mid n \geq 0$ & $m \geq 0\}$

$L_3 = L_1 \cap L_2 = \{a^m b^n \mid n \geq 0$ & $m \geq 0$ is Regul

3.b i0

Regular Linear Grammar and FA Equivalence

Let $M = (\{q_0 q_1 \ldots q_n\}, \Sigma, \delta, q_0, F)$ be a DFA.

The Equivalent Grammar can be Constructed from given DFA Such that production corresponded to transition

By the transitions terminating we can encountered the final State

Let

$G = (\{A_0, A_1 \ldots A_n\}, \Sigma, P, A_0)$

where P is the Set of production Rule

1, $A_i \to aA_j$ is a production Rule if

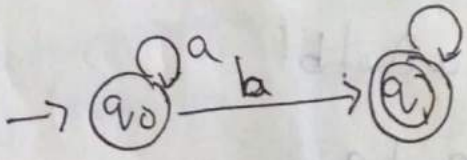
$\delta(q_i, a) = q_i$ & $q_i \in F$

2, $A_i \to aA_j$ and $A_i \to a$ are productio rule if $\delta(q_i a) = a_i$ where $q_i \in F$

Thus given grammar is accepted by DFA M

Problem on FA to Regular Grammar.

eg 4.3 Construct Regular Grammar for given DF



Soln

$G = (V, T, P, S)$ $\quad q_0 = A_0, \; q_1 = A_1$

$V = \{A_0, A_1\}$

$T = \{a, b\}$

$A_0 \rightarrow aA_0 \quad A_1 \rightarrow aA_1 / a$

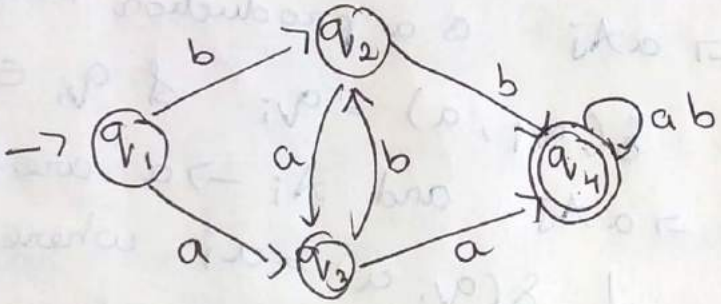
$A_0 \rightarrow bA_1 \quad A_1 \rightarrow bA_1 / b$

$A_0 \rightarrow b$

$A_0 \rightarrow aA_0 / bA_1 / b$

$A_1 \rightarrow aA_1 / bA_1 / a / b$

② Construct a Regular Grammar for given FA



Soln

$V = (V, T, P, S)$ where

$V = \{A_0, A_1, A_2, A_3\}$

$q_1 = A_0$

$q_2 = A_1$

$q_3 = A_2$

$q_4 = A_3$

Production

$A_0 \rightarrow bA_1 / aA_2$

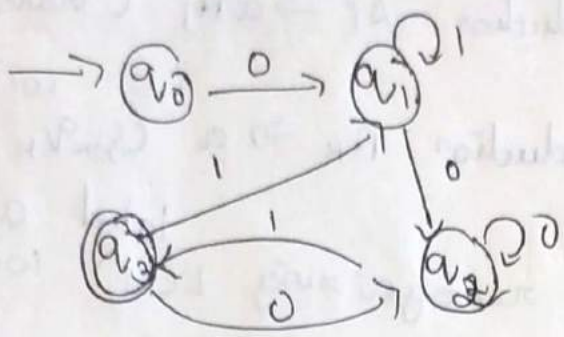
$A_1 \rightarrow bA_3 / aA_2 / b$

$A_2 \rightarrow bA_1 / bA_3 / a$

$A_3 \rightarrow aA_3 / bA_3 / a / b$

3-b-4/10-b

③ Construct Regular grammar for.



Soln

$V = \{ A_0, A_1, A_2, A_3 \}$

$A_0 \rightarrow 0A_1$

$A_1 \rightarrow 1A_1 \mid 0A_2$

$A_2 \rightarrow 0A_2 \mid 1A_3 \mid 1$

$A_3 \rightarrow 0A_2 \mid 1A_1$

Construction of FA from Regular Grammar.

Let $G = (V, \Sigma, P, A_0)$ be Regular grammar

We can Construct DFA M whose

① States correspond to Variable

② Initial State correspond to A_0

③ Transition M Correspond to production P

If there is a production from $A_i \rightarrow a$ the corresponding transition terminal at a new State (final State)

DFA ⇒ $M = (\{q_0, q_1 \cdots q_n q_f) \Sigma, \delta, q_0 \{q_f\}$

Thus δ defined

① Each production $A_i \to a A_j$ (trans from q_i to q_j with label a)

② Each production $A_k \to a$ (δ to q_k to q_f with label a)

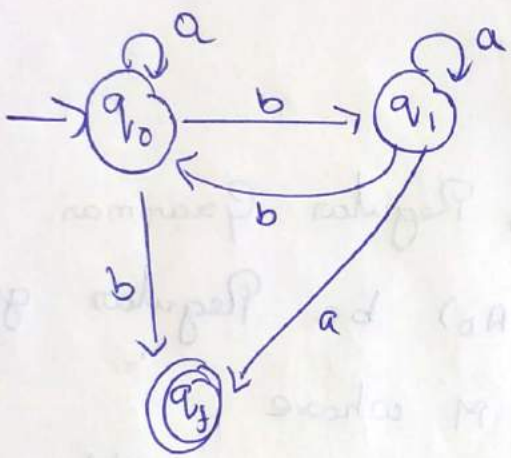
Construct a FA recognizing $L(G)$ where G is the grammar

$S \to aS \mid bA \mid b$

$A \to aA \mid bS \mid a$

Soln

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



Context Free Grammar :-

⇒ Context Free grammar is a formal Grammar which is used to generate all possible String in a given formal Language

⇒ CFG can be define by four tuples as

$G = (V, T, P, S)$

V/N → Non terminal

T → Terminal

P → Production Rule

S → Start Symbol

⇒ If production Rule

$A \Rightarrow \alpha$

$\alpha \in (V \cup T)^*$

eg $A \Rightarrow \varepsilon$

$A \Rightarrow BCD$

Context

① $aAb \rightarrow bb$

a → left context of A

b → Right Context of A

② $\varepsilon A \varepsilon$

No Context

Capabilities of CFG

⇒ CFG is used to describe most of the programming languages

⇒ If the grammar is properly designed then an efficient parser can be constructed automatically

⇒ CFG is capable of describing nested structures like balanced parentheses, matching begin-end, corresponding if then-else & so on.

Properties of CFG:-

① Based on number of String it generates

If CFG is generating finite number of String then CFG is Non-Recursive.

If CFG can generate infinite number of String then the grammar is said to be Recursive grammar.

② Based on number of derivation tree.

If there is only 1 derivation tree then the CFG is unambiguous

If there are more than 1 derivation tree then CFG is ambiguous.

eg Recursive Grammar:-

① $S \to SaS$

$S \to b$

$L = \{b, bab, bababab\}$

② $S \to Aa$

$A \to Ab/c$

$L = \{ca, cba, cbba \ldots\}$

eg Non Recursive Grammar:-

$S \to Aa$

$A \to b/c$

$L = \{ba, ca\}$

Context Free Language:-

In formal Language theory a Context Free language is Langage generated by some CFG

The set of all CFL is identical to the set of Languages accepted by pushdown automata

Conversion of Language to Grammar.

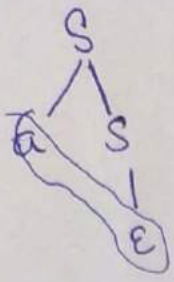
$L = \{a^n / n \geq 0\}$

Soln

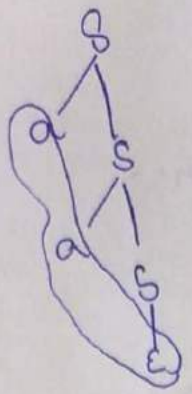
$L = \{\varepsilon, a, aa \ldots\}$

$S \to aS / \varepsilon$

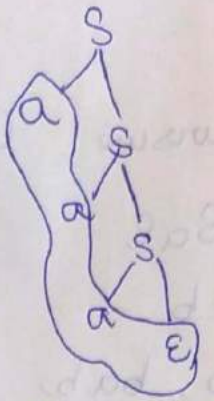
3. b 1 2



$\psi = a$



$\psi = aa$



ψ aaa

② L = { any combination of a's & b's }

Sol:

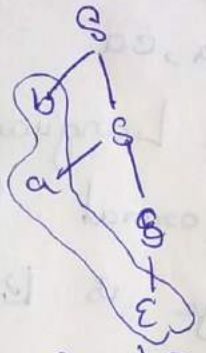
$L = (a+b)^*$

$L = \{ \varepsilon, a, b, ab, ba, aa, abb \ldots \}$

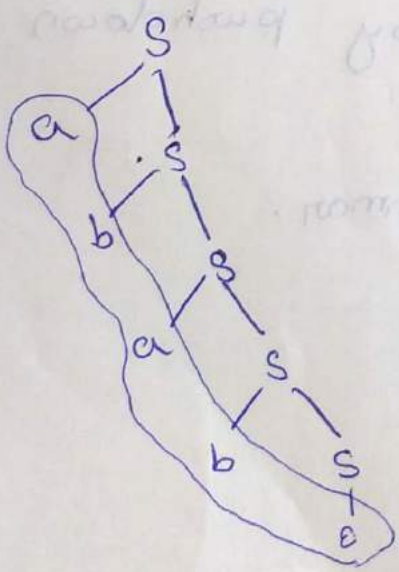
$S \rightarrow aS \mid bS \mid \varepsilon$

$\psi = a$

$\psi = ab$



$\psi = ba$



abab

Generating grammers from Language

1) $L = \{a^n b^m \mid n \geq 1, m \geq 0\}$

$n = 1 \text{ to } \infty$

$a^n \Rightarrow L = \{a, aa\}$

$L = aA / a$

$b^m \rightarrow b^0 \Rightarrow 1 = \varepsilon$

$L = \{\varepsilon, b, bb, bbb \ldots\}$

$B \rightarrow bB / \varepsilon$

b

$S \rightarrow AB$

$A \rightarrow aA / a$

$B \rightarrow bB / \varepsilon$

---

2) $L = \{a^n b^m \mid (n+m) \text{ is even}\}$

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

even + even = even

odd + odd = even

$S \rightarrow AB + aAbB$

$A \rightarrow aaA / \varepsilon$

$B \rightarrow bbB / \varepsilon$

NOTE

$L = (a)^* = \{\varepsilon, a, aa, aaa \ldots\}$ $A \rightarrow aaA / \varepsilon$

$(aa)^* = \{\varepsilon, aa, aaaa, \ldots\}$

$a(aa)^* = \{a, aaa, aaaaa \ldots\}$

3 $L = \{ a^n b^m \mid (n+m) \text{ is odd} \}$

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

$(E + 0 = 0)$ $(0 + \varepsilon \neq 0)$

$S \to AbB \mid aAB$

$A \to aaA \mid \varepsilon$

$B \to bbB \mid \varepsilon$

---

(4) $L = \{ a^n b^m \mid n \geq 3 \quad m \geq 2 \}$

$n = 3 \quad m = 2$

$L = \{ aaabb, aaaabbb \ldots \}$

$S \to aaaAbbB$

$A \to aA \mid \varepsilon$

$B \to bB \mid \varepsilon$

---

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

$n = 1$

$L = \{ \varepsilon, ab, aabb \ldots \}$

$S \to aSb \mid \varepsilon$

③ $L = \{ a^n b^m \mid n \geq m \}$

Soln

$L(G) = \{ ab, aab, aabb, aaabb \ldots \}$

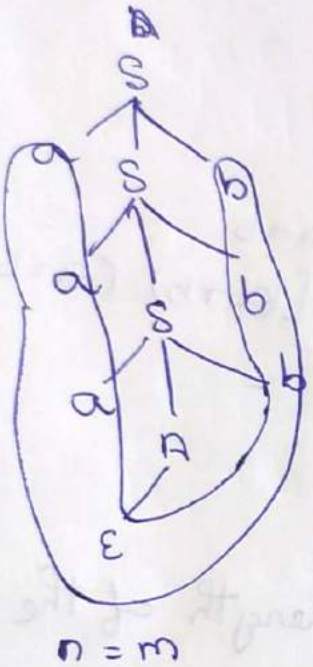
$S \rightarrow aAb \mid aSb$

$A \rightarrow aA \mid \varepsilon$

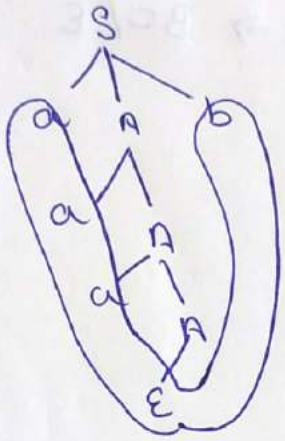
[NOTE $\Rightarrow$ $A \rightarrow aA \mid \varepsilon$

$A \Rightarrow aA \Rightarrow aaA \Rightarrow aaA$

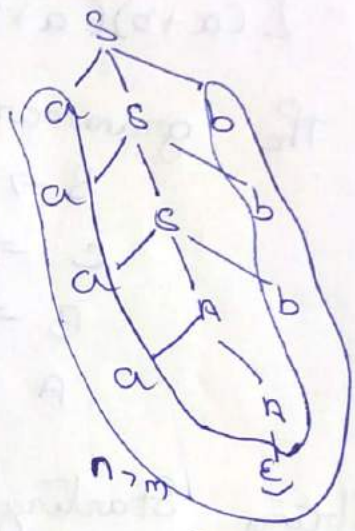
$L = a^+$



$n = m$



$n > m$



$n > m$

④ Find the grammar that generate all String of a's & b's where each String is Starting and ending with same Symbol

$RE = a(a+b)^* a + b(a+b)^* b + a + b$

Soln

The grammar is

$S \rightarrow aAa \mid bAb \mid a \mid b$

$A \rightarrow aA \mid bA \mid \varepsilon$

$[(a+b)^* = a A$

any combination of b

3. b (1 4)

5) Starting and Ending with diff Symbol

R.E =

6) Length of String is odd (a & b input)

R E $= [(a+b)(a+b)]^* (a+b)$

Sol

$(a+b) \Rightarrow A \rightarrow a/b$

$(a+b)(a+b) = B \Rightarrow AA$

$[(a+b)(a+b)]^* \Rightarrow C \Rightarrow BC/\varepsilon$

$A = a+b$

$C = [(a+b)(a+b)]^*$

the grammar is

$S \Rightarrow AC$

$C \rightarrow BC/\varepsilon$

$B \rightarrow AA$

$A \rightarrow a/b$

(7) String Starting with zero and total length of the String is even or if String Starting with 1 then total length of the String is odd (Σ 0, 1)

Sol Regular Expression

{0 + odd} = even

+

{1 + even} = odd

$RE = 0[(0+1)((0+1)(0+1))^*] +$

$1[((0+1)(0+1))^*]$

$(0+1) \Rightarrow A \rightarrow 0/1$

$(0+1)(0+1) \Rightarrow B \rightarrow AA$

3.b 13

$[(0+1)(0+1)]^* = C \to BC/\varepsilon$

The grammar is

$S \to 0Ac/1C$

$C \to BC/\varepsilon$

$B \to AA$

$A \to 0/1$

II Converter of Language to Grammar

① Develop a grammar for the language

$L_1 = \{ wc w^R, w \text{ is in } (a,b)^* \}$

where w = word, c = Single character c w^R = coord

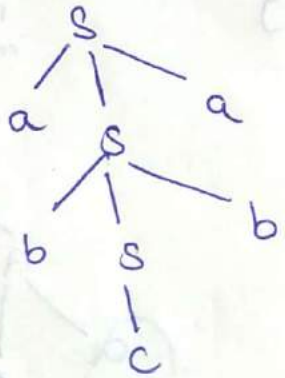
Soln

$L = \{abcba, aabcbaa \ldots\}$

The grammar

$S \to aSa/bSb/c$

input String abcba



abcba

④ Develop a grammar for the following Language.

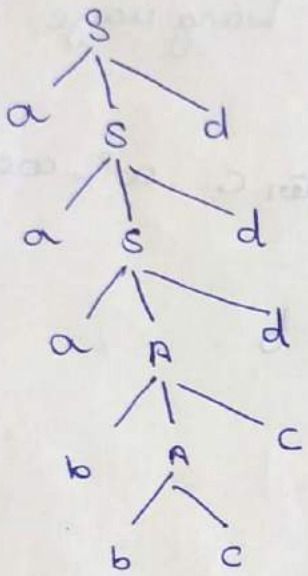
$L = \{ a^n b^m c^m d^n$ where $m, n \geq 1 \}$

Sol:

$S \rightarrow a S d \mid a A d$

$A \rightarrow b A c \mid bc$

$L = \{ abbccd, aaabcddd \cdots \}$



aaabbccddd

⑤ $L = \{ a^n b^n c^m d^m$ where $m, n \geq 1 \}$

$m \& n \geq 0$

$n=0 = a^0 b^0 c^1 d^1$ not applicable

$m=1 = cd$

$G_1 = \{ (S, A, B)(a\ b\ c\ d) \}$

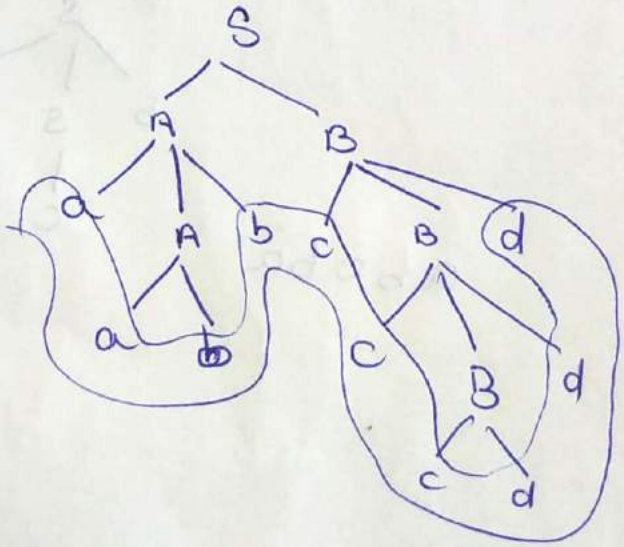
$S \rightarrow A B$

$A \rightarrow aAb \mid ab$

$B \rightarrow cBd \mid cd$

$m = 3 \quad n = 2$

aabbcccddd



Derivation

A grammar derives a String by beginning with the Start Symbol & repeatly replacing a Nonterminal by the Right Side of a production for that Non terminal is called as derivation

A replacement according to a production is Known as Derivation

Two type of Derivation

1, Left most Derivation
2, Right most Derivation / Canonical Derivation

① Left most Derivation

In this derivation the left most Nonterminal is replaced at every Step

eg Consider the production $E \rightarrow E+E \mid E*E \mid id$ The input String id + id + id

Soln $E \rightarrow E+E$

$\rightarrow E+E+E$

$\rightarrow id+id+E$

$\Rightarrow id+id+id$

② Right most Derivation Right

In this derivation the left most nonterminal is expanded first

$E \rightarrow E+\underline{E} \rightarrow E+E+\underline{E}$

$E+E+id$

$E+id+id \rightarrow id+id+id$

3. bCIT

Derive the String 'aabbabba' for Left Most derivation & Right most derivation using CFG given by

$S \rightarrow aB \mid bA$

$A \rightarrow a \mid aS \mid bAA$

$B \rightarrow b \mid bS \mid aBB$

Soln Left Most Derivation

S

aB

aaBB

aabB

aabbS

aabbaB

aabbabbS

aabbabbA

aabbabba

Derivation

$S \rightarrow aB$

$B \rightarrow aBB$

$B \rightarrow b$

$B \rightarrow bS$

a

$S \rightarrow bS$

$B \rightarrow bS$

$S \rightarrow bA$

$S \rightarrow a$

a a b b S

a a b b a b S

$S \to aB$

$B \to bS$

Right Most Derivation

$S \to aB$

$B \to aBB$

$B \to bS$

$S \to bA$

$A \to a$

$B \to bS$

$S \to bA$

$A \to a$

S

aB

aaBB

aaBbS

aaBbbA

aaBbba

aabSbba

aabbAbba

aabbabba

② Derive the String 1000111 for left most & Right most derivation using CFG where

$V = \{S, T\}$ $T = \{0, 1\}$

$P \to \{ S \to T00T$

$T \to 0T | 1T | \varepsilon \}$

Left Most

S

T00T

1T00T

10T00T

10ε00T

$S \to T00T$

$T \to 1T$

$T \to 0T$

$T \to \varepsilon$

3.b (18)

(4)

1000T T → 1T

10001T T → 1T

10001111T T → ε

100111

Right Most Derivation

S → S → TOOT

TOOT T → 1T

TOO1T T → 1T

TOO11T T → 1T

TOO111T T → ε

TOO1111ε T → 1T

TOO11111ε T → 0T

1TOO11111ε T → ε

10TOO11111ε

10OO111

3-b (19)

Construct the derivation tree for the String aabb abba for CGF given by

$S \to aB \mid bA$

$A \to a \mid aS \mid bAA$

$B \to b \mid bS \mid aBB$

Sol:

S

aB

aaBB

aabbS

aabbaB

aabbabS

aabbabbA

aabbabba

$S \to aB$

$B \to aBB$

$B \to b$

$B \to bS$

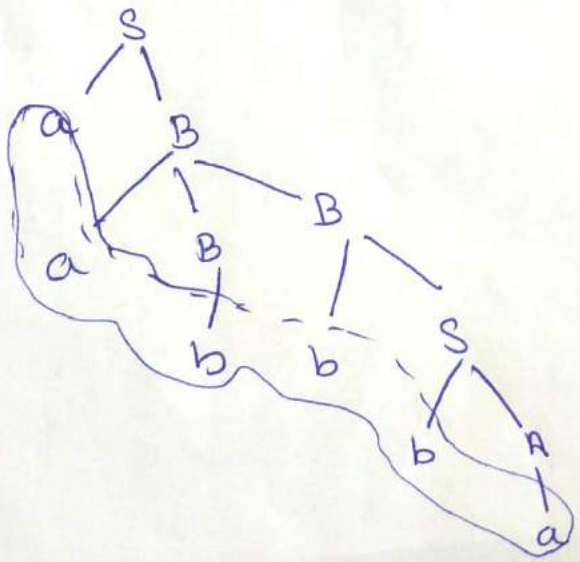
$S \to aB$

$B \to bS$

$S \to bA$

$A \to a$

aa bb a bb a



3.b(20)

Sentential Forms:-

A Sentential form is the Start Symbol 'S' of a grammar (or) any String $(V \cup T)^*$ that can be derived from 'S'

Consider the Grammar $V = \{S, B\}$, $T = \{a, b\}$ $P = \{S \to aS, S \to B, B \to bB, B \to \varepsilon\}$ and input String abb

A derivation using this grammar might look like this

$S \to aS$

aB

abB

$abbB$

abb

$S \to B$

$B \to bB$

$B \to bB$

$B \to \varepsilon$

Sentential form

$\{S, aS, abB, abbB, abb\}$

Method to find whether a String belong to a grammar or not

① Start with the Symbol and Choose the closest production that matches to the given String

② Replace the Value with its most appropriate production. Repeat the process until the String is generated / until no other production left

Verify whether its the Grammar

$S \rightarrow 0B / 1A$

$A \rightarrow 0 / 0S / 1AA / \varepsilon$

$B \rightarrow 1 / 1S / 0BB$

Generate the String 00110101

Sol

$S \rightarrow 0\underline{B}$

$\rightarrow 0\ 0\underline{BB}$

$\rightarrow 00\underline{1}\ \underline{B}$

$\rightarrow 0011\ \underline{S}$

$\rightarrow 00110\ \underline{B}\ S$

$\rightarrow 00110\ 0\ B$

$\rightarrow 00110101$

$S \rightarrow 0B$

$B \rightarrow 0BB$

$B \rightarrow 1$

$B \rightarrow 1S$

$S \rightarrow 0B$

$B \rightarrow 1S$

$S \rightarrow 0B$

$B \rightarrow 1$

00110101 =

Language:-

Set of String generate by any grammar is known as language.

Identify the Lang generate by grammar.

① $A \to aA/a$ find Lang

Soln $A \to a$

$\therefore L = \{a$

$A \to aA$

$L = \{aa, aaa \ldots$

$L = \{a, aa, aaa \ldots$

$L = a^+$

$S \to bS \mid \varepsilon$

Soln $S \to \varepsilon$

$L = \{\varepsilon$

$S \to bS$

$L = \{b, bb, bbb \ldots$

$\therefore L = \{\varepsilon, b, bb, bbb \ldots\}$

$L = a^*$

4) $S \to bSAB$, $A \to a$, $B \to b$

$S \to AB$

$L = \{ab, \}$

5) $S \to aS \mid bS \mid a$

Son

① $S \to aS$

$L = \{aaa\}$

$S \to aS \Rightarrow S \to aS$

$aaS \Rightarrow S \to a$

aaa

② $S \to aS$

$L = \{ab$

$S \to aS$

$ab \}$

aba

$S \to bS$

$S \to \varepsilon$

③ $S \to a$

$L = a$

④ $S \to bS$

$\to baS$

$\to baa$

$L = \{a, aaa, aba, baa \cdots$

$L = (a+b)^* a$

5) $S \to aS \mid bS \mid \varepsilon$

① $S \to \varepsilon$

$L = \{\varepsilon$

2 $S \to aS$

$\to a \cdot \varepsilon$

$= a$

$L = \{\varepsilon, a$

3-b(22)

3) $S \to bS$

$\to b \cdot \varepsilon$

$\to b$

$L = \{\varepsilon, a, b$

4) $S \to b^a S$

$\to abS$

$\to ab\varepsilon$

$\to ab$

5) $S \to b^{-S}$

$\to baS$

$\to ba\varepsilon$

$\to ba$

$L = \{\varepsilon, a, b, ab, ba, bb, aa \ldots$

$(a+b)^*$