Unit-2. Finite Automata

2.1 Definition of Automata

2.2 Why study Automata Theory?

2.2.1 Introduction to finite Automata

2.2.2 Structural representations

2.2.3 Automata and Complexity

2.3 Descriptions of Finite Automata, Transition Systems, Transition Functions

2.4 Deterministic Finite Automata (DFA)

2.5 Nondeterministic Finite Automata (NFA)

2.6 The Equivalence of DFA and NFA

2.7 Minimization of DFA

2.8 Finite Automata with ε-Moves

2.9 Melay and Moore Machines: Definition and Examples

2.10 Applications of Finite Automata

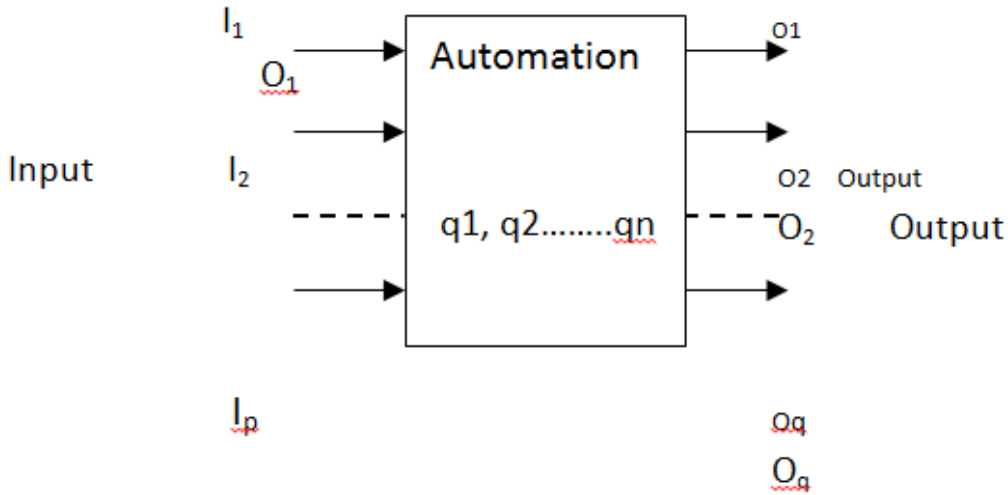
2.1 DEFINATION OF AUTOMATA:

Automation is defined as a system in which energy, materials and information are transformed, transmitted and used for performing some functions without direct participation of man.

Examples:

- automatic machine tools
- automatic packing machines and
- automatic photo printing machines

Model of a discrete automation



The characteristics of Automation

- *Input*. For every discrete instants of the given time t_1, t_2…… , by considering input values as I_1, I_2……., each of which can take a finite number of fixed values from the input alphabet ∑,are applied to the input side of model shown in fig 2.1.
- *Output*. The O_1, $O_{2,}$,……. and O_q are the outputs of the automaton model, and every of which can take finite numbers of fixed values from the output O.
- *States*. At any instantaneous of time the automaton can be in one of the states q_1,q_2,…..q_n.
- *State relation*. The automaton's next state at any instant of time is determined by the present state and the present input.
- *Output relation*. Output is associated to either state only or to both the input and the state. Note that at any instant of time the automaton is in some State. After reading an input symbol, the automaton moves to a next state which is specified by the state relation.

2.2 WHY STUDY AUTOMATA THEORY?

- ✓ The study of automata theory is an essential measure of the core of Computer Science.
- ✓ Automata are helpful in making model which are used for Software,
- ✓ For, designing digital circuits as well as checking the behavior of digital circuit.
- ✓ For, Lexical Analyzer of the usual compiler that breaks the input text into logical unit as key words, identifier, and punctuation marks.
- ✓ For, searching large text may be in file or on web to find out occurrence of words, phrases or any other given patterns.
- ✓ For verifying system of all types that have a finite number of individual state such as communication protocol or protocol for secure exchange of information.
- ✓ For, natural language processing.

2.2.1 INTRODUCTION TO FINITE AUTOMATA

- Automata theory is that the study of abstract computing machines or devices that are a useful model for several important kinds of hardware and software.

Example: we could implement it in hardware as a circuit or as an easy form of program which will make decisions looking only at a limited amount of data or using the position within the code itself to make the choice.

2.2.2 STRUCTURAL REPRESENTATIONS

- **Grammars**
- **Regular Expressions**

Grammars

Grammars are useful models in designing software which processes data with a recursive structure.

Example: **Parser** (the component of a compile)

It deals with the recursively nested features of the standard programing language like expressions, arithmetic, conditional and so on.

Example: a grammatical rule like E-> E + E / E * E states that an expression can be formed by taking any two expressions and connecting them by a plus or multiplication sign.

- **Regular Expressions**

  Regular Expressions likewise denote the structure of data especially text strings. The patterns of strings describe are exactly the same as what can be described by finite automata. The style of such expressions varies significantly from the grammars.

  Example: The UNIX style regular expression [A-Z][a-z]*[A-Z][A-Z] represents capitalized words followed by a space and two capital letters. This expression represents patterns in text that could be a city or state.

  Example: Maharashtra or MS.

2.2.3 AUTOMATA AND COMPLEXITY

Automata are essential for the study of the limits of computation.

Two important issues:

- What can a computer do at all?

  The study for this is called **decidability** and the problems that can be solved by computer are called **decidable**.

- What can a computer do efficiently?

  The study for this is called **intractability** and the problems that can be solved by a computer using no more time than some slowly growing function of the size of the input are called **tractable**.

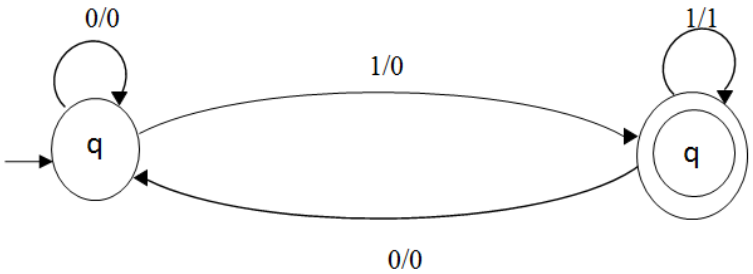
2.3 DESCRIPTIONS OF FINITE AUTOMATA, TRANSITION SYSTEMS, TRANSITION FUNCTIONS:

DESCRIPTIONS:

- **Finite automaton**, is a mathematical model consist of Finite set of status & set of transitions from state to state that occurs on an input symbol choose from an alphabet.
- **A directed graph** called a transition diagram and is associated with finite automata as follows:
- **The vertices of graph** corresponds to states of finite automata if there is a transitions from q to state p on input k then there is an arc labeled k from state
- **The finite automata accepts** the string x if sequence of transition corresponds to symbol of x. leads to from start state to an accepting state (final state).
- **The starting state** is indicated by an arrow label start .final state or acceptor state is indicated by double circle or encircle. Final state or acceptor state is indicated by double circle or encircle.

TRANSITION SYSTEMS:

A transition graph or a transition system is defined as a finite directed labeled graph in which each vertex or node denotes a state and the directed edges specify the state transition and the edges between states are labeled with “input / output".



Definition: A transition system

A transition system is a 5-tuple $(Q, \sum, \delta, q_0, F)$,where

(a) Q is the finite nonempty set of states, $\sum$ is an input alphabet, and F is the set of final states.

(b) q_0 is starting state and q_0 is in Q and q_0 is nonempty and

(c) δ is a transition function mapping as $\delta : Q \times \sum^* \rightarrow Q$

- The graph starts at the vertex q_0, goes along a set of edges, and reaches the vertex q_1.
- The w is the concatenation of the label of all the edges encountered.

Definition: A transition system accepts a string

A transition system **accepts a string denoted** as **w** in $\sum^*$

if (a) there exists a path which started from some initial state, which goes along the arrows, and get terminates at some final state, and

(b) the path values obtained by concatenation of all edge labels of the path is equal to w.

❖ **ACCEPTABILITY OF A STRING BY A FINITE AUTOMATON**

Definition:

1. The acceptability of a string by the final state.
2. A string x is accepted by a finite automaton $M=(Q,\sum,\delta,q_0,F)$ if $\delta(q_0,x)=q$ for some $q \varepsilon F$.

Note: A final state is also called as accepting state.

2.4 DETERMINISTIC FINITE AUTOMATA (DFA)

Definition:

A Deterministic Finite Automation can be represented by a 5-quin tuple $(Q, \sum, \delta, q_0, F)$, where

Q represents a finite nonempty set of states.

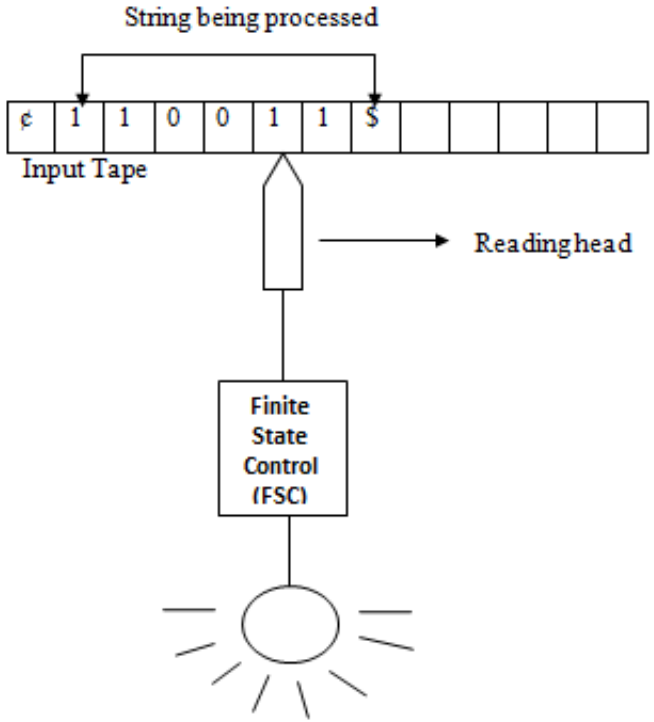
$\sum$ is a finite nonempty set of inputs called input alphabet.

δ is a function which maps $Q \times \sum$ into Q and is usually called direct transition function. δ is also called the next state function.

$q_0 \varepsilon Q$ is the initial state and

F (is subset of Q) is the set of final states. Assumed as there may be more than one final state.

Block diagram of a finite automaton

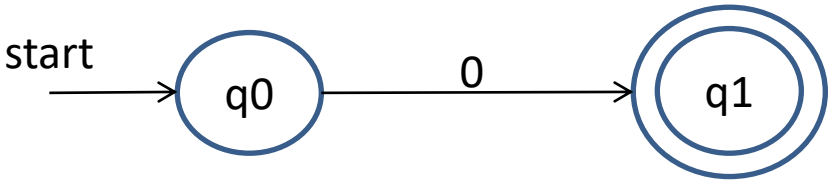


❖ The Various components are as follows:

- ***Input tape:*** The input tape is divided into squares, each square containing a single symbol from the input alphabet ∑. The end square of the tape contain end markers ¢ at the left end and $ at the right end. Absence of end markers indicates that the tape is of infinite length. The left-to-right sequence of symbols between the end-markers is the input string to be processed.
- ***Reading head:*** The head examines only one square at a time and can move one square either to the left or to the right. We restrict the movement of R-head only to the right side.
- ***Finite control****:* The input to the finite control will be usually, symbol under the R-head, say a, or the present state of the machine, say q, to give the following outputs (a) A motion of R-head along the tape to the next square (In some a null move, i.e. R-head remaining to the same square is permitted); (b)the next state of the finite state machine given by δ(q, a).

Example:

A finite automata for language containing string having only 0 over an $\sum=\{0\}$ $\sum=^{*}\{0\}$



2.5 NON DETERMINISTIC FINITE AUTOMATA (NDFA/NFA):

Nondeterministic Finite Automata is a five tuple $(Q, \sum, \delta, q_0, F)$ where;

Q is a finite nonempty set of states.

$\sum$ is a finite nonempty set of i/p symbol.

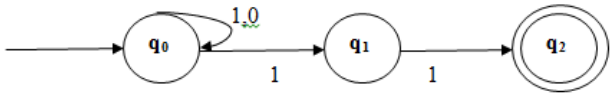
δ Is a transition function mapping from $Q \times \sum \rightarrow 2^{Q.}$

q_0 belongs to Q is initial state.

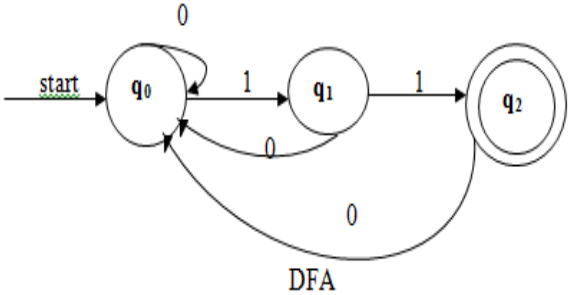
F is final state. F is subset of Q.

Note: The finite automation model allow zero, one, or more transitions from a state on same input symbol is called NFA

Draw DFA and NFA for string ending with 11.



$\sum^*$={011, 01011,010111,101011,.....} 0 NFA



DFA

M=({q_0, q_1, q_2}, {0,1}, δ,q_0,{q_2})

Where , δ is state transition function &defined as,

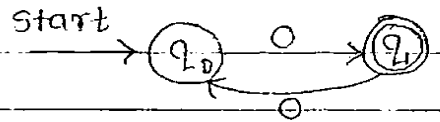
δ	0	1
q_0	q_0	q_1
q_1	q_0	q_2
(q_2)	q_0	ø

③ Design a finite automata having the strin of odd length over an $\Sigma = \{0\}$ / $\Sigma = \{1\}$ or even

$\Rightarrow$ $\{0\}^{n}_{odd}$

① $\Sigma^* = \{0, 000, 00000, \ldots\}$

L	w
1	0
3	000
5	00000



$A = (Q, \Sigma, \delta, q_0, F)$

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

where,

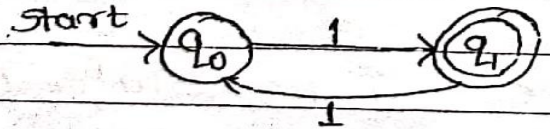
δ is transition fun mapping as,

δ	0
$\rightarrow q_0$	q_1
(q_1)	q_0

Strings having odd number of 1's

odd

② $\Sigma^* = \{1, 111, 11111, \ldots\}$



$A = (Q, \Sigma, \delta, q_0, F)$

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

where, δ is transition fun mapping as,

δ	1
$\rightarrow q_0$	q_1
(q_1)	q_0

Strings having even number of 0's

even

(3) $\Sigma^* = \{\epsilon, 00, 0000, \ldots\ldots\}$

start 

$A = (Q, \Sigma, \delta, q_0, F)$

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

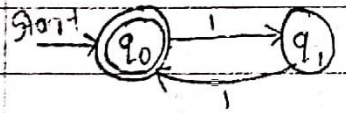
where,

δ is transition fun mapping as,

δ	0
$\rightarrow (q_0)$	q_1
q_1	q_0

Strings having even number of 1's

even (4) $\Sigma^* = \{\epsilon, 11, 1111, \ldots\}$



L	w
0	ϵ
2	11
4	1111
.	..

$A = (Q, \Sigma, \delta, q_0, F)$

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

where,

δ is transition fun mapping as,

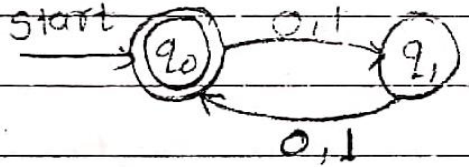
δ	1
$\rightarrow (q_0)$	q_0, q_1
q_1	q_0

Strings having even length over given alphabet

even

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

$\Rightarrow$ Strings - $\Sigma = \{\epsilon, 01, 10, 00, 11 - - - \}$



$A = (Q, \Sigma, \delta, q_0, F)$

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

where,

δ is transition fun is mapping as,

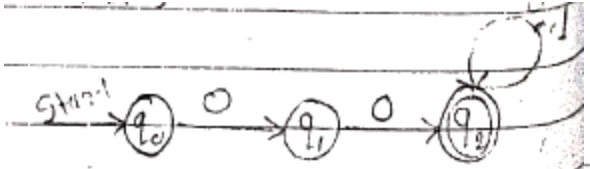
δ	0	1
$\rightarrow (q_0)$	q_1	q_1
q_1	q_0	q_0

$\rightarrow$ transition Table.

(8) FA for the language containing strings starting with two consecutive zero's. (must 2 zero starting)

→ $\delta 0 n^n$

$\Sigma^* = \{00, 0011, 00101, \ldots\}$



$A = (Q, \Sigma, \delta, q_0, F)$

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

where,

δ is transition fun mapping as,

δ	0	1
→ q_0	q_1	ϕ
q_1	q_2	ϕ
(*) q_2	q_2	q_2

⑦ Design a FA for the language having strings in which occurences of 0's & 1's are even with cardinality even.

1)

$\Rightarrow \Sigma^* = \{\epsilon, 00, 11, 0011, 001100, \ldots\}$

$A = (Q, \Sigma, \delta, q_0, F)$

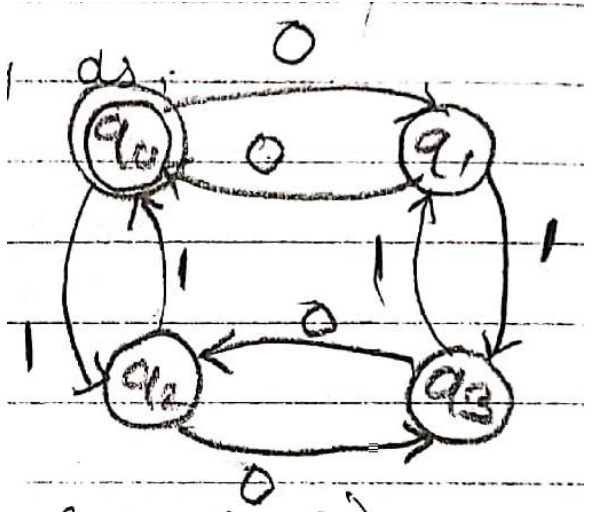
$= (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_0\})$

where,

δ is transitin fun mapping as:

δ	0	1
$\rightarrow (q_0)$	q_1	q_1
q_1	q_0	q_0

↳ Transition table.



(8) FA for the language containing strings starting with two consecutive zero's. (must 2 zero starting)

$\Rightarrow 00(0+1)^*$

$\Sigma^* = \{00, 000, 0011, 00101, \ldots\}$

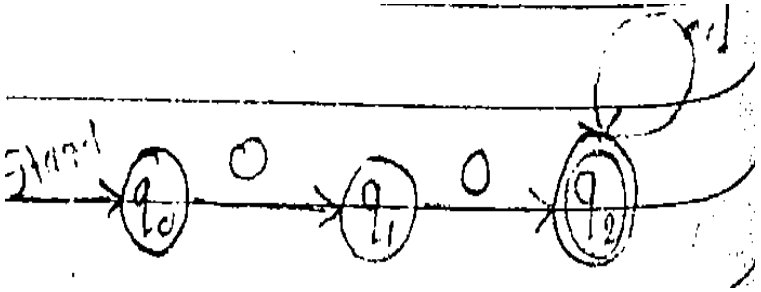
$A = (Q, \Sigma, \delta, q_0, F)$

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

where,

δ is transition fun mapping as

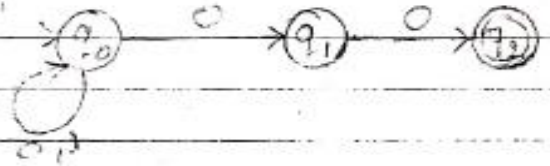
δ	0	1
$\rightarrow q_0$	q_1	ϕ
q_1	q_2	ϕ
(*) q_2	q_2	q_2



FA for the language containing string ending with two consecutive zero's.

Soln:-

$\Sigma^* = \{10100, 100, 10100100, \ldots\}$



$A = (Q, \Sigma, \delta, q_0, F)$

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

where,

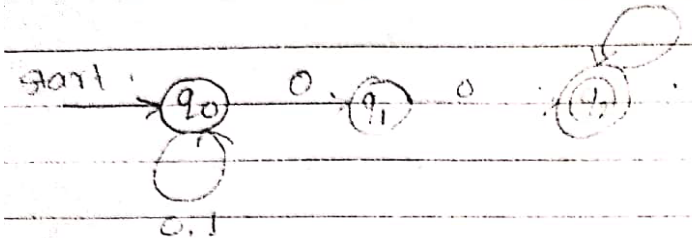
δ is transition fun mapping as,

δ	0	1
$\rightarrow q_0$	$\{q_1, q_0\}$	q_0
q_1	q_2	ϕ
(*) q_2	ϕ	ϕ

(10) FA for the language having the string with 2 consecutive zero's.

NFA

$\Sigma^* = \{ 0100011, 0110100, 001100, \ldots \}$



$A = (Q, \Sigma, \delta, q_0, F)$

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

where,

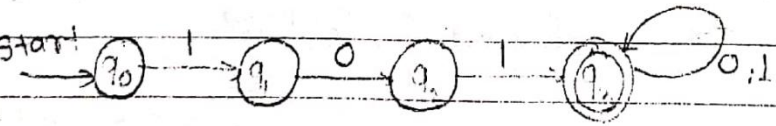
δ is transition fun mapping as,

δ	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	q_0
q_1	q_2	ϕ
(q_2)	q_2	q_2

(11) FA for the language having starting string with 101.

⇒ Soln-

$\Sigma^* = \{101, 10111, 101000, \ldots\}$



$A = (Q, \Sigma, \delta, q_0, F)$

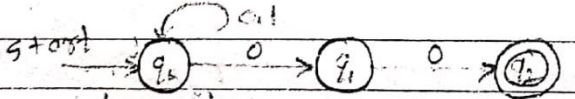
$= (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, q_0, \{q_3\})$

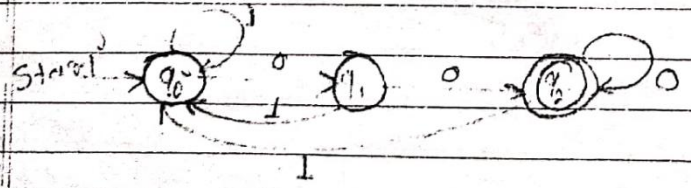
where,

δ is transition fun mapping as,

δ	0	1
$\rightarrow q_0$	ϕ	q_1
q_1	q_2	ϕ
q_2	ϕ	q_3
(q_3)	q_3	q_3

H.W.① Design a DFA which excepts the string ending with consecutive zero (00) over an $\Sigma = \{0, 1\}$





$A = (Q, \Sigma, \delta, q_0, F)$

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

where,

δ is transition fun mapping as,

δ	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_2	q_0
(q_2)	q_2	q_0

→ Transition table.

② Design a DFA having strings ending with 101.

⇒ Soln -

→ NFA

$\Sigma^* = \{ 00101, 11011101, 01001101 \ldots \}$

$A = (Q, \Sigma, \delta, q_0, F)$

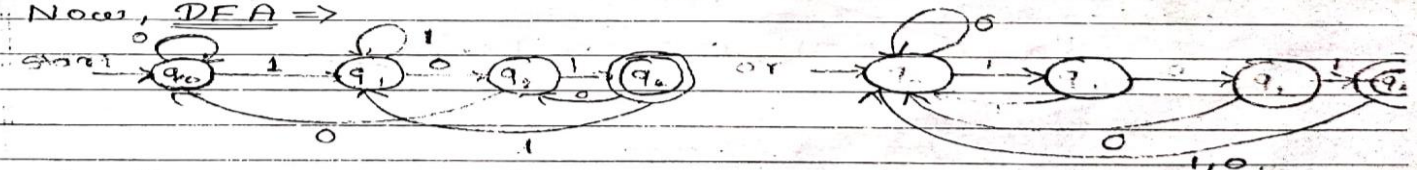
$= (\{q_0, q_1, q_2, q_3\}, \{0,1\}, \delta, q_0, \{q_3\})$

where,

δ is transition fun mapping as,

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

Now, DFA =>



or

$A = (Q, \Sigma, \delta, q_0, F)$

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

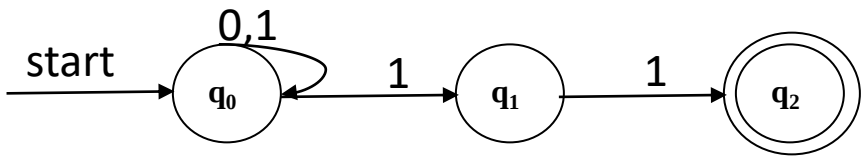
where,

δ is transition fun mapping as.

δ	0	1
$\rightarrow q_0$	q_0	q_1
q_1	q_2	q_1
q_2	ϕ	q_3
(q_3)	q_2	q_1

Draw DFA for string ending with 11

$\sum^*$={ 11, 011, 111, 0011, 1111, 0111,1011, 01011, 010111, 101011, …… }



2.6 The Equivalence of DFA and NFA

- NFA's are generally easier to program than DFA.
- Remarkably, for any NFA N there is a DFA D exists,

  such that L(D) = L(N), and vice versa.
- It means, the language accepted by the DFA is same that accepted by the NFA.
- An algorithm converts any non-deterministic finite state automaton (NFA) to an equivalent deterministic finite state automaton (DFA).
- An automaton B can be generically constructed from automaton A.

Algorithm converts NFA to its equivalent DFA:

Given, NFA N = (Σ, Q, q0, F, δ) we have to construct equivalent DFA D= (Σ, Q', q'0, F', δ') by using following steps.

Step 1. Initially Q' = ∅.

Step 2. Add q0 to Q'.

Step 3. For each state in Q find the set of possible states for each input symbol using N's transition table, δ. Add this new set of states to Q', if it is not already present in it.

Step 4. F' is the set of final states of D. F' will contain all the states in Q' that contain in F

Consider Example: NFA $N = (\Sigma, Q, q_0, F, \delta)$

Where, $\Sigma = \{a, b, c\}$

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

$F = \{q_2\}$

δ the transition table is given bellow.

State/Input	**a**	**b**	**c**
q_0	$\{q_0, q_1\}$	q_0	q_2
q_1	$\emptyset$	q_2	$\emptyset$
***q_2**	$\emptyset$	$\emptyset$	$\emptyset$

By following the steps of algorithm of NFA to DFA conversion, the equivalent DFA $D= (\Sigma, Q', q'_0, F', \delta')$ of a given NFA:

$\Sigma = \{a, b\}$

$Q' = \{q_0, [q_0, q_1], q_2, [q_0, q_2]\}$

$q'_0 = q_0$

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

δ' is:

State/Input	**a**	**b**	**c**
q_0	$[q_0, q_1]$	q_0	q_2
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_2]$	q_2
***q_2**	–	–	–
***$[q_0, q_2]$**	$[q_0, q_1]$	q_0	q_2

2.7 MINIMIZATION OF DFA:

Minimization of DFA is a process to convert a given DFA to its equivalent DFA with minimum number of states.

We construct an automaton with minimum number of states equivalent to a given automaton M.

As our interest lies only in strings accepted by M, what really matters is whether a state is a final state or not. We define some relations in Q.

Definition: Two states q_1 and q_2 are equivalent (denoted by $q_1 \equiv q_2$) if both $\delta(q_1, x)$ and $\delta(q_2, x)$ are final states ,or both of them are non final states for all $x \in \sum^*$.

As it is difficult to construct $\delta(q_1, x)$ and $\delta(q_2, x)$ for all x is in $\sum^*$ (there are infinite number of strings in $\sum^*$).

Definition: Two states q_1 and q_2 are k-equivalent ($k \geq 0$) if both $\delta(q_1, x)$ and $\delta(q_2, x)$ are final states or both non final states for all strings x of length k or less. In particular, any two final states are 0-equivalent and any two non final states are also 0-equivalent.

Consider, DFA M = (Q, Σ, q0, δ, F) recognizes a language L. An equivalent minimized DFA M'= (Q', Σ, q0, δ', F') can be constructed for language L with following steps.

Step 1: Partition the Q (set of states) into two sets, one set will contain all final states and other set will contain non-final states. This partition is called P_0.
Step 2: Assign k = 1 for initialization.
Step 3: Find out P_k by partitioning the different sets of P_{k-1}. Each set of P_{k-1}, take all possible pair of states. Divide the sets into different sets in $P_{k,}$ if two states of a set are distinguishable.
Step 4: Partitioning process will stop when $P_k = P_{k-1}$ (There is no change in partition)
Step 5: All states of one set are merged into one. Number of states in minimized DFA will be equal to number of sets in P_k.

Two states (p, q) are distinguishable in partition P_k if for any input symbol a, δ(p, a) and δ (q, a) are in different sets in partition P_{k-1}.

Two states (p, q) are equivalent if $\delta(p, w) \in F$ and $\delta(q, w) \in F$

$\delta(p, w) \notin F$ and $\delta(q, w) \notin F$

For the minimization of DFA we have to find equivalence relationship from **0** equivalent to **n** number of equivalent till we get the same last two states.

If |w|=0 then p, q are referred as 0 equivalent

If |w|=1 then p, q are referred as 1 equivalent

:

If |w|=n then p, q are referred as n equivalent.

If |w|=n then p, q are referred as n equivalent.

States/Input	a	b
q0	q1	q2
q1	q1	q3
q2	q1	q2
q3	q1	q4
***q4**	q1	q2

Table to find equivalent relation:

Equivalence	States Partition	
0 equivalent	{q0, q1, q2, q3} {q4}	Last 2 states are same
1-equivalent	{q0, q1, q2} {q3} {q4}	
2-equivalent	{q0, q2} {q1} {q3} {q4}	
3-equivalent	{q0, q2} {q1} {q3} {q4}	

The transition table of minimized DFA of a given DFA is given as follows.

States/Input	a	b
q0q2	q1	q0q2
q1	q1	q3
q3	q1	q4
***q4**	q1	q0q2

2.8 FINITE AUTOMATA WITH E-MOVES:

- Model of non-deterministic finite automata may be entered to include transition on empty symbol ε

  formally non-deterministic automata with ε moves is defined as quintuple.

  M= (Q , ∑ , δ , q_0 ,F) with all components as before but ð the transition function maps as,

  $\delta: Q \times \sum \times \{\varepsilon\} \times 2^Q$.

* Finite Automata with ϵ-moves :-

Model of NFA may be entered to include transition on empty string (ϵ, ^, λ). Formally NFA with ϵ-moves is defined as quin tuple as,

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

Q - is finite non-empty set of states.

Σ - is finite non-empty set of input symbols.

δ - is transition function mapping as,

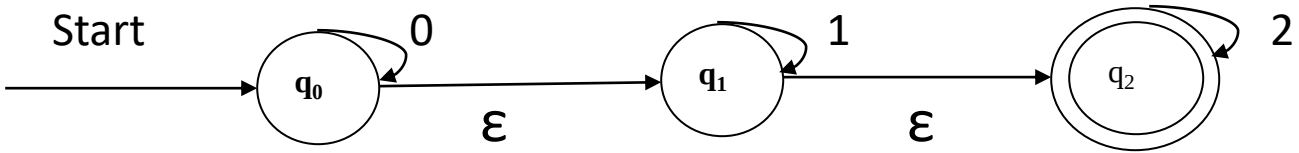
$\delta : Q \times \Sigma \times \{\epsilon\} \rightarrow 2^Q$

q_0 - is starting state, $q_0 \in Q$

F - is set of final state, $F \subseteq Q$

Example::

Transition diagram which accepts string consisting of **any 0's followed any no. of 1's followed any no. of 0's.**



M= ($\{q_0, q_1, q_2\}$, ∑, δ :∑ , ∪ {ε}, q_0, $\{q_2\}$) is finite state machine

where δ is state machine function & defined as,

δ	ε	0	1	2
→ q_0	q_1	q_0	ø	ø
q_1	q_2	ø	q_1	ø
q_2	ø	ø	ø	q_2

* ε - CLOSURE of q :-

ε - CLOSURE of q is defined as the set of vertices 'p', such that their is a path from q to p labeled ε, where p is in P.

ε-closure of q_0 ={q_0,q_1,q_2}

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

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

ε-closure ($\mathbf{q_2}$) = {q_2}

* ε - CLOSURE of q :-

ε - CLOUSURE of q is defined as the set of vertices 'p', such that their is a path from q to p labeled ε, where p is in P.

* $\bar{\delta}$ (Extended transition function) :-

δ is defined as,

① $\bar{\delta}(q, \varepsilon) = \varepsilon$ - CLOUSURE (q),

② For 'w' in Σ^* and a in Σ,

$\bar{\delta}(q, wa) = \varepsilon$-CLOUSURE (P), where,

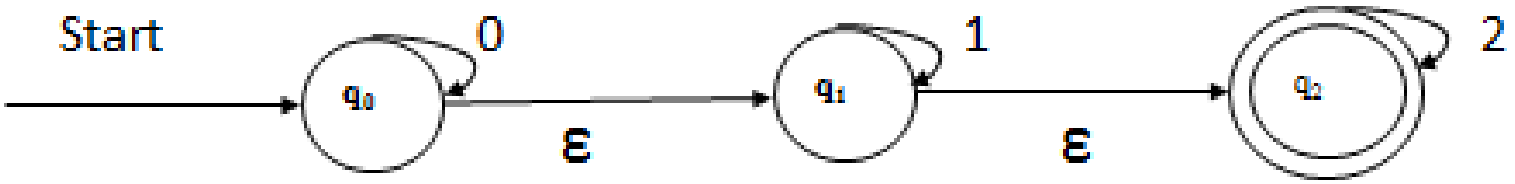
$P = \{p$ / for some r in $\bar{\delta}(q, w)$, p is in $\delta(r, a)\}$

It is convinient to extend δ & $\bar{\delta}$ two all set of states of by,

③ $\delta(R, a) = \bigcup_{q \text{ in } R} \delta(q, a)$

$\bigcup_{q \text{ in } R} \delta(q, a)$ or

④ $\bar{\delta}(R, w) = \bigcup_{q \text{ in } R} \bar{\delta}(q, w)$.



1. $\overline{\delta}(q_0, \varepsilon) = \varepsilon\text{-closure}(q_0)$
   $= \{q_0, q_1, q_2\}$
2. $\overline{\delta}(q_0, 0) = \varepsilon\text{-closure}(\delta(q_0, 0))$
   $= \varepsilon\text{-closure}(\delta(\delta\, q_0, \varepsilon), 0)$
   $= \varepsilon\text{-closure}(\delta(q_0, q_1, q_2, q_3), 0)$
   $= \varepsilon\text{-closure}(\delta(q_0, 0) \cup \delta(q_1, 0) \cup (q_2, 0))$
   $= \varepsilon\text{-closure}\ q_0 \cup \emptyset \cup \emptyset$
   $= \varepsilon\text{-closure}\ \{q_0\}$
   $= \{q_0, q_1, q_2\}$

3. $\overline{\delta}(q_0,1) = \varepsilon\text{-closure}(\delta(\overline{q_0},1))$

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

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

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

$= \delta(q_0,0) \cup \delta(q_0,1) \cup \delta(q_2,1)$

4. $\overline{\delta}(q_0,2) = \varepsilon\text{-closure}\ \delta(q_0,2)$

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

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

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

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

$= \varepsilon\text{-closure}\ \emptyset \cup \emptyset \cup q_2$

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

$= \{q_2\}$

5. $\overline{\delta}(q,0)$ = ε-closure $\delta(\overline{q},0)$

= ε-closure $\delta(\overline{\delta}(q,\varepsilon),0)$

= ε-closure $\delta(\{q_1,q_2\},0)$

=ε-closure $\delta(\{q_1,q_2\},0)$

=ε-closure $\delta(q,0) \cup \delta(q_2,0)$

=ε-closure $\varnothing \cup \varnothing$

= $\varnothing$

6. $\overline{\delta}(q_1,1)$ = ε-closure $\delta(q,1)$

= ε-closure $\delta(\delta(q,\varepsilon),1)$

=ε-closure $\delta(q_1,1) \cup \delta(q_2,1)$

=$\{q_1, q_2\}$

7. $\overline{\delta}\ (q_1,2) = \varepsilon\text{-closure}\ \delta\ (\overline{q_1},2)$

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

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

$= \varepsilon\text{-closure}\ \delta\ \emptyset \cup q_2$

$= \varepsilon\text{-closure}\ \delta\ \{q_2\}$

$= q_2$

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

8. $\overline{\delta}(q_0, 001122) = \varepsilon\text{-closure } \delta(q_0, \overline{0}01122)$

$= \varepsilon\text{-closure } \delta\ \overline{(\delta}(q_0,0),0ø122)$

$= \varepsilon\text{-closure } \delta\ \{q_0,q_1,q_2\},0,01122$

$= \varepsilon\text{-closure } \delta\ (q_0, 01122)$

$= (\{q_0, q_1, q_2\},01122)$

$= \varepsilon\text{-closure } \delta\ (\overline{(}\{q_0,q_1,q_2\},0)1122)$

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

$= \varepsilon\text{-closure } \delta\ (q_0 \cup ø \cup ø, 1122)$

$= \varepsilon\text{-closure } \delta(q_0, 1122)$

$= (\{q_0,q_1,q_2\},1122)$