Formal Languages & Automata Theory

UNIT-1

→ <u>Automata Theory</u>:- It is the study of abstract computing devices or machines. The main goal is to describe precisely the boundary b/n what a computing machine could do and what it could not do.

Automata Theory is an exciting, theoretical branch of computer science where it established its roots during the 20th century, as mathematicians began developing machines which imitated certain features of man, completing calculations more quickly and reliably.

The word automation itself denotes the automatic processes carrying out the production of specific processes. Through automata, computer scientists are able to understand how machines compute functions and solve problems.

Automata are abstract machines that perform computations on an input by moving through series of states.

→ Every device has low level calculations that we called as abstract machines.

input $\xrightarrow{\text{takes}}$ [Abstract Machine] $\xrightarrow{\text{produces}}$ output.

For examples, when we take switch,

OFF state ⟶ [Switch] ⟶ ON state.
push

→ These type of abstract machines will be used to implement algorithms efficiently. The abstract machines we call it as "automata". If this machine implements in a finite no. of states then it is called "Finite Automata".

→ Finite Automata is a abstract machine which takes the input and processes it and gives output.

→ Introduction to finite automata:-

Finite automata are useful model for many kinds of s/w & H/w

Some of the most important kinds are

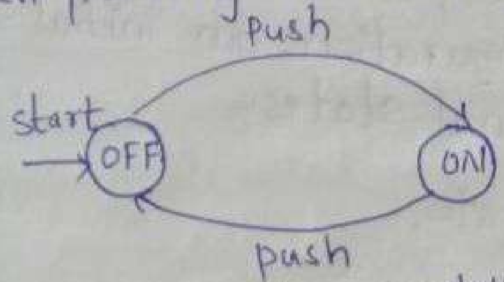
1. Software for designing and checking the behavior of digital circuits.
2. The "lexical analyzer" of a typical compiler, i.e the compiler component that breaks the i/p next into logical units such as identifiers, keywords and punctation.
3. Software for scanning large bodies of text, such as collection of web pages, to find occurrences of words, phrases or other patterns.
4. Software for verifying the systems of all types that have a finite no of distinct states such as communication protocols or protocols for secure exchange of information.

→ The finite automata deals with the finite no of states. The purpose of a state is to remember the relevant portion of the system's history. The advantage of having only a finite no. of states is that we can implement the system with a fixed set of resources.

→ Ex:-
Lets implement finite automata on a switch. The device remembers whether it is the "ON" state or the "OFF" state and it allows the user to press a button whose effect

2023.06.27 08:47

is different, depending on the state of the switch. That is, if the switch is in the "OFF" state, then pressing the button changes it to the "ON" state and if the switch is in the "ON" state, then pressing the same button turns it to "off" state.



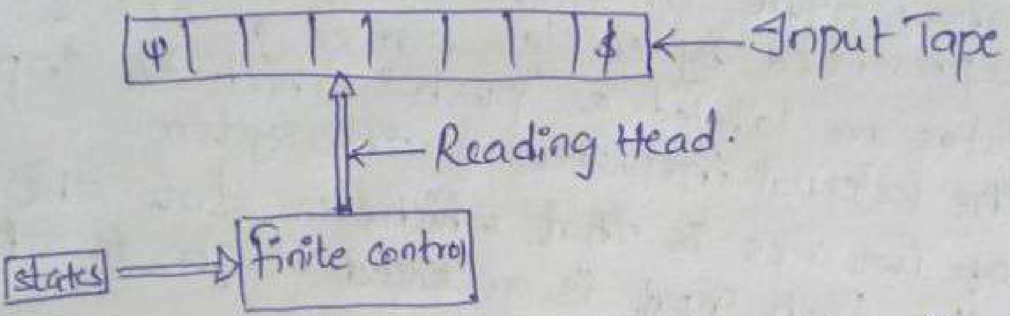
A Finite Automaton modelling of an ON|OFF Switch.

→ From the above figure, in finite automata modeling the states are represented by circles named as off & ON. Arcs b/n states are labeled as "push" which are inputs represents the external influences on the system. The intent of two arcs is that whichever state the system is in, when the Push input is received it goes to other state

→ Structural Representation of finite Automata (FA):-

A Finite Automata consists of a finite memory called input tape, a finite-nonempty set of states, an input alphabet, a read-only head, a transition function which defines the change of configuration, an initial state & a finite-non empty set of final states.

→ Model of finite automata:-



→ The input tape is divided into cells and each cell contains one symbol from the input alphabet.

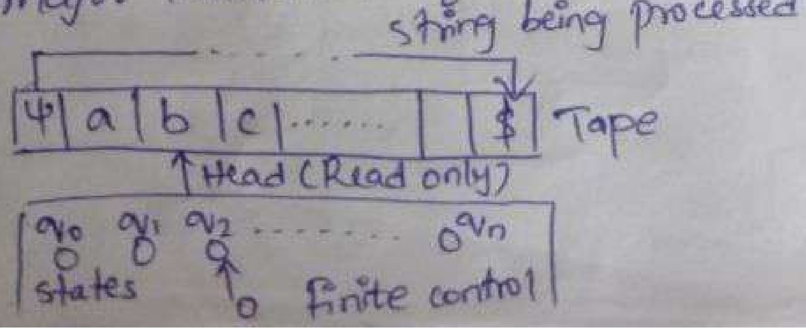
→ The symbol 'φ' is used at the leftmost cell and symbol $ is used at the rightmost cell to indicate the beginning and end of the input tape.

→ The head reads one symbol on the input tape & finite control controls the next configuration.

→ The head can read either from left-to-right or from right-to-left one cell at a time. The head can't write and can't move backward.

→ So, FA can't remember its previous read symbols. This is major limitation of FA.

Ex:-



→ Automata and complexity:-

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

1. What can a computer do at all? This study is called "decidability" and the problems that can be solved by computer are called "decidable".
2. What can a computer do efficiently? This study 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".

2023.06.27 08:47

→ The Central concepts of Automata Theory:-

→ Alphabet:-
An alphabet is a finite, non-empty set of symbols denoted by Σ.

Ex:- 1. $\Sigma = \{0,1\}$ → the binary alphabet

2. $\Sigma = \{a, b, \dots z\}$ → set of lowercase letters.

3. $\Sigma = \{A, B, C, \dots Z\}$ → set of uppercase letters.

4. Σ = set of all ASCII characters.

→ String:-
A string is a finite collection of symbols chosen from alphabet.

Ex:- 001011 is a string in set of binary alphabet $\Sigma = \{0,1\}$.
abcaabd is a string in set of lowercase alphabet $\Sigma = \{a, b, \dots z\}$

→ Empty String:-
It is a string with zero occurrences of symbols denoted by 'ϵ'. The length of empty string is zero.

→ Length of a string:-
The length of string is calculated by no. of symbols in a given string. If 'w' is a string then its length is denoted by $|w|$.

Ex:-

1. If w = abbc, then $|w| = 4$

2. ϵ is the empty string and has length zero.

→ Powers of an alphabet:-
If Σ is an alphabet, we can express the set of all strings of a certain length by using exponential notation.

2023.06.27 08:47

→ We define Σ^k to be the set of strings of length k, each of whose symbols is in Σ.

Ex:-

$\Sigma^0 = \epsilon$.

If $\Sigma = \{a, b, c\}$ then $\Sigma^1 = \{a, b, c\}$,

$\Sigma^2 = \{aa, bb, cc, ab, bc, ca, ba, cb, ac\}$.

$\Sigma^3 = \{aaa, bbb, ccc, aab, aac, abb, abc \ldots\}$.

→ Kleene closure (or) star closure:-

The set of all strings over an alphabet Σ is denoted by Σ^* & defined as $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$

Ex: If $\Sigma = \{a, b\}$. Then find Σ^*

Sol:- $\Sigma^0 = \{\epsilon\}$, $\Sigma^1 = \{a, b\}$, $\Sigma^2 = \{aa, bb, ab, ba\}$ and so on.

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$

$\therefore \Sigma^* = \{\epsilon, a, b, aa, bb, ab, ba, \ldots\}$.

→ ~~Kleene~~

→ Positive closure:-

If Σ is an alphabet then positive closure Σ is denoted by Σ^+ and defined as $\Sigma^+ = \Sigma^* - \{\epsilon\}$ (excludes empty string ϵ from set of strings. $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$

$\Sigma^* = \Sigma^+ \cup \{\epsilon\}$.

Ex:- If $\Sigma = \{a, b\}$ then find Σ^+.

Sol:- $\Sigma^+ = \{a, b, aa, bb, ab, ba, \ldots\}$.

2023.06.27 08:47

→ Concatenation of strings:-

Let x any y be strings. Then xy denotes the concatenation of x and y.

Ex:- Let $x = 1101$ & $y = 0011$ then

$xy = 11010011$, $yx = 00111101$

→ Language:-

Language is set of all strings over alphabet. If Σ is a alphabet and $L \subseteq \Sigma^*$ then 'L' is a language over Σ.

Ex:-

1. The language of all strings consisting of n 0's followed by n 1's is $L = \{\epsilon, 01, 001, 0111, \dots\}$ for some $n \geq 0$.
2. The set of strings 0's and 1's with an equal number each: $L = \{\epsilon, 01, 0011, 000111, 1001, 0101 \dots\}$
3. The set of binary numbers whose value is a prime:

   $L = \{10, 11, 101, 111, 1011 \dots\}$
4. Σ^* is a language for any alphabet Σ.
5. ϕ, is a empty language over any alphabet.
6. $\{\epsilon\}$, the language consisting of only the empty string is also a language over any alphabet. Note that $\phi \neq \{\epsilon\}$.

2023.06.27 08:47

→ Finite Automata Representation:-

We represent Finite Automata by 5 tuples $(Q, \Sigma, \delta, q_0, F)$

1. Q is a finite, non-empty set of states.
2. Σ is a an input alphabet
3. δ is a transition function which maps $Q \times \Sigma \rightarrow 2^Q$ i.e the head reads a symbol in its present state and moves into the set of next state(s). ~~2^Q is the power set of Q.~~
4. $q_0 \in Q$ known as initial state.
5. $F \subseteq Q$ known as final states.

→ Notations used for representing 5-tuples :-

1. Initial state is represented by a state within a circle and arrow entering into a circle.

$\rightarrow (q_0)$ (q_0 is a initial state).

2. Final state is represented by double circles.

$((q_1))$ (q_1 is a final state).

3. The hang state is represented by 'ϕ' within a circle

(ϕ)

4. Other states are represented by statename within a circle.
5. A directed edge with label shows the transition (or) move suppose 'p' is the present state and 'q' is the next state on input symbol 'a' then this is represented as

$(p) \xrightarrow{a} (q)$

6. A directed edge with more than one label shows the transitions or moves. Suppose 'p' is present state and 'q' is the next state on input-symbols 'a_1' or 'a_2' or 'a_3' ... 'a_n' then this is represented as

$(p) \xrightarrow{a_1, a_2, a_3 \cdots a_n} (q)$

→ Transition functions :-

We have 2 types of transition functions.

1. Direct Transition function
2. Indirect Transition function.

→ 1. Direct Transition function:-

When the input is a symbol then transition function is known as direct transition function denoted by 'δ'.

Ex:-

$\delta(P, a) = q$ (where 'p' is present state and 'q' is next state) known as one step transition.

→ 2. Indirect Transition function:-

When input ~~string~~ is a string then transition function is known as indirect transition function denoted by 'δ''.

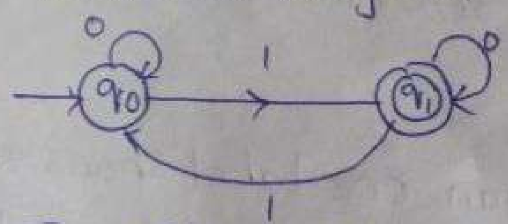
Ex:-

$\delta'(P, w) = q$ (where 'p' is a present state and 'q' is a next state after $|w|$ transitions. It is known as multiple step transition.

→ Finite Automata can be represented in 2 ways:

1. Directed Graph with States - Transition Diagram.
2. Transition Table.

→ Transition Diagram:-



→ Transition table :-

	0	1
q_0	q_0	q_1
q_1	q_1	q_0

→ states of finite Automata:-

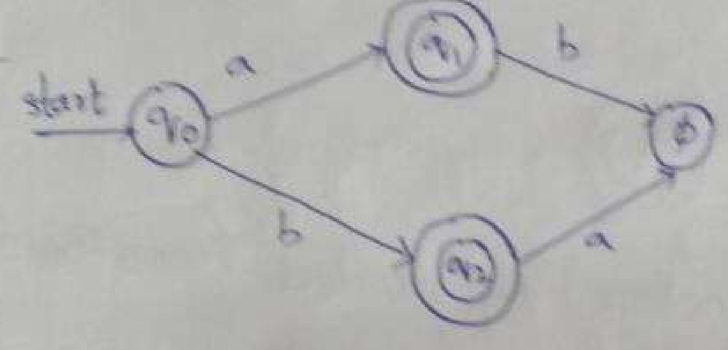
1. Initial state:- From this state processing starts and initial state can be only one

2. Final state:- It is the end of transition where the execution of string is ended. We can have more than one final state.

3. Non-final states: Except final states all other states are known as Non-final states.

4. Hang states: These are the states not included in δ after reaching these states, FA sits in idle mode. Hang states are denoted by ϕ.

Ex:-

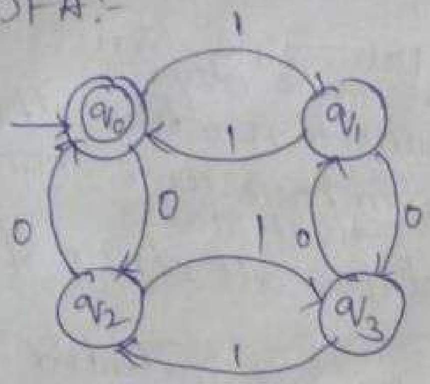


where q_0: Initial state.

$\left.\begin{matrix} q_1 \\ q_2 \end{matrix}\right\} \rightarrow$ final states

ϕ : Hang state

→ DFA:-



Check the acceptability of String 101101 for DFA

Sol:-

	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 path:-

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

final state.

→ Using δ Transition function i.e Extended Transition function

$\hat{\delta}(q_0, 101101)$

$\rightarrow \delta(\hat{\delta}(q_0, 1), 01101)$

$\rightarrow \delta(q_1, 01101)$

$\rightarrow \delta(\hat{\delta}(q_1, 0), 1101)$

$\rightarrow \delta(q_3, 1101)$

$\rightarrow \delta(\hat{\delta}(q_3, 1), 101)$

$\rightarrow \delta(q_2, 101)$

$\rightarrow \delta(\hat{\delta}(q_2, 1), 01)$

$\delta(q_3, 01)$

$\delta(\hat{\delta}(q_3, 0), 1)$

$\delta(q_1, 1)$

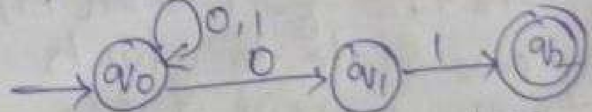
$\delta(\hat{\delta}(q_1, 1), \epsilon)$

$\delta(q_0, \epsilon)$

$= q_0$ → final state

Given string 101101 is acceptable.

→ NFA:-



Check the acceptability of string 00101

Sol:-

Transition table :-

$$\to q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_1 \xrightarrow{1} \boxed{q_2}$$

	0	1
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	ϕ	$\{q_2\}$
q_2	ϕ	ϕ

. δ Transition function i.e extended transition function.

× $\hat{\delta}(q_0, 00101)$

not acceptable

$\delta(\hat{\delta}(q_0, 0), 0101)$

$\delta(q_0, 0101)$

$\delta(\hat{\delta}(q_0, 0), 101)$

$\delta(q_0, 101)$

$\delta(\hat{\delta}(q_0, 1), 01)$

$\delta(q_0, 01)$

$\delta(\hat{\delta}(q_0, 0), 1)$

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

$\delta(\hat{\delta}(q_0, 1), \epsilon)$

$\delta(q_0, \epsilon)$

$= q_0$ ×

$\hat{\delta}(q_0, 00101)$

$\hat{\delta}(\hat{\delta}(q_0, 0), 0101)$

$\delta(q_0, 0101)$

$\delta(\hat{\delta}(q_0, 0), 101)$

$\delta(q_0, 101)$

$\delta(\hat{\delta}(q_0, 1), 01)$

$\delta(q_0, 01)$

$\delta(\hat{\delta}(q_0, 0), 1)$

$\delta(q_1, 1)$

$\delta(\hat{\delta}(q_1, 1)\, \epsilon)$

$\delta(q_2, \epsilon)$

$= q_2$ final state

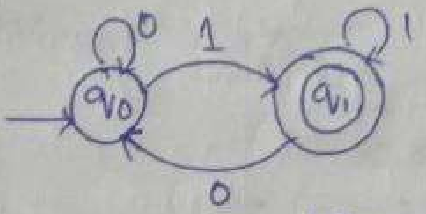
→ Deterministic Finite Automata:-

The term "Deterministic" refers to the fact that on each input there is one and only one state to which the automaton can transition from its current state.

→ A DFA consists of 5 tuple $(Q, \Sigma, \delta, q_0, F)$

→ Transition systems:-

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



DFA

→ Acceptability of a string by a finite Automaton:-

A string x is accepted by a finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ if $\delta(q_0, x) = q$ for some $q \in F$.

A final state is called as accepting state.

A Transition table is a conventional tabular representation of transition function δ that takes two arguments and returns a value.

→ Extended Transition function:-

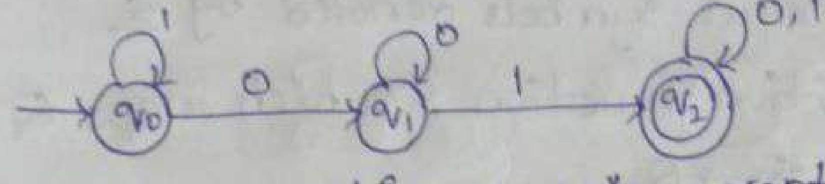
The extended Transition function is a function that takes a state q and string w and returns state p. The state that the automaton reaches when starting in state q and processing the sequence of i/p's w.

It is represented by $\hat{\delta}$.

→ $\hat{\delta}(q, \epsilon) = q$ i.e if we are in state 'q' and read no input then we are still in state 'q'.

$w = xa$ then $\hat{\delta}(q, w) = \delta(\hat{\delta}(q, x), a)$

Ex:-



check whether string 1101 is accepted by automaton or not - Check the acceptability of string i. 1101
ii, 0010
iii, 1000

Sol:- Transition Table:-

	0	1
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_2	q_2

Transition Diagram:-

$\rightarrow (q_0) \xrightarrow{1} (q_0) \xrightarrow{1} (q_0) \xrightarrow{0} (q_1) \xrightarrow{1} ((q_2))$

final state

ii) For : 1000

$\rightarrow (q_0) \xrightarrow{1} (q_0) \xrightarrow{0} (q_1) \xrightarrow{0} (q_1) \xrightarrow{0} (q_1)$

Didn't reached final state, string not accepted.

→ $\hat{\delta}(q_0, 1101)$

$= \delta(\hat{\delta}(q_0, 1), 101)$

$= \delta(q_0, 101)$

$= \delta(\hat{\delta}(q_0, 1), 01)$

$= \delta(q_0, 01)$

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

$= \delta(q_1, 1)$

$\delta(\hat{\delta}(q_1, 1), \epsilon)$

$\delta(q_2, \epsilon) = q_2$ (final state)

As the given string reached final state

∴ 1101 is acceptable.

2023.06.27 08:47

→ DFA (Deterministic finite Automata):-

A DFA consists of 5 tuple $(Q, \Sigma, \delta, q_0, F)$

1. A finite non-empty set of states denoted by Q
2. A finite set of input symbols denoted by Σ
3. A direct transition function δ which maps $Q \times \Sigma$ into Q i.e $\delta: Q \times \Sigma \rightarrow Q$.
4. $q_0 \in Q$ is the initial state
5. $F \subseteq Q$ is the final states.

→ NFA (Non-Deterministic finite Automata):-

In NFA, for a particular input symbol, the machine can move to any combination of the states in the machine. In otherwords, the exact state to which the machine moves cannot be determined.

→ NFA consists of 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

1. Q is a finite, non empty set of states.
2. Σ is a finite set of input symbols.
3. δ is the transition function mapping $Q \times \Sigma$ into 2^Q which is the power of Q, the set of all subsets of Q.

$$\delta: Q \times \Sigma \rightarrow 2^Q$$

[2^Q is taken in NDFA b'coz in case of NFA from a state transition can occur to any combination of Q states]

4. $q_0 \in Q$ is the initial state
5. $F \subseteq Q$ is the set of final states.

→ Note:- The difference b/n NFA & DFA is, in 'δ' for DFA the outcome is a state, i.e an element of Q. For NFA the outcome is subset of Q.

2023.06.27 08:47

→ Acceptability of string in DFA:-

Let DFA $M = (Q, \Sigma, \delta, q_0, F)$ and input string $w \in \Sigma^*$.

The string w is accepted by M if and only if $\delta(q_0, w) = q_f$ where $q_f \in F$.

When w is accepted by M then the execution of string w ends in final state and this execution is known as successful otherwise unsuccessful.

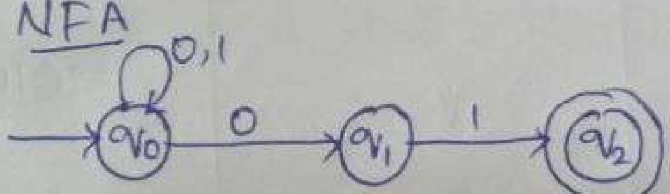
→ Acceptability of string in NFA:-

Let NFA $M = (Q, \Sigma, \delta, q_0, F)$ and input string $w \in \Sigma^*$

The string is accepted by M if and only if

$\delta(q_0, w) = \{q_i : q_i \in F, \text{for some } i = 0, 1, 2 \ldots n\}$.

Ex:- NFA



$\delta(q_0, 00101)$

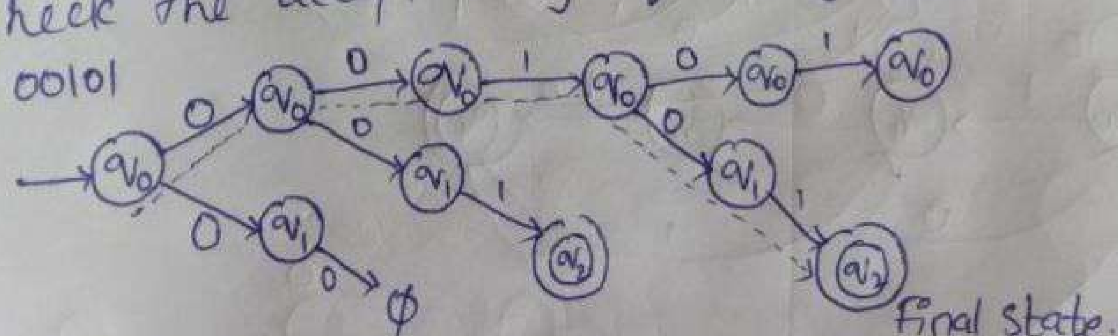
$\delta(\delta(q_0, 0), 0101)$

Transition Table:-

	0	1
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	ϕ	$\{q_2\}$
q_2	ϕ	ϕ

check the acceptability of string 1. 00101 2. 01010

1. 00101



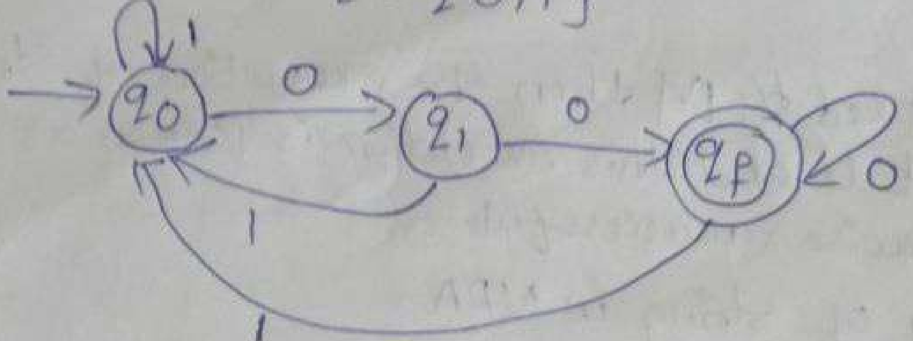
$\rightarrow q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_1 \xrightarrow{1} q_2$

∴ Given string 00101 reached final state. String is acceptable.

①

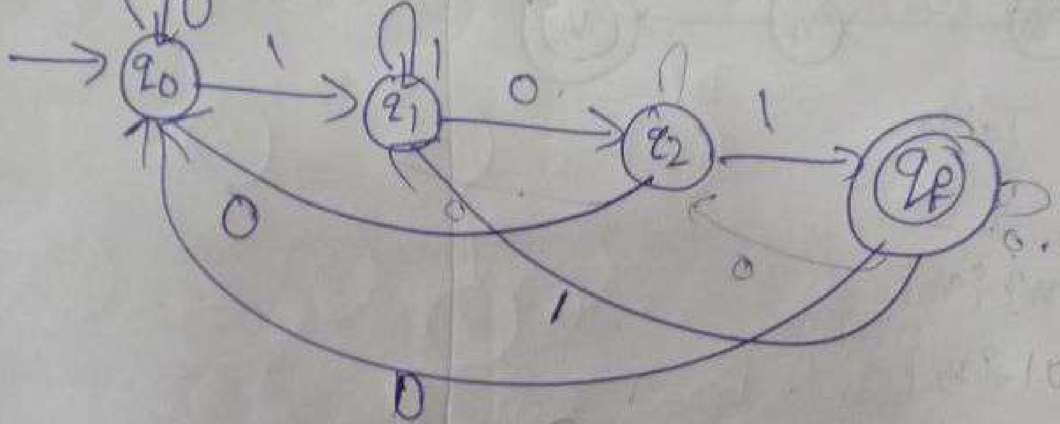
$\Sigma = \{0, 1\}$

end with 00
01
10
11

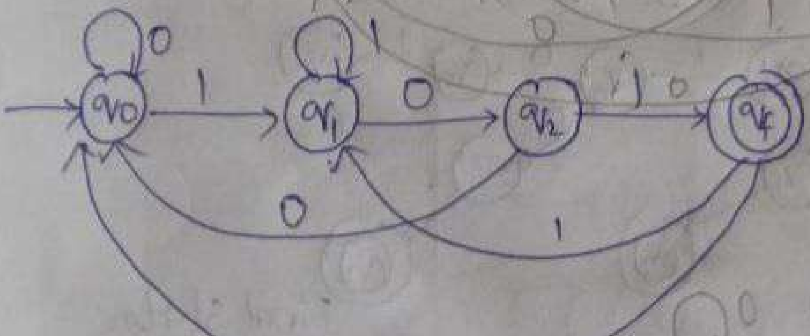


End with 101

0101

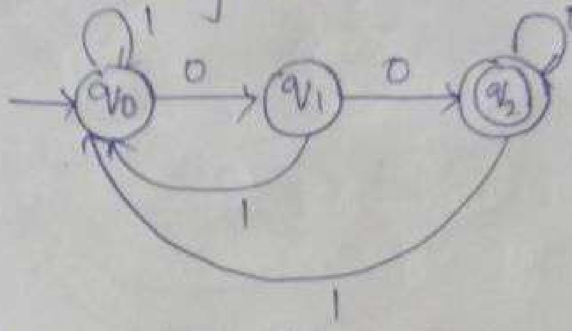


→ 101

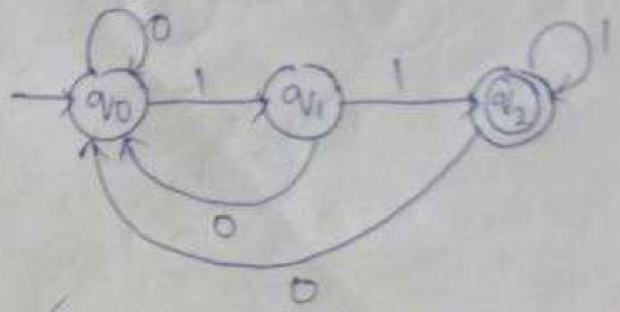


① Design an DFA to accept strings with 0's & 1's such that the string ends with 00, 11, 10, 01

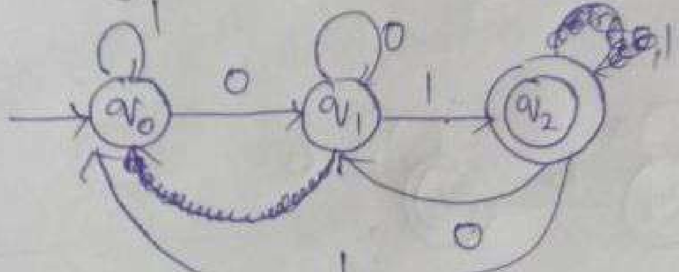
Sol:- → String ends with 00



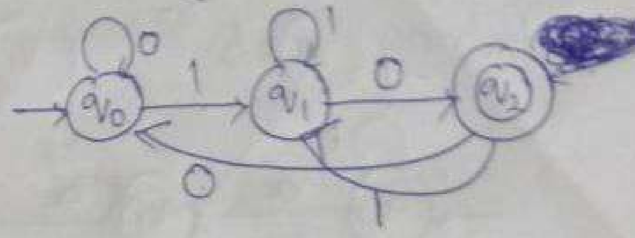
→ string ends with 11



→ String ends with 01



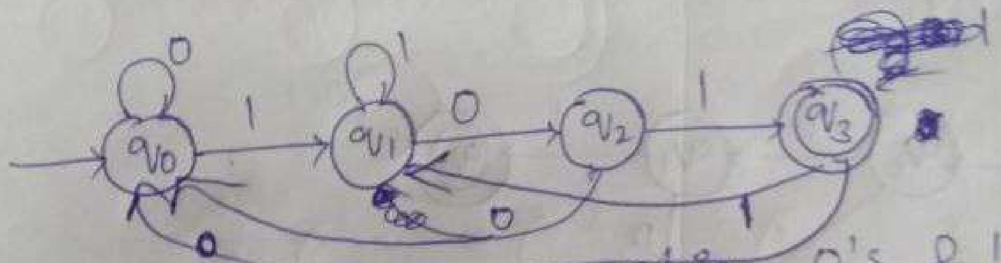
→ string ends with 10



→ Design DFA that accepts strings ending with 101

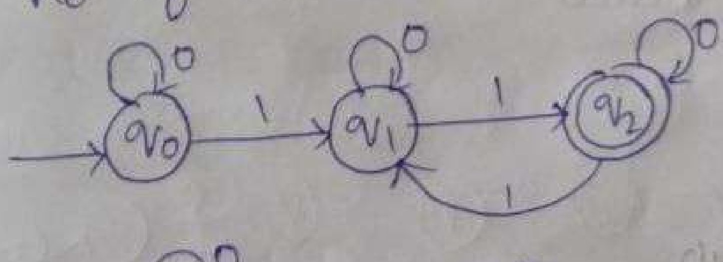
111
000
001
010
100
101
110
011

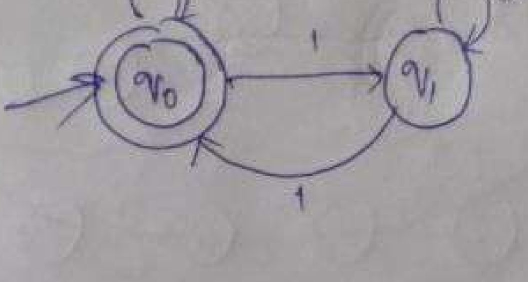
Sol:-



→ Design DFA that accepts string 0's & 1's such that no of 1's should be even.

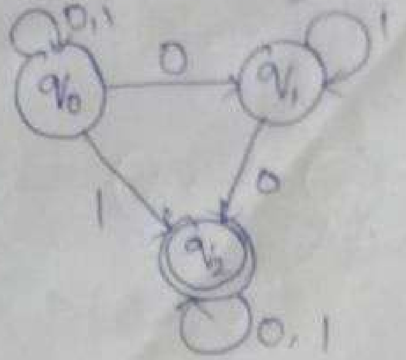
Sol:-



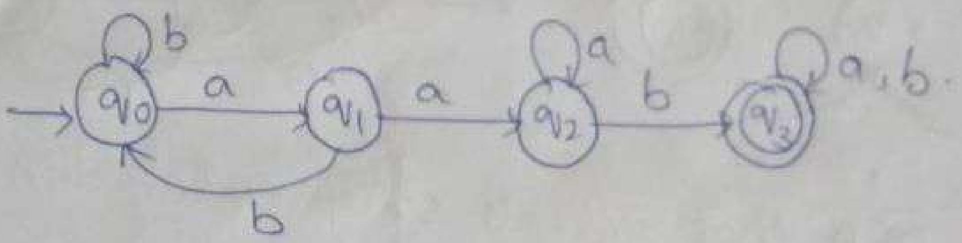


2023.06.27 08:47

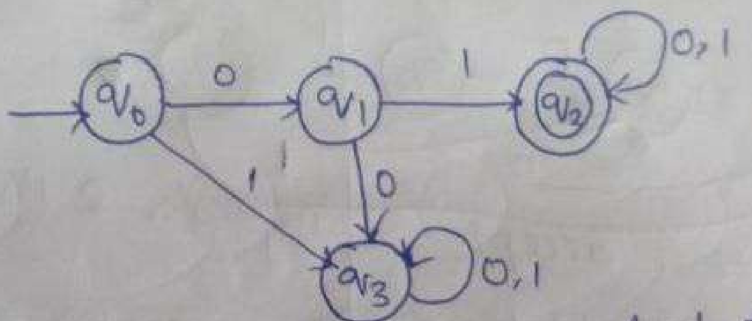
→ Design a DFA that recognizes set of all strings with 0's and 1's starting with prefix either 0 or 1.



→ Design DFA that accepts string a's and b's containing substring aab.



→ Prefix 01.

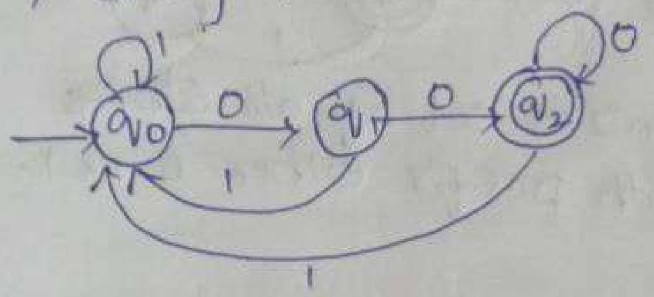


Here q_3 is called Dead state.

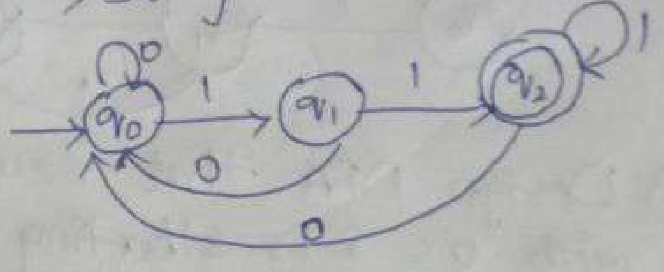
→ The Language of DFA:-

① Design an DFA to accept strings with 0's and 1's such that the string ends with 00, 11, 01, 10.

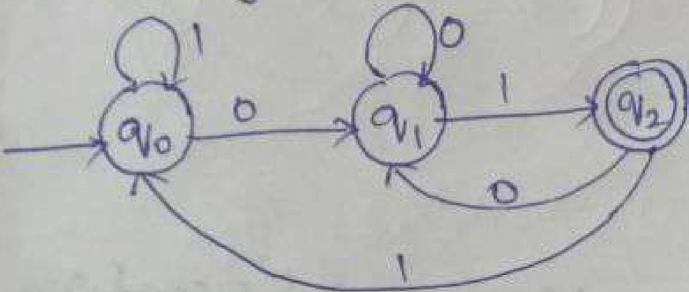
→ string ends with 00



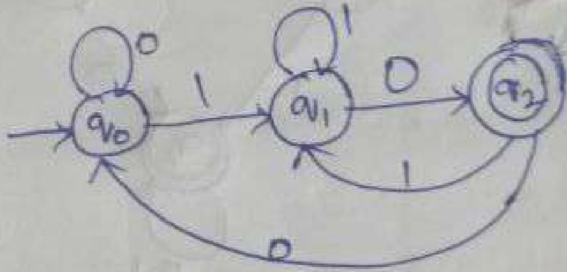
→ string ends with 11



→ String ends with 01

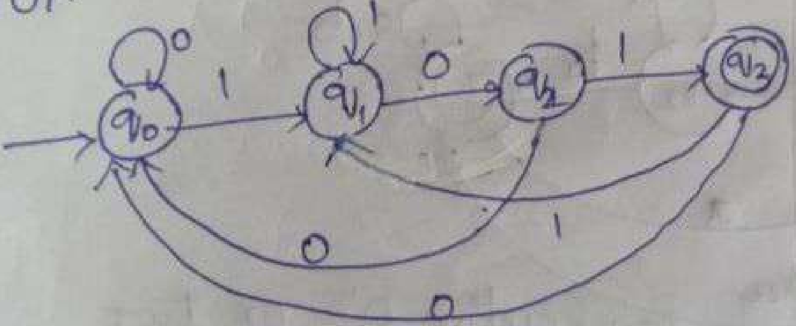


→ string ends with 10

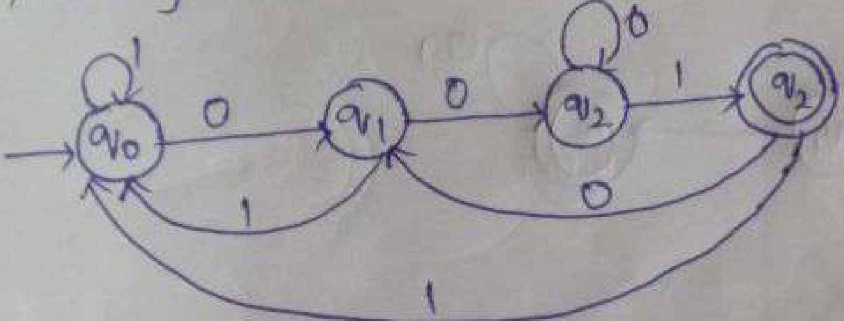


② Design DFA that accepts strings ~~with DFA~~ ends with 101.

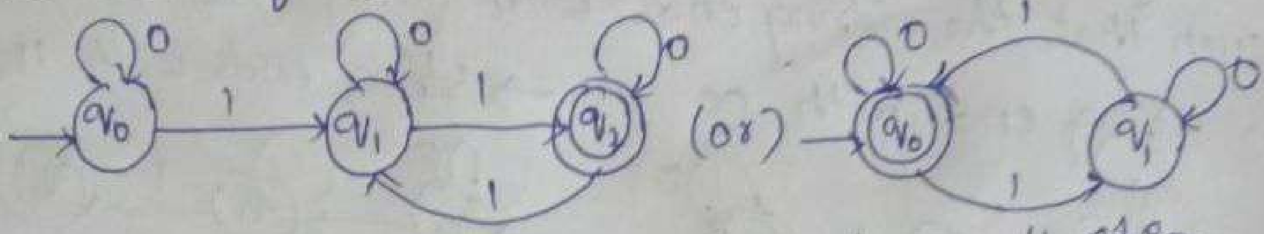
111
000
001
010
100
101
110
011



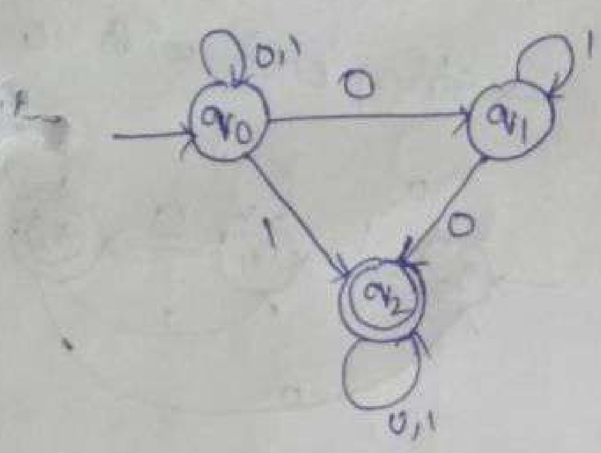
→ string ends with 001



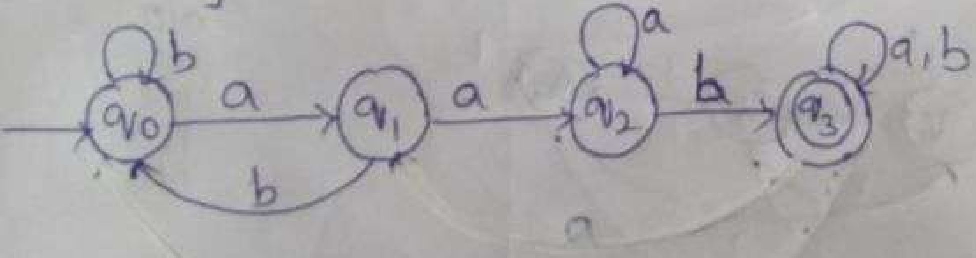
③ Design DFA that accepts string 0's & 1's such that no's of 1's should be even.



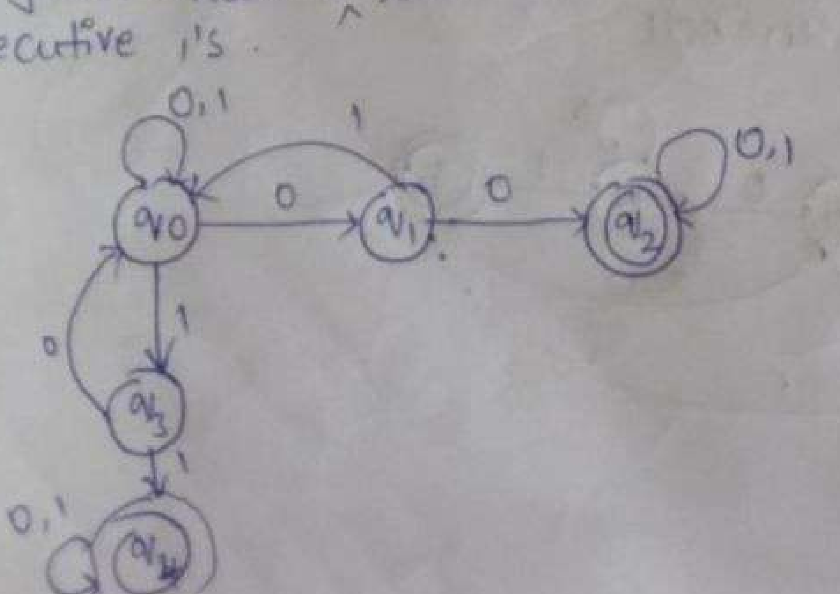
④ Design DFA that recognizes set of all strings with 0's & 1's starting with prefix either 0 or 1.



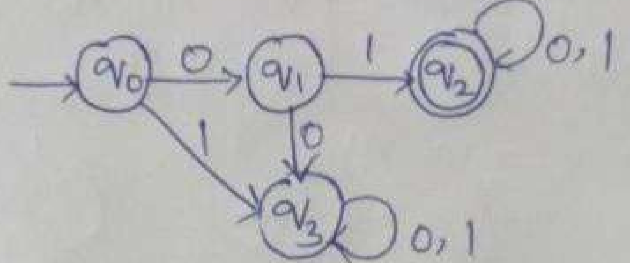
⑤ Design DFA that accepts string a's & b's containg substring aab.



⑥ Design DFA with either two consecutive 0's or two consecutive 1's.

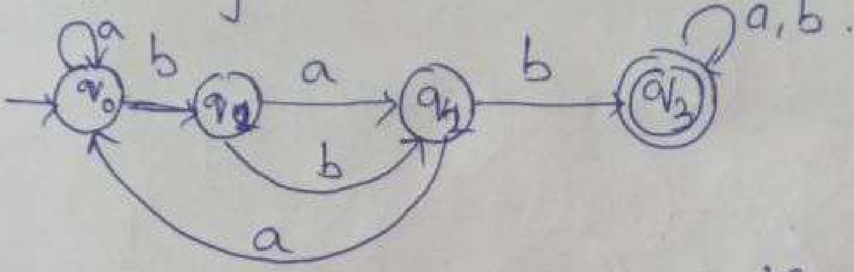


(7) String starting with prefix 01.



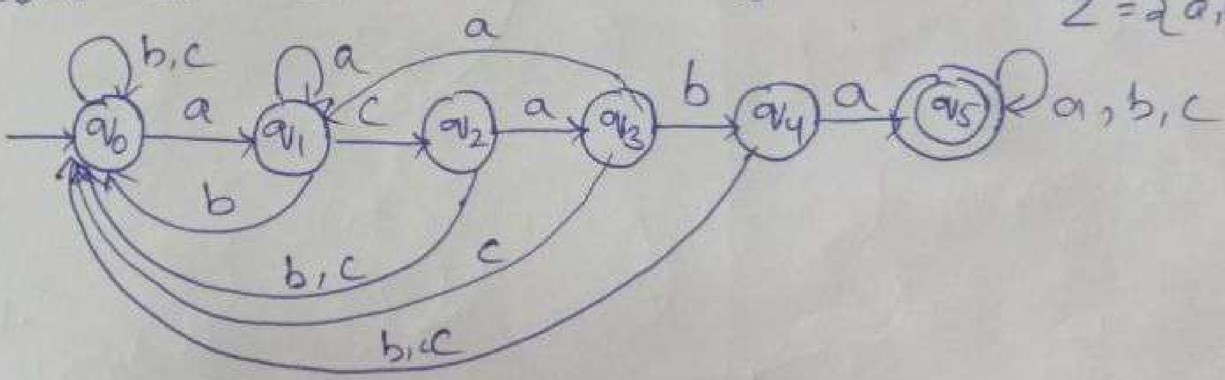
q_3 is a dead state

(8) Design DFA that accepts string a's & b's such that substring bab (or) bb.

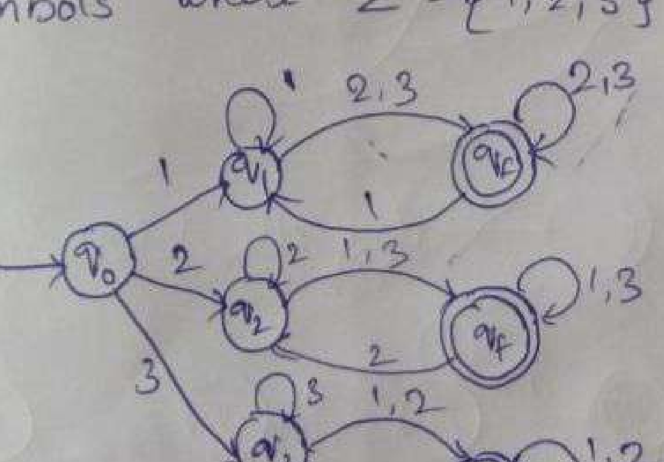


(9) Construct a DFA with substring acaba

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



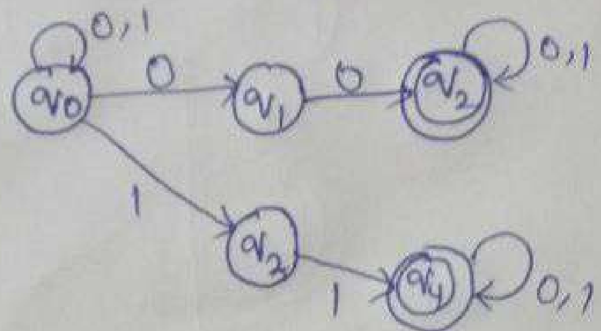
(10) DFA with strings beginning & ending with different symbols where $\Sigma = \{1, 2, 3\}^*$.



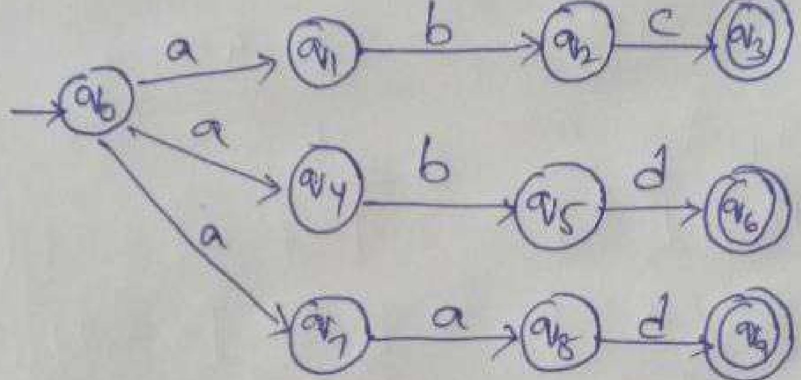
1 111 2 2 2 3 3 3 2 2 2

3 1 1 1 3 3 2 2 2 3 3

→ Design NFA which accepts strings with 0's & 1's such that string ~~ends with~~ contains 2 consecutive 0's or 1's.



→ NFA that accepts only the strings abc, abd & aad.



2023.06.27 08:47

→ Equivalence of DFA & NFA.

→ Construction of DFA equivalent to given NFA.

Sol:- Let given NFA is $M = (Q, \Sigma, \delta, q_0, F)$ and equivalent DFA is $M' = (Q', \Sigma, \delta', q_0', F')$.

step1:- $Q' = 2^Q$, the set of all ~~states~~ subsets of Q.

step2:- $q_0' = [q_0]$ the initial state.

step3: F' is the set of all subsets of Q which are having final state of given NFA.

step4: construction of δ'. We start the construction of δ' for $[q_0]$. We continue by considering only states appearing earlier under i/p columns and constructing δ' for such states. We stop when no more new states appear under the i/p columns.

→ Construct DFA equivalent to $M = (\{q_0, q_1\}, \{0,1\}, \delta, q_0, \{q_0\})$

δ is given by.

	0	1
→ q_0	$\{q_0\}$	$\{q_1\}$
q_1	$\{q_1\}$	$\{q_0, q_1\}$

Sol:- DFA $M' = (Q', \Sigma, \delta', q_0', F')$, (i) $Q' = 2^Q$

ii) $Q' = 2^Q = \{\phi, [q_0], [q_1], [q_0, q_1]\}$, $q_0' = [q_0]$

iii) $F' = \{[q_0], [q_0, q_1]\}$ as these are the only states containing q_0 which is final state in given NFA.

iv) δ' is defined by state table.

	0	1
$[q_0]$	$[q_0]$	$[q_1]$
$[q_1]$	$[q_1]$	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$

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

→ $\delta([q_0, q_1], 1) = \delta(q_0, 1) \cup \delta(q_1, 1)$
$= [q_1] \cup [q_0, q_1] = [q_0, q_1]$

2023.06.27 08:47

→ Construct DFA equivalent to $M = \{\{p, q, r, s\}, \{0,1\}, \delta, \{p\}, \{s\}\}$ for the given Transition table given below.

	0	1
→ p	{p, q}	{p}
q	{r}	{r}
r	{s}	ϕ
(s)	{s}	{s}

Sol:-

δ' for DFA is ~~design~~ defined by Transition table

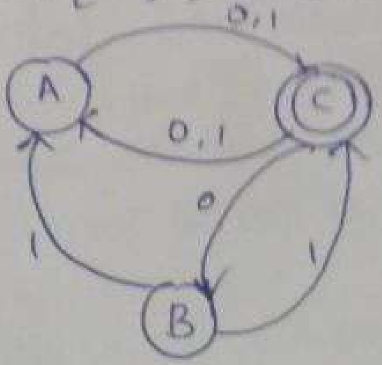
	0	1
→ [p]	[p, q]	[p]
[p, q]	[p, q, r]	[p, r]
[p, r]	[p, q, s]	[p]
[p, q, r]	[p, q, r, s]	[p, r]
[p, q, s]	[p, q, r, s]	[p, r, s]
[p, s]	[p, q, s]	[p, s]
[p, q, r, s]	[p, q, r, s]	[p, r, s]
[p, r, s]	[p, q, r, s]	[p, r, s]

→ $Q' = \{\phi, [p], [q], [r], [s], [p,q], [p,r], [p,s], [q,r], [q,s], [r,s], [p,q,r], [p,q,s], [p,r,s], [q,r,s], [p,q,r,s]\}$.

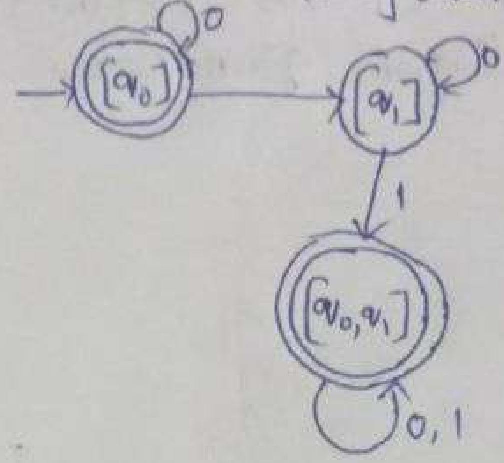
→ $F' = \{[s], [p,s], [q,s], [r,s], [p,q,s], [p,r,s], [q,r,s], [p,q,r,s]\}$.

→ Transition Diagram:-

→ Construct DFA equivalent to M = ({A, B, C}, {0, 1}, δ, {A}, {C}) where Transition diagram given below.



→ Transition Diagram:-



→ ② Construct DFA equivalent to $M = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, \{q_0\}, \{q_2\})$ where Transition Table is given below.

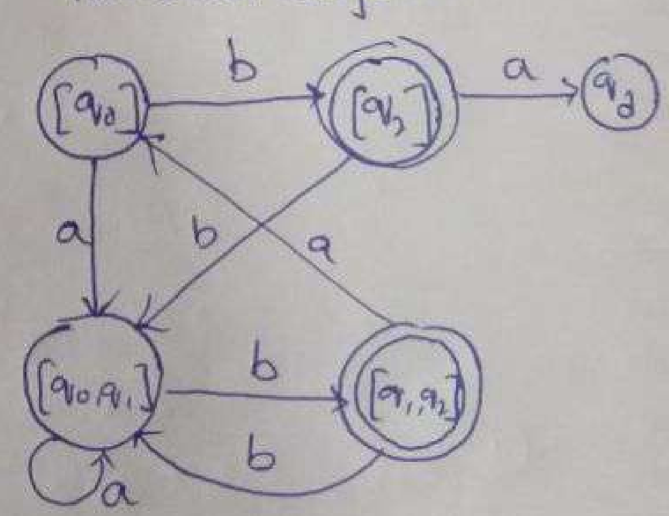
	a	b
→ q_0	$\{q_0, q_1\}$	$\{q_2\}$
q_1	$\{q_0\}$	$\{q_1\}$
q_2	ϕ	$\{q_0, q_1\}$

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

1. $Q' = 2^Q \Rightarrow 2^3 = \{\phi, [q_0], [q_1], [q_2], [q_0, q_1], [q_0, q_2], [q_1, q_2], [q_0, q_1, q_2]\}$.
2. $[q_0]$ is initial state
3. $F' = \{[q_2], [q_0, q_2], [q_1, q_2], [q_0, q_1, q_2]\}$
4. δ' is defined by state table

	a	b
→ $[q_0]$	$[q_0, q_1]$	$[q_2]$
$[q_2]$	ϕ $[q_d]$	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_1, q_2]$
$[q_1, q_2]$	$[q_0]$	$[q_0, q_1]$

Transition diagram.



→ Finite Automata with Epsilon Transitions:-

An FA with ϵ-Transition is one that is allowed to make a transition spontaneously, with or without receiving an i/p symbol. Like the Non-determinism added to a simple DFA, this new capability does not expand the class of languages that can be accepted by an FA, but does give us some added "programming convenience".

→ Definition:-

An NFA with ϵ-moves in a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ with all components as NFA but δ, the transition function maps $Q \times \{\Sigma \cup \{\epsilon\})$ to 2^Q.

Ex:-

→ ϵ-closure(q):

It is the set of all vertices P such that there is a path from q to P labelled ϵ.

In the above $\epsilon\text{-closure}(q_0) = \{q_0, q_1, q_2\}$

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

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

→ Converting NFA with ϵ-moves to NFA without ϵ-moves:-

Let $M = (Q, \Sigma, \delta, q_0, F)$ is NFA with ϵ-moves

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

→ step 1: Find ϵ-closures for all states.

→ step 2: find extended transition function

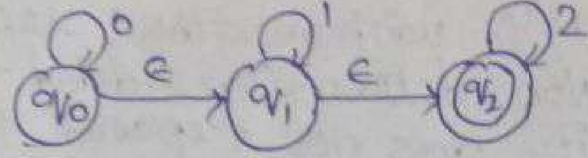
$$\hat{\delta} : Q \times \Sigma^* \longrightarrow 2^Q$$

$$\hat{\delta}(q_0, \epsilon) = \epsilon\text{-closure}(q_0)$$

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

→ step 3: set of final states F' consists of all states whose ϵ-closure contains a final state in F.

→



ε = empty string

ϕ = Empty language

Note ϕ = {ε}

	0	1	2	ε
$\rightarrow q_0$	$\{q_0\}$	ϕ	ϕ	$\{q_1\}$
q_1	ϕ	$\{q_1\}$	ϕ	$\{q_2\}$
q_2	ϕ	ϕ	$\{q_2\}$	ϕ

Sol:-

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

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

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

$\hat{\delta}(q_0,0) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_0,\epsilon),0))$

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

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

$= \epsilon\text{-closure}(q_0 \cup \phi \cup \phi) = \{q_0, q_1, q_2\}$

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

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

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

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

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

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

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

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

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

$= \{q_2\}$

$\hat{\delta}(q_1,0) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_1,\epsilon),0))$

$= \epsilon\text{-closure}(\delta(q_1,q_2),0)$

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

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

$\hat{\delta}(q_1,1) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_1,\epsilon),1))$

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

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

$= \epsilon\text{-closure}(q_1\cup\phi) = \{q_1,q_2\}$

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

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

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

$= \epsilon\text{-closure}(q_2\cup\phi) = \cancel{\{q_1,q_2\}}\ \{q_2\}$

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

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

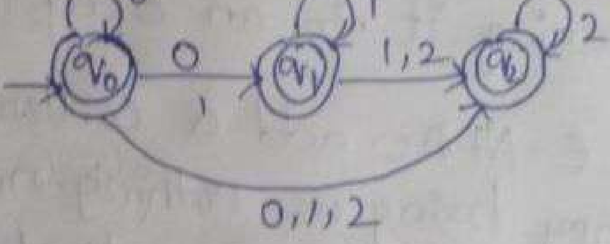
$\hat{\delta}(q_2,1) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_2,\epsilon),1))$

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

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

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

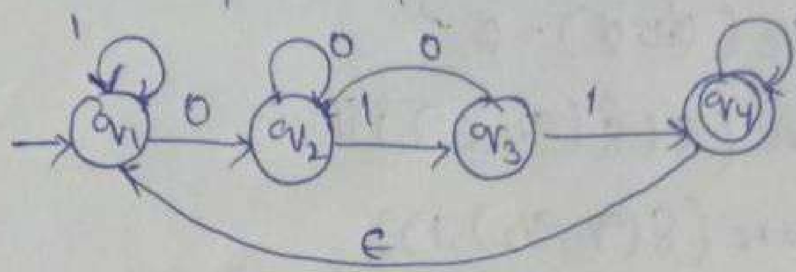
	0	1	2
$\rightarrow q_0$	$\{q_0,q_1,q_2\}$	$\{q_1,q_2\}$	$\{q_2\}$
q_1	ϕ	$\{q_1,q_2\}$	$\{q_2\}$
q_2	ϕ	ϕ	$\{q_2\}$



As ϵ-closure of q_0 & q_1 contain the final state q_2, q_0 & q_1 are made as final states.

→ Epsilon Transitions:-

The finite machine below recognizes strings that contain a repeated pattern.



The above machine accepts any string the contains 1 or more repetitions of the substring 001. To make it possible for the machine to accept inputs that contain more than one repetition of the pattern 011, the machine makes use of a special type of transition called Epsilon transition.

→ Non-Determinism:-

Epsilon Transitions make machine non-deterministic. When a machine enters a state that has an Epsilon transition coming from it, the machine can decide whether to stay in the state and await the next i/p symbol or it can immediately take the epsilon transition and go a new state.

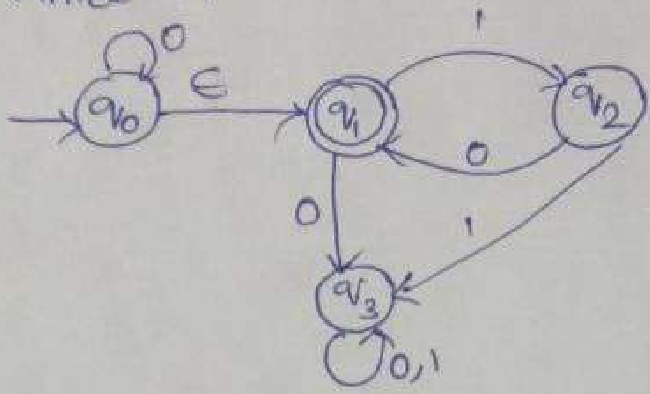
NFA accepts its input if there is some path through the machine driven by i/p that results in the machine making it to an accepting state.

→ ε-NFAs add a convenient feature but (in a sense) they bring us nothing new. They do not extend the class of languages that can be represented.

→ Both NFA's and ε-NFA's recognize exactly the same languages.

→ Finite n

(2)



	0	1	ϵ
$\rightarrow q_0$	$\{q_0\}$	ϕ	$\{q_1\}$
q_1	$\{q_3\}$	$\{q_2\}$	ϕ
q_2	$\{q_1\}$	$\{q_3\}$	ϕ
q_3	$\{q_3\}$	$\{q_3\}$	ϕ

Sol:-

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

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

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

$\epsilon\text{-closure}(q_3) = \{q_3\}$