UNIT - I

Automata Theory and Compiler Design

Introduction :

Automata : It is used to solve the computational problems in discrete or continous system.

Automata Theory is the study of abstract computing devices.

It is a branch of the theory of computation & study of abstract machines.

The abstract machine is called automata.

abstract → Invisible.

Applications of Automata : AI, distributed system, speach recognition, Vending Machine, traffic lights & board games, computer applications - Finite Automata, Pushdown automata, turining machine.

Example : speach Recognition.

This is also best example of automata.

Let's open youtube, click on speaker button & speack something, here we are giving voice as input & getting the result based on our input voice.

We can not able to see the process.

The most Powerful automata is turing Machine.

Formal Language :

we use the language to communicate with others.

In similar way to communicate with abstract machine, we have formal language.

The languages are defined by Chomsky based on the grammar.

According to chomsky, there are four(4) types of languages.

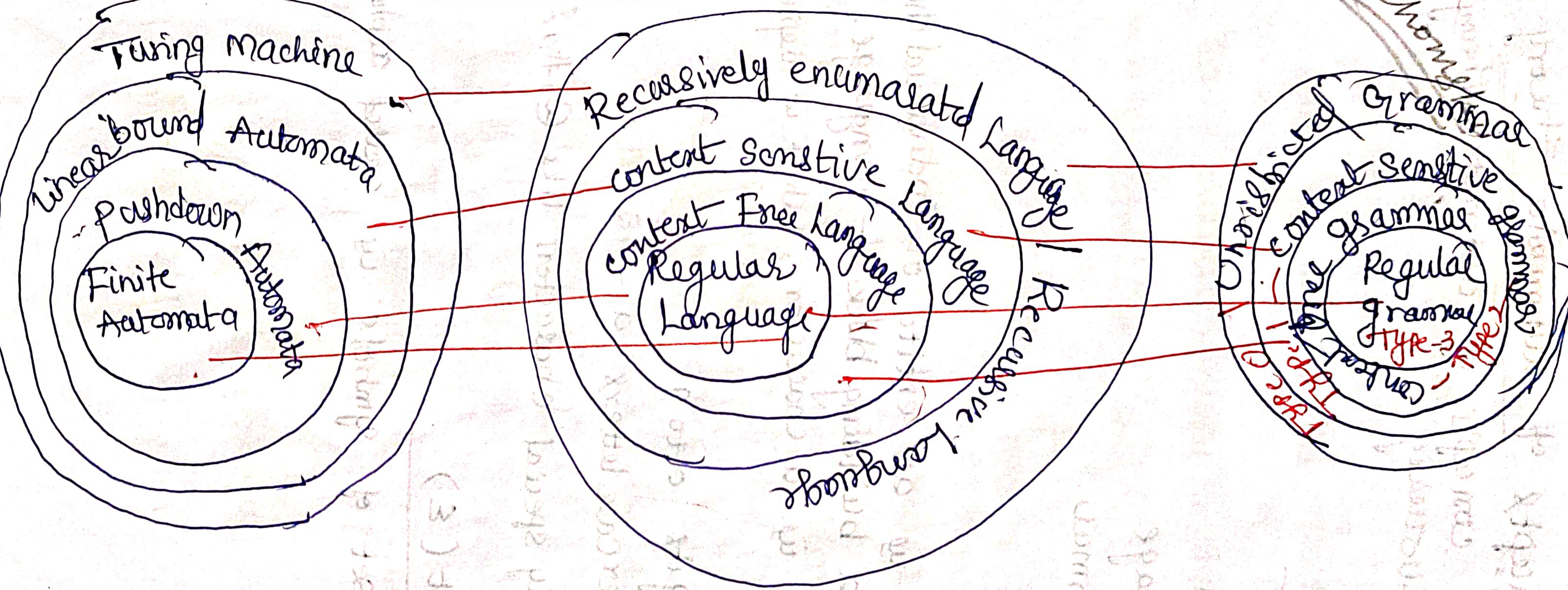
Noam Chomskey Hierarchy :

According to chomsky hierarchy, grammar is divided into 4 types as follows.

1. Type 0 — Unrestricted Grammar
2. Type 1 — context sensitive grammar
3. Type 2 — context free grammar
4. Type 3 — Regular grammar.

Grammar Type	Grammar Accepted	Language Accepted	Automation
Type 0	Unrestricted Grammar	Recursively enumerated Language	Turing Machine
Type 1	Context-sensitive grammar	context-sensitive Language	Linear bounded Automata
Type 2	Context Free grammar	Context free Language	Pushdown automata
Type 3	Regular grammar	Regular Language	Finite Automata,

Noam Chomskey Hierarchy of Language :



3

Central concepts of automata theory:

The central concepts of automata theory consist the following

1. Symbol.
2. Alphabet
3. String
4. Language
5. Grammar.

1. Symbol :

Symbol is an entity or individual character
symbol is a building block of language.
symbol is a character, which is represented with

digits 0 to 9
or Lower case letters a --- z
or Any special characters like @, #, $ ---

2. Alphabet (Σ) :

A set of symbols is called alphabet

3. String : String is a finite set of Sequence of symbols over alphabet (Σ)

String is represented with <u>w</u> (word)

Ex: aabac, 01001, #@$

aabac - is a String over an alphabet $\Sigma = \{a, b, c\}$

01001 - String over $\Sigma = \{0, 1\}$ i.e Binary no.s

$#@$ - string over $\Sigma = \{\#, @, \$\}$

<u>Empty String</u>: String with zero (0) occurance of symbol (no symbols) is called empty string.

It is denoted by ϵ (Epsilon).

i.e $\Sigma = \{\ \}$

<u>Length of string</u>: Number of symbols present in the String.

Length of the string is denoted by $|w|$

Ex: $w = 010110$

$|w| = 6$.

Power of alphabet:

It is denoted by Σ^K, is the set of strings of length K.

If Σ is an alphabet, we can express set of all string of certain length from that alphabet by using exponential notation.

i.e Σ^K.

$\Sigma = \{0,1\}$ has 2 symbols.

1) $K=0$, $\Sigma^0 = \{\}$ or ϵ – Null string, length 0

2) $K=1$, $\Sigma^1 = \{0,1\}$ $(2^1 = 2)$ – string length with 1

3) $K=2$, $\Sigma^2 = \{00, 01, 10, 11\}$ $(\because 2^2 = 4)$
string length with 2

4) $K=3$, $\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$
$(2^3 = 8)$ – string length with 3

Kleene closure (Σ^*) or Kleene Star

The set of strings over alphabet Σ is denoted by Σ^* i.e kleene closure.

i.e $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3$ ---- Σ^n

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

$\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, \text{---} \Sigma^n\}$

Kleene Pluse (Σ^+).

The set Σ^+ is the infinite set of all possible strings of all possible lengths over Σ excluding empty string (ϵ)

$$\Sigma^+ = \Sigma^* - \Sigma^0$$

or

$$\Sigma^+ = \Sigma^* - \epsilon$$

i.e $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 - - - -$

String : Def.
- empty string
- Length of string
- Power of alphabet
  - Kleene star
  - Kleene closure (Σ^*)
  - Kleene pluse (Σ^+).

4. Language : Collection of strings over alphabet is called Language.

It is represented with L.

Language can be
1. Finite Language
2. Infinite. Language

① L = { length of the string 2 over $\Sigma = \{a, b\}$

L = { aa, ab, ba, bb }.

This is finite Language

(3)

$L = \{$ on $\Sigma = \{a, b\}$, every string starting with a $\}$

$\therefore L = \{a, aa, ab, aba, aaa, ----\}$

This is infinite language.

String $\Rightarrow$ Language.

5. Grammar (G) :

The rules which describes about the language is called the grammar. It is represented G.

G is a collection of (4) tuples.

Grammar is used to determine if a language is syntactically correct.

$$G = (V, T, P, S)$$

Where

V — Finite set of variables/ non terminals

We use capital letters for Variables, A, B, C —

T — Terminals (we use small letters for terminals a, b, c --

P - Production rules.

S = Start symbol.

Finite automata (FA) :

Finite automata is an abstract computing device.

The automata is called finite automata if it has finite number of states.

(or) FA is a model with finite no. of states

It is a mathematical model of a system with discrete

→ inputs
→ outputs
→ States

and set of transitions.

Finite automata abstract machines used to recognize patterns in input sequences, forming the basis for understanding regular languages in Computer Science.

Applications of FA :

1. Artificial intelligence & NLP
2. compiler design (Lexical & syntax analysis).
3. Text Processing & Pattern matching
4. Robotics and control systems.
5. Designing digital circuits.
6. Used in verification

Finite automata is described as 5 tuples :

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

where

Q = set ~~of~~ all possible states / finite set of states.

Σ = set of inputs (Alphabet)

δ = Transition Function.

q_0 = Initial State, $q_0 \in Q$.

F = Final state, $F \subseteq Q$

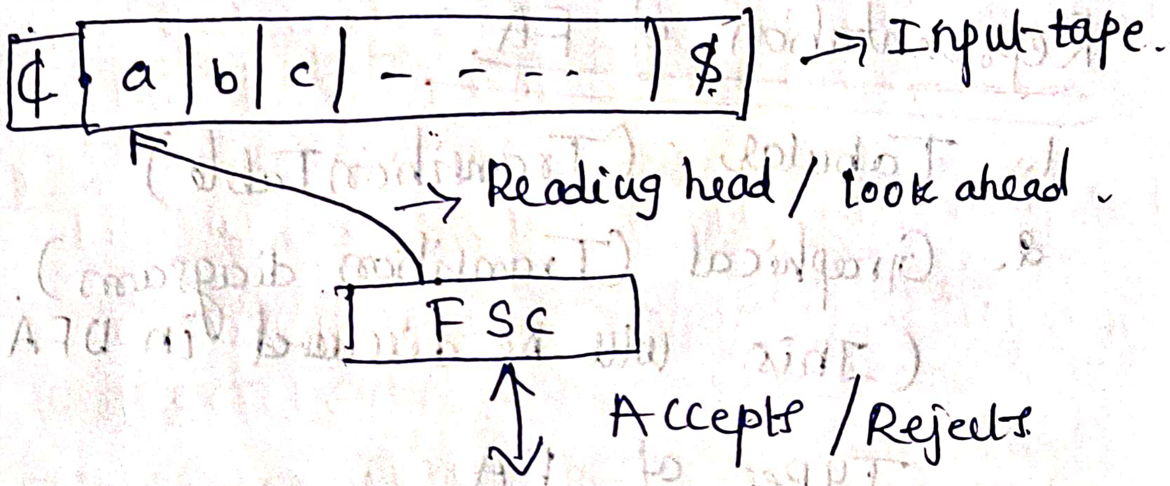
Transition Fun = $\boxed{\delta : Q \times \Sigma = Q}$.

Dr. K. Bhargavi

Finite automata Model or Block diagram of FA.

Finite automata has 3 Parts.

1. Input tape
2. Reading head
3. Finite state control (FSC).



Input tape : linear tape having number of cells. Each cell placed with single i/p symbol. End markers are ¢ and $.

It is a linear sequential tape.

String placed left to right sequence.

Tape Reader :

It reads one block at a time and can move one ~~either~~ from left to to right direction.

At the beginining of the operation the head is always at the leftmost block.

Each input symbol is placed in each c[ell]

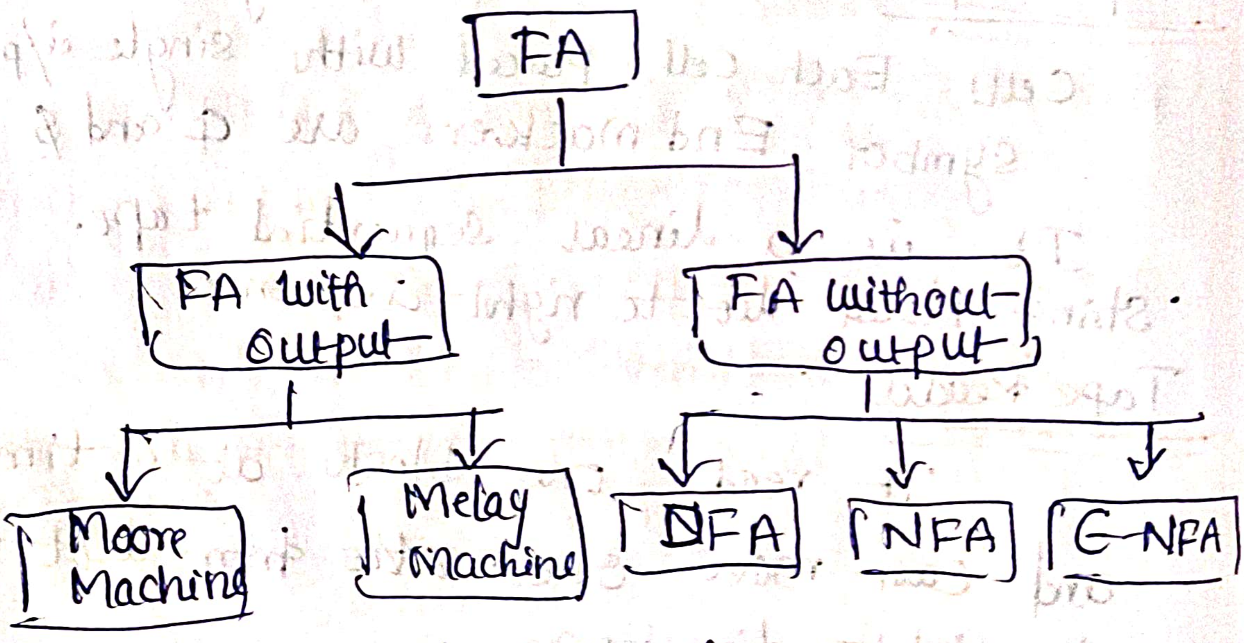
FSC : Finite control that determines the state of the automation and also control the movement of the head.

It decides the next state of receiving particular input from input tape.

Representation of FA

1. Tabular (Transition Table)
2. Graphical (Transition diagram)
   (This will be discussed in DFA).

Types of FA



1. Deterministic FA (DFA)
2. Non-Deterministic FA (NFA)
3. NFA with ϵ-moves/transitions (ϵ-NFA)

1. Deterministic Finite automata (DFA) :

2. Deterministic refers to Uniqueness of computation.

A finite automata said to be deterministic if and only if the machine reads one input string symbol at a time.

→ In DFA, only one path from the current state to next state.

→ DFA, can not accepts null moves. i.e cannot change its state without any input character.

→ DFA's are deterministic because each string and state sequence is unique.

→ A DFA accepts a string if the final state reached is an accepting state.

Formal definition of DFA :

DFA is described as 5 tuple notation same as FA.

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

where Q = Finite set of states
Σ = finite set of i/p symbols
q_0 = Initial state
δ = Transition Function
F = Final state.

$$\delta : = Q \times \Sigma \Rightarrow Q.$$

Representation of DFA

1. Transition table
2. Transition diagram / Transition graph

Transition table :

It is basically a tabular representation of the transition function. This takes 2 arguments.

→ state
→ a symbol.

and returns a value i.e next state.

δ	a	b	c
Q_1			
Q_2			
⋮			
Q_6			

— Input symbols (Σ)

States: $Q_1, Q_2, \ldots, Q_6$

Rows corresponds to states
columns corresponds to input symbols (Σ)
Entries are next-state.

Initial state is marked by arrow ($\rightarrow$)
Final state is marked by * or circle.

Example.

δ	0	1
$\rightarrow q_0$	q_0	q_1
q_1	q_2	q_1
$*q_2$	q_1	q_2

here q_0 = initial state = q_0
F = Final state = q_2.

Transition diagram

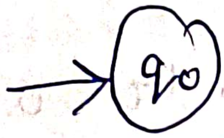
It is a directed graph associated with the vertices of the graph corresponds to the state of finite automata.

State is represented by vertices & labeled with input character

Ex: q

Initial state is represent with arrow
$\rightarrow \bigcirc$ initial state.

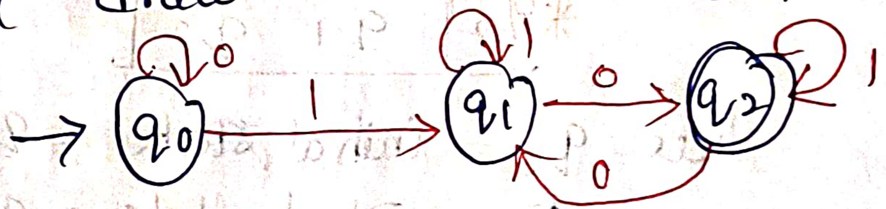
Ex: q_0 is



Final state is represented with double circle.

Ex: 

consider the previous transition table and draw the transition graph.



inputs are 0,1 , i.e $\Sigma = \{0,1\}$

States $\rightarrow q_0, q_1, q_2$

$\therefore$ DFA $\doteq M = (Q, \Sigma, \delta, q_0, F)$

$Q = (q_0, q_1, q_2)$

$\Sigma = \{0,1\}$, δ = Transition Function

q_0 = initial state $= q_0$

F = Final state $= q_2$

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

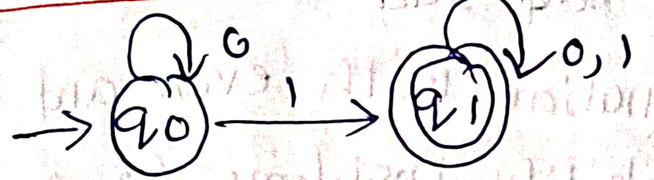
$\delta(q_0, a) = q_0$, $\delta(q_0, 1) = q_1$

$\delta(q_1, 0) = q_2$, $\delta(q_1, 1) = q_1$

$\delta(q_2, 0) = q_1$, $\delta(q_2, 1) = q_2$

$\delta(q_0, a) = q_0$ i.e q_0 on a input is q_0

Ex: 1.



convert into Transition table.

Sol:

$Q = (q_0, q_1)$

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

q_0 = Initial state $= q_0$

F = Final state $= q_1$

δ	0	1
q_0	q_0	q_1
q_1	q_1	q_1

$\delta \Rightarrow$ $\delta(q_0, 0) = q_0$, $\delta(q_0, 1) = q_1$

$\delta(q_1, 0) = q_1$, $\delta(q_1, 1) = q_1$

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

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

DFA Problems :

Dr. K. Bhargavi

1. Start with
2. End with
3. Start and End with
4. Length (equal 2, ≤ 2, ≥ 2)
5. Even and odd
6. Combination with even and odd
7. Divisibility Problems (2, 3, 4, 5)
8. Substring

1) Design / construct a DFA which accepts String starting with 0, over $\Sigma = \{0,1\}$

Solu: Given $\Sigma = \{0,1\}$

Step 1: Write the Language for given

$L = \{\underbrace{0}_{\text{Length 1}}, \underbrace{01, 00}_{\text{Length 2}}, \underbrace{000, 001, 010, 011}_{\text{Length 3}} \dots\} \Rightarrow$ Infinite Language

Step 2: Find Minimum Length of String from the language.

$L = \{0, 01, 00, 000, 001, 010, \dots\}$

1 2 3

Minimum length of String $|W| = 1$

Step 3: Find the no. of states required for DFA

No. of states required = Min. Length of String + 1

$= 1+1 = 2$

2 states are required.

Let 2 states are q_0, q_1

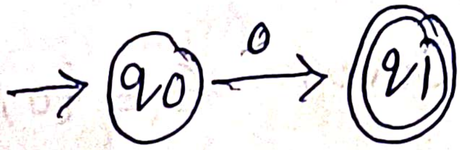
i.e $Q\ (q_0, q_1)$

Let Initial state $= q_0$

$L = \{0, 01, 00, 000, 001, 010, 011, \ldots\}$

$L = \{\Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \Sigma^4 \cup \ldots\}$

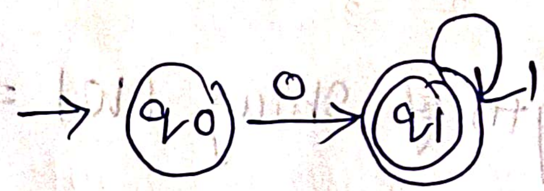
String 0 –



started from q0 & reads/input 0 & reaching to q1, which is Final state.

Next string is 01 } length of string is 2 & after reading this string, should reach to the final state.

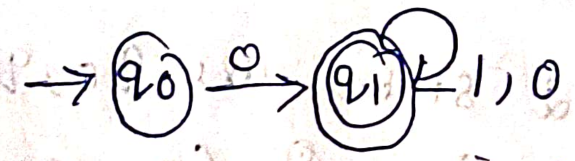
→q0 →0 – q1 → 1 → q1



i.e q0 reads the i/p 0 & reaches to q1; q1 reads the i/p 1 and reaches to q1 only which is final state. can't take on q1 on 1 to q0 because not reaching final state.

Read 00 string :

q0 → 0 → q1 → 0 → q1



can't take q1 on 0 is q0 because q0 is not final state.

000 $\Rightarrow$ $q_0 \to 0 \to q_1 \to 0 \to q_1 \to 0 \to q_1$

which is accepting.

010 $\Rightarrow$ $q_0 \xrightarrow{0} q_1 \xrightarrow{1} q_1 \xrightarrow{0} q_1$ — which is Final state.

Now let's take any string which starts with '0'

01001

$\Rightarrow q_0 \to 0 \to q_1 \to 1 \to q_1 \to 0 \to q_1 \to 0 \to q_1 \to 1 \to q_1$

which satisfying. i.e reaching q_1 final state.

Note:
Need to check for DFA is, every state should have the path for given set of input symbols.

2 states $\Rightarrow$ $Q = (q_0, q_1)$

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

$\therefore$ q_0 should have path for 0 & 1

q_1 should have path for 0 & 1

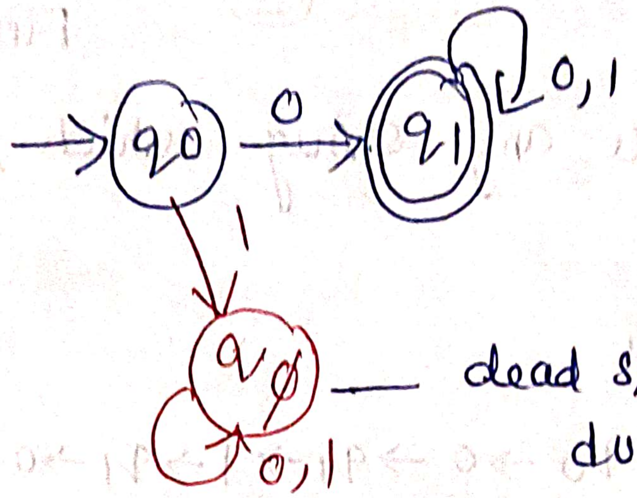
① q_0 < 0 — q_1, 1 — No Path

② q_1 < 0 — q_1, 1 — q_1

q_1 on 1 we don't have path.

∴ consider the dead state(q_ϕ) take the path q_1 on 1.

So the complete DFA is



— dead state / trap state / dummy state.

if we give q_ϕ on $0 \to q_\phi$

q_ϕ on $1 \Rightarrow q_\phi$

Write the Transition function for DFA

$\delta(q_0, 0) = q_1$ $\quad \delta(q_0, 1) = q_\phi$

$\delta(q_1, 0) = q_1$ $\quad \delta(q_\phi, 1) = q_1$

$\delta(q_\phi, 0) = q_\phi$ $\quad \delta(q_\phi, 1) = q_\phi$

Transition ~~diagram~~ table

δ	0	1
$\to q_1$	q_1	q_ϕ
q_2	q_2	q_2
q_ϕ	q_ϕ	q_ϕ

Q2) Design DFA, which accepts all strings start with 01, over $\Sigma = \{0,1\}$

Soln: Start with 01, $\Sigma = \{0,1\}$

$L = \{01, 010, 011, 0100, 0101, \ldots\}$

Minimum length of string = 2

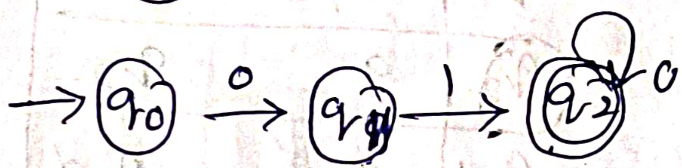
No. of states requires = 2+1 = 3

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

$q_0 = q_0$.

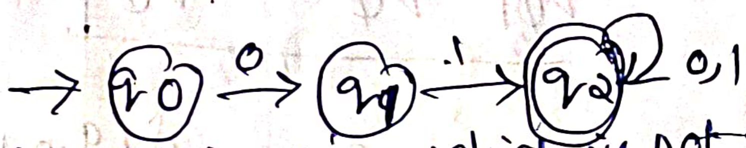
01

010



After reading reaching Final state

011



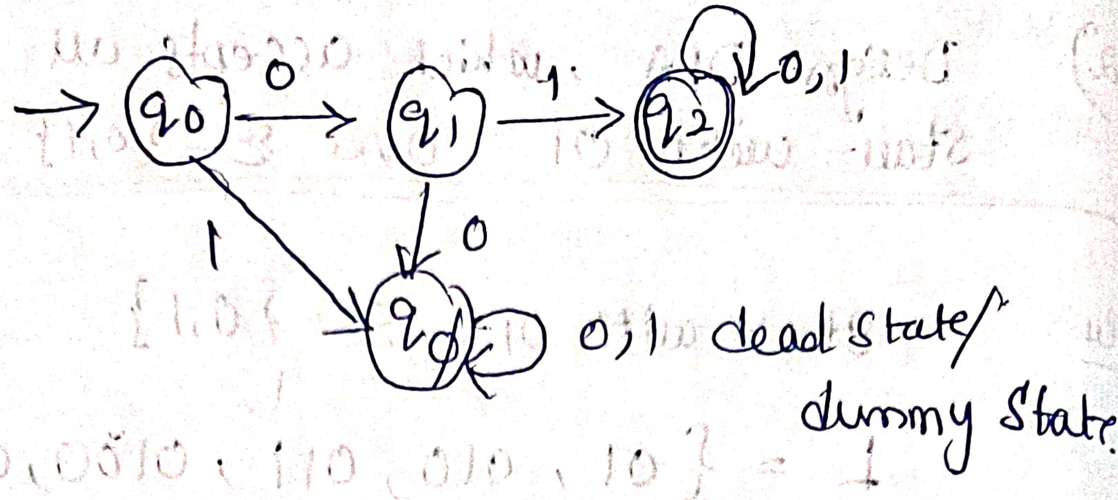
~~can't take q_2 on 1 to q_1, which is not final state~~

0100, 0101 also satisfying the above diagram.

But not complete diagram.

$q_0 \begin{cases} 0 - q_0 \\ 1 - ? \end{cases}$; $q_1 \begin{cases} 0 - ? \\ 1 - q_2 \end{cases}$ $q_2 \begin{cases} 0 - q_2 \\ 1 - q_2 \end{cases}$

Don't have path for q_0 on 1 & q_1 on 0, so consider dead state



Now q_0 on 1 is q_ϕ

q_1 on 0 is q_ϕ.

Transition table for above one

δ	0	1
$\rightarrow q_0$	q_1	q_ϕ
q_1	q_ϕ	q_2
q_2	q_2	q_2
q_ϕ	q_ϕ	q_ϕ

$q_0 = q_0$

$F = q_2$.

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

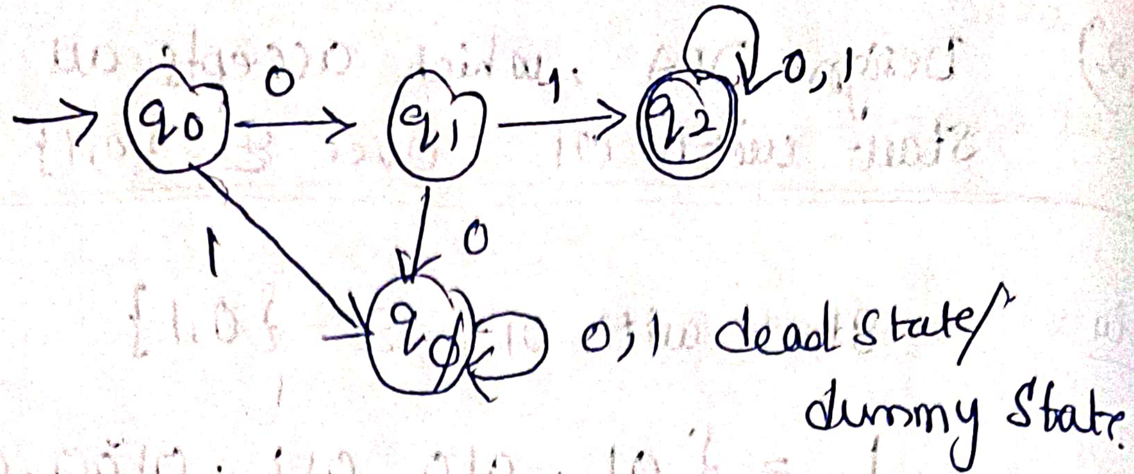
$\delta(q_1, 0) = q_\phi$, $\delta(q_1, 1) = q_2$

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

$\delta(q_\phi, 0) = q_\phi$, $\delta(q_\phi, 1) = q_\phi$

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

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



Now q_0 on 1 is $q\phi$

q_1 on 0 is $q\phi$

Transition table for above one

δ	0	1
$\rightarrow q_0$	q_1	$q\phi$
q_1	$q\phi$	q_2
q_2	q_2	q_2
$q\phi$	$q\phi$	$q\phi$

$q_0 = q_0$

$F = q_2$

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

$\delta(q_1, 0) = q\phi$, $\delta(q_1, 1) = q_2$

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

$\delta(q\phi, 0) = q\phi$, $\delta(q\phi, 1) = q\phi$

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

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

Q3) construct the DFA, which accepts end with 0, over $\Sigma = \{0,1\}$.

Soln: End with 0

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

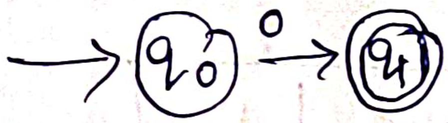
$L = \{0, 00, 10, 110, 100, 010, 1010, 101010, ---\}$

Minimum Length = 1

No. of states require = 1+1 = 2

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

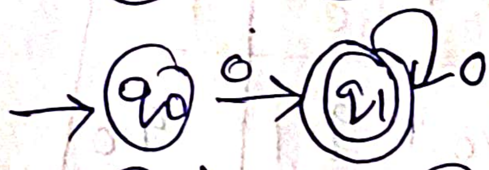
$q_0 = q_0$

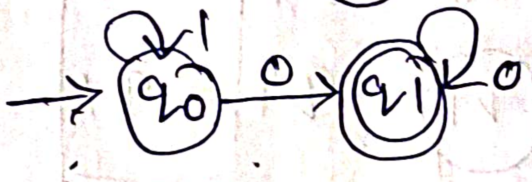
0 

q_1 on $0 \to q_0$ not possible

q_0 is not Final state

$\therefore q_0 \xrightarrow{0} q_1$ which is Final state

00 

10 

100, 010 are satisfying.

consider 1010 

$q_0 \to 1 \to q_0 \to 0 \to q_1 \to 1 - q_0 \to 0 - q_1$

Take 101010

$q_0 \to 1 \to q_0 \to 0 \to q_1 \to 1 \to q_0 \to 0 \to q_0 \to 0 \to q_1$

satisfying

Check complete DFA or not

$q_0 \begin{cases} 0 - q_1 \\ 1 - q_0 \end{cases}$ having path for $\Sigma = \{0,1\}$.

$q_1 \begin{cases} 0 - q_1 \\ 1 - q_1 \end{cases}$ ∴ complete DFA

Transition Function (δ)

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

Transition diagram

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

q_0 = Initial state
q_1 = Final state
$\Sigma = \{0,1\}$

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

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

Dr. K. Bhargavi

Q4 Design DFA, start with 0 & end with 1 over $\Sigma = \{0,1\}$

Soln: Start with 0 & End with 1

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

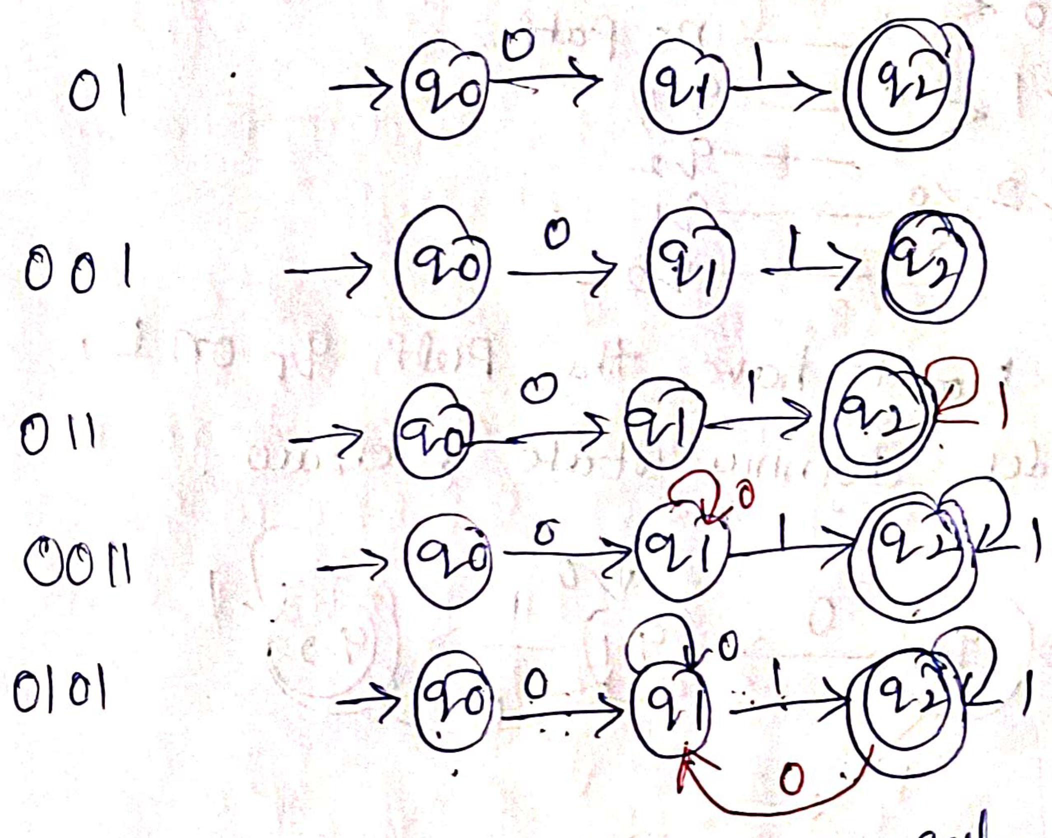
$L = \{01, 001, 011, 0011, 0101, 01001;$

Minimum length = 2

No. of states required = 2+1 = 3.

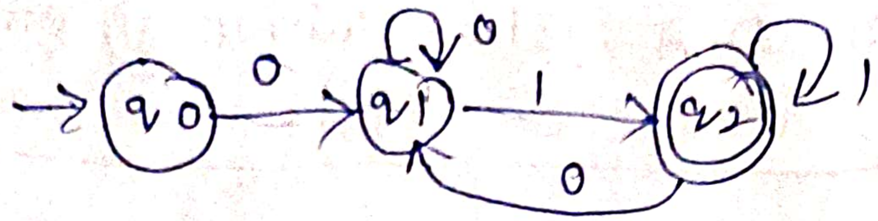
let $Q = \{q_0, q_1, q_2\}$.

$q_0 = q_0$, $F = q_2$



→ cannot take for q_0 self already path exist, q_2 cannot be

here cannot take on q_2 self because, should end with 1 only. cannot go to q_0 also.



Check one string $\underline{0}01001\underline{1}$, start 0 & end 1

$q_0 \to 0 \to q_1 \to 0 \to q_1 \to 1 \to q_2 \to 0 \to q_1 \to 0$

$q_1 \to 1 \to q_2 \to 1 \to q_2$

Reaching final string after reading that

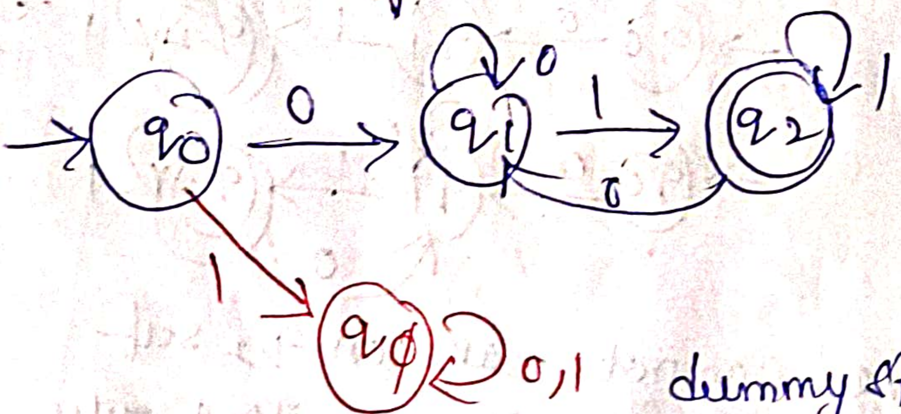
Check complete DFA or not

q_0 < 0 — q_1 ; 1 — no Path.

q_1 < 0 — q_1 ; 1 — q_2

q_2 < 0 — q_1 ; 1 — q_2

Don't have the Path q_0 on 1, consider dummy state & draw.



dummy state.

Transition function (δ)

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

$\delta(q_1, 0) = q_1$ $\delta(q_1, 1) = q_2$

$\delta(q_2, 0) = q_1$ $\delta(q_2, 1) = q_2$

Transition Table

δ	0	1
$\rightarrow q_0$	q_1	q_ϕ
q_1	q_1	q_2
$* q_2$	q_1	q_2
q_ϕ	q_ϕ	q_ϕ

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

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

Q Design DFA starts with 1 and end with 0

Sol:

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

Start with 1 & End with 0.

$L = \{10, 100, 110, 1000, 1010, 1100, \ldots\}$

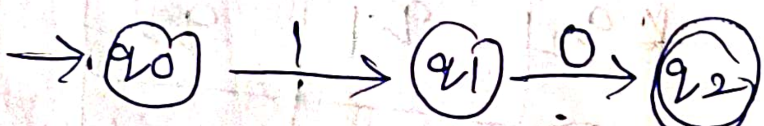
Min length = 2

No. of states required $= 2+1 = 3$

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

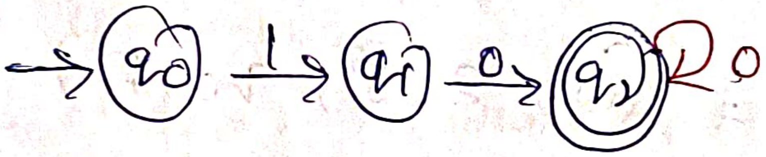
$q_0 = q_0$, $F = q_2$

10



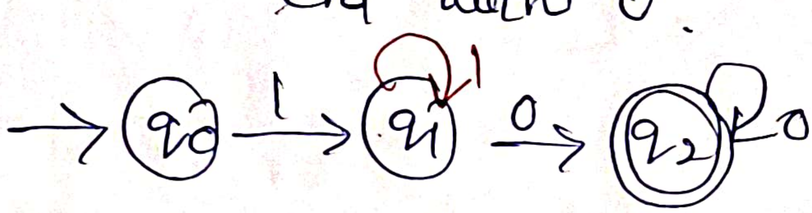
After reading reaching final state

100.



can it take on q_1 because path exist q_1 ~~on~~ 0 is q_2. on q_2 is possible because it can end with 0.

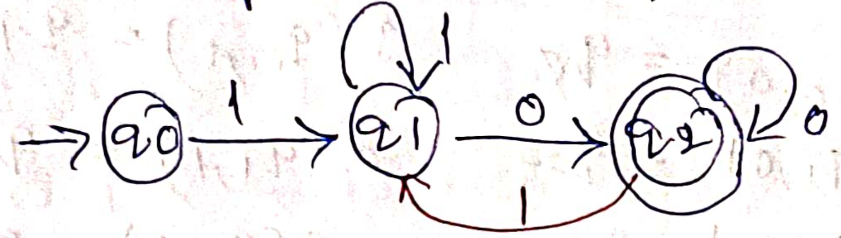
110



we can't take on q_0, already path exist, can't take on q_2, should end with 0.

$\therefore$ q_1 on 1 is q_1

1000 satisfying same diagram.



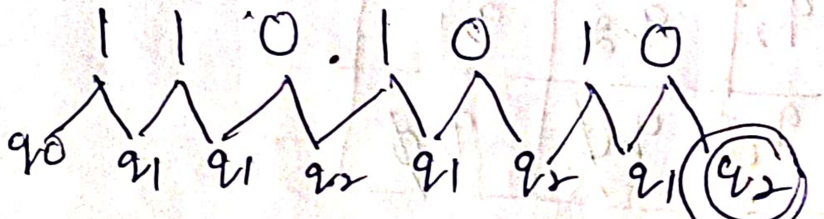
1010

can't take self loop on q_2
because string should end with '0'
only.

$q_0 \to 1 \to q_1 \to 0 \to q_2 \to 1 \to q_1 \to 0 \to q_2$

which is satisfying.

consider one string $\boxed{1101010}$

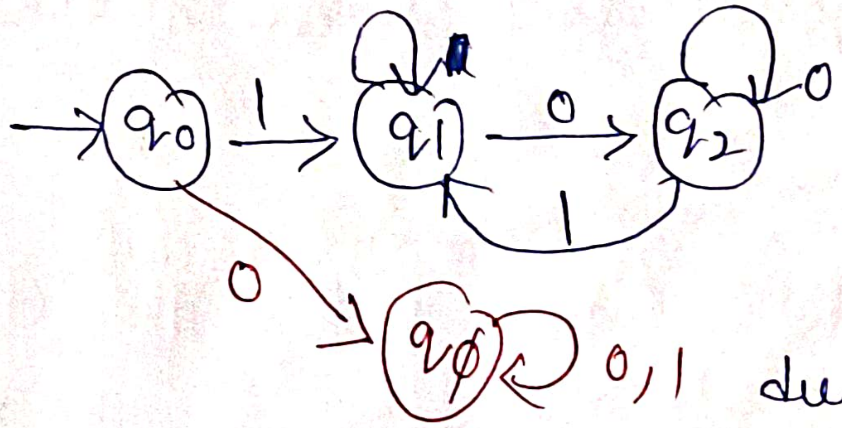


satisfying

Check complete DFA or not.

q_0 does not have the path for '0'
q_1 & q_2 have the path for 0 & 1
∴ consider dummy state for

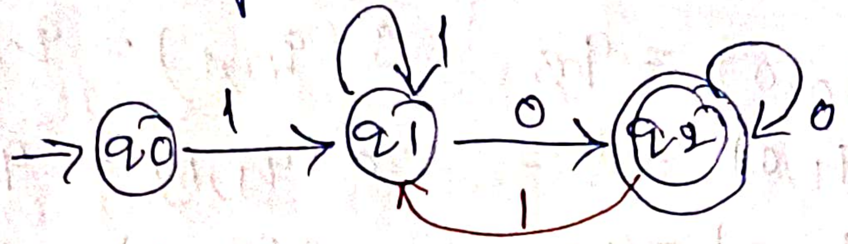
$\boxed{q_0 \text{ on } 0 \text{ is } q_\phi}$



dummy state.

1000 satisfying same diagram.

1010



can't take self loop on q_2 because string should end with '0' only.

$q_0 \rightarrow 1 \rightarrow q_1 \rightarrow 0 \rightarrow q_2 \rightarrow 1 \rightarrow q_1 \rightarrow 0 \rightarrow q_2$

which is satisfying.

consider one string. $\boxed{1101010}$

1 1 0 1 0 1 0

q_0 q_1 q_1 q_2 q_1 q_2 q_1 q_2 satisfying

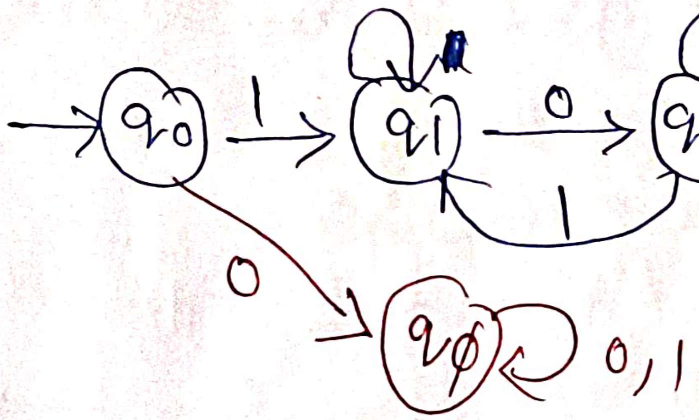
Check complete DFA or not.

q_0 does not have the path for '0'

q_1 & q_2 have the path for 0 & 1

∴ consider dummy state for

$\boxed{q_0 \text{ on } 0 \text{ is } q_\phi}$



dummy state.

Transition Function:

$\delta(q_0, 0) = q_\phi$ $\quad$ $\delta(q_0, 1) = q_1$

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

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

Transition Table

δ	0	1
$\rightarrow q_0$	q_0	q_1
q_1	q_2	q_1
$* q_2$	q_2	q_1
q_ϕ	q_ϕ	q_ϕ

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

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

Q6 Design DFA end with 10, $\Sigma = \{0,1\}$

Sol: End with 10

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

$L = \{10, 010, 110, 0110, 0010, 1010, 1110, \ldots\}$

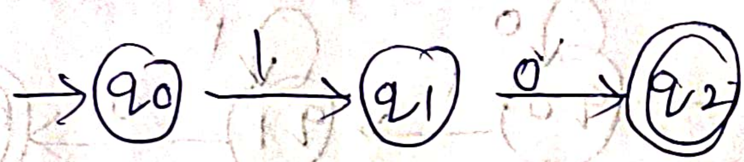
minimum length = 2

No. of states required = 2+1 = 3

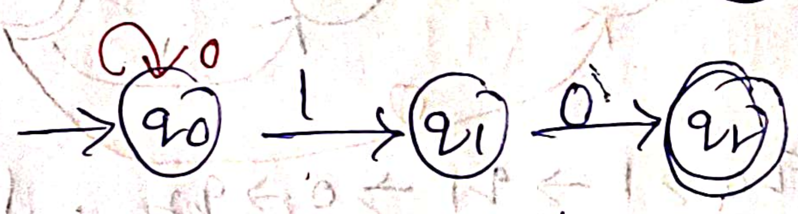
Let $Q = \{q_0, q_1, q_2\}$

$q_0 = q_0$, $F = q_2$

10



010



here can't take for q_1, q_2 also so q_0 on 0 is q_0 & moving forward & reaching final state q_2

110

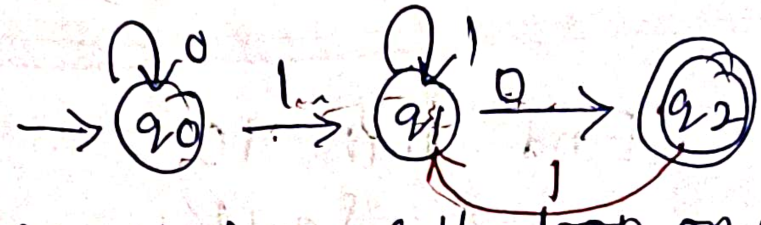
can't take q_0, already path exist

q_2 can't take string end with 10

so q_1 on 1 = q_1

Above diagram is satisfying 0110 also

1010



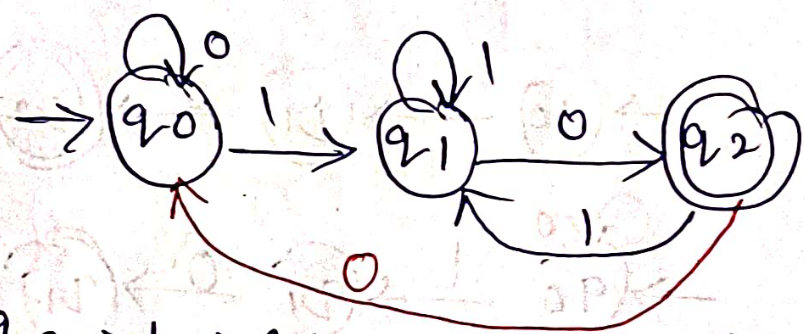
can't take self loop on q_2 for 1 or 0 bez satisfying 1011, voilating rule.

Consider q_2 on 1 $\Rightarrow q_1$

$q_0 \rightarrow 1 \rightarrow q_1 \rightarrow 0 \rightarrow q_2 \rightarrow 1 \rightarrow q_1 \rightarrow 0 \rightarrow q_2$

Reaching Final state.

10010.



$q_0 \rightarrow 1 \rightarrow q_1 \rightarrow 0 \rightarrow q_2$, here can't take self loop for q_2, satisfying 1000 also. So take q_2 on 0 to q_0

$q_0 \rightarrow 1 \rightarrow q_1 \rightarrow 0 \rightarrow q_2 \rightarrow 0 \rightarrow q_0 \rightarrow 1 \rightarrow q_1 \rightarrow 0 \rightarrow q_2$

satisfying.

It is complete DFA also, having path (single) for 0 & 1 for all state

Transition Table

δ	0	1
$\rightarrow q_0$	q_0	q_1
q_1	q_2	q_1
$*q_2$	q_0	q_1

Transition Function

$\delta(q_0, 0) = q_0$

$\delta(q_0, 1) = q_1$

$\delta(q_1, 0) = q_2$

$\delta(q_1, 1) = q_1$

$\delta(q_2, 0) = q_0$

$\delta(q_2, 1) = q_1$

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

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

Q7 Design DFA, which accepts string length equal to 2 over $\Sigma = \{a, b\}$

Soln:

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

length = 2, Σ^2 over $\Sigma = \{a, b\}$

i.e $L = \{aa, ab, ba, bb\}$

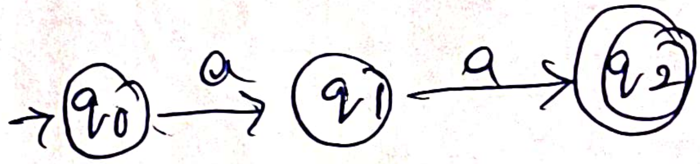
Finite Language

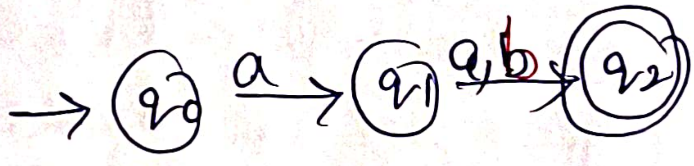
Min. Length = 2

No. of states required = 2+1 = 3

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

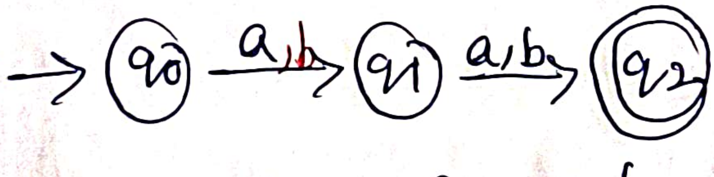
$q_0 = q_0$, $F = q_2$

aa 

ab 

reading a from q_0 on a = q_1 reached now read b, reach final state

$\therefore$ q_1 on b = q_2 ⇒ final state.

ba 

Can't take q_0 on b because reaching q_1, which is not final state. so q_0 on b = q_1

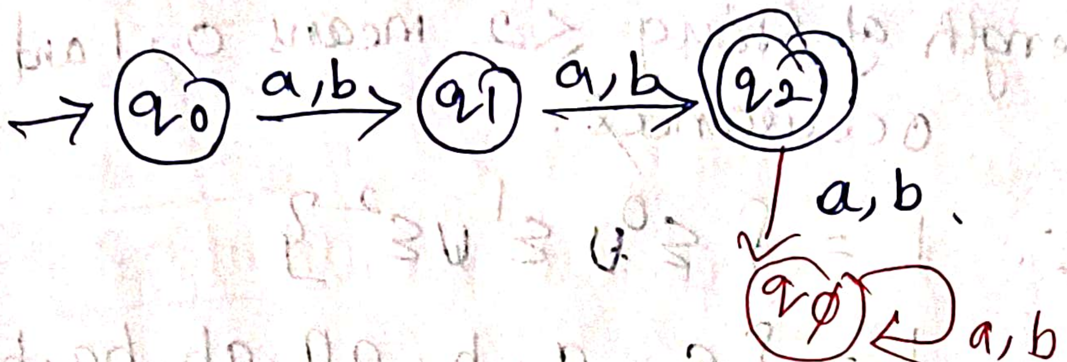
Also satisfying bb.

check for DFA or not

q_0 & q_1 having path for a & b

but not q_2.

consider q_ϕ, dead state on q_2 for a, b



Transition Function (δ):

$\delta(q_0, a) = q_1$ $\delta(q_0, b) = q_1$

$\delta(q_1, a) = q_2$ $\delta(q_1, b) = q_2$

$\delta(q_2, a) = q_\phi$ $\delta(q_2, b) = q_\phi$

Transition diagram

δ	a	b
$\rightarrow q_0$	q_1	q_1
q_1	q_2	q_2
$*q_2$	q_ϕ	q_ϕ
q_ϕ	q_ϕ	q_ϕ

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

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

Design DFA, having string ≤ 2 over (Q8)
$\Sigma = \{a, b\}$

Soln:

Length of string $\leq 2 \Rightarrow$ Finite Language

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

Length of string ≤ 2 means 0, 1 and 2 occurances.

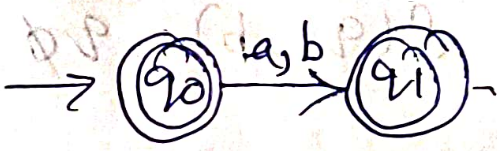
$\therefore L = \{\Sigma^0 \cup \Sigma^1 \cup \Sigma^2\}$

$L = \{\epsilon, a, b, aa, ab, ba, bb\}$

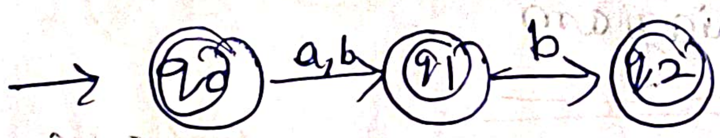
$[\because \Sigma^0 = \epsilon]$

ϵ

a or b



ab



ba

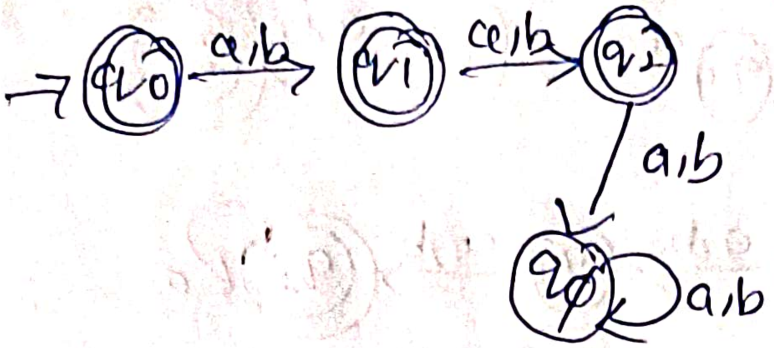
The above one, accepts the strings bb and aa also.

For q_0 and q_1, have the path for a & b

Don't have q_2 for a & b path,

i.e Not in DFA.

consider dummy/dead state for q_2 on a & b



dead state.

δ	a	b
$\rightarrow q_0$	q_1	q_1
q_1	q_2	q_2
q_2	q_ϕ	q_ϕ

$\delta(q_0, a) = q_1 \quad \delta(q_0, b) = q_1$
$\delta(q_1, a) = q_2 \quad \delta(q_1, b) = q_2$
$\delta(q_2, a) = q_\phi, \delta(q_2, b) = q_\phi$.

$M = (Q, \Sigma, \delta, q_0, F)$.
$M = ((q_0, q_1, q_2), \{a,b\}, \delta, q_0, \{q_0, q_1, q_2\})$.

Q. Construct DFA, where length of string is ≥ 2 over $\Sigma = \{a, b\}$

Soln:

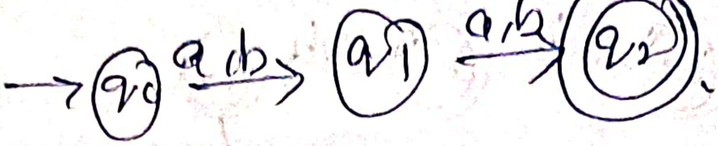
Length string ≥ 2. $\Rightarrow$ Infinite Language.
$\Sigma = \{a, b\}$.

$L = \{aa, ab, ba, bb, abb, aba, baa, \ldots\}$
min Length $= 2$
No. of states required $= 2+1 = 3$
Let $Q = \{q_1, q_2, q_3\}$.

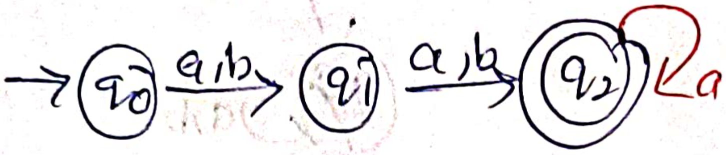
aa

ab

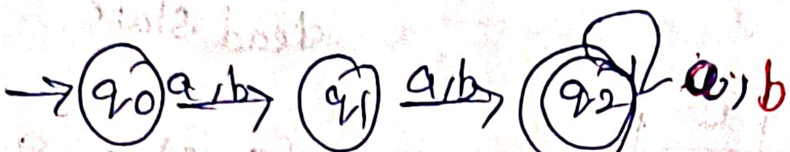
ba



aba



abb



This can accept bab, bbb, baba ----

also complete DFA

i-e having all states for a, b paths.

δ	a	b
$\rightarrow q_0$	q_1	q_1
q_1	q_2	q_2
q_2	q_2	q_2

$\delta(q_0, a) = q_1 \quad \delta(q_0, b) = q_1$

$\delta(q_1, a) = q_2, \delta(q_1, b) = q_2$

$\delta(q_2, a) = q_2 \ \delta(q_2, b) = q_2$

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

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

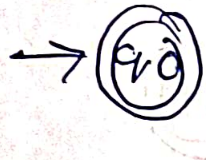
Dr. K. Bhargavi

Q10 DFA accepts even length string :

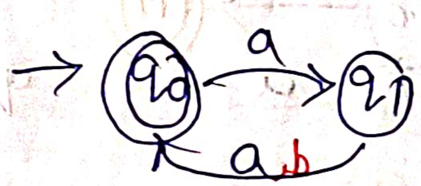
$\Sigma = \{a, b\}$

$L = \{\epsilon, aa, ab, ba, bb, aaaa, abab, baba, bbbb, \ldots\}$ → Infinite Language.

i.e $\boxed{n \bmod 2 = 0}$

ϵ  without reading any input, it reaches to final state. ϵ

aa 

ab 

ba 

Satisfying all other strings also.

Draw the transition table and Write

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

$q_0 = q_0$, $F = q_0$.

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

Where $\delta(q_0, a) = q_1$, $\delta(q_0, b) = q_1$

$\delta(q_1, a) = q_0$, $\delta(q_1, b) = q_0$.

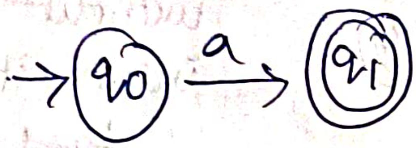
(11) Design DFA accepts the string is Odd, Σ=

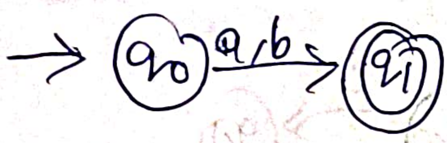
$\Sigma = \{a, b\}$

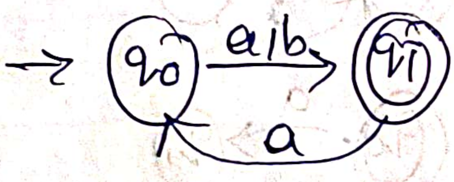
String is odd length.

i.e $L = \{a, b, aaa, aba, abb, baa, \ldots\}$

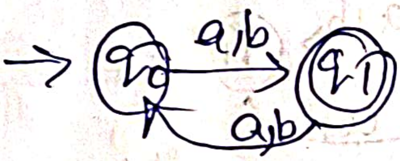
→ Infinite Language.

a 

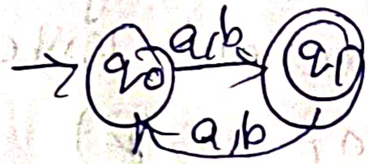
b 

aaa 

$q_0 - a - q_1 - a - q_0 - a - q_1$

aba 

$q_0 \to a - q_1 \to b - q_0 - a - q_1$

baa 

$q_0 \to b \to q_1 - a \to q_0 - a - q_1$

δ	a	b
$\to q_0$	q_1	q_1
q_1	q_0	q_0

$\delta(q_0, a) = q_1 \quad \delta(q_0, b) = q_1$

$\delta(q_1, a) = q_0, \delta(q_1, b) = q_0$

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

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

(12) construct DFA, which accepts Even no. of a's & Even no. of b's over $\Sigma = \{a,b\}$

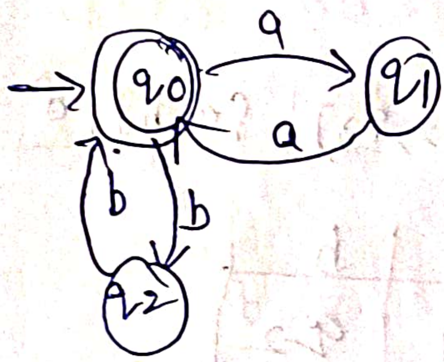
Even no. of a's & Even no. of b's.

$L = \{ \epsilon, aa, bb, abab, aabb, bbaa, baba, abab, --- \}$

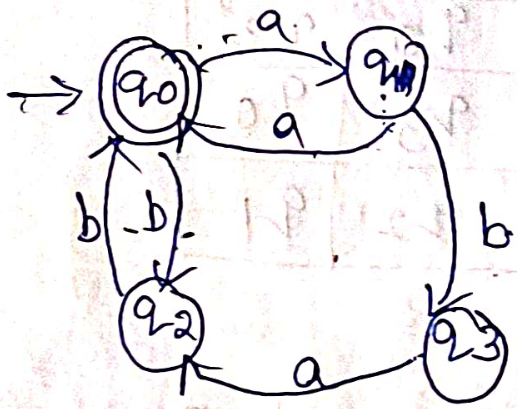
ϵ

aa

bb.



ab ab.

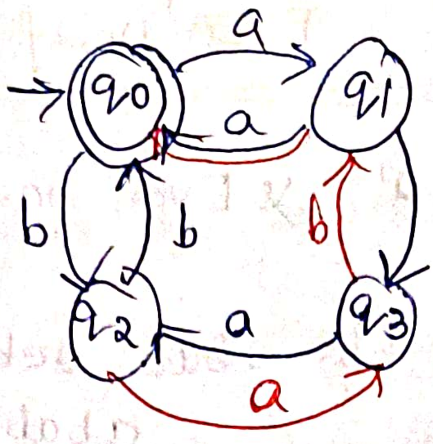


$q_0 \to a - q_1 \to b \to q_3 - a - q_2 - b \to q_0$

bbaa & aa bb will satisfy the same above diagram.

Dr. K. Bhargavi

baba



$q_0 - b \rightarrow q_2 - a - q_3 - b - q_1 - a - q_0$

$q_0 = q_0$, $F = q_0$.

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

$M = ((q_0, q_1, q_2, q_3), \{a, b\}, \delta, q_0, q_0)$

where

δ	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_0	q_3
q_2	q_3	q_0
q_3	q_2	q_1

Similarly

a) Even no. of a's and Even no. of b's = q_0 Final state

b) odd no. of a's & ~~odd~~ Even no. of b's = $q_1 \rightarrow F$

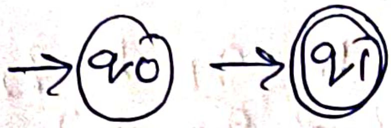
c) Even no. of a's & odd no. of b's = $q_2 \rightarrow F$

d) odd no. of a's & odd no. of b's = $q_3 \rightarrow F$

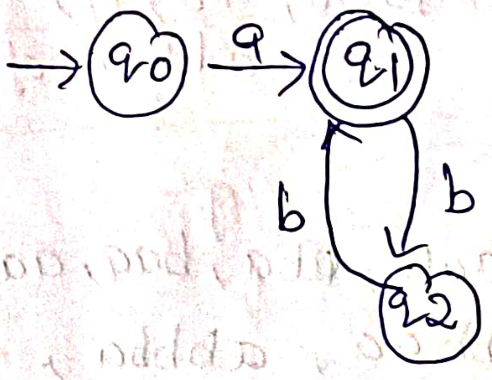
(b) Add no. of a's & Even no. of b's

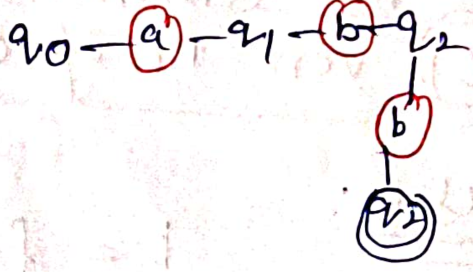
L = { a, abb, bba, bab, abbbb, ---, abbaa, }

a

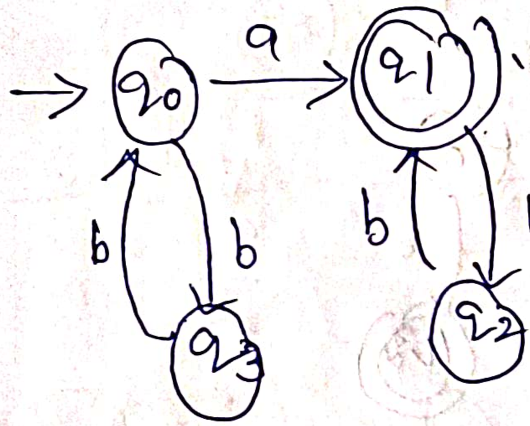


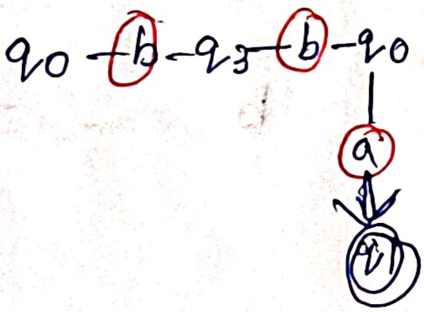
abb



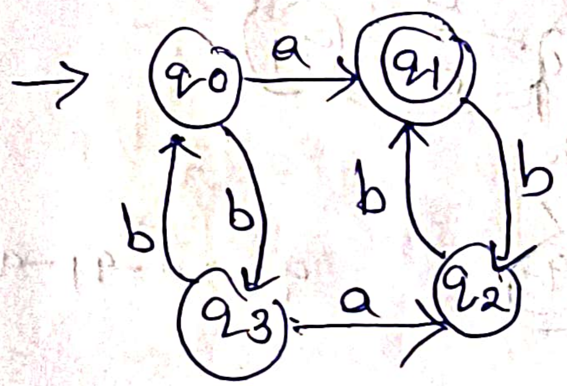


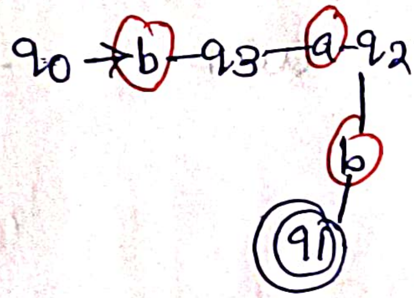
bba



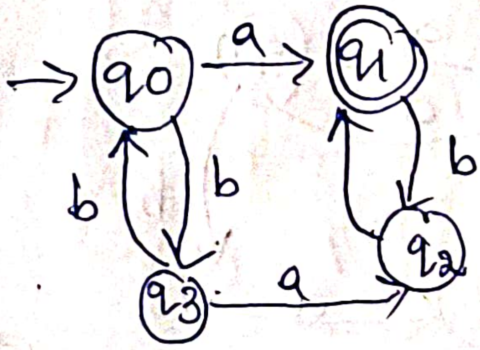


bab



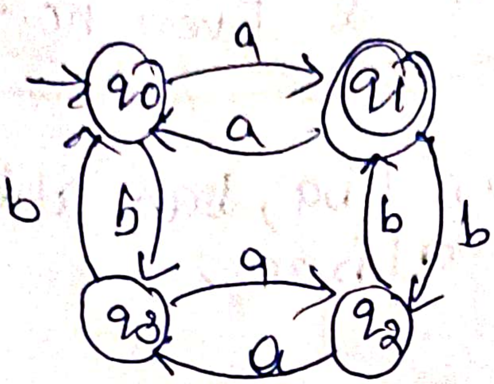


abbbb



babbb

Write transition table for the DFA



$q_0 \rightarrow b \rightarrow q_3 \rightarrow a \rightarrow q_2 - b - q_1 - b - q_2 - b \rightarrow q_1$

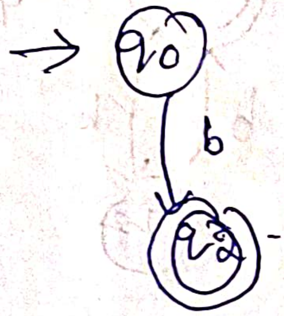
$M = ((q_0, q_1, q_2, q_3), \{a, b\}, \delta, q_0, q_1)$

c) Even no. of a's and odd no. of b's

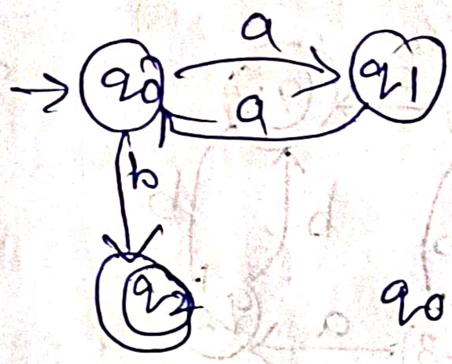
Soln:

$L = \{b, aab, aba, baa, aaaab, bbbaa, abbba, \text{---}\}$

b

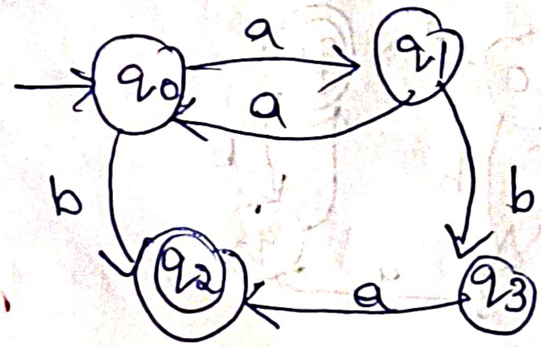


aab



$q_0 \rightarrow a - q_1 - a - q_0 - b - q_2$

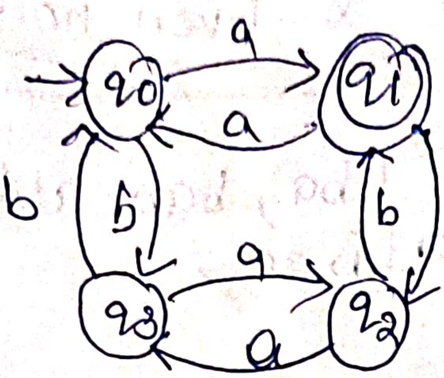
aba



$q_0 - a \rightarrow q_1 - b - q_3 - a - q_2$

babbb

Write transition table for the DFA



$q_0 \to b \to q_3 \to a \to q_2 - b - q_1 - b - q_2 - b \to q_1$

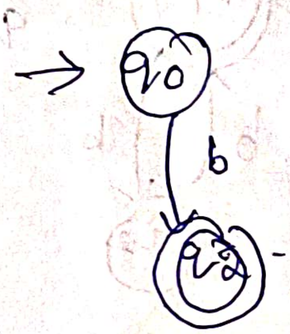
$M = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \delta, q_0, q_1)$

(c) Even no. of a's and odd no. of b's

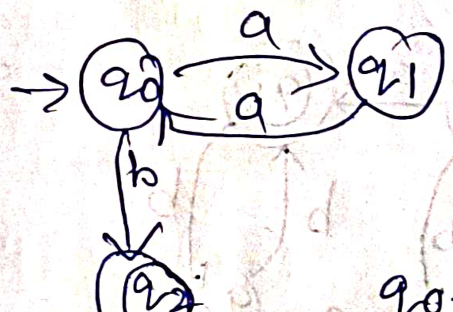
Soln:

$L = \{b, aab, aba, baa, aaaab, bbbaa, abbba, ---\}$

b

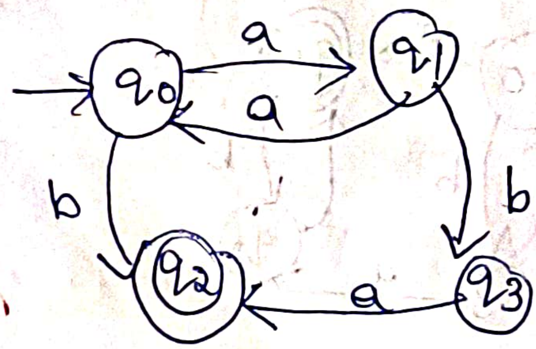


aab



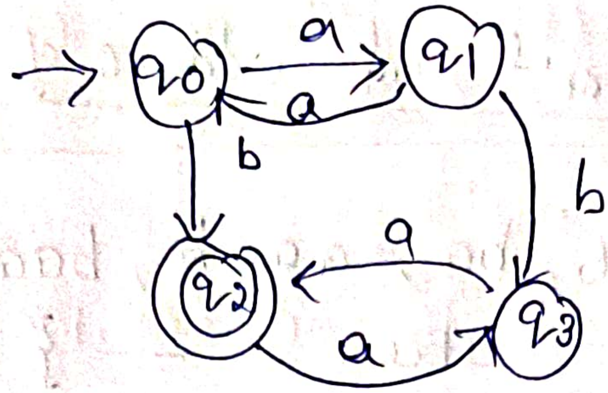
$q_0 \to a - q_1 - a - q_0 - b - q_2$

aba



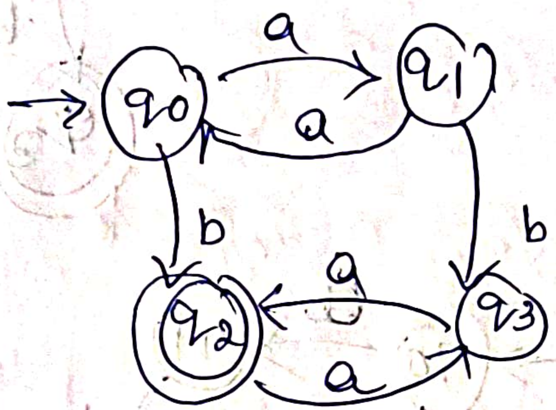
$q_0 - a \to q_1 - b - q_3 - a - q_2$

baa



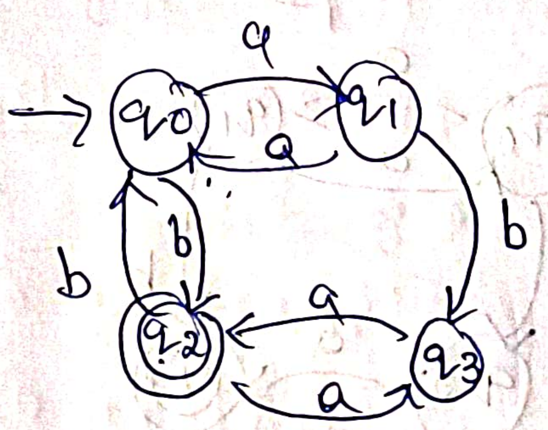
$q_0 - b \to q_2 \to a - q_3 - a - q_2$

aaaab



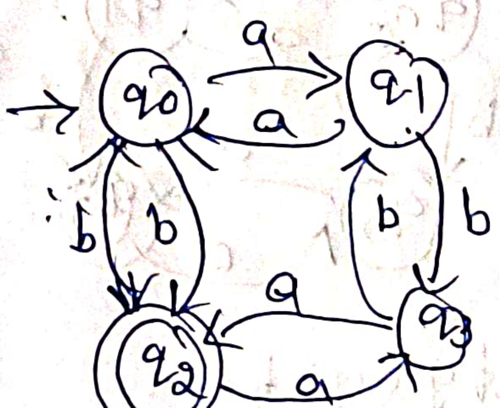
$q_0 - a - q_1 - a - q_0 \to a - q_1 - a - q_0 - b - q_2$

bbbaa



$q_0 \to b \to q_2 \to b - q_0 \to b \to q_2 \to a \to q_3 - a - q_2$

abbba



$q_0 - a - q_1 - b - q_3$

$b - q_1$

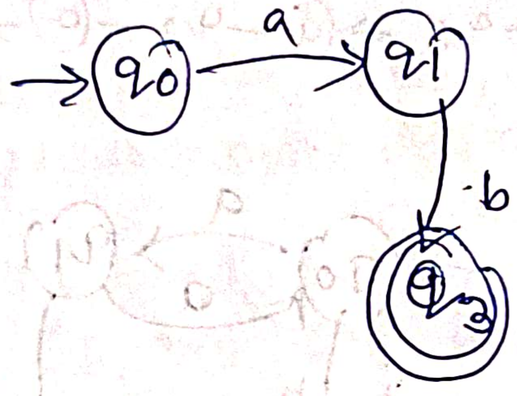
$b - q_3$

$a - q_2$

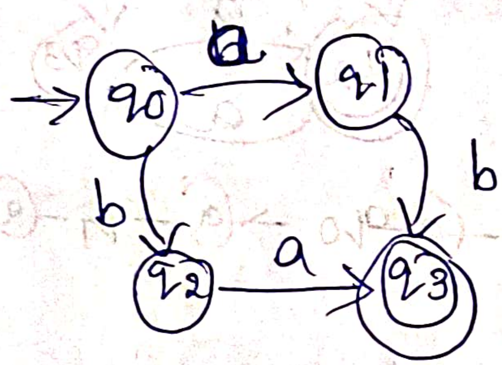
(d) Odd no. of a's and odd no. of b's

L = { ab, ba, aaab, baaa, abbb, abaa ..... }

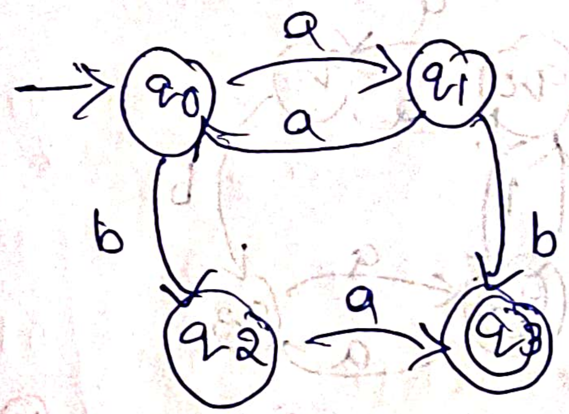
ab



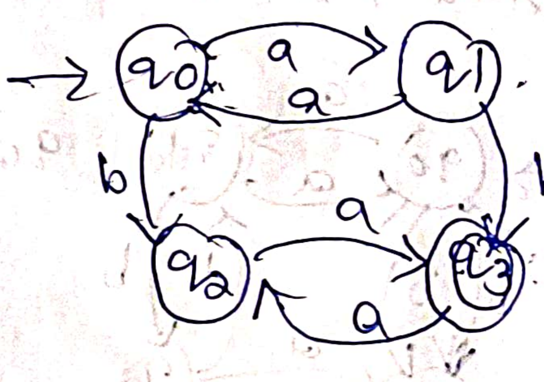
ba



aaab

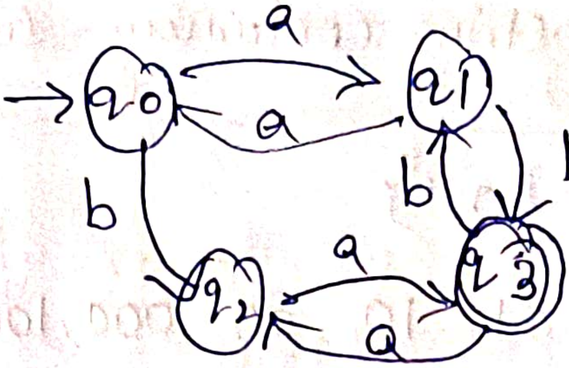


baaa

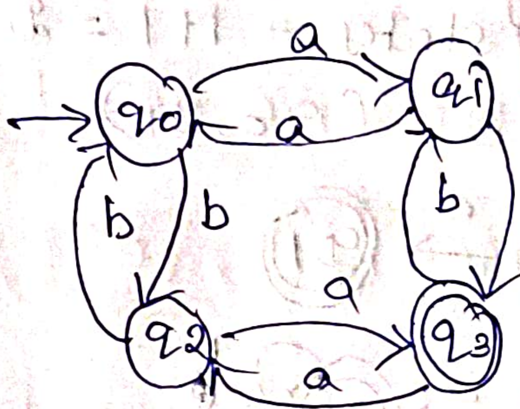


Satisfying baaa

abbb



bbba



Observed that, the final states are

q_0 — Even no. of a's & Even no. of b's

q_1 — Odd no. of a's & Even no. of a's.

q_2 — Even no. of a's & odd no. of b's

q_3 — odd no. of a's & odd no. of b's

(13) DFA accepts string containing substring 0

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

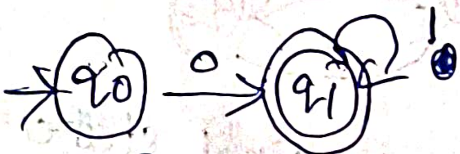
$L = \{0, 01, 10, 00, 000, 100, 001, 101, \ldots\}$

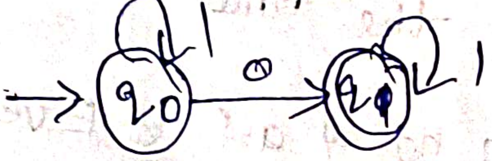
Min length = 1

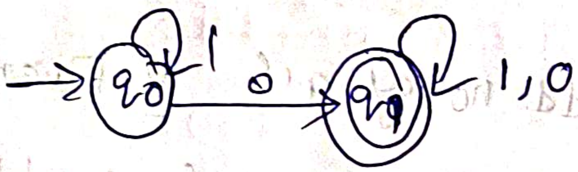
No. of states = 1+1 = 2

Let $Q = (q_0, q_1)$.

0 

01. 

10 

000 

Satisfying all other strings also.

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

$\delta(q_0, 0) = q_1 \quad \delta(q_0, 1) = q_0$

$\delta(q_1, 0) = q_1 \quad \delta(q_1, 1) = q_1$

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

(14) DFA accepts strings containing substring 00.

L = {00, 000, 001, 100, 1001, 1100, 0011, 10100, ---}

Σ = {0,1}.

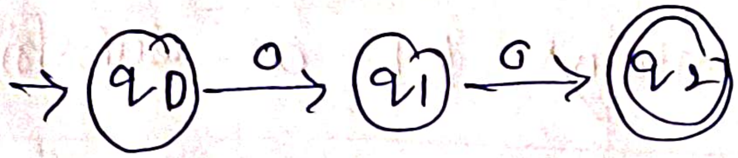
00 :- Min. length = 2

No. of states = 2+1 = 3

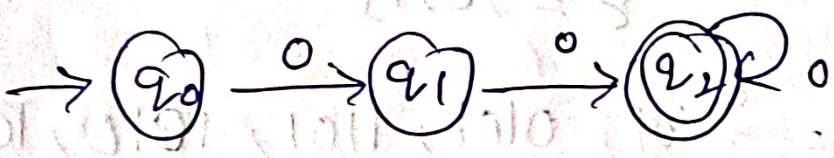
Let Q = (q_0, q_1, q_2)

Let q_0 = q_0

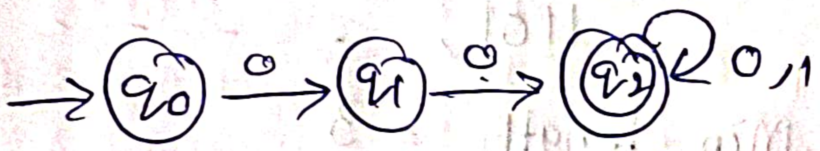
00



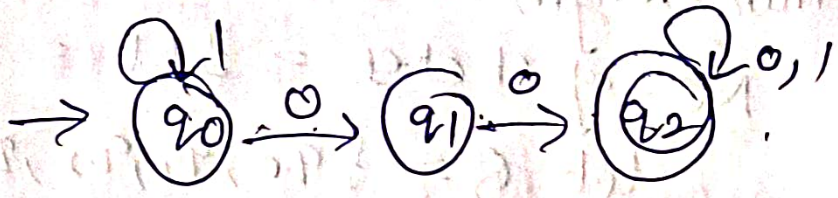
000



001

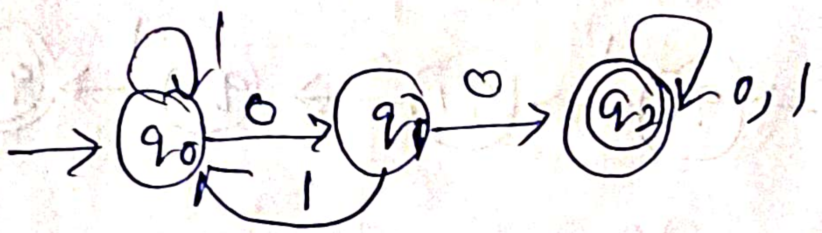


100



Satisfying 1001, 1100, 11001

101100



$q_0 \to (1) \to q_0 \to (0) - q_1 - (1) \to q_0 - (1) - q_0 - (0) - q_1$

$(0) \to q_2$

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

$\delta(q_0,0) = q_1 \quad \delta(q_0,1) = q_0$

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

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

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

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

15 String contains 101 as substring for DFA

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

$L = \{101, 0101, 1101, 11010, 1011, 01010, 1101, \ldots\}$

Min. Length $= 3$

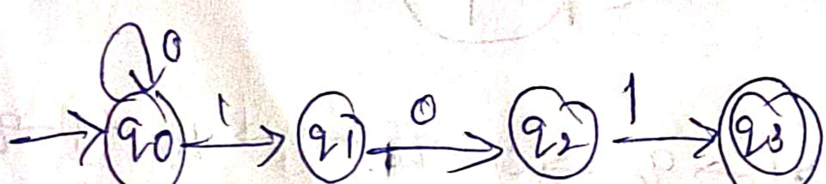
No. of states $= 3+1 = 4$

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

$q_0 = q_0$, $F = q_3$

101

0101



1101

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

$\delta(q_0,0) = q_1 \quad \delta(q_0,1) = q_0$

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

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

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

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

15 String contains 101 as substring ~~for~~ DFA

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

$L = \{101, 0101, 1101, 1010, 1011, 01010,$
$1101, - - - - \}$

Min. length $= 3$

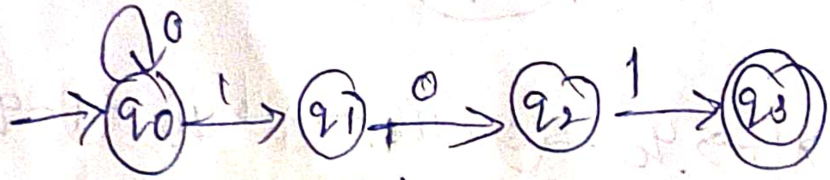
No. of states $= 3+1 = 4$

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

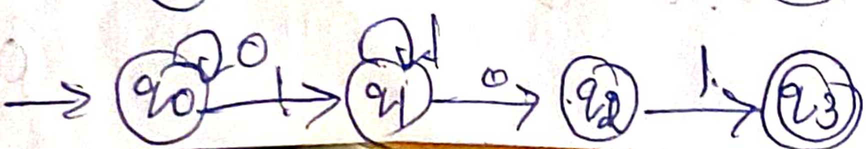
$q_0 = q_0$, $F = q_3$

101

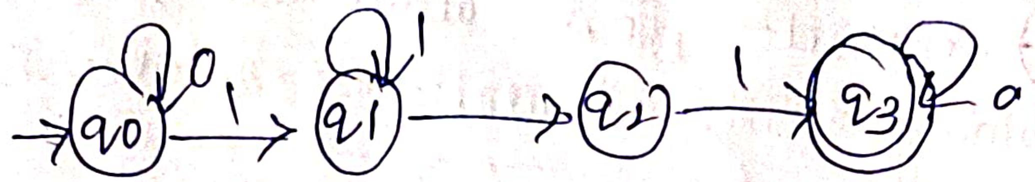
0101



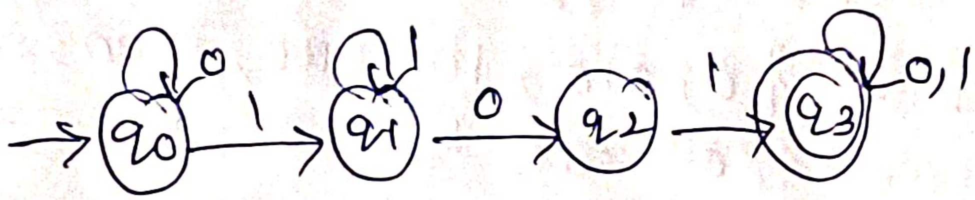
1101



1010



1011



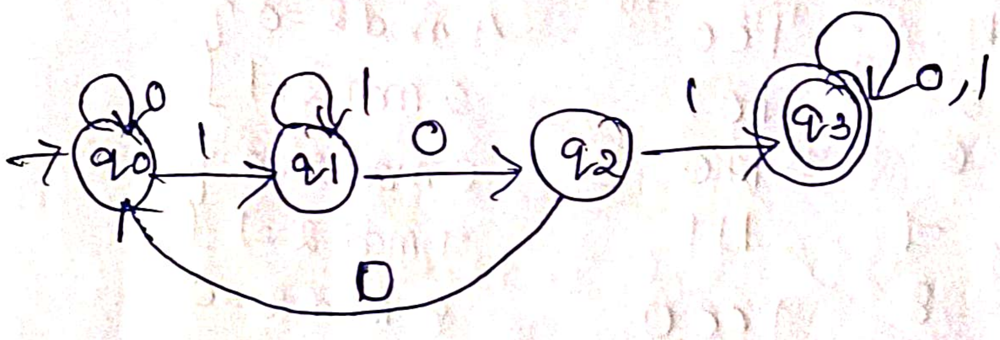
01010

$q_0 - 0 - q_0 - 1 - q_1 \rightarrow 0 - q_2 \rightarrow 1 \rightarrow q_3 - 0 - q_3$

100101

$q_0 \rightarrow 1 \rightarrow q_1 \rightarrow 0 - q_2 \rightarrow 0 \rightarrow q_0 \rightarrow 1 - q_1 - 0 - q_2$

$1 - q_3$



Dr. K. Bhargavi

(16) construct DFA which accepts all binary strings divisable by 2 over $\Sigma = \{0, 1\}$

$L = \{0, 010, 100, 110, 1000, ---\}$
$\quad\quad 0 \quad 2 \quad 4 \quad 6 \quad 8 \quad ---$

$L = \{ w/w \quad n \bmod 2 = 0 \}$.

Decimal no.	Binary no.	Remainder
0	000	$0 \% 2 = 0$
1	001	$1 \% 2 = 1$
2	010	2 mod 2 = 0
3	011	3 mod 2 = 1
4	100	4 mod 2 = 0
5	101	5 mod 2 = 1
6	110	6 mod 2 = 0
7	111	7 mod 2 = 1
8	1000	8 mod 2 = 0.

$0 \% 2 = 0$
$1 \% 2 = 1$.

$2 \% 2 = 0$
$3 \% 2 = 1$

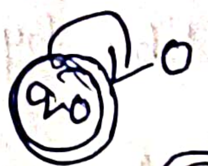
here sequence of 0 & 1

Let $q_0 \rightarrow 0$
$q_1 \rightarrow 1$

q_0 accepts the binary no. with remainder 0
q_1 " " " remainder 1.

Note: we can identify no. of states required by examining the remainders.

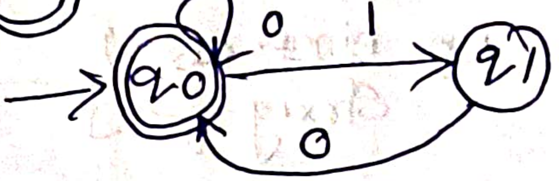
$L = \{0, 010, 100, 110, 1000 ---\}$



(reach to q0)

0-0

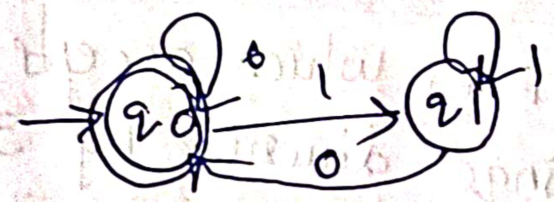
1-001



(After reading string reach q_1 state)

2-010 same the above one

3-011 - reminder 1 → reach to state q_1 after reading the state.



4-100 - remainder 0 - q0 state need to reach

5-101 - remainder 1 - q1 state "

6-110 - remainder 0 - q0 "

⇒ same above diagram satisfy.

Now consider the Language

$L = \{0, 010, 100, 110, 1000 ---\}$

after reading the strings from the language need to reach final state i.e q_0

$F = q_0$

$010 \Rightarrow q_0 \to 0 \to q_0 \to 1 - q_1 \to 0 - q_0$

$100 \Rightarrow q_0 \to 1 \to q_1 \to 0 \to q_0 \to 0 \to q_0$

$110 \Rightarrow q_0 \to 1 \to q_1 \to 1 \to q_1 \to 0 \to q_0$

All the strings are satisfying.

Take any string divisible by 2

i.e 1010 binary string of } — 10

Check for 1010

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

satisfying.

17Q) construct DFA, which accepts set of binary strings divisible by 3 over $\Sigma = \{0,1\}$

Soln: Identify the no. of states required.

$L = \{ w / w \bmod 3 = 0 \}$

$L = \{ 0, 3, 6, 9, 12, 15, --- \}$

Language in Binary digits as

$L = \{ 0, 011, 100, 1001, 1100, --- \}$

Decimal no.	Binary no.	Remainder
0	000	0 — q_0 $0 \% 3 = 0$
1	001	1 — q_1 $1 \% 3 = 1$
2	010	2 — q_2 $2 \% 3 = 2$
3	011	0 — q_0
4	100	1 — q_1
5	101	2 — q_2

6	110	0	q_0
7	111	1	q_1
8	1000	2	q_2
9	1001	0.	

All possible remainders are ⇒ 0, 1, 2

Let assume that 0, 1 & 2 are representing states q_0, q_1 & q_2

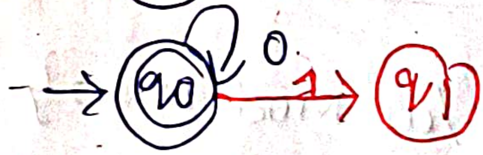
$0 \rightarrow q_0$
$1 \rightarrow q_1$
$2 \rightarrow q_2$

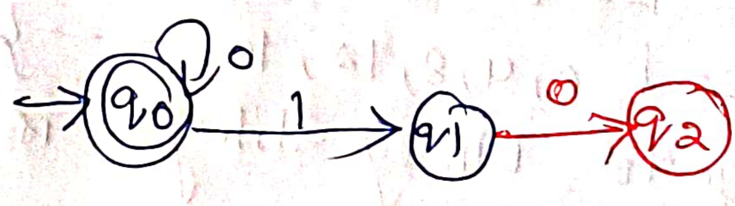
q_0 accepts the string binary no. with remainder 0
q_1 " " " " 1
q_2 " " " " 2

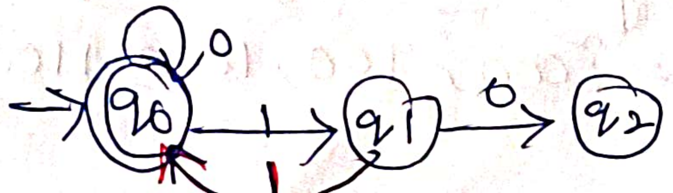
Remainder 0 no's are = 0, 3, 6 → q_0
Remainder 1 no's are = 1, 4, 7 → q_1
Remainder 2 no's are = 2, 5, 8 → q_2

Construct the DFA from the table.

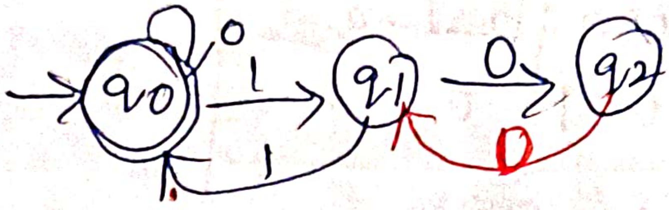
000 (q_0)  (Reaching q_0)

001 (q_1)  (Reaching q_1)

010 (q_2)  (Reaching q_2)

011 (q_0)  (Reaching q_0)

100 — Reach q_1 state



101 — Reach q_2 state.

Now check for

$L = \{0, 011, 110, 1001, 1100 ---\}$

Reading all the strings from the above Language, should reach to final state q_0..

Ex: 1100
q_0 q_1 q_0 q_0 (q_0) which is final state.

Q3) DFA for divisible by 4 over $\Sigma = \{0,1\}$

Sol :
$L = \{w / w \bmod 4 = 0\}$

$L = \{0, 4, 8, 12, 16, ---\}$

Language for binary strings is

$L = \{000, 100, 1000, 1100, ---\}$

Divisible no.	Binary no.	Remainder	
0	000	$0 \% 4 = 0$	q_0
1	001	$1 \% 4 = 1$	q_1
2	010	$2 \% 4 = 2$	q_2
3	011	$3 \% 4 = 3$	q_3
4	100	$4 \% 4 = 0$	q_0
5	101	$5 \% 4 = 1$	q_1
6	110	$6 \% 4 = 2$	q_2
7	111	$7 \% 4 = 3$	q_3
8	1000	$8 \% 4 = 0$	q_0

All possible remainders are 0, 1, 2 & 3.

Let assume that

0 → q_0 state

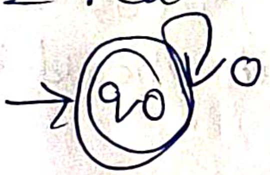
1 → q_1 state

2 → q_2 state

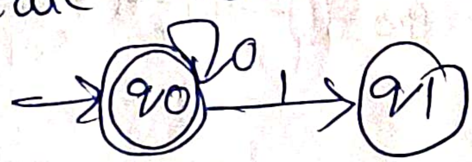
3 → q_3 state.

construct the DFA based on the above table.

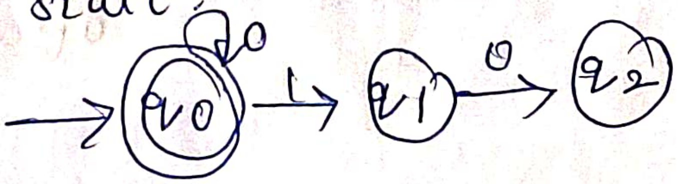
000 – reaches to q_0



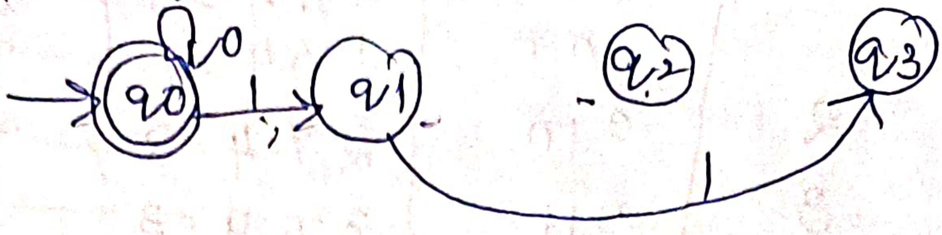
001 – q_1 state need to reach



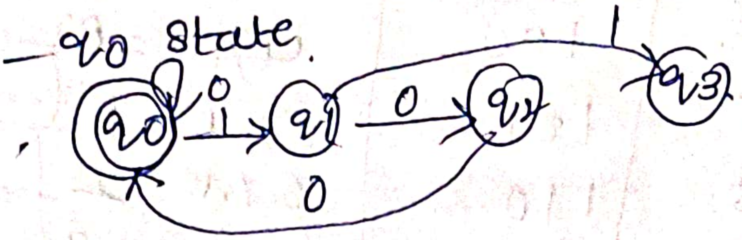
010 – q_2 state.



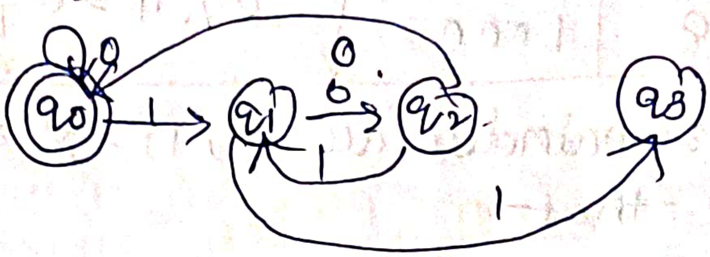
011 — q_3 state.



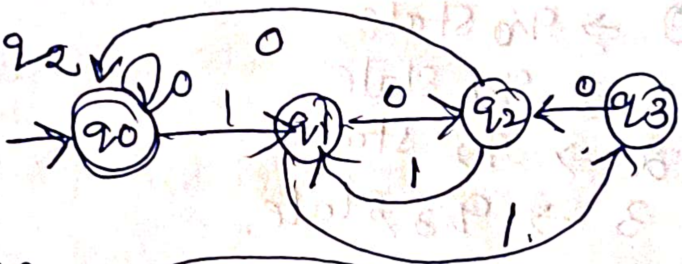
100 — q_0 state.



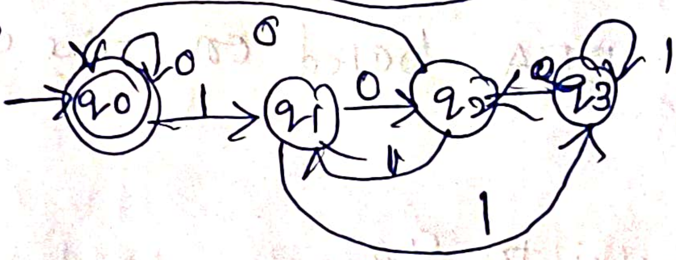
101 — q_1 state



110 — q_2



111 — q_3



Check for $L = \{000, 100, 1000, \text{——}\}$

After reading every string it should reaches to final state.

1100 $\Rightarrow q_0 - 1 - q_1 - 1 - q_3 - 0 - q_2 \to 0 - q_0$

Reached the final state after reading string

Divisible by 5 over $\Sigma = \{0,1\}$

Soln: $L = \{w / w \bmod 5 = 0\}$

$L = \{0, 5, 10, 15, ---\}$

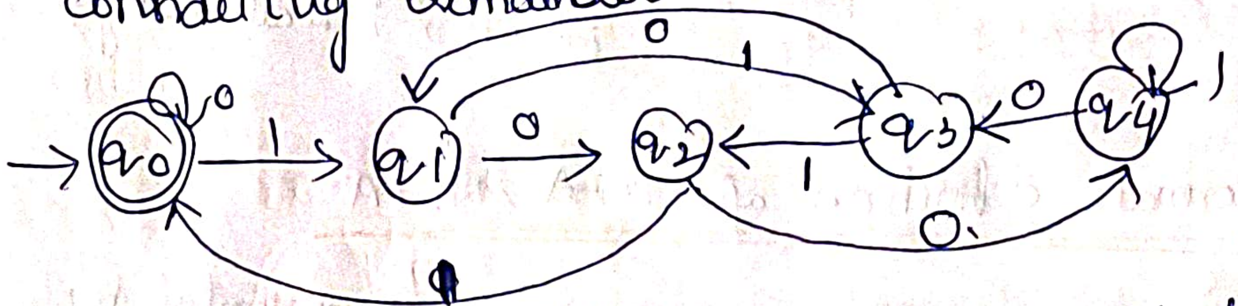
language for binary no.s :

$L = \{000, 101, 1010, 1111, ---\}$

Divisible no	Binary no.	Remainder	
0	000	0 Mod 5 = 0	q_0
1	001	1 mod 5 = 1	q_1
2	010	2 mod 5 = 2	q_2
3	011	3 mod 5 = 3	q_3
4	100	4 mod 5 = 4	q_4
5	101	5 mod 5 = 0	
6	110	6 mod 5 = 1	
7	111	7 mod 5 = 2	
8	1000	8 mod 5 = 3	
9	1001	9 mod 5 = 4	
10	1010	10 mod 5 = 0.	

states.

Draw the DFA for above binary no, by considering remainders for concern states.



Check for Language, after reading all the strings need to reach final state

Ex : 1010

$q_0 \rightarrow 1 - q_1 \rightarrow 0 - q_2 \rightarrow 1 - q_0 - 0 - q_0$

Reached to final state.

Divisible by 5 over $\Sigma = \{0,1\}$

Soln: $L = \{ w / w \bmod 5 = 0 \}$

$L = \{ 0, 5, 10, 15, \ldots \}$

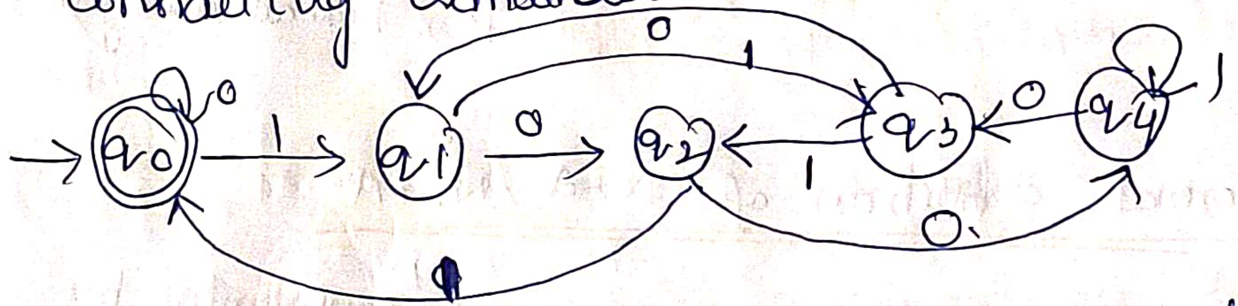
language for binary no's:

$L = \{ 000, 101, 1010, 1111, \ldots \}$

Divisible no	Binary no.	Remainder
0	000	0 mod 5 = 0 → q_0
1	001	1 mod 5 = 1 → q_1
2	010	2 mod 5 = 2 → q_2
3	011	3 mod 5 = 3 → q_3
4	100	4 mod 5 = 4 → q_4
5	101	5 mod 5 = 0
6	110	6 mod 5 = 1
7	111	7 mod 5 = 2
8	1000	8 mod 5 = 3
9	1001	9 mod 5 = 4
10	1010	10 mod 5 = 0.

states.

Draw the DFA for above binary no, by considering remainder for concern states.



Check for Language, after reading all the strings need to reach final state

Ex: 1010

$q_0 \rightarrow 1 - q_1 \rightarrow 0 - q_2 \rightarrow 1 - q_0 - 0 - q_0$

Reached to final state

NFA (Non-deterministic Finite automata)

NFA stands for non deterministic finite automata.
The Finite automata (FA) is called NFA, when there exist multiple / many paths from one state to another state for a specific input.

→ It is easy to construct NFA, than DFA for a regular language.

→ Every NFA is not DFA, but each NFA can be translated into DFA.

Applications of NFA :

Probabilistic algorithm.
genetic algorithms
Simulated annealing
NLP, Lexical analysis

Formal definition of NFA /NDFA

An NFA can be expressed/represented by a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where -

Q — Finite set of states

Σ — Is a finite set of ~~sym~~ symbols called .

δ — Transition Function where

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

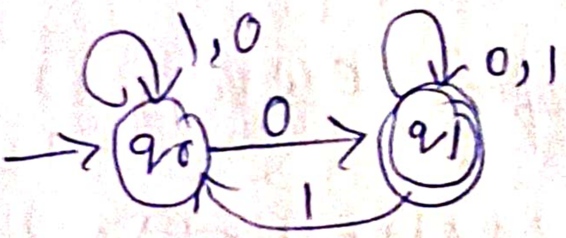
2^Q is power set of Q, transition can occur to any combination of Q states.

q_0 — Initial state $(q_0 \in Q)$

F — Set of final state $(F \subseteq Q)$.

An NFA is represented by digraphs called state diagram. (Similar to DFA).

Example :



having q_0 state path for 0 & 1

$$q_0 < \begin{matrix} 0 - q_1 \text{ and } q_0 \text{ (2 paths) (both states)} \\ 1 - q_0 \end{matrix}$$

$$q_1 < \begin{matrix} 0 - q_1 \\ 1 - q_0, q_1 \text{ (2 paths)} \end{matrix}$$

18) Construct NFA, start with 0 over $\Sigma = \{0,1\}$.

Solu:

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

Transition diagram:

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

$L = \{0, 01, 00, 010, 001, 011, ---\}$

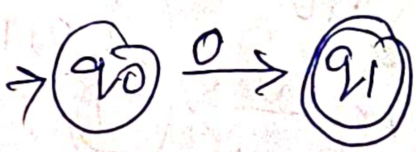
Min Length = 1.

~~Vet~~ No. of states = 2 (1+1)

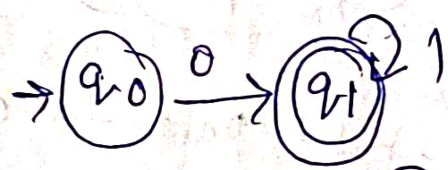
let $Q = \{q_0, q_1\}$

$q_0 = q_0$

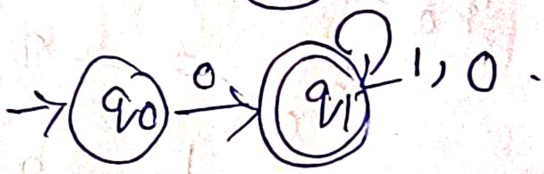
0



01



00



all other strings are satisfying

Transition Table

$F = q_1$

δ	0	1
$\rightarrow q_0$	q_1	ϕ
q_1	q_1	q_1

$\delta(q_0, 0) = q_1$

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

$\delta(q_1, 0) = q_1$

$\delta(q_1, 1) = q_1$

No path can represent by ϕ of -

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

(Q) Start with 01 over $\Sigma = \{0,1\}$

Soln :

$L = \{01, 010, 011, 0101, 0110, 0111, \ldots\}$

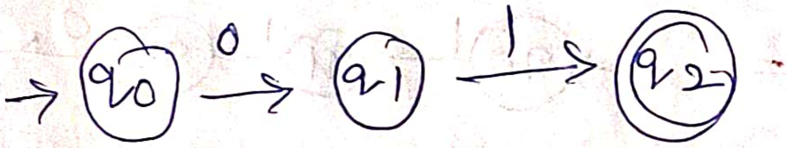
Min. Length = 2

No. of states = 2+1 = 3

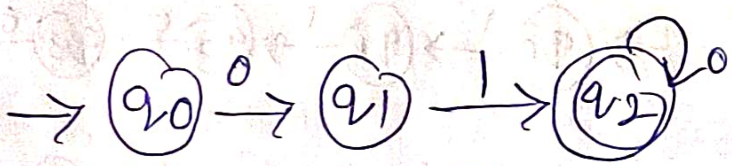
let $Q = \{q_0, q_1, q_2\}$

let $q_0 = q_0$

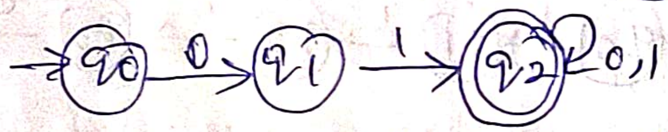
0



010



011



End we can consider any no. of 0's & 1's.

0110, 0111 are satisfying.

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

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

δ	0	1
$\rightarrow q_0$	q_1	ϕ
q_1	ϕ	q_2
q_2	q_2	q_2

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

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

(30) starts with 1100 over $\Sigma = \{0,1\}$

Soln;

$L = \{1100, 11000, 11001, 110011, ---\}$

Min length = 4

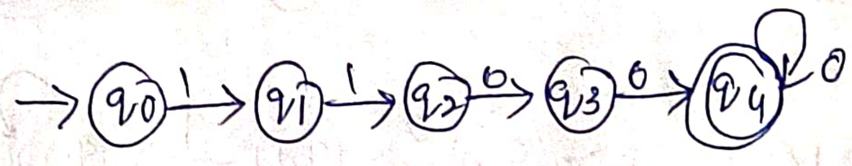
No. of states = 4+1 = 5

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

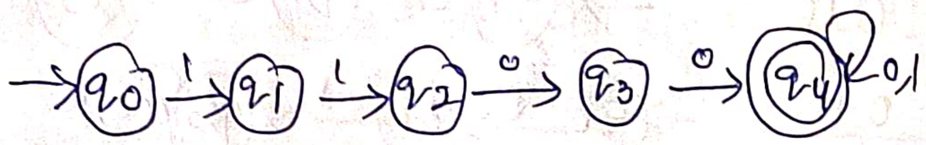
$q_0 = q_0$, $F = q_4$

1100 =

11000 =



110001 =



Accepts all other strings from language.

δ	0	1
$\rightarrow q_0$	$\emptyset$	q_1
q_1	$\emptyset$	q_2
q_2	q_3	$\emptyset$
q_3	q_4	$\emptyset$
q_4	q_4	q_4

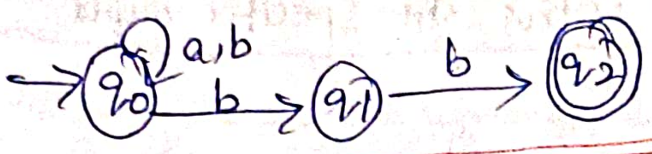
$M = (Q, \Sigma, \delta, q_0, F)$

$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0,1\}, \delta, q_0, q_4)$.

DFA — only one path for specific input
NFA : Many paths for specific input

DFA	NFA
1. It stands for deterministic finite automata	It stands for non deterministic finite automata
2. It is not easy to construct	It is easy to construct
3. Transition function $\delta : Q \times \Sigma \to Q$	Transition function $\delta : Q \times \Sigma \to 2^Q$.
4. No transition for ϵ	ϵ-transitions are allowed.
5. For each state, transition on all possible symbols should be defined.	Not require
6. Dead state may be required	Dead state may not be required.
7. It require more space compare to NFA.	It require less space compared to DFA.
8. DFA allows only one move for single i/p alphabet	There can be choice (more than one move) for single input alphabet.
9. All DFA's are NFA	Not all NFA are DFA

1Q Convert the following NFA to DFA.



Procedure :

Step 1: Construct Transition table / diagram for the given question.

Step 2 : Define given machine M.

Step 3 : Construct DFA table as

→ consider only the initial state with given input symbol.

→ construct the transition for the initial state

→ if any new states are arrived, then consider new state & construct transitions.

Step 4 : Repeat until no more states available.

Construct DFA & definition of M.

Step 5 : Identify the final state for DFA.

check the final state of NFA, then where it appears in TT of DFA will be the final state for DFA

Step 6 : Check the string acceptance for automata.

Solution

Step 1 : Transition table for given NFA

δ	a	b
$\rightarrow q_0$	q_0	$\{q_0, q_1\}$
q_1	—	q_2
q_2	—	—

Step 2 : Translate the NFA table into DFA transition table.

δ	a	b
$\rightarrow q_0$	q_0	$\{q_0, q_1\}$
$\{q_0, q_1\}$	q_0	$\{q_0, q_1, q_2\}$
$* \{q_0, q_1, q_2\}$	q_0	$\{q_0, q_1, q_2\}$

State New from the initial state is $\{q_0, q_1\}$, consider as the state as next one & find transition for $\{q_0, q_1\}$ on a & b

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

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

$\Rightarrow$ again new transition $\{q_0, q_1, q_2\}$, so consider as new state

Transitions for $\{q_0, q_1, q_2\}$ on a & b

$$\delta(\{q_0, q_1, q_2\}, a) = \delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a)$$
$$= q_0 \cup \phi \cup \phi$$
$$\text{or } q_0 \cup - \cup -$$
$$= q_0.$$

$$\delta(\{q_0, q_1, q_2\}, b) = \delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b)$$
$$= \{q_0, q_1\} \cup \{q_2\} \cup \phi$$
$$= \{q_0, q_1, q_2\}$$

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

Step 3

Final state (F) for DFA

NFA final state $= q_2$

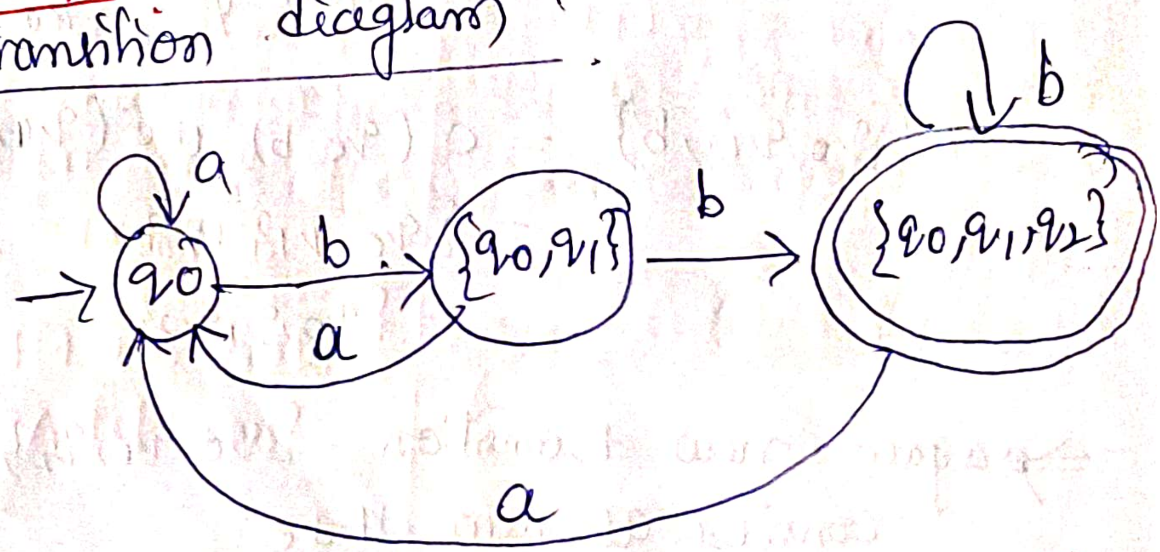
$$Q = (q_0, \{q_0, q_1\}, \{q_0, q_1, q_2\})$$

$F = q_2$, which is existing in $\{q_0, q_1, q_2\}$

$\therefore$ Final state for DFA $= \{q_0, q_1, q_2\}$

Step 4

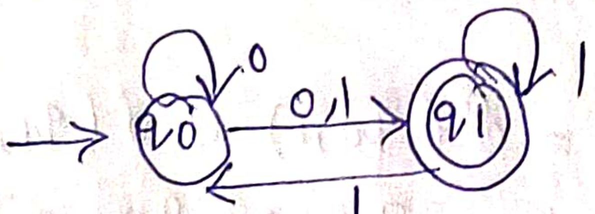
Transition diagram:



Each state have the path for a, b input symbols.

It is complete DFA.

2Q: Convert NFA to DFA of



Step1 : Transition Table for NFA

δ	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	q_1
q_1	—	$\{q_0, q_1\}$

Step 2 : Transition table for DFA, consider the Initial state transition from NFA

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

New state arrived $\{q_0, q_1\}$, then find the transition for $\{q_0, q_1\}$ on 0 & 1

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

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

q_1 is new state, consider as new state find the transitions.

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

Step 3

Final state of NFA (F) = q_1

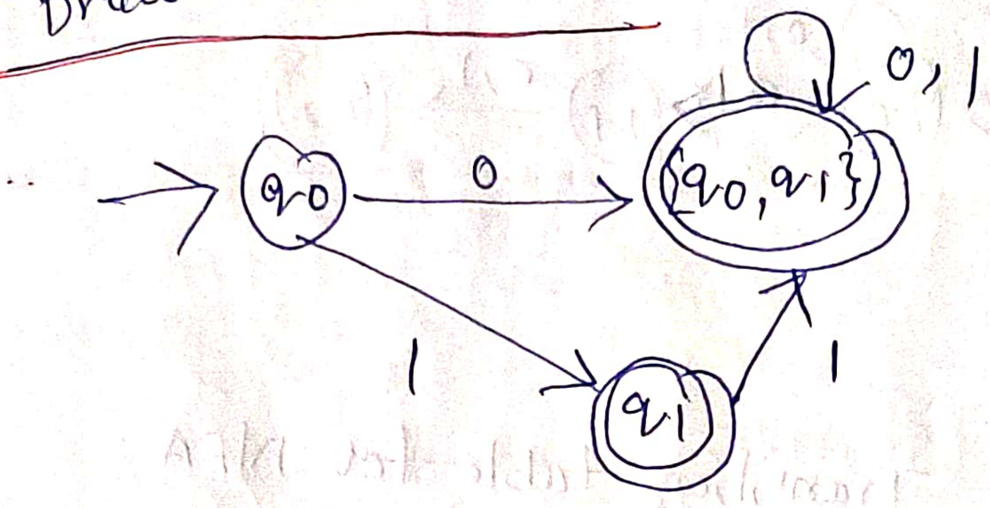
F of DFA is where q_1 exists in the states, all become

i.e Two states have q_1

i.e F = q_1 and $\{q_0, q_1\}$

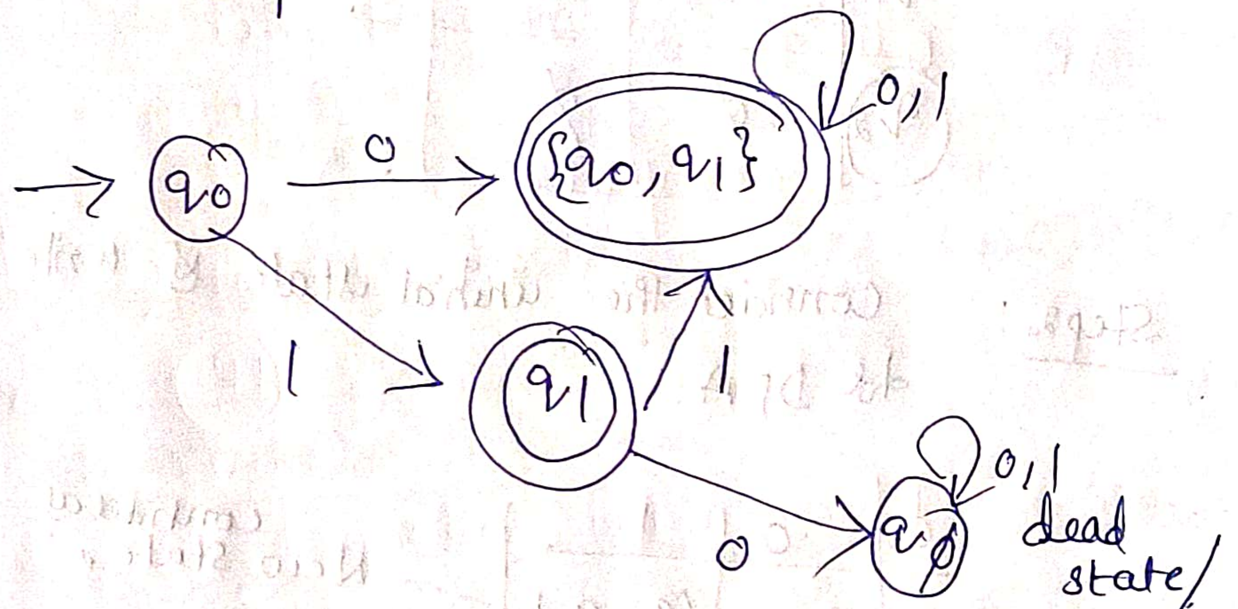
$q_0 = q_0$, $\Sigma = \{0, 1\}$

Step4
Draw the DFA :

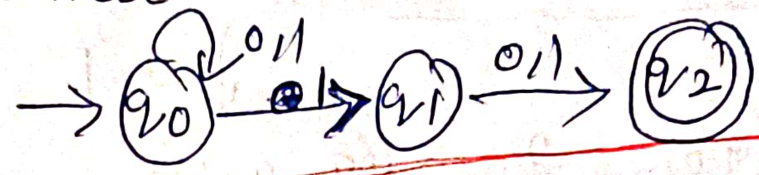


q_1 does not have the path for 0 input.

$\therefore$ consider the dead state q_ϕ for 0 input



38 Convert NFA to DFA for



Soln

Step 1: Transition table for NFA

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

Step 2: Consider the initial state & write TT for DFA.

δ	0	1
$\rightarrow q_0$	q_0	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_2\}$	q_0	$\{q_0, q_1\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
[illegible]	[illegible]	[illegible]

consider as New State.

$\delta((q_0, q_1), 0) = \delta(q_0, 0) \cup \delta(q_1, 0)$
$= q_0 \cup q_2$

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

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

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

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

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

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

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

$\delta((q_0, q_1, q_2), 0) = \delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0)$
$= q_0 \cup q_2 \cup \phi$

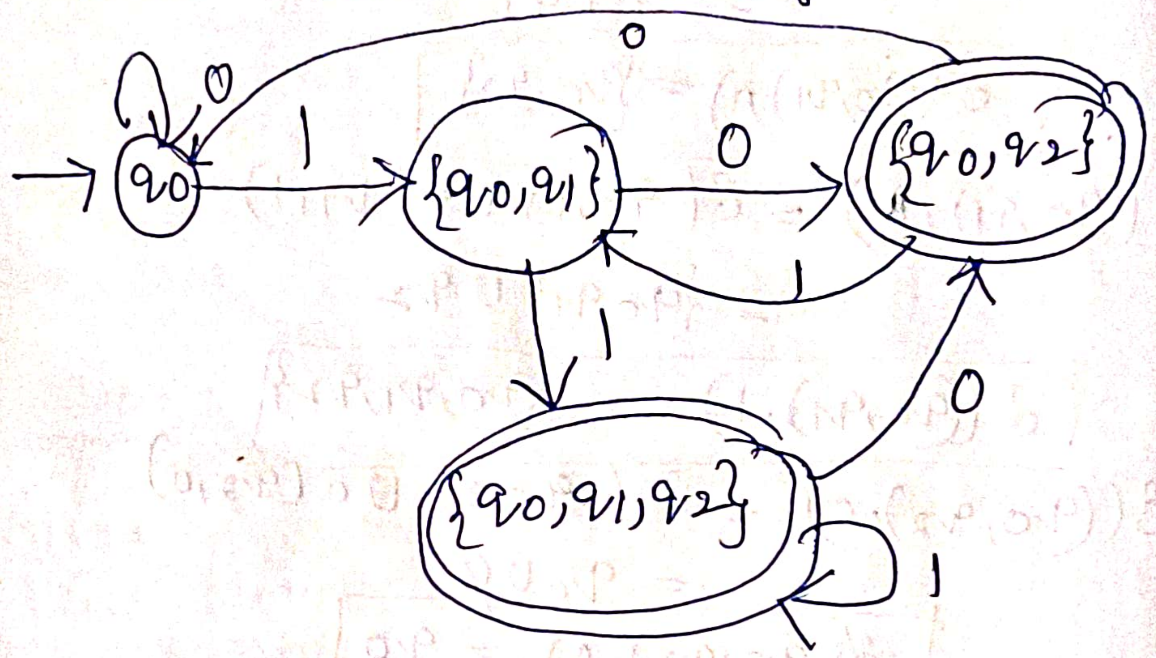
$\boxed{\delta((q_0, q_1, q_2), 0) = \{q_0 q_2\}}$

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

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

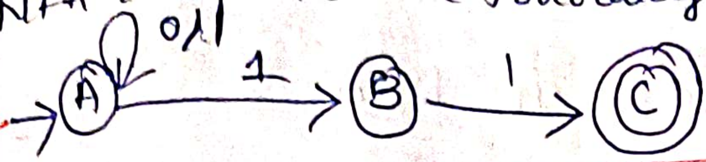
NFA final state is q_2
→ q_2 consisting in $\{q_0, q_2\}$ & $\{q_0, q_1, q_2\}$, so these become the final states.

Draw the transition diagram for DFA

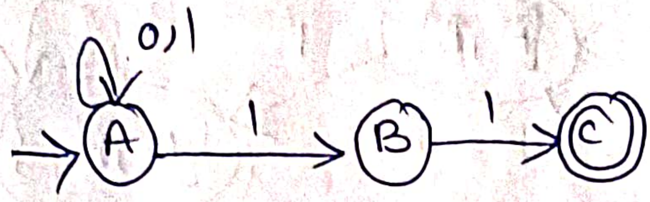


All the states have the path for 0 & 1.

∴ complete DFA.

Q4 

solution

step1 : 

TT for above

δ	0	1
$\rightarrow$ A	A	{A,B}
B	$\emptyset$	C
C	$\emptyset$	$\emptyset$

step2 : TT for DFA

δ	0	1
$\rightarrow$ A	A	{A,B}
{A,B}	A	{A,B,C}
* {A,B,C}	A	{A,B,C}

$$\delta((A,B),0) = \delta(A,0) \cup \delta(B,0) = A \cup \emptyset = A$$

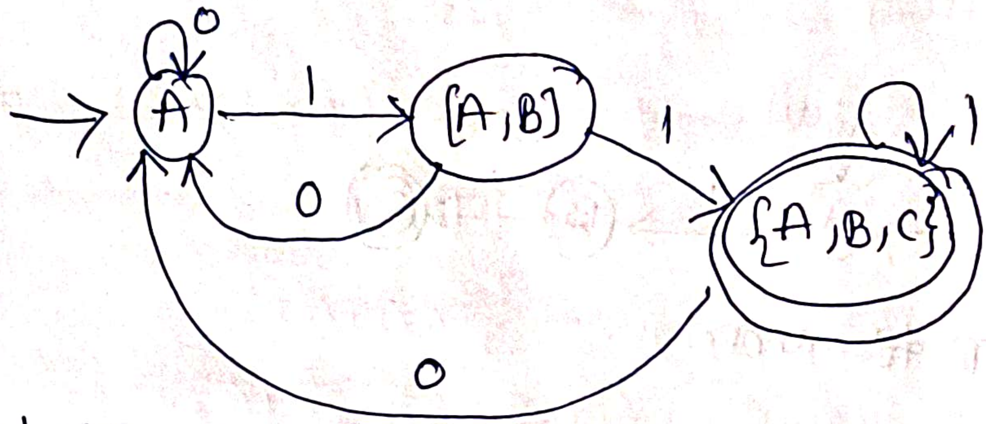
$$\delta((A,B),1) = \delta(A,1) \cup \delta(B,1) = \{A,B\} \cup C = \{A,B,C\}$$

$$\delta(\{A,B,C\},0) = \delta(A,0) \cup \delta(B,0) \cup \delta(C,0) = A \cup \emptyset \cup \emptyset = A$$

$$\delta((A,B,C),1) = \delta(A,1) \cup \delta(B,1) \cup \delta(C,1) = \{A,B\} \cup C \cup \emptyset = \{A,B,C\}$$

Step 3

Transition diagram for DFA



Step 4

Final state :

Final state for NFA is C

In DFA , C is having {A,B,C}

$\therefore$ {A,B,C} is Final State = F

$M = ((A, \{A,B\}, \{A,B,C\}), \{0,1\}, \delta, A, \{A,B,C\})$

5Q. convert the following NFA to DFA

~~State = {q0, q1}~~

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

Solution is Question No 2 refer

NFA with ∈ (∈-NFA)

An epsilon - non deterministic finite automation (∈-NFA) is a type of finite automation, that allows transitions without any input symbols.

i.e empty string, ∈.

Epsilon (∈) Transition is also known as an empty move or empty transition.

Formal definition of ∈-NFA :-

∈-NFA is represented with 5 tuple $(Q, \Sigma, \delta, q_0, F)$ where

Q – is a finite set of states

Σ – Set of symbols called alphabet

$\delta : Q \times (\Sigma \cup \epsilon) \rightarrow 2^Q$ as transition function.

q_0 – Initial state $(q_0 \in Q)$

F – Final state $(F \subseteq Q)$

ϵ- closure :

ϵ-closure is a set of states, Which can be reached from one state with ϵ moves / transitions including itself.

Examples: Find the ϵ-closure of the following.

(1Q).

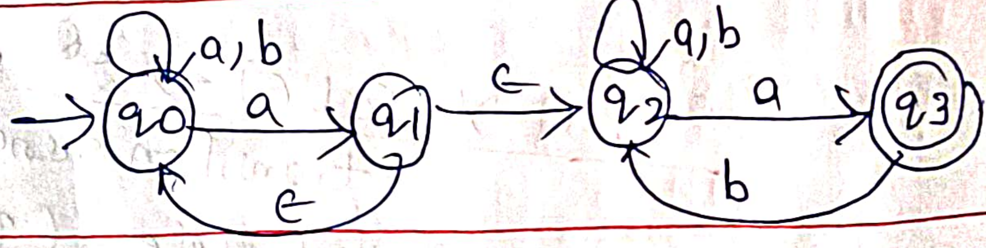
Soln. ϵ-closure$(q_0) = \{q_0, q_1, q_2\}$

Note: q_0 the state itself is states reached from q_0 ~~on~~ ϵ moves.

ϵ-closure$(q_1) = \{q_1, q_2\}$

ϵ-closure$(q_2) = \{q_2\}$.

(2Q)



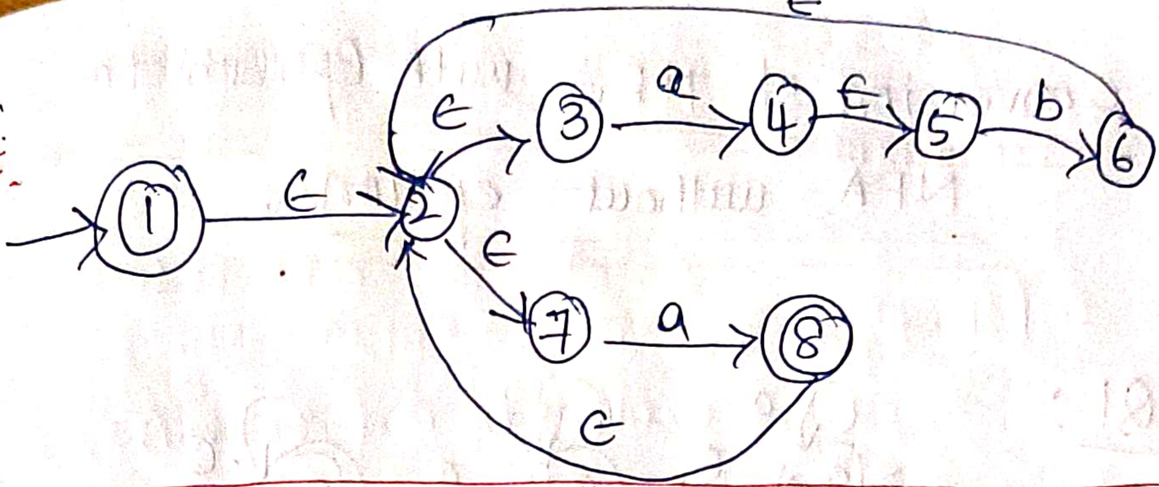
Soln: ϵ-closure$(q_0) = \{q_0\}$

ϵ-closure$(q_1) = \{q_0, q_2, q_1\}$

ϵ-closure$(q_2) = \{q_2\}$

ϵ-closure$(q_3) = \{q_3\}$

39:



Soln:

ε-closure(1) = {1, 2, 3, 7}

ε-closure(2) = {2, 7, 3}

ε-closure(3) = {3}

ε-closure(4) = {4, 5}

ε-closure(5) = {5}

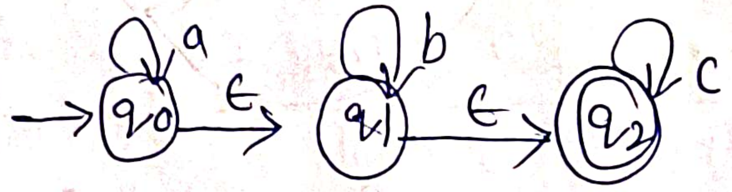
ε-closure(6) = {6, 2, 3, 7}

ε-closure(7) = {7}

ε-closure(8) = {8, 2, 3, 7}

Conversion of NFA with Epsilon to NFA without epsilon.

Q1:



Procedure:

Step 1: write the transition table (TT) for ϵ-NFA. (δ)

Step 2: Find the ϵ-closure of all states.

Step 3: Transition table for NFA

Note: Extended transition function i.e δ' (ϵ-closure on δ moves)

Step 4: Identify the Final state for NFA.

Check ϵ-closures of all states, if any of the state consisting final state of ϵ-NFA, becomes the Final state.

Step 5: Transition diagram for NFA.

Solution :

Step1 : TT for ϵ-NFA

δ	a	b	c	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.

Step 2 : Calculation of ϵ-closure.

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

Step 3 : Transition table for NFA (δ')

δ'	a	b	c
$\rightarrow q_0$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	q_2.
q_1	ϕ	$\{q_1, q_2\}$	q_2
q_2	ϕ	ϕ	q_2

δ' $\Rightarrow$ ϵ-closure on δ moves.
transitions

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

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

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

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

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

$\boxed{\delta'(q_0, a) = \{q_0, q_1, q_2\}}$

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

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

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

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

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

$\boxed{\delta'(q_0, b) = \{q_1, q_2\}}$

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

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

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

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

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

$\boxed{\delta'(q_0, c) = \{q_2\}}$

$\delta'(q_1, a) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_1); a)$

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

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

$\delta'(q_1, a) = \epsilon\text{-closure}(\phi \cup \phi) = \phi$

$\delta'(q_1, b) = \epsilon\text{-closure}(\delta[\epsilon\text{-closure}(q_1), b)$

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

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

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

$\boxed{\delta'(q_1, b) = \epsilon\text{-closure}(q_1) = \{q_1, q_2\}}$

$\delta'(q_1, c) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_1), c)$

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

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

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

$\boxed{\delta'(q_1, c) = \epsilon\text{-closure}(q_2) = \{q_2\}}$

Similarly

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

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

$\boxed{\delta'(q_2, a) = \epsilon\text{-closure}(\phi) = \phi}$

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

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

$\boxed{\delta'(q_2, b) = \epsilon\text{-closure}(\phi) = \phi}$

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

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

$\boxed{\delta'(q_2, c) = \epsilon\text{-closure}(q_2) = q_2}$

Step 4 : Final state for NFA

In ε-NFA Final State $= q_2$

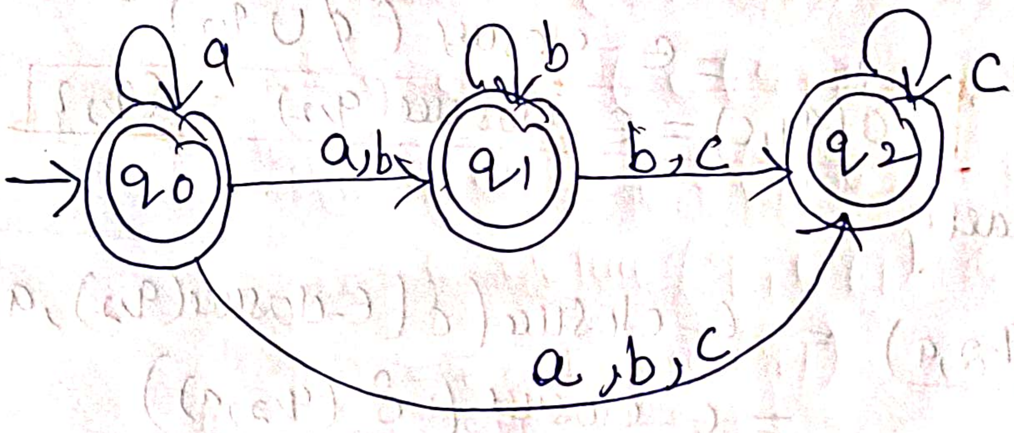
↳ q_0 ε-closure contain final state q_2

↳ q_2 " " "

↳ q_1 " " "

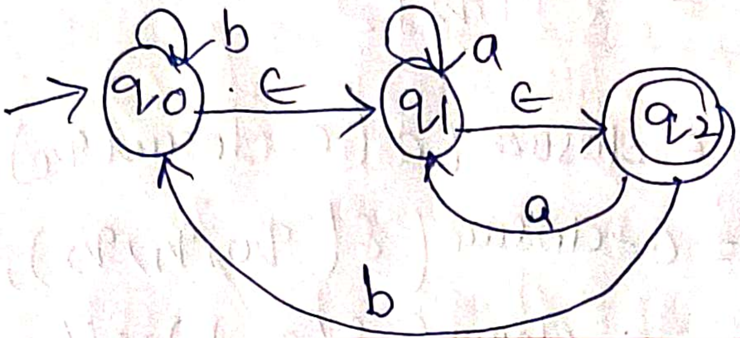
∴ All states q_0, q_1 & q_2 are final states.

Step 5 Transition diagram for NFA



Dr. K. Bhargavi

Q2: Conversion of NFA with ϵ-moves to NFA without ϵ-moves for the following



Soln: ① TT for given ϵ-NFA

δ	a	b	ϵ
q_0	ϕ	q_0	q_1
q_1	q_1	ϕ	q_2
q_2	q_1	q_0	ϕ

Step 2:

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

Step 3: Find Extended transitions (δ')

$$\boxed{\cancel{\delta'(q_0, a) = \epsilon\text{-closure}}\ \delta'(\cancel{\epsilon\text{-closure}(q_0), a})}$$

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

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

$\boxed{\delta'(q_0,a) = \{q_1, q_2\}}$

$\delta'(q_0,b) = \epsilon\text{-closure}[\delta(\epsilon\text{-closure}(q_0), b)]$

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

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

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

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

$\boxed{\delta'(q_0,b) = \{q_0, q_1, q_2\}}$

$\delta'(q_1,a) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_1), a)$

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

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

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

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

$\boxed{\delta'(q_1,a) = \{q_1, q_2\}}$

$\delta'(q_1,b) = \epsilon\text{-closure}(\delta(\epsilon\text{closure}(q_1), b)$

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

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

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

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

$\boxed{\delta'(q_1,b) = \{q_0, q_1, q_2\}}$

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

$$\boxed{\delta'(q_2, a) = \{q_1, q_2\}}$$

$\delta'(q_2, b) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_2), b)$
$= \epsilon\text{-closure}(\delta(q_2, b))$
$= \epsilon\text{-closure}(q_0)$

$$\boxed{\delta'(q_2, b) = \{q_0, q_1, q_2\}}$$

Draw the T. table & Find the final state.

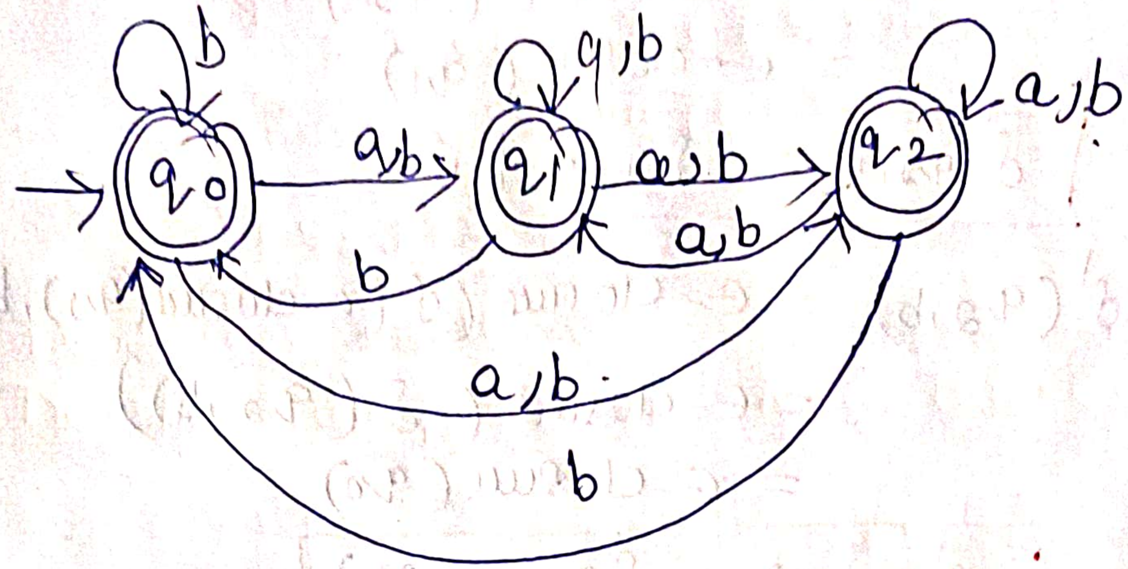
q_2 consisting ϵ-closure of q_0, q_1 & q_2

$\therefore$ q_0, q_1 & q_2 become final state

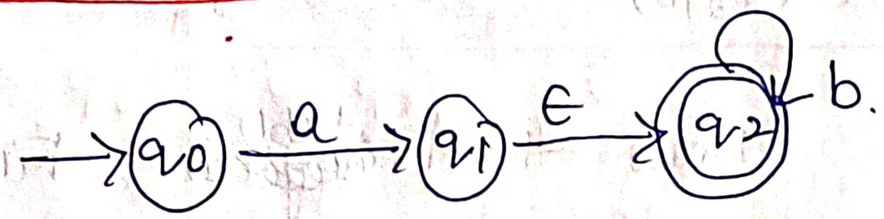
δ'	a	b
$\rightarrow q_0$	$\{q_1, q_2\}$	$\{q_0, q_1, q_2\}$
q_1	$\{q_1, q_2\}$	$\{q_0, q_1, q_2\}$
q_2	$\{q_1, q_2\}$	$\{q_0, q_1, q_2\}$

Final states are q_0, q_1 and q_2.

Transition diagram :



3Q)



Convert NFA with ∈ to without ∈ NFA

Sol: Transition table for given ∈-NFA

δ	a	b	ϵ
$\rightarrow q_0$	q_1	ϕ	ϕ.
q_1	ϕ	ϕ	q_2
q_2	ϕ	q_2	ϕ.

Find ∈-closure

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

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

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

Find extended transitions (δ') for all the states for i/p symbols.

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

$\boxed{\delta'(q_0 A) = \{q_1, q_2\}}$

$\delta'(q_0, b) = \epsilon\text{-closure}(\delta[\epsilon\text{-closure}(q_0), b))$
$= \epsilon\text{-closure}(\delta(q_0), b)$
$= \epsilon\text{-closure}(\phi)$

$\boxed{\delta'(q_0, b) = \phi}$

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

$\boxed{\delta'(q_1, a) = \phi}$

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

$\boxed{\delta'(q_1, b) = q_2}$

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

$\boxed{\delta'(q_2, a) = \phi}$

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

$$\boxed{\delta'(q_2, b) = q_2}$$

Transition table &
Transition diagram for NFA with δ'

δ'	a	b
$\rightarrow q_0$	$\{q_1, q_2\}$	$\emptyset$
q_1	$\emptyset$	q_2
q_2	$\emptyset$	q_2

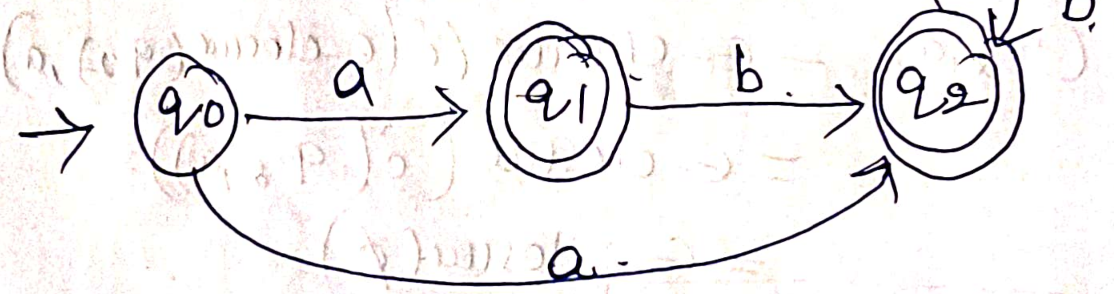
Final states for NFA:

ϵ-NFA final state is q_2

ϵ-closure of q_1 & q_2 have q_2

$\therefore$ q_1 & q_2 are final states

Transition diagram



Unit-I
Chapter-II – Regular Expressions (RE)

- Introduction to Regular Language.
- Regular expressions.
- Algebric laws for RE.
- conversion of FA to RE
- conversion of RE to FA.
- Pumping Lemma for RL.

Regular Expressions (RE):

The language accepted by Finite automata can be described by simple expressions called Regular expressions.

RE's are most effective way of represent the language.

RE's are used to generate patterns or strings

If the strings of an language satisfied by RE's are called Regular Language (RL).

Applications: Data validation
compiler

Regular Language is a language that can be defined by
→ DFA
→ NFA
→ NFA with ϵ
Expressions

Identity Rules :

Rule 1 : $\phi + r = r$

Rule 2 : $\phi \cdot r = r \cdot \phi = \phi$

Rule 3 : $\epsilon \cdot r = r \cdot \epsilon = r$

Rule 4 : $r + r = r$

Rule 5 : $r^* \cdot r^* = r^*$

Rule 6 : $r^* \cdot r = r \cdot r^*$

Rule 7 : $(r^*)^* = r^*$

Rule 8 : $(\epsilon + r^* r) = (\epsilon + r \cdot r^*) = r^*$

Rule 9 : $(PQ)^* = P(QP)^*$

Rule 10 : $\epsilon^* = \epsilon$ and $\phi^* = \epsilon$

Rule 11 : $R + R = R$

Rule 12 : $(P+Q)R = PR + QR$ and
$R(P+Q) = RP + RQ$.

Q1 : Write the regular expression for the language consisting of all a's over the set $\Sigma = \{a\}$.

Solu : $L = \{\epsilon, a, aa, aaa, \ldots\}$

Represents the kleene closure of a

$$\boxed{RE = a^*}$$

Q2) write the regular expression for the language consisting of all the strings of a, except the null over set $\Sigma = \{a\}$.

Sol: $L = \Sigma^* - \Sigma^0$ or $\Sigma^* - \epsilon$ i.e Kleen plus

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

Kleene plus of a.

$$\boxed{RE = a^+}$$

Q3: Accepting all the strings containing any no. of a's and any no. of b's

Sol: $L = \{\epsilon, a, b, ab, aa, ba, bb, aba, ---\}$.

All a's, b's including ϵ

$$\therefore \boxed{RE = (a+b)^*}$$

Languages

- Finite Language
  - ↓
  - For finite consider the union.
- Infinite Language.

Regular Expression is defined as.

(a) ϵ is RE corresponding to the Lang $L = \{\epsilon\}$

Null string with 'o' Length

(b) ϕ is RE corresponding to the Lang $L = \{\ \}$

i.e empty string

(c) x is RE corresponding to the $L = \{x\}$.

(Language with only one string)

If x is RE over Language $L(x)$

y is RE over Language $L(y)$

then

(i) $x+y$ is RE corresponding to $L(x) \cup L(y)$

where $\boxed{L(x) \cup L(y) = L(x \cup y)}$.

(ii) xy is RE corresponding to $L(x).L(y)$

where $\boxed{L(x.y) = L(x).L(y)}$

(iii) R^* is RE corresponding to $\boxed{L(R^*) = (L(R))^*}$

No string $= \{\ \} = \phi$.

Length of string 0 = empty string

$\Rightarrow L = \{\epsilon\}$.

(1) write a RE for length of 1 over $\Sigma = \{a, b\}$

$L = \{a, b\}$. $\Rightarrow$ a & b both

$\therefore a+b$.

$$\boxed{RE = (a+b)}$$

(2) write RE for string with length 2 exactly over $\Sigma = \{a, b\}$

Sol: $L = \{aa, ab, bb, ba\}$ → Finite Language.

$RE = (aa + ab + bb + ba)$

$= a(a+b) + b(a+b)$

$$\boxed{RE = (a+b)(a+b)}$$

3Q) String length with 3.

Sol: Length of 2 exactly $RE = (a+b)(a+b)$.

$2+1 \Rightarrow 3$.

Length exactly $3 = RE = (a+b)(a+b)(a+b)$

4) string with only one a.

Soln:- String can have only one a, remaining can be any no. of b's.

Any no. of b's $\Rightarrow \epsilon, b, bb, bbb$ ---

$RE = b^*$

$RE = b^* a b^*$

5) At least 2 a's

$L = \{aa, ab, ba, bb, abb, aba, aab, \text{---}\}$

Soln: Minimum 2 a's & also can have any no. of a's & any no. of b's

a's & b's with $n_a(w) = 2$ (i.e no. of a's = 2).

For Any no. of a's & any no. of b's

$= RE = (a+b)^*$.

No. of a's = 2.

$\therefore$ RE = Any no. of a's & b's a Any no. of a's & b's a Any no. of a's & b's

$RE = (a+b)^* a (a+b)^* a (a+b)^*$

6. Exactly 2 a's

Soln: $L = \{aa, baa, bbaa, aab, \text{---}\}$

$\boxed{RE = b^* a b^* a b^*}$ here Exactly $n(a)(w) = 2$

7 At most 2 a's

Sol:

$W \leq 2$. (Length 0, 1 & 2)

i.e (0 no. of a's + 1 no. of a's + 2 no. of a's)

0 no. of a's = b^*

1 no. of a's = $b^* a b^*$

2 no. of a's = $b^* a b^* a b^*$

$RE = b^* + b^* a b^* + b^* a b^* a b^*$

Note : Maximum 2 a's (0, 1 & 2)

(8). Write a RE for string that contain even no. of a's.

Sol:

$L = \{\epsilon, aa, aab, babab, b, bb, bbb, ---\}$

$n_a(W) \text{ Mod } 2 = 0$ $\{2, 4, 6, 8 ---\}$

$L = \{0, 2, 4, 6, ---\}$

$0 \rightarrow \epsilon$

exactly 2 a's $\Rightarrow b^* a b^* a b^*$ — getting null ϵ, b, bb.

$RE = b^* + (b^* a b^* a b^*)^*$

b^*: ϵ, b, bb, bbb

$(b^* a b^* a b^*)^*$: ϵ, aa, aab, babab

Q) Find RE for the string consist odd no. of a's

Sol^n : $L = n_a(w) \bmod 2 = 1$

no. of a's $= 1 + 0 = 1$ if $* = 0$
no. of a's $= 1 + 2 = 3$ if $* = 1$
no. of a's $= 1 + 4 = 5$ if $* = 2$
no. of a's $= 1 + 6 = 7$ if $* = 3$

$$RE = b^* a b^* (b^* a b^* a b^*)^*$$

or

$$RE = (b^* a b^* a b^*) b^* a b^*$$

Q. Find RE for even length string over $\Sigma = \{a, b\}$

Soln: $L = \{0, 2, 4, 6, ---\}$, $n \bmod 2 = 0$

$L = \{\epsilon, aa, ab, ba, bb, bbaa, abab, ---\}$

$0 \Rightarrow \epsilon$.
$2 \Rightarrow aa, ab, ba, bb = (a+b)(a+b)$
$(4) \Rightarrow (a+b)(a+b)(a+b)(a+b)$.

i.e $(a+b)^{2n}, n \geq 0$

$$\boxed{\therefore RE = ((a+b)(a+b))^*}$$

Note: - Repeating set of string with length 2 over and over.

RE: String start with ba

Sol: RE = ba (Any no. of a's & Any no. of b's)

$$RE = ba(a+b)^*$$

RE: Start with a & end with b

Sol: RE = a $\underline{\text{any no. of a's, b's}}$ b

$$RE = a\,\underline{(a+b)^*}\,b$$

$$\therefore RE = a(a+b)^*b.$$

Write RE, Begin or end with either 00 or 11

Sol: Begin with 00 or 11 = L_1

End with 00 or 11 = L_2.

$L = L_1 \cup L_2$. or $L = L_1 + L_2$.

L_1 = (00 + 11) any no. of a's & any no. of b's

L_2 = Any no. of a's & any no. of b's (00+11)

$$L_1 = (00+11)(0+1)^*$$

$$L_2 = (0+1)^*(00+11)$$

$$L = L_1 + L_2$$

$$\boxed{L = (00+11)(0+1)^* + (0+1)^*(00+11)}$$

write RE, at least one a, one b followed by one c.

$$RE = a^+ b^+ c^+$$

RE, 3rd character from right end is a.

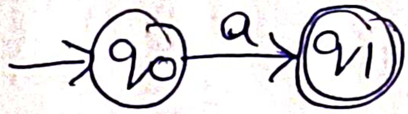
Soln: Any a's any b's $\underline{a}$ $\underline{a \text{ or } b}$ $\underline{a \text{ or } b}$

$$RE = (a+b)^* a (a+b)(a+b)$$

convert RE to FA

Basic RE's.

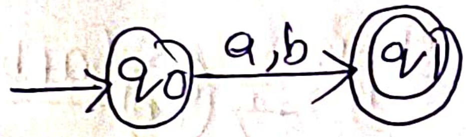
RE = a



RE = ab

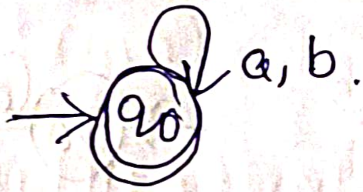
RE = a+b

or



RE = a*

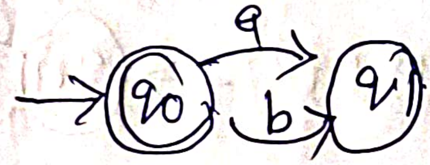
RE = (a+b)*



RE = (a+b)+

L = {a, b, ab, ba, bb, aa, aab, ---}

RE = (ab)*



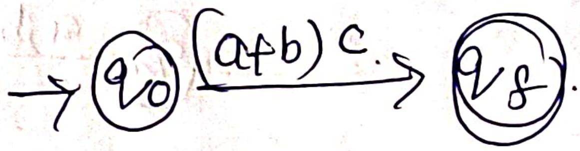
L = {ϵ, ab, abab, ---}

⊛ Input given the Regular expression (RE)
Output will be FA.

Q1: Convert the RE into DFA
$(a+b)c$.

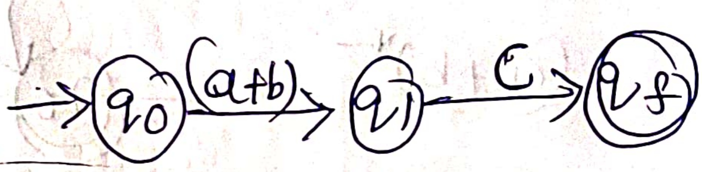
Soln: Let take 2 states q_0 & q_f

Step 1:



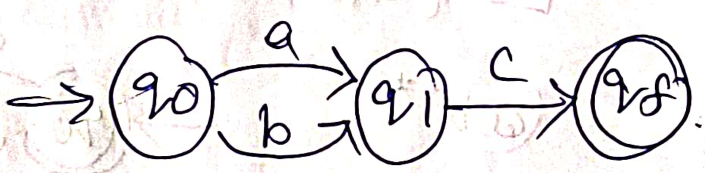
Expand / split into 2 part $(a+b)$ & c.

Step 2:



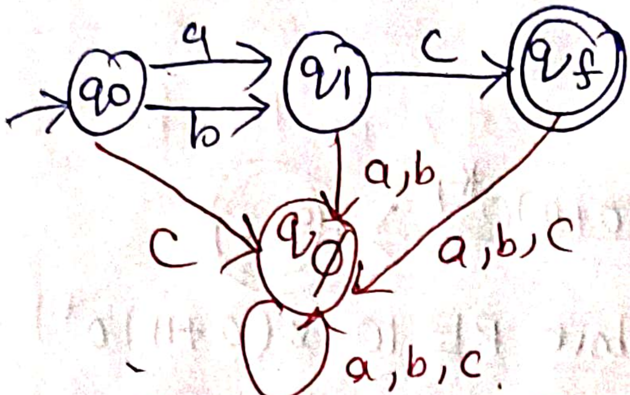
Now split $(a+b)$ also
i.e a or b.

Step 3:



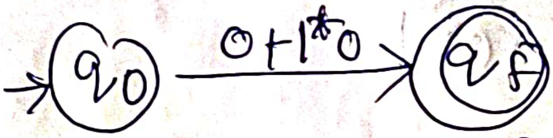
$Q = \{q_0, q_1, q_2\}$, $\Sigma = \{a, b, c\}$

Check all the states have the path of set of symbols a, b & c.
if not consider dead state q_ϕ.



Q2 : convert $(0 + 1^*0)$ into DFA

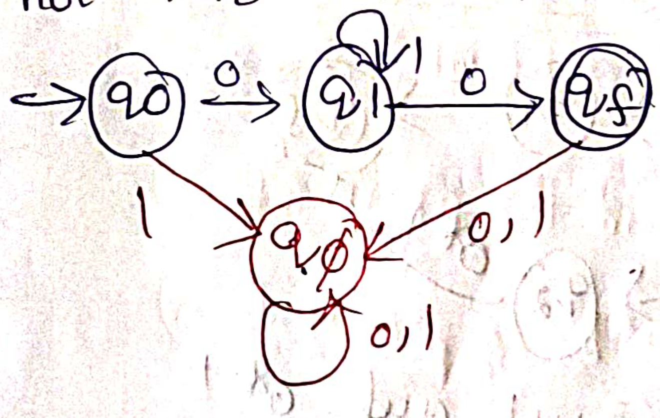
Sol:-



Expand above one of RE

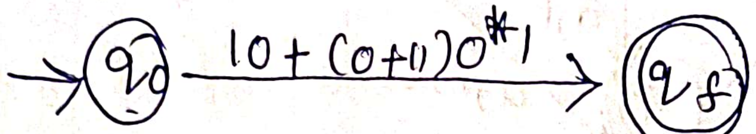
, expand 1^*0.

Check the paths for q_0, q_1 & q_f of 0,1 if not exist consider the dead state.

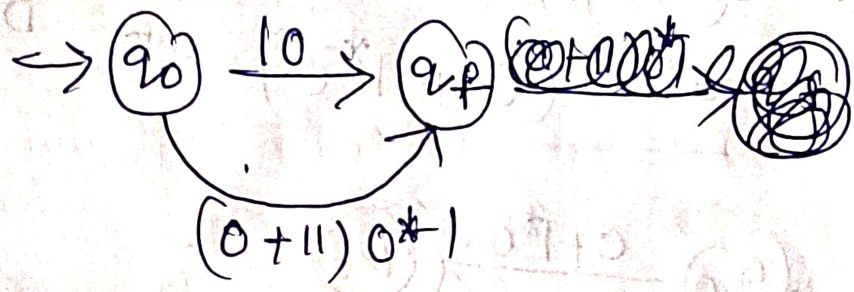


3Q: convert RE $10 + (0+11)0^*1$ into DFA.

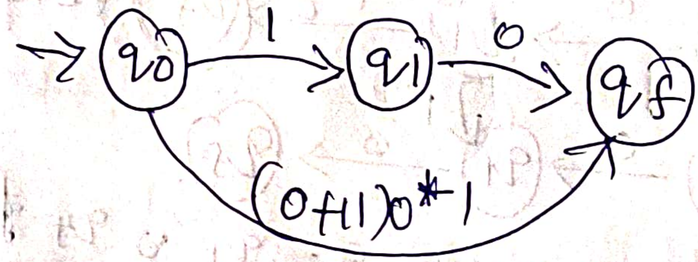
Sol:



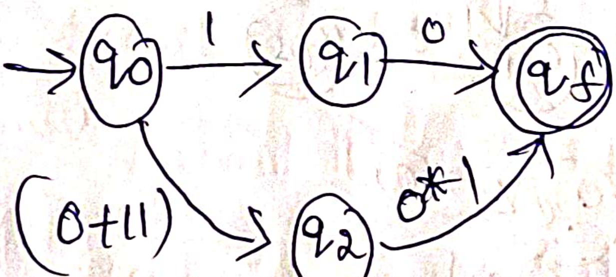
Expand the above RE $10 + (0+11)0^*1$



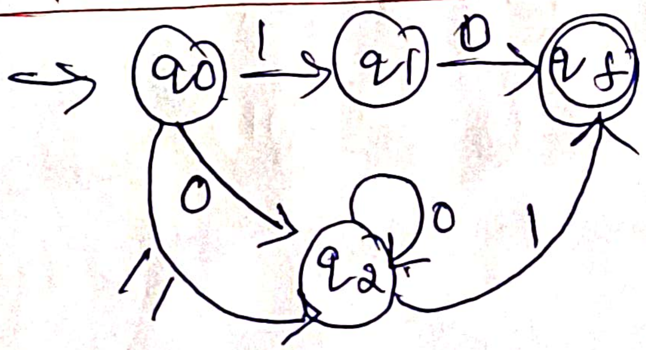
Expand 10



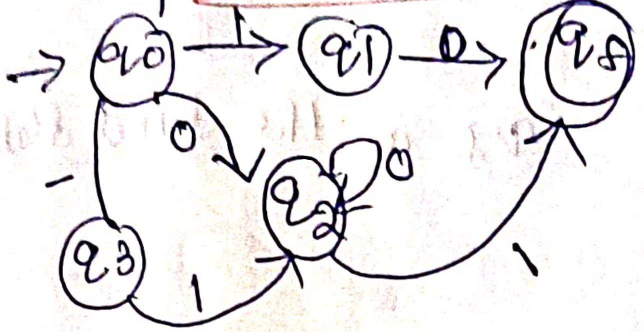
Expand $(0+11)0^*1$



Exand $(0+11)$ and 0^*1



Expand 11 :



.Check the paths for all the states of 0 & 1.

The above one is not in DFA.

It is in NFA.

Now convert NFA into DFA.

Draw the TT for NFA

δ	0	1
q_0	q_2	$\{q_1, q_3\}$
q_1	q_f	ϕ
q_2	q_2	q_f
q_3	ϕ	q_2
q_f	ϕ	ϕ

Consider the initial state., check new state arrived.

$(q_1, q_3) \rightarrow$ $\delta((q_1, q_3), 0) = \delta(q_1, 0) \cup \delta(q_3, 0)$

$= q_f \cup \phi = q_f$

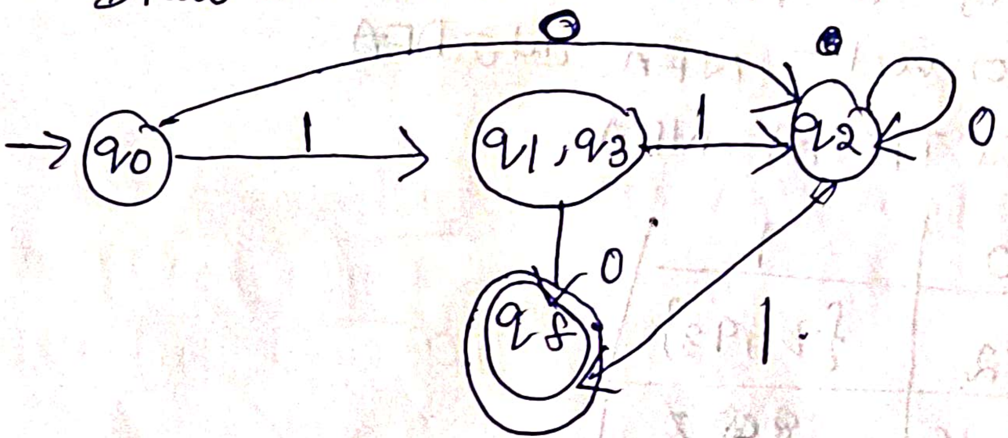
$\delta((q_1, q_3), 1) = \delta(q_1, 1) \cup \delta(q_3, 1)$

$= \phi \cup q_2 = q_2$.

δ	0	1
→q_0	q_2	$\{q_1, q_3\}$
$\{q_1, q_3\}$	q_f	q_2
q_2	q_2	q_f
q_f	ϕ	ϕ

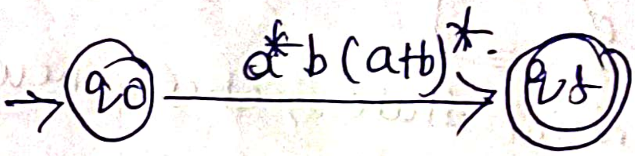
q_f is the final state.

Draw the DFA for the above TT

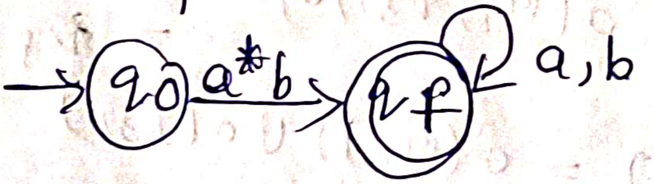


4Q). $a^* b\ (a+b)^*$

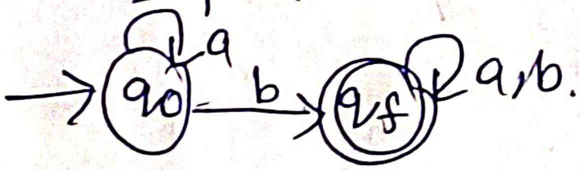
Sol^n:



Expand the above RE

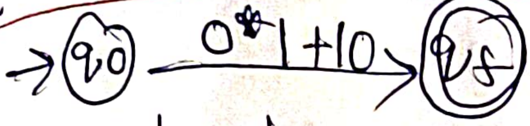


Expand a^*b

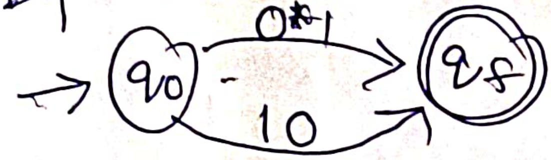


8). 0^*1+10 Convert into FA.

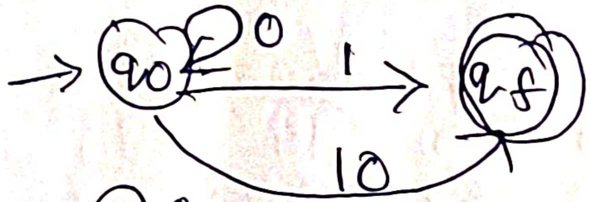
Sol:



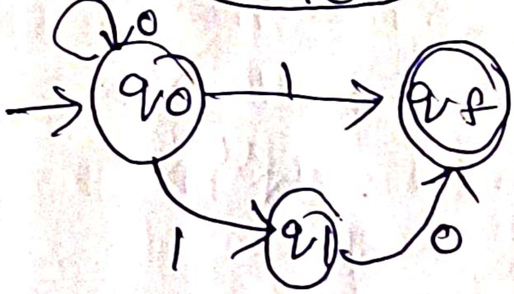
Expand above one.

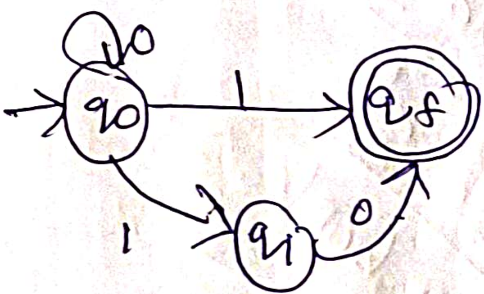


, expand 0^*1



, Now expand 10





Not complete DFA
Consider dead state.
it is in NFA.
If the given question
is given only FA
leave it, if DFA
convert into DFA.

Convert FA to RE :

Steps to convert FA to RE

Solution : 1) Equation for each state based on incoming edges.

2) Add ϵ to initial state.

3) Simplify the equation using ARDEN's theorm and find the final state for RE

Conditions :

1) FA should not contain ϵ Transitions.
2) FA should have only one initial state.

ARDEN's Theorm :

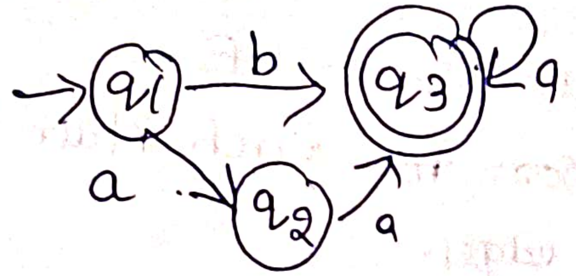
$R = Q + RP$ has unique solution

$R = QP^*$

if P & Q are RE over Σ & P does not contain ϵ.

$$R = Q + RP$$
$$= Q + QP^* \cdot P$$
$$= Q(\epsilon + P^* \cdot P)$$
$$= QP^* \Rightarrow \text{hence it is proved}$$

Q1) convertion of the following FA to RE.



Solution:

Check the conditions (i) only one initial state

(ii) no ϵ - transitions.

both are true.

Equation for each state and add ϵ to the equation of initial state.

q_1 state :

→ no incoming edges.

→ Add ϵ to the initial state to q_1

$\therefore q_1 = \epsilon$ — 1

q_2 state : For q_2 one incoming edge

$q_1 \xrightarrow{a} q_2$

$\therefore q_2 = q_1 a$ — 2

q_3 state : Incoming from q_1 — $q_1 b$

incoming from q_2 — $q_2 a$

incoming self $q_3 \rightarrow q_3 a$

$\therefore q_3 \rightarrow q_1 b + q_2 b + q_3 a$ — 3

Solve the equations.

Apply ① & ② in eq^n ③

$q_3 = q_1 b + q_2 a + q_3 a$.

$q_3 = \epsilon . b + q_1 aa + q_3 a$

$q_3 = \epsilon . b + \epsilon . aa + q_3 a$

$q_3 = \epsilon (b + aa) + q_3 a$

$q_3 = (b + aa) + q_3 a$.

$R = Q + RP. \Rightarrow R = Q P^*$

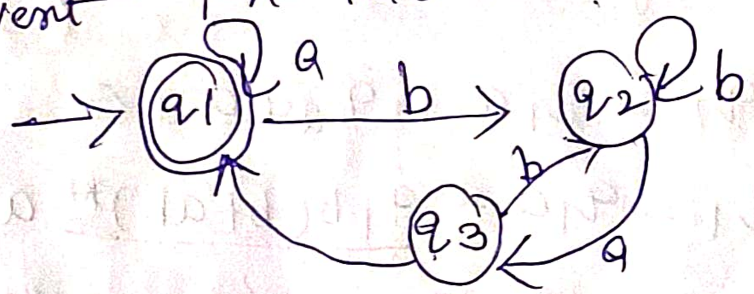
$R = q_3$, $Q = (b + aa)$, $P = a$

$\therefore R = QP^* \Rightarrow q_3 = (b+aa) a^*$

$\boxed{\therefore RE = (b+aa) a^*}$

Final state is q_3.

2Q). Convert FA into RE.



Sol:
$q_1 = q_1 a + q_3 a + \epsilon$ — ①
$q_2 = q_1 b + q_2 b + q_3 b$ — ②
$q_3 = q_2 a$ — ③

q_1 is final state.

Write q_3 in q_1. ie Apply (3) - in (1)

$$q_1 = q_1 a + q_3 a + \epsilon \quad \text{—} (1)$$

$$q_1 = q_1 a + q_2 a \cdot a + \epsilon \qquad (\because q_3 = q_2 a)$$

$$q_1 = q_1 a + q_2 aa + \epsilon \quad \text{—} (4)$$

Apply (3) eqn in (2)

$$q_2 = q_1 b + q_2 b + q_3 b$$

$$q_2 = q_1 b + q_2 b + q_2 ab.$$

$$q_2 = q_1 b + q_2 (b + ab).$$

$$R = Q + RP$$

$$q_2 = R, \quad Q = q_1 b, \quad P = (b + ab)$$

$$R = QP^* \Rightarrow \boxed{q_2 = q_1 b . (b + ab)^*} \quad \text{—} (5)$$

Apply (5) in (4)

$$q_1 = q_1 a + q_2 aa + \epsilon$$

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

$$q_1 = q_1 (a + b(b + ab)^* aa) + \epsilon.$$

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

$$R = Q + RP \Rightarrow R = QP^*$$

$$R = q_1, \quad Q = \epsilon, \quad P = a + b(b + ab)^* aa$$

$$R = \epsilon (a + b(b + ab)^* aa)^*$$

$$\boxed{RE = (a + b(b + ab)^* aa)^*}$$

Pumping Lemma

Pumping Lemma is used to prove that a Language is not regular language.

The word pumping refers to generating many input strings by pushing a symbol in an input string repeatedly.

Pumping Lemma Theorm :

If L is a regular Language, then there exist a constant n (Pumping Length) such that for every string w in L.

where $|w| \geq n$

$w \rightarrow$ length of string
$n \rightarrow$ pumping length.

We can divide w in to 3 strings $w = xyz$ that

(i) $|y| > 0$
(ii) $|xy| \leq n$
(iii) for all $K \geq 0$ the string xy^Kz is also belongs to L.

Q1. Prove that language is non regular, for $L = \{a^n b^n / n > 0\}$ over $\Sigma = \{a, b\}$

Soln: Need to prove $L = \{a^n b^n / n > 0\}$ is not a regular language.

Apply the pumping Lemma Theorm.

$L = \{ab, aabb, aaabbb, ---\}$

$n = 1 \Rightarrow ab$

$n = 2 \Rightarrow aabb$

$n = 3 \Rightarrow aaabbb$

if n = pumping length = 5

then select the string which is ≥ 5

$w = aaabbb$

$|w| = 6$, $|w| \geq n$

i.e $6 \geq 5$ (True).

split $w = xyz$.,

let $x = aa$, $y = ab$, $z = bb$

(i) $|y| > 0 \Rightarrow |ab| = 2 > 0$ True

(ii) $|xy| \leq n \Rightarrow |aa.ab| \Rightarrow 4 \leq 5$ True.

(iii) $k \geq 0$ for $xy^kz \in L$.

if $k = 0 \Rightarrow aa(ab)^0.bb = aabb \in L$

if $k = 1 \Rightarrow aa(ab)^1.bb \Rightarrow aaabbb \in L$

if $K=2 \Rightarrow aa(ab)^2bb$

$\Rightarrow aa\,aabbbb \notin L$

voilating the rule

$\therefore$ The given language is not a regular language.

2Q). P.T Language is non Regular for
$L = \{a^p \mid p \text{ is a prime}\}$.

Prime no's are $\Rightarrow$ 2, 3, 5, 7, ---

if $p=2 \Rightarrow a^2 = aa$

$p=3 \Rightarrow a^3 = aaa$

$p=5 \Rightarrow a^5 = aaaaa$

$\therefore L = \{aa, aaa, aaaaa, ---\}$

Let $n=3$, then consider $w = aaa$

$|w| = 3 \geq 3$ True.

$w = xyz$, $x = a$, $y = a$, $z = a$.

(i) $|y| > 0$, $|a| = 1 > 0$ True.

(ii) $|x.y| \leq n$, $|a.a| = 2 < 3 \Rightarrow$ True.

(iii) for $k \geq 0$, $xy^kz \in L$

if $k=0$, $a(a)^0(a) = aa \in L$

$k=1$, $a(a)a = aaa \in L$

$k=2$, $a(a)^2a = aaaa \notin L$

String does not belongs to Language.

$\therefore$ The given language is not a regular. Language.

3Q). S.T $L = \{0^{n^2} \mid n > 0\}$ is not a regular Language

Soln:

Given $L = \{0^{n^2} / n \geq 0\}$.

if $n = 1 \Rightarrow 0^{1^2} = 0$

$n = 2 \Rightarrow 0^{2^2} = 0^4 = 0000$

$n = 3 \Rightarrow 0^{3^2} = 0^9 = 000000000$

$L = \{0, 0000, 000000000, ----\}$

Let $n = 2$. then w should ≥ 2

consider $w = 0000$.

$|w| \geq n \Rightarrow 4 \geq 2$ True

divide the string w in to 3 parts

$x = 0, \ y = 0, \ z = 00$

(i) $|y| > 0$, $|0| = 1 > 0$ true.

(ii) $|x.y| \leq n$, $|0.0| = 2 \leq 2 \Rightarrow$ True

(iii) $k \geq 0$, $xy^k z \in L$

if $k = 0 \Rightarrow 0(0)^0 0 \Rightarrow 00 \notin L$.

$\therefore$ The given language is not a regular language.