10/03/22

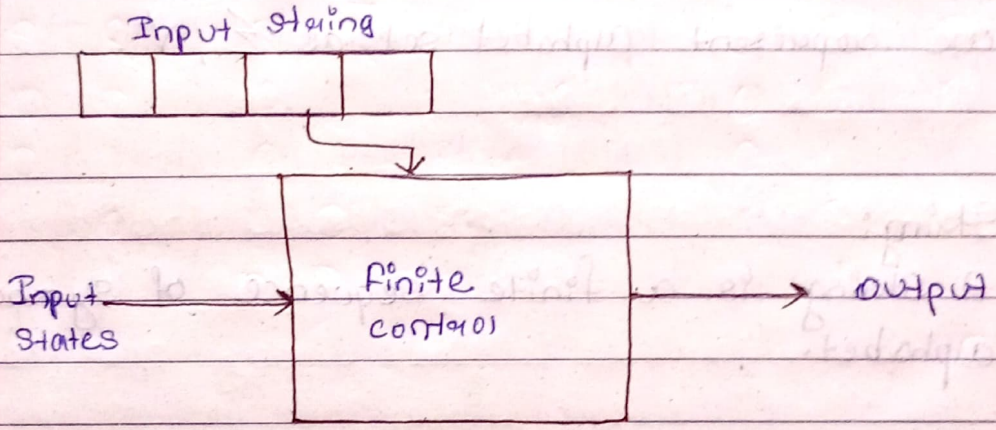
Introduction

Formal Language :

The mode of communication with machine properly such that any work can be done automatically is called as formal language.

Automata :

An automatic machine or an automaton were any task can be perform without the direct interface of the human being.



Components of an Automatic machine

1) State
2) Input
3) Finite control
4) state transition
5) Output

Unit-I

12/03/21

Mathematical Preliminaries

→ Alphabet
→ String
→ Language
→ Grammer
→ Operations on String

Alphabet:
A finite no. of symbols used to frame any string of a language.

we represent alphabet set as Σ, V.

String:
A string is a finite sequence of symbols or alphabet.

The symbols are represented or written next to each other without any separation.

The string is represented by (w) lower case w and length of string by $|w|$.

Ex:
$w = 0101$ $\quad |w| = 4$

we can write empty string has $\in, \{\}, 1$.

Language:
It is denoted by L.

set of strings formed by same set of alphabets

Grammer :

Set of rules used to generate a string of a language is called as grammer.

It is consisting of set of syntax is.

It is represented by G(V, T, P, S) with 4 tuples V, T, P, S.

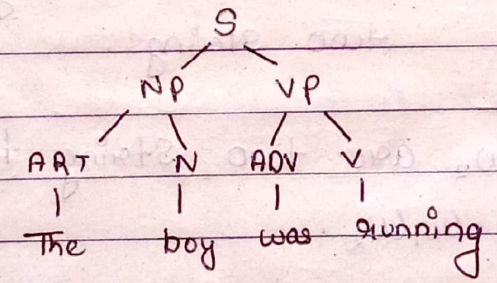
V stands for Variables (all capital letters)

T stands for terminal (except capital letters, all lower cases, digits & symbol).

P stands for productions (rules)

S stands for Starting Symbol.

Ex :



$\Sigma = \{a, \ldots, z, A, \ldots, Z\}$

V = S, NP, VP, ART, N, ADV, V

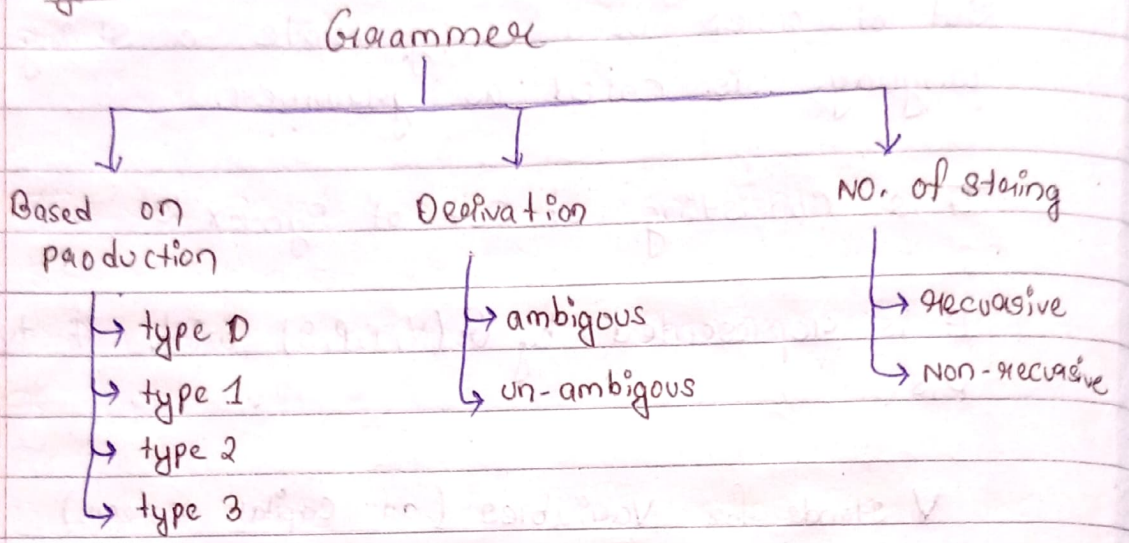
T = {The, boy, was, running}

P = S → NP, VP NP → ART, N VP → ADV, V

ART → The N → boy ADV → was V → running

S = S

Types



Operation on String

→ Concatenation, Union, Closure and transpose

Concatenation (.) : It is the binary operation b/w two strings.

If ω_1 & ω_2 are two string then the concate string is $\omega_1\omega_2$.

Ex: $\omega_1 = 01$ $\omega_2 = 11$

$\omega_1 \cdot \omega_2 = \omega_1\omega_2 = 0111$

Union (+): Binary operation b/w two strings
ω_1 & ω_2 are two strings
$\omega_1 + \omega_2$ are union

Ex: $\omega_1 = 01$ $\omega_2 = 11$

$\omega_1 + \omega_2 = (01 + 11)$

Closure : Uniary Operation

Finding the set of all possible Strings is called as closure operation.

It is the power set of a String.

There are two types of closure :

1) Kleens closure (*)
2) positive closure (+)

$$\Sigma^+ = \Sigma^* - \{\wedge\}$$

Ex : $\Sigma = \{0\}$

1) $\Sigma^* = \wedge, 0, 00, 000, 0000, ----$
2) $\Sigma^+ = 0, 00, 000, 0000, ----$

14/03/21

Transpose : If ω is given string then ω^T is stands for transpose of ω.

Ex :

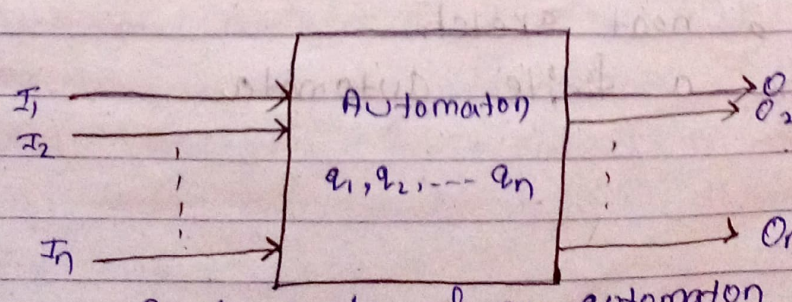
$\omega = \omega_1 \omega_2 \omega_3 --- \omega_n$

$\omega^T = \omega_n ---- \omega_3 \omega_2 \omega_1$

$\omega = abbbab$

$\omega^T = babbba$

* Basic Characteristics of an Automaton



Basic model of an automaton

→ Input → output → output relation
→ State → state relation

Input : Finite no. of fixed values from the set of alphabet provided to the automaton on which the operations are carried out is called as input.

Output : Finite no. of fixed values from the set of alphabet after the operation is called as output.

State : At any instant of time the status of the automaton is known as state.

State relation : The next state of an automaton is determine by the presence state and the present input is called as state relation.

Output relation : The output of an automaton depends upon present input only or both state and input.

IQ) 1) Differentiate b/w Kleen's closure and positive closure.
2) Describe basic model of an automaton with a neat sketch.
3) Define a finite automata

Finite Automata: An automaton with finite no. of states and input is called as finite automata or finite state machine.

FA or FSM is the abbreviation.

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

where, Q = the set of finite states $= \{q_1, q_2 \dots q_n\}$
Σ = set of finite input symbol
q_0 = starting state of the automaton
F = set of final states / stop states
δ = transition function

$$Q \times \Sigma \rightarrow Q$$

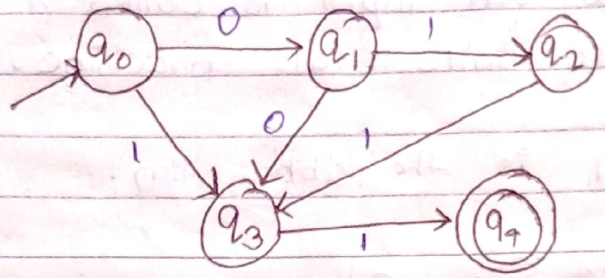
present state, present input, next state

δ (p.state, p.input) → next state
$\delta(q_i, a) \longrightarrow q_j$

Transition represent in 3 ways:
1) Transition function (δ (Pstate, pinput) → next state)
2) Transition graph
3) Transition Table

Transition graph: It is a directed, ~~leveled~~ labelled graph.
- Each node or vertex of the graph represent a state and depicted as a circle (O).
- Arc - represents the transition
- label - indicates input/output or simply input
- Final state - represented by (◎) double circle.
- Start state (→).

Ex



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

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

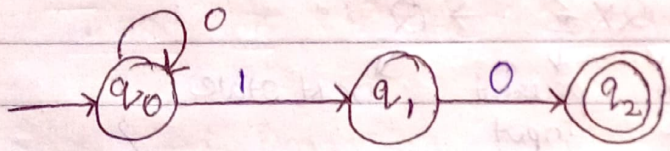
$q_0 = q_0$

$F = \{q_4\}$

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

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

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



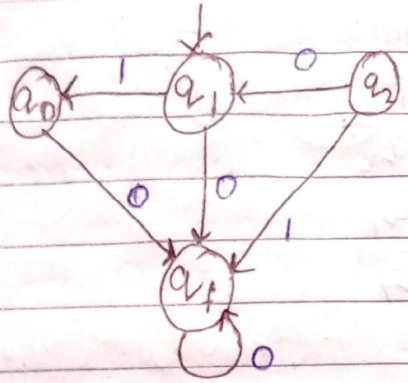
Transition Function

$\sigma(q_0, 0) \vdash q_0$ $\sigma(q_0, 1) \vdash q_1$

$\sigma(q_1, 0) \vdash q_2$

Transition Table

Pstate	next state 0	1
→ q_0	q_0	q_1
q_1	q_2	-
(q_2)	-	-



Transition Table

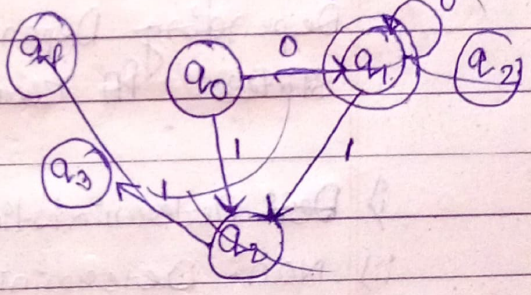
pstate	next state 0	next state 1
q_0	q_f	-
→ q_1	q_f	q_0
q_2	q_1	q_f
(q_f)	q_f	-

Transition function

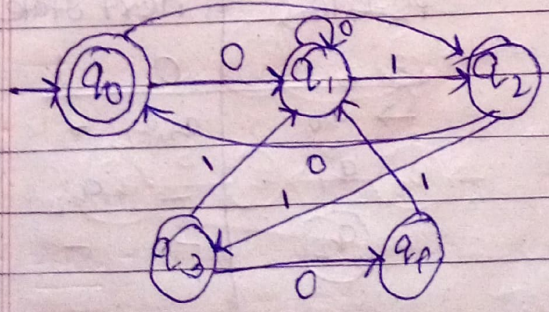
$\delta(q_0,0) \vdash q_f$
$\delta(q_0,0) \vdash q_f$
$\delta(q_1,1) \vdash q_0$
$\delta(q_2,0) \vdash q_1$
$\delta(q_2,1) \vdash q_f$
$\delta(q_f,0) \vdash q_f$

pstate	next state 0	next state 1
→ (q_0)	q_1	q_2
q_1	q_1	q_2
q_2	q_0	q_3
q_3	q_f	q_1
q_f	-	q_1

Transition graph



Transition graph

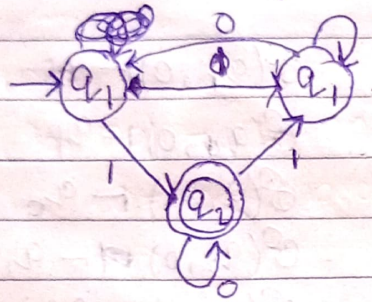


Transition function

$\delta(q_0,0) \vdash q_1$
$\delta(q_0,1) \vdash q_2$
$\delta(q_1,0) \vdash q_1$
$\delta(q_1,1) \vdash q_2$
$\delta(q_2,0) \vdash q_0$
$\delta(q_2,1) \vdash q_3$
$\delta(q_3,0) \vdash q_f$
$\delta(q_3,1) \vdash q_1$
$\delta(q_f,1) \vdash q_1$

pstate	next state	
	0	1
$\rightarrow q_0$	q_1	q_2
q_1	q_0	q_1
(q_2)	q_2	q_1

Transition graph



Transition function

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

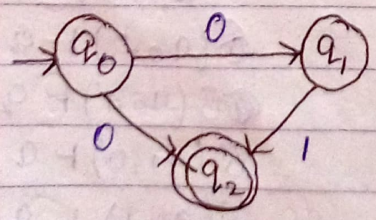
Types of Finite Automata

~~Depending~~ Depending upon the no. of next states FA can be of two types :

i) ~~Deri~~ Deterministic FA (DFA)
ii) NON- Deterministic FA (NFA)

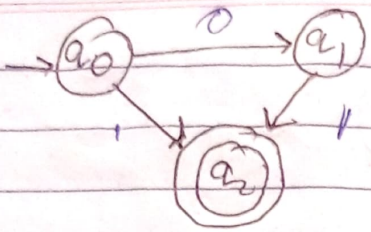
IQ* i) Differentiate b/w DFA & NFA

1)



Pstate	next state	
	0	1
$\rightarrow q_0$	q_1, q_2	-
q_1	-	q_2
(q_2)	-	-

2)



Pstate	next state	
	0	1
→ q_0	q_1	q_2
q_1	-	q_2
(q_2)	-	-

Deterministic FA:- On any i/p symbol if a single next state is determined. It is called as DFA.

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

$\delta : Q \times \Sigma \to Q$

NO null string is entertained

NON- Deterministic FA:- more than one next state will be determined on any i/p symbol is called NFA.

NFA 2^Q

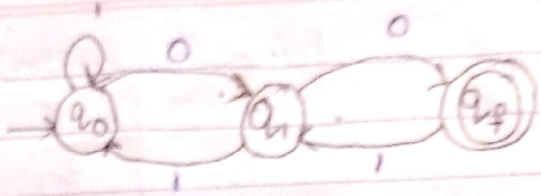
$M = (Q, \Sigma, \delta, q_0, F)$

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

null string is entertained.

Acceptance of a String

A string is said to be accepted by a finite automata, if after reading all the symbols it will reach to the final state.



1001

$(q_0, 1001) \vdash (q_0, 001) \vdash (q_1, 01) \vdash (q_f, 1) \vdash q_1$

$\therefore$ 1001 is not accepted by FA.

100

$(q_0, 100) \vdash (q_0, 00) \vdash (q_1, 0) \vdash q_f$

$\therefore$ $\omega = 100$ accepted by FA.

101010

$(q_0, 101010) \vdash (q_0, 01010) \vdash (q_1, 1010) \vdash (q_0, 010) \vdash$
$(q_1, 10) \vdash (q_0, 0) \vdash q_1$

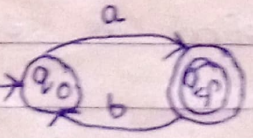
$\therefore$ 101010 is not accepted by FA.

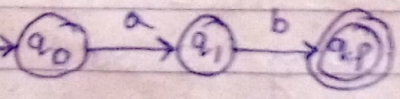
1010100 → accepted by FA.

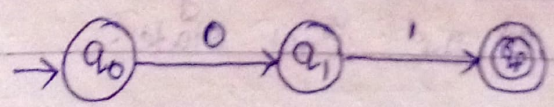
*Design of a FA :

1Q. Design a FA ~~that~~ accepts the string $\omega = 01$

(a) Accepts $\wedge$ (null) = 

(b) Accepts (a+b) = 

(c) Accepts (ab) = 



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

$\Sigma = (0,1)$

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

$q_0 = q_0$

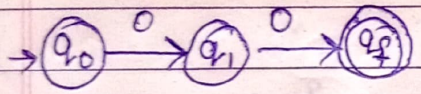
$F = \{q_f\}$

2Q. Design a FA that accepts the strings consisting of exactly two 0's.

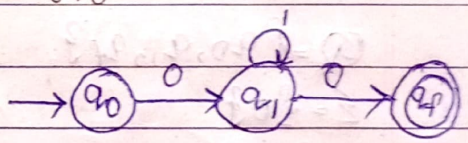
$\Sigma = \{0,1\}$

00, 010, 001, 100, 01^*0, 1^*00, 001^*

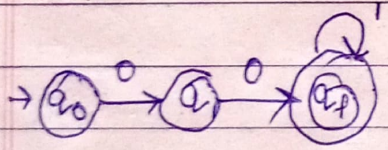
00



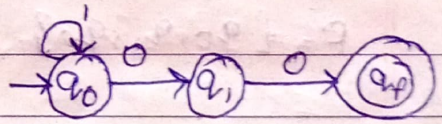
010

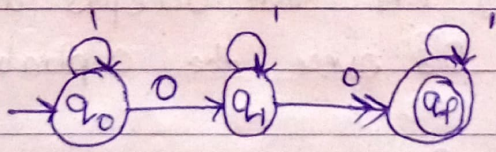


001



100





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

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

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

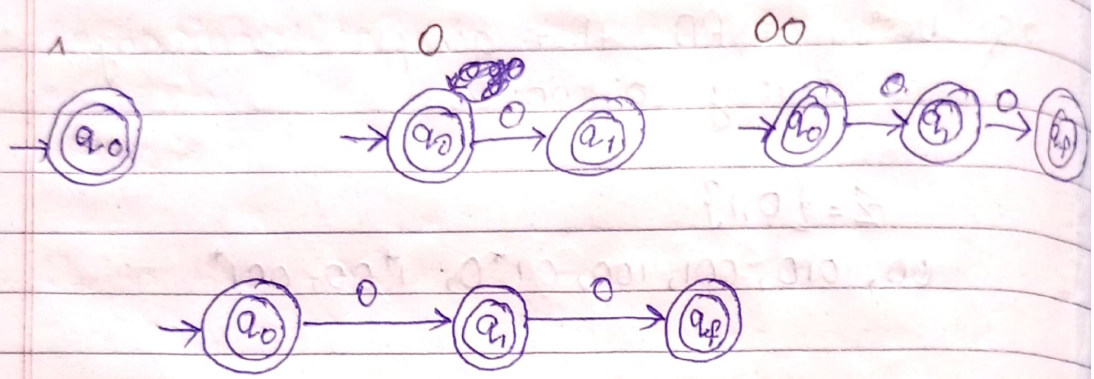
$q_0 = q_0$

$F = \{q_f\}$

3Q. Design a FA that accepts all the string with maxm 2 0's over the alphabet set 0.

$\Sigma = \{0\}$

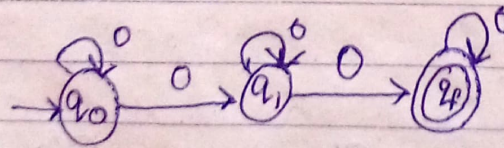
00, 0, ∧



$Q = \{q_0, q_1, q_f\}$
$\Sigma = \{0\}$
$\delta = \{ \delta(q_0, 0) \vdash q_1, \ \delta(q_1, 0) \vdash q_f \}$
$q_0 = q_0$
$F = \{q_0, q_1, q_f\}$

4Q. Design a FA that accepts all the string with minm 2 0's over the alphabet set 0.

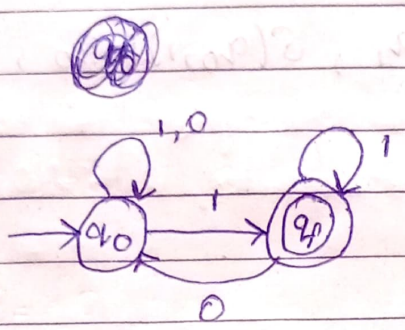
$\Sigma = \{0\}$



$Q = \{q_0, q_1, q_f\}$
$q_0 = q_0$
$F = \{q_f\}$
$\delta = \{ \delta(q_0, 0) \vdash q_0, q_1, \ \delta(q_1, 0) \vdash q_1, q_f, \ \delta(q_f, 0) \vdash q_f \}$
$\Sigma = \{0\}$

5 Q. Design a FA that accepts all the strings ending with 1 over the alphabet set 0, 1.

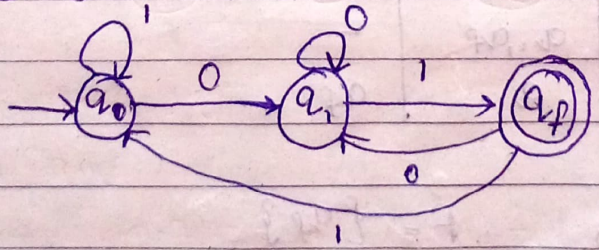
$\Sigma = \{0, 1\}$ 1, 01, [illegible], 0*1, 11, 1*1
01*01, 0*1*0*1.



$Q = \{q_0, q_1\}$
$\Sigma = \{0, 1\}$
$\delta = \{ \delta(q_0, 1) \vdash q_0, q_f \;,\; \delta(q_0, 0) \vdash q_0 \;,$
$\delta(q_f, 1) \vdash q_f \;,\; \delta(q_f, 0) \vdash q_0 \}$
$F = \{q_f\}$
$q_0 = q_0$

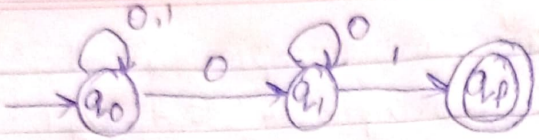
6 Q. Design a FA that accepts all the strings ending with 01 over the alphabet set 0, 1.

$\Sigma = \{0 1\}$



(DFA)

$(q_0, 01100101) \vdash (q_1, 1100101) \vdash (q_f, 100101) \vdash$
$(q_0, 00101) \vdash (q_1, 0101) \vdash (q_1, 101) \vdash (q_f, 01)$
$\vdash (q_1, 1) \vdash q_f$



(NFA)

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

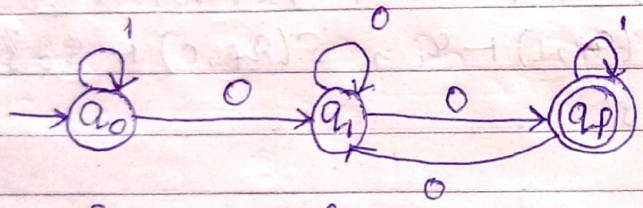
$\Sigma = \{0, 1\}$

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

$q_0 = q_0$

$F = \{q_f\}$

7Q. Design the FA that accepts all the strings consisting of 2 consuctive zeros's over the alphabet set 0, 1.



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

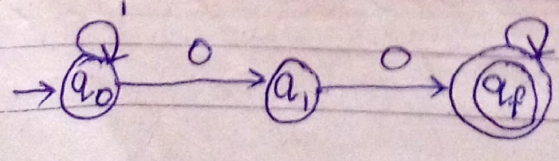
$\Sigma = \{0, 1\}$

δ -

	0	1
q_0	q_1	q_0
q_1	q_1, q_f	-
q_f	q_1	q_f

$q_0 = q_0$ $\quad F = \{q_f\}$

8Q. Design a FA that accepts all the strings only 2 consecutive 0's over the $\Sigma = \{0, 1\}$



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

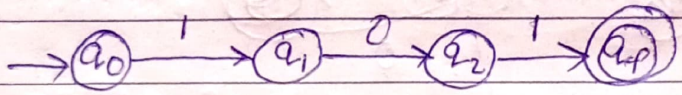
$\Sigma = \{0, 1\}$

$q_0 = q_0$ $F = \{q_f\}$

$\delta =$

	0	1
q_0	q_1	q_0
q_1	q_f	-
q_f	-	q_f

9Q. Design a FA that accepts the string 101.



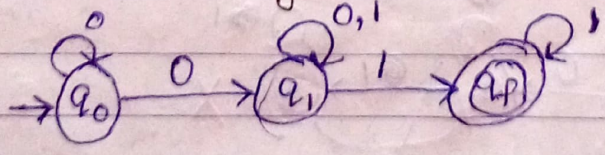
$Q = \{q_0, q_1, q_2, q_f\}$

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

$\delta =$

	0	1
q_0	-	q_1
q_1	q_2	-
q_2	-	q_f
q_f	-	-

10Q. Design a FA that accepts the string starting with 0 and ending with 1



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

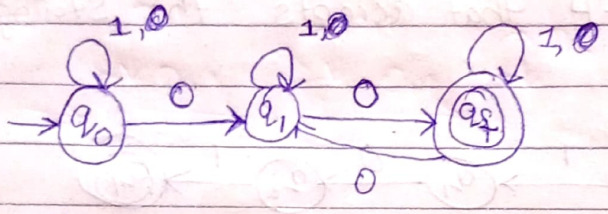
$\Sigma = \{0, 1\}$

$q_0 = q_0$ $F = \{q_f\}$

δ =

	0	1
q_0	q_0, q_1	-
q_1	q_1	q_1, q_f
q_f	-	q_f

11Q. Design an FA that accepts all the strings with even no. of zero's over $\Sigma = \{0, 1\}$



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

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

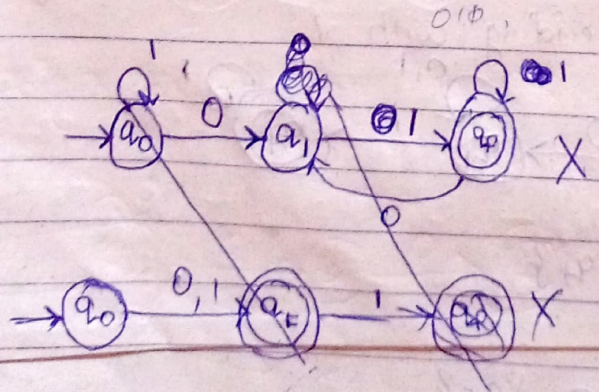
δ =

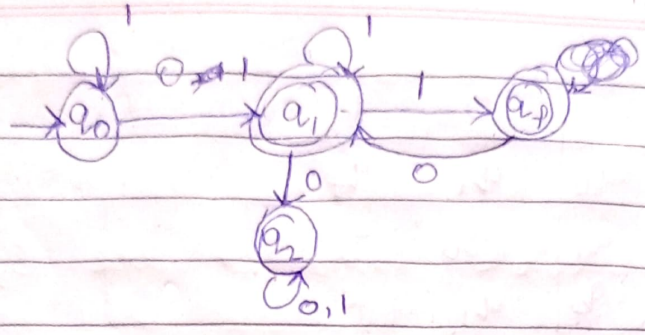
	0	1
q_0	q_1	q_0
q_1	q_f	q_1
q_f	q_f	q_f

12Q. Design a FA that accepts all the string over the $\Sigma = \{0, 1\}$ such that not containing two consecutive 0's.

010, 0101, 1010

10010101010





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

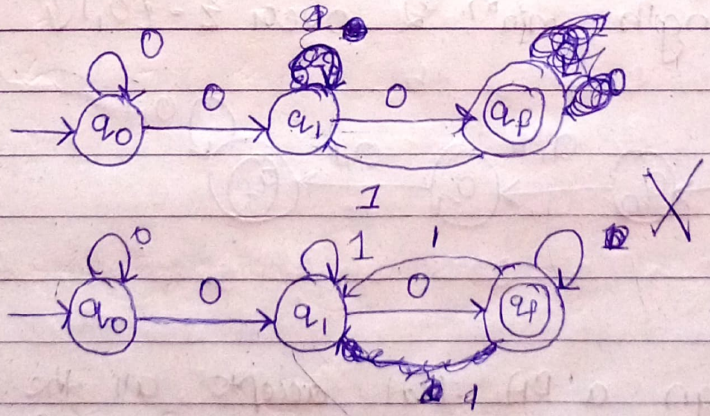
$\Sigma = \{0, 1\}$ $\quad q_0 = q_0$ $\quad F = \{q_1, q_f\}$

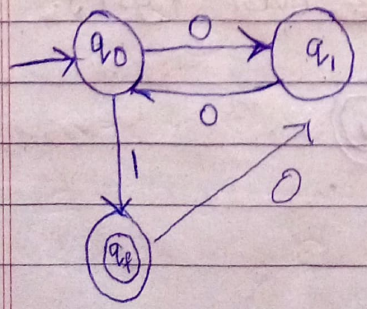
$\delta =$

	0	1
q_0	q_1	q_0, q_1
q_1	q_2	q_1, q_f
q_2	q_2	q_2
q_f	q_1	–

13Q. Design a FA that accepts all the strings with even 0's ~~and~~ followed by a single 1.

010, 001, 100, 10000, 01000





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

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

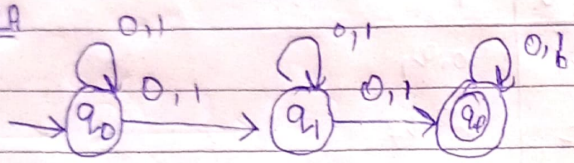
$F = \{q_f\}$

δ-

	0	1
q_0	q_1	q_f
q_1	q_0	-
q_f	q_1	-

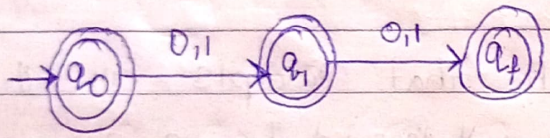
14 Q. Design a FA that accepts all the strings of length ~~atleast~~ max^m 2 over the alphabet set 0,1.

NFA

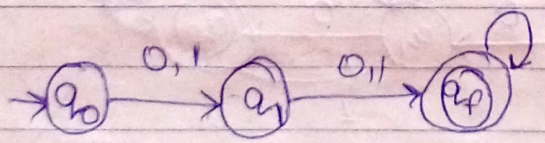


DFA

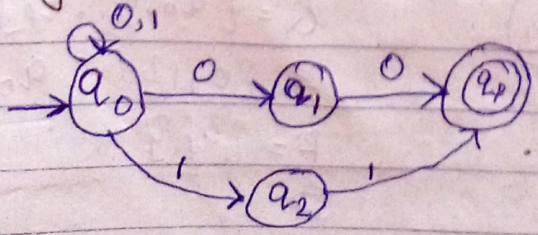
$\{\wedge, 0, 1, 10, 01, 00, 11\}$



15 Q. Design a FA that accepts all the strings of length min^m 2 over $\Sigma = \{0,1\}$



16 Q. Design a FA that accepts all the strings ending with 00 or 11 over $\{0,1\}$.



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

$q_0 = q_0$ $\quad F = \{q_f\}$ $\quad \Sigma = \{0, 1\}$

$\delta =$

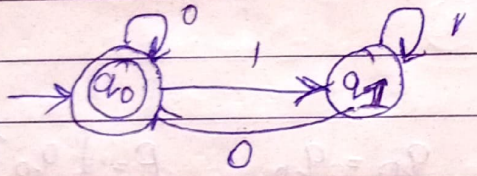
	0	1
q_0	q_0, q_1	q_0
q_1	q_f	-
q_2	-	q_f
q_f	-	-

of length

*

17Q. Design a DFA that will accepts all the binary strings divisible by 2 over $\{0, 1\}$.

0000, 0010, 0100, 0110, 1000, 1010



Step 1 Identify the possible remainder
Those are 0, 1

Strings

Step 2 Identify the string behaviour for accepts .ie., the string ended with 0.

Step 3 Compute the no. of states for each of remainder .i.e., State generation,

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

ge

Step 4 Draw the diagram

$Q = \{q_0, q_1\}$ $\quad q_0 = q_0$ $\quad F = \{q_0\}$

$\delta =$

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

18Q. Design a DFA that accepts all the binary string divisible by 3 over $\{0,1\}$

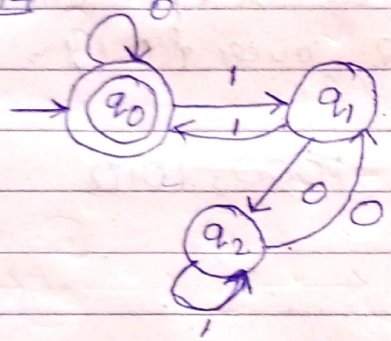
Step 1 possible remainder are 0, 1, 2

Step 2 The String behaviour - ~~ending with 0,1~~ NO such behaviour

Step 3 no. of states are

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

Step 4



0 - 0000
3 - 0011
6 - 0110
9 - 1001
12 - 1100

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

$\delta =$

	0	1
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_1	q_2

19Q. Design a DFA that accepts all the binary string divisible by 5 over $\{0,1\}$

Step 1 possible remainder are 0, 1, 2, 3, 4.

00 01 10 11 100

Step 2 String behaviour - NO such behaviour

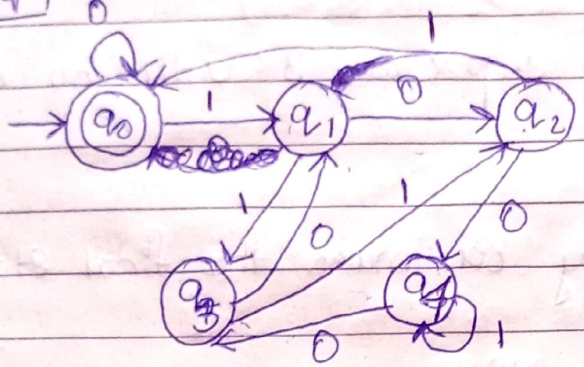
0 - 0000
5 - 0101
10 - 1010
15 - 1111
20 - 10100
~~25 = 11001~~

Step 3 no. of states are

$q_0 \rightarrow 0$ $q_1 \rightarrow 1$ $q_2 \rightarrow 2$ $q_3 \rightarrow 3$

$q_4 \rightarrow 4$

Step 4



1011 0

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

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

$\delta =$

	0	1
q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_0
q_3	q_1	q_2
q_4	q_3	q_4

E-closure of NFA

Step 1:- Intialize E-closure of a State is to itself.

E-closure $(q) = \{q\}$

Step 2:- Find out the next state from the current State on E-transition

$\delta(q, t)$

Step 3 :- Make the next state equal to the Current State.

Current state = next state.

Step 4 :- Repeat step 2 and 3 until no new state is found.

Step 5 :- Identify and mark the final state.

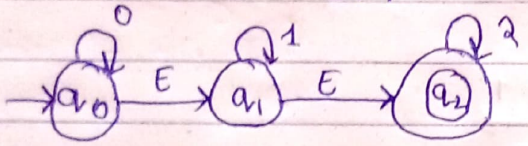
$f_A + \epsilon$ = NFA

↓

NFA without ϵ

↓

DFA



Q. find out ϵ-closure of each state :-

ϵ-closure $(q_0) = \{q_0, q_1, q_2\}$

ϵ-closure $(q_1) = \{q_1, q_2\}$

ϵ-closure $(q_2) = \{q_2\}$

0- transition :-

$\delta(q_0, 0) = \epsilon$-closure $(\delta(q_0, \epsilon), 0)$

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

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

$= \epsilon$-closure $(q_0, \emptyset, \emptyset)$

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

$\delta(q_1, 0) = \epsilon$-closure $(\delta(q_1, \epsilon), 0)$

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

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

$= \epsilon$-closure $(\emptyset, \emptyset) = \emptyset$

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

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

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

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

1- transition:-

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

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

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

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

$= \{q_1, q_2\}$ ①

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

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

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

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

$= \{q_1, q_2\}$

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

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

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

2- transition:-

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

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

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

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

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

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

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

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

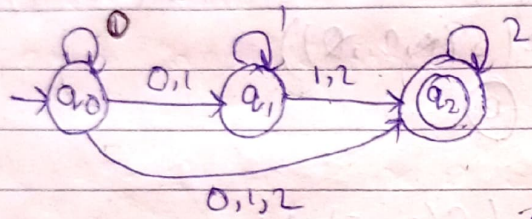
$= E\text{-closure}(\delta(q_1,q_2))$

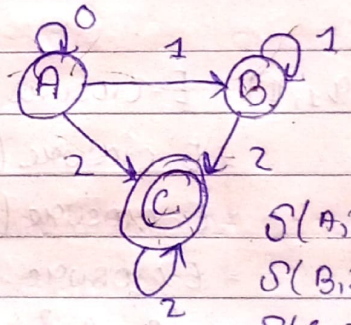
$= E\text{-closure}(q_2) = \{q_2\}$

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

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

$= E\text{-closure}(\delta(q_2)) = \{q_2\}$



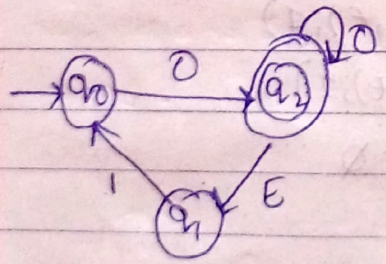


$\delta(A,0) = A$ $\quad \delta(A,1) = B$ $\quad \delta(A,2) = C$

$\delta(B,0) = \emptyset$ $\quad \delta(B,1) = B$ $\quad \delta(B,2) = C$

$\delta(C,0) = \emptyset$ $\quad \delta(C,1) = \emptyset$ $\quad \delta(C,2) = C$

Q.



E-closure $(q_0) = \{q_0\}$

E-closure $(q_1) = \{q_1\}$

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

0-transition:

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

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

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

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

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

$= E\text{-closure}(\delta(\theta)) = \theta$

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

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

$= E\text{-closure}(\delta(q_2) = \theta \{q_1, q_2\}$

1-transition:

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

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

$= E\text{-closure}(\delta(\theta)) = \theta$

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

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

$= E\text{-closure}(\delta(q_0)$

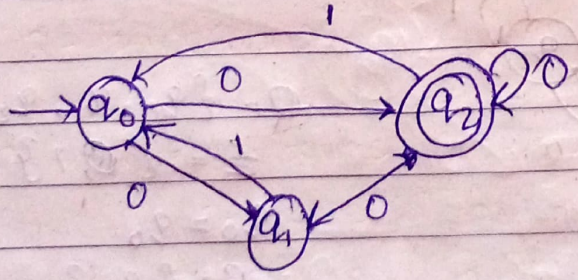
$= E\text{-closure}\{q_0\} = \{q_0\}$

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

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

$= E\text{-closure}(\delta(\theta)) = E\text{-closure}(q_\theta)$

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



Conversion of NFA to DFA

Step 1:- Assume the NFA = M
$\{Q, \Sigma, \delta, q_0, F\}$

DFA = $M' = \{Q', \Sigma, \delta', q_0, F'\}$

Step 2:- Initialise $Q' = \{q_0\}$

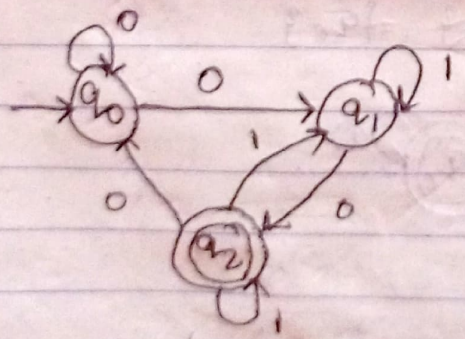
Step 3:- Find out the transition of q_0 on each input symbols.

Step 4:- If any new state is encountered then add it to Q'.

Step 5:- Find out the transition of each new state on each input symbols.

Step 6:- Add the new state to Q' and repeat step 4 and 5 until no new next state is found.

Ex



$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{0, 1\}$
$q_0 = q_0$
$F = \{q_2\}$

	0	1
→ q_0	q_0, q_1	-
q_1	q_2	q_1
(q_2)	q_0	q_2, q_1

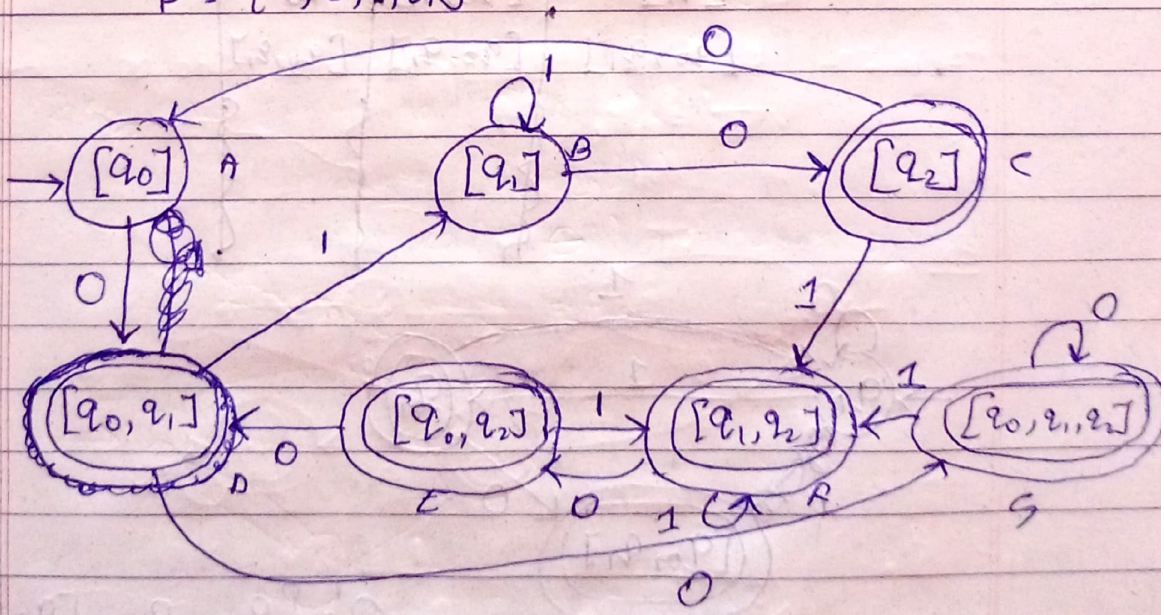
1) $Q', \Sigma, \delta', q_0, F'$

2) $Q' = \{q_0\}$

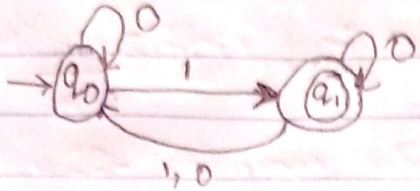
3) $\delta' =$

P.State	Next State	
	0	1
$\rightarrow q_0$	$[q_0, q_1]$	-
$\blacktriangle [q_0, q_1]$	$[q_0, q_1, q_2]$	$[q_1]$
$[q_1]$	$[q_2]$	$[q_1]$
$[q_0, q_1, q_2]$	$[q_0, q_1, q_2]$	$[q_1, q_2]$
$[q_2]$	$[q_0]$	$[q_1, q_2]$
$[q_1, q_2]$	$[q_0, q_2]$	$[q_1, q_2]$
$[q_0, q_2]$	$[q_0, q_1]$	$[q_1, q_2]$

- $Q' = \{A, B, C, D, E, F, G\}$
  $\Sigma = \{0, 1\}$ $q_0 = q_0$
  $F' = \{C, E, F, G\}$



Q.



$Q = \{q_0, q_1\}$ $\Sigma = \{0, 1\}$ $q_0 = q_0$ $F = \{q_1\}$

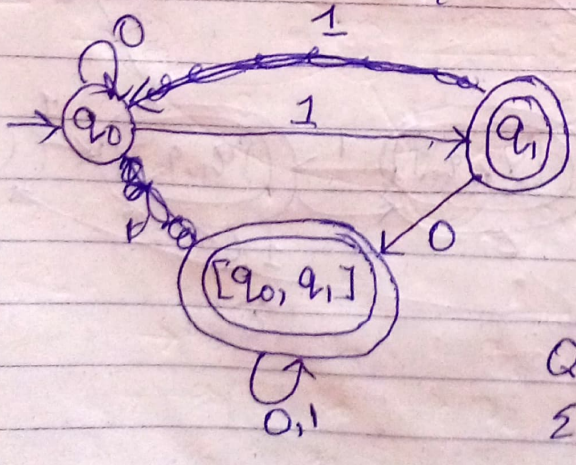
$\delta =$

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

$Q' = \{q_0\}$

$\delta' =$

P.state	next state	
	0	1
$\rightarrow [q_0]$	q_0	q_1
$[q_1]$	$[q_0, q_1]$	q_0
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$

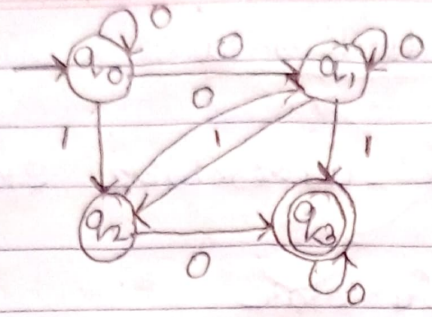


$Q' = \{q_0, q_1, [q_0, q_1]\}$

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

$F' = \{q_1, [q_0, q_1]\}$

Q.



$Q = \{q_0, q_1, q_2, q_3\}$ $\Sigma = \{0, 1\}$ $q_0 = q_0$ $F = \{q_3\}$

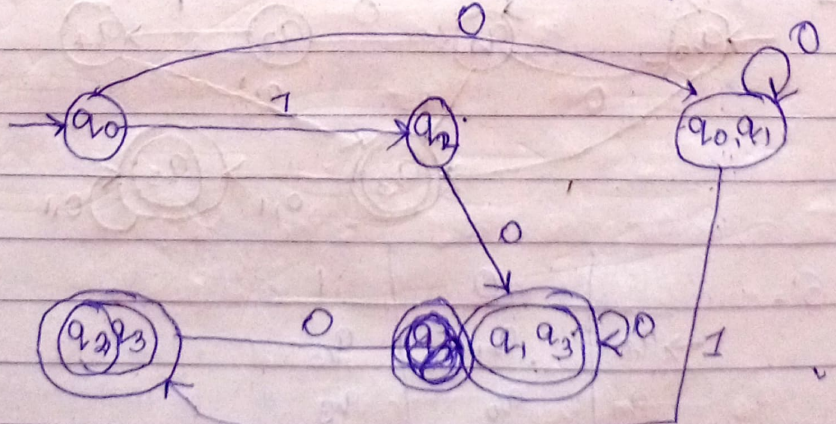
$\delta =$

	0	1
$\rightarrow q_0$	q_0, q_1	q_2
q_1	q_1	q_2, q_3
q_2	q_3, q_1	-
(q_3)	q_3	-

$Q' = \{q_0\}$

$\delta' =$

P. State	Next State (0)	Next State (1)
$\rightarrow [q_0]$	$[q_0, q_1]$	$[q_2]$
$[q_2]$	$[q_3, q_1]$	-
$[q_0, q_1]$	$[q_0, q_1]$	$[q_2, q_3]$
$[q_1, q_3]$	$[q_1, q_3]$	-
$[q_2, q_3]$	$[q_1, q_3]$	-
~~$[q_1, q_3]$~~	~~$[q_1, q_3]$~~	~~$[q_2, q_3]$~~



$Q = \{q_0, q_2, [q_0, q_1], [q_2, q_3], [q_1, q_3]\}$

$\Sigma = \{0, 1\}$ $q_0 = q_0$ $F = \{[q_1, q_3], [q_2, q_3]\}$

Minimization of Automata

Step 1:- Remove the unreachable state.

Step 2:- Remove dead state.

Step 3:- Identify the repetation & eliminate it by renaming.

Step 4:- Split the set of states into two set such as final set & non-final set.

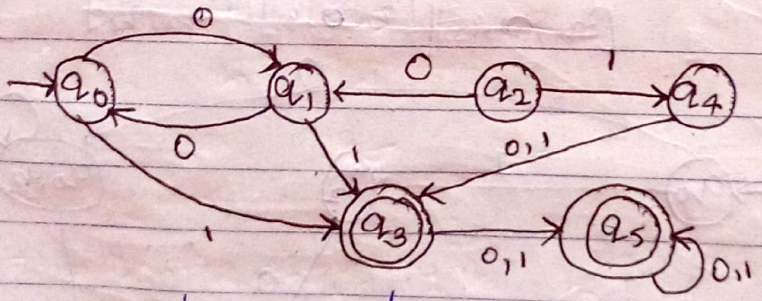
$\pi_0 = Q$ $\quad \pi = Q_1 \quad Q_2$

final $\quad$ non final

Step 5:- Find out the next state from each state on all the inputs and split it based on transition.

Step 6:- Repeat the previous steps until $\pi_k = \pi_{k+1}$

Ex



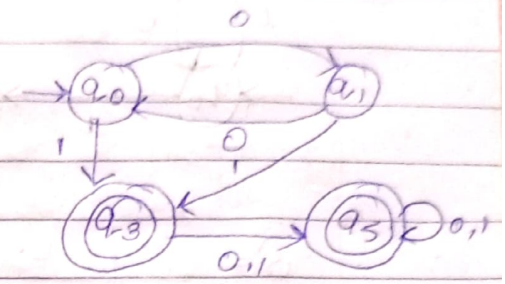
$\delta =$

	0	1
$\rightarrow q_0$	q_1	q_3
q_1	q_0	q_3
q_2	q_1	q_4
(q_3)	q_5	q_5
q_4	q_3	q_3
(q_5)	q_5	q_5

① Let us remove q_2, q_4 which are unreachable state.

$\delta =$

	0	1
$\rightarrow q_0$	q_1	q_3
q_1	q_0	q_3
(q_3)	q_5	q_5
(q_5)	q_5	q_5

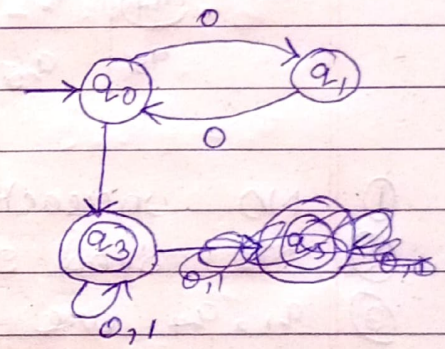


② No dead state

③ q_3, q_5 are same. So, we remain $q_5 = q_3$

$\delta =$

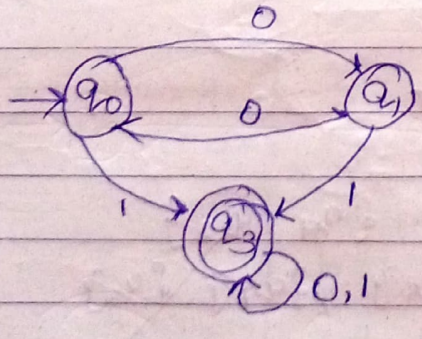
	0	1
$\rightarrow q_0$	q_1	q_3
q_1	q_0	q_3
(q_3)	q_3	q_3



④ $\pi_0 = \{q_0, q_1, q_3\}$

⑤ $\pi_1 \Rightarrow \{Q_1 = \{q_0, q_1\}, \quad Q_2 = \{q_3\}\}$

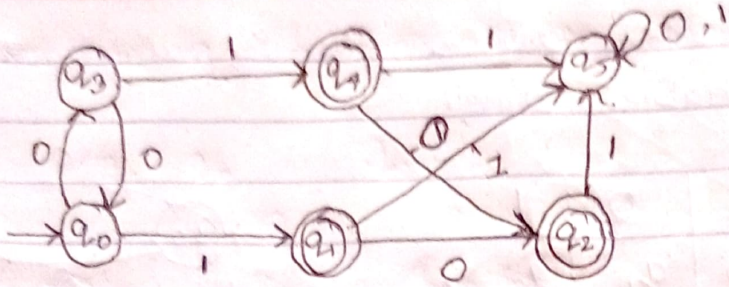
⑥ $\pi_2 = \{q_0\}\{q_1\}\{q_3\}$



$Q = \{q_0, q_1, q_3\} \quad q_0 = q_0 \quad F = \{q_3\}$

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

Q.



δ =	0	1
$\rightarrow q_0$	q_3	q_1
(q_1)	q_2	q_5
(q_2)	-	q_5
q_3	q_0	q_4
(q_4)	q_2	q_5
q_5	q_5	q_5

① NO unreachable state

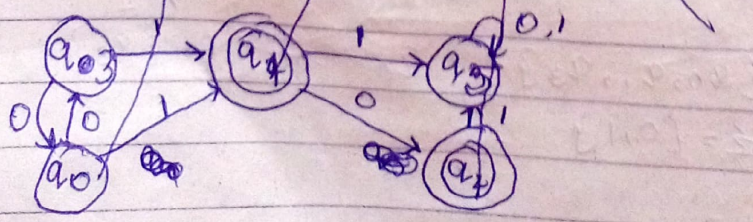
② NO dead state

③ q_1, q_4 are same. so, we remain $q_1 = q_4$

δ =	0	1
q_0	q_3	q_4
q_4	q_2	q_5
q_2	-	q_5
q_3	q_0	q_4
q_5	q_5	q_5

④ $\pi_0 = \{q_0, q_3, q_5\}$ $\{q_4, q_2\}$

$= \{q_0\}, \{q_3\}$ $\{q_5\}$ $\{q_4\}$ $\{q_2\}$

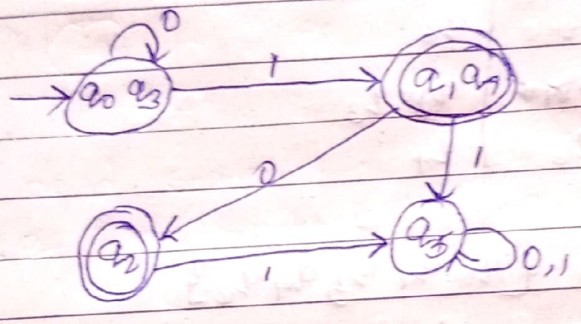


④ $Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$

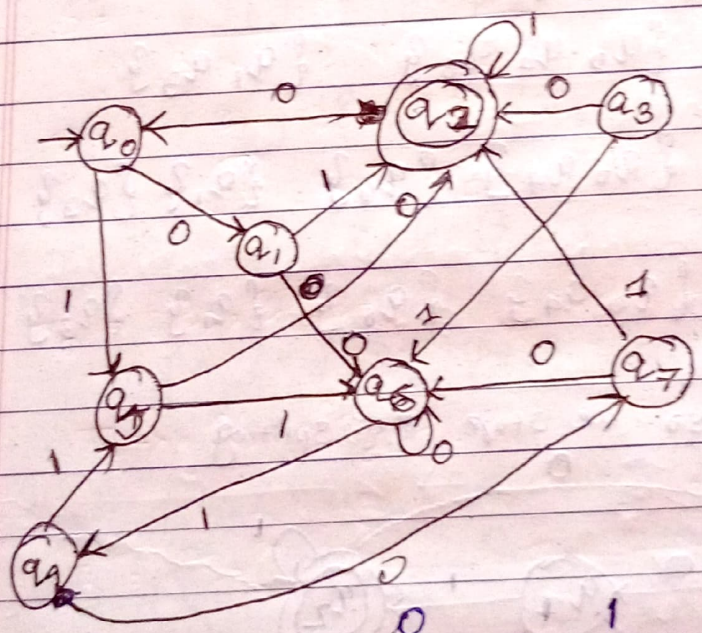
$\pi_0 = \{q_1, q_2, q_4\}\ \{q_0, q_3, q_5\}$

$\pi_1 = \{q_1, q_4\}\ \{q_0, q_3, q_5\}\ , \{q_2\}$

$\pi_2 = \{q_1, q_4\}\ \{q_2\}\ \{q_0, q_3\}\ \{q_5\}$



Q.



$\delta =$		0	1
$\rightarrow$	q_0	q_1	q_5
	q_1	q_6	q_2 -
	(q_2)	q_0	q_2
	q_3	q_2	q_6 -
	q_4	q_5	q_7
	q_5	q_2	q_6 -
	q_6	q_6	q_4
	q_7	q_6	q_2 -

① Remove q_3 because it is unreachable.

②

③ $q_1 = q_7$ $q_3 = q_5$

	0	1
→ q_0	q_1	q_3
q_1	q_6	q_2
(q_2)	q_0	q_2
q_3	q_2	q_6
q_4	q_1	q_3
q_6	q_6	q_4

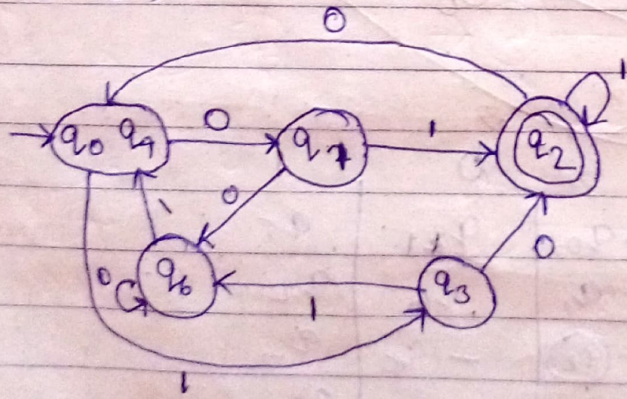
$\pi_0 = \{q_2\}\ \{q_0 q_1 q_3 q_4 q_6\}$

$\pi_1 = \{q_2\}\ \{q_0 q_4 q_6\}\ \{q_1 q_3\}$

$\pi_2 = \{q_2\}\ \{q_0 q_4\}\ \{q_6\}\ \{q_1\}\ \{q_3\}$

$\pi_3 = \{q_2\}\ \{q_0 q_4\}\ \{q_6\}\ \{q_1\}\ \{q_3\}$

$\pi_2 = \pi_3$, so we stop splitting



Q.

	0	1
$\rightarrow q_0$	q_0	q_3
q_1	q_2	q_5
q_2	q_3	q_4
q_3	q_0	q_5
q_4	q_0	q_6
q_5	q_1	q_4
(q_6)	q_1	q_3

$\pi_0 = \{q_6\} \{q_0\ q_1\ q_2\ q_3\ q_4\ q_5\}$

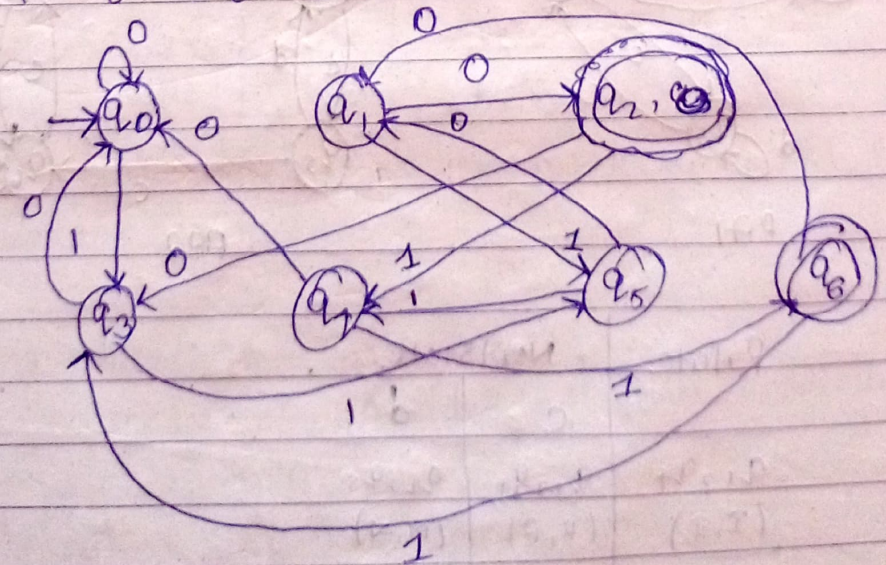
$\pi_1 = \{q_6\} \{q_0\ q_1\ q_2\ q_3\ q_5\} \{q_4\}$

$\pi_2 = \{q_6\} \{q_4\} \{q_0\ q_1\ q_3\} \{q_2\ q_5\}$

$\pi_3 = \{q_6\} \{q_4\} \{q_0\} \{q_1, q_3\} \{q_2, q_5\}$

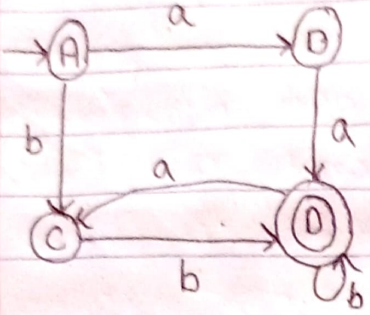
~~$\pi_4 = \{q_6\} \{q_4\} \{q_0\} \{q_2\} \{q_5\} \{q_1\} \{q_3\}$~~

$\pi_4 = \{q_6\} \{q_4\} \{q_0\} \{q_1\} \{q_3\} \{q_2\} \{q_5\}$

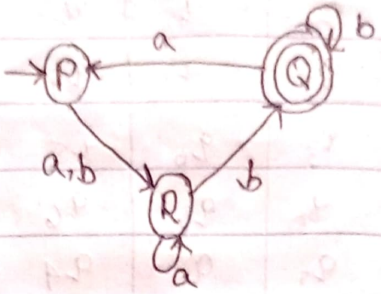


Equivalence of two Finite Automata.

Q.



FA1

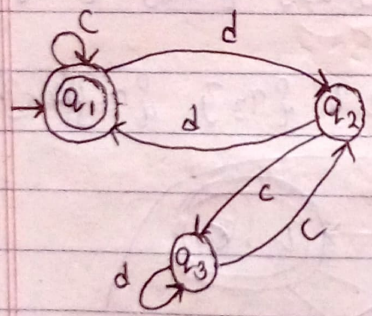


FA2

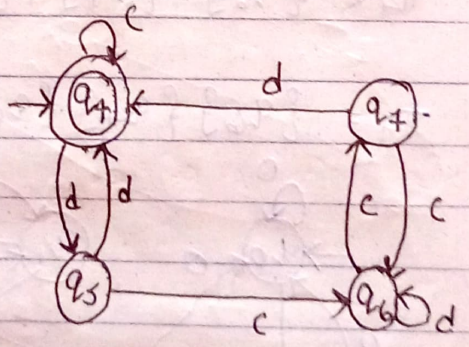
pstate	Nextstate	
	a	b
A, P	B, R	C, R
(I,I)	(N,N)	(N,N)
B, R	D, R	—, Q
(N,N)	(F,N)	(-,N)

Stop as we got unmatched pair. So, FA1 is not equivalent to FA2.

Q.



FA1

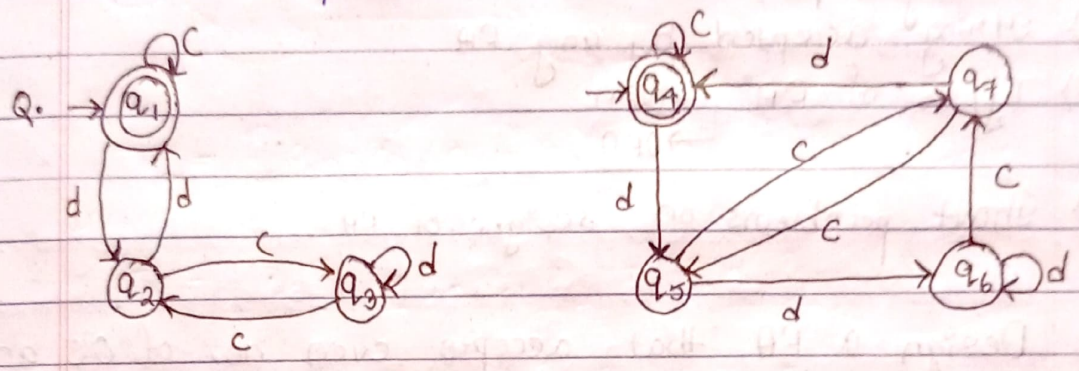


FA2

Pstate	Nextstate	
	c	d
q_1, q_4	q_1, q_4	q_2, q_5
(I,I)	(F,F)	(N,N)
q_2, q_5	q_3, q_6	q_1, q_4
(N,N)	(N,N)	(F,F)

q_3, q_6 (N,N)	q_2, q_7 (N,N)	q_3, q_6 (N,N)
q_2, q_7 (N,N)	q_3, q_6 (N,N)	q_1, q_4 (F,F)

No more pairs to be explored. All the pairs obtained in the table are matched pair. Hence, both FA1 and FA2 are equivalent.



pstates	Next state	
	c	d
q_1, q_4 (I,I)	q_1, q_4 (F,F)	q_2, q_5 (N,N)
q_2, q_5 (N,N)	q_3, q_7 (N,N)	q_1, q_6 (F,N)

Stops as we got unmatched pair. So, FA1 is not equivalent to FA2.

Long Questions

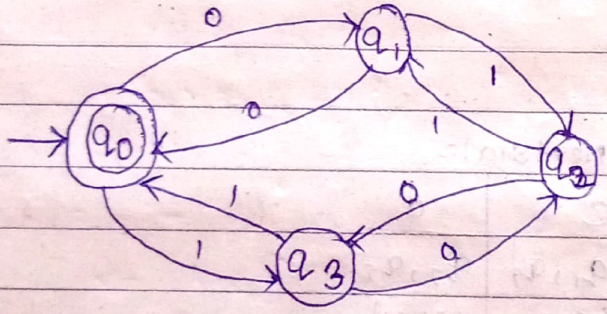
1) Basic model of FA (block diagram, 5 points - describe)
2) Design any DFA
3) Eliminating ε from NFA
4) NFA to DFA Conversion
5) Minimization of Automata

6) Equivalence of two FA

Short Questions

1) Define a FA (defination, 5-tupies)
2) Diff^n b/w NFA and DFA
3) Diff^n b/w Kleen's closure and positive closure.
4) String acceptance
5) String operation
6) String accepted by any FA
7) Types of FA → NFA, → DFA
8) Short problems on design of FA

Q) Design a FA that accepts even no. of 0's and even no. of 1's over $\Sigma = \{0,1\}$



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

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

$\delta =$

	0	1
q_0	q_1	q_2
q_1	q_0	q_3
q_2	q_3	q_0
q_3	q_2	q_1

Unit - II

09/04/22

Regular Set & Regular Expression

Regular set:- A set that depends upon the variable or symbols and denotes the values of regular expression is called as regular set.

It is denoted by the 'R'.

Ex:- {a,b,ab} {1,11,111} {1,a,aa,aaa,----}

Regular set has several properties based on the string operations:-

i) Class of regular languages or regular sets are closed under union operation.

ii) Concatenation of two regular set is regular.

iii) This closed under closure operation.

iv) The complement of regular set is also regular.

Regular Expressions:- It is the representation of string in an algebaic fashion.

It is a sequence of pattern that describes a string.

It is similar to an arithematic expression.

Any language accepted by FA is represented using regular expression.

Ex:-	Set	Expression
	{10}	R = 10
	{0,1}	R = 0+1
	{01,10}	R = 01+10

$\{1, ab\}$	$R = 1 + ab$
$\{abb, ab, ba\}$	$R = abb + ab + ba$
$\{1, 0, 00, 000 \cdots 0^n\}$	$R = 0^*$ (Kleen's closure)
$\{a, aa, aaa, a \cdots a^n\}$	$R = aa^*$ or a^+

1Q. Find out the regular expression for any language L = set of all strings of 0's and 1's ending with 00.

$$R = (0+1)^* 00$$

2Q. L = set of all strings of 0's and 1's begining with 01.

$$R = 01(0+1)^*$$

3Q. L = set of all strings over a's and b's of any length.

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

4Q. L = set of all strings over 0 and 1 and starting with the 0 and ending with 1.

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

5Q. $L = \{\wedge, 11, 1111, 111111, \cdots\}$

$$R = (11)^*$$

6Q. L = set of strings starting with ab and ending ab.

$R = ab(a+b)^* ab$

7Q. L = set of strings consisting of even no. of 0's over the $\Sigma = \{0\}$

$R = (00)^*$

8Q. L = even no. of a's followed by even no. of b's.

$R = (aa)^* (bb)^*$

9Q. L = even no. of b's followed by odd no. of as

$R = (bb)^* a (aa^*)$

10Q. L = a followed by any no. of b's.

$R = (ab^*)^*$

11Q. $L = \{1^{2n+1} / n > 0\}$

$R = 1(11)^+$

12Q. $L = \{w \in \{a,b\} / w$ has only one a$\}$

$L = \{a, ab, [illegible], abb, ba \cdots\}$

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

13Q. $L = \{aa, a^5, a^8, a^{11}, \cdots\}$

$R = aa(aaa)^*$

Identities of Regular Expression

1) $\emptyset + R = R$
2) $\emptyset R = R\emptyset = \emptyset$
3) $\wedge R = R\wedge = R$
4) $\wedge^* = \wedge\emptyset^* = \wedge$
5) $R + R = R$
6) $R^* R^* = R$
7) $RR^* = R^*R$
8) $(R^*)^* = R^*$
9) $\wedge + RR^* = R^* = \wedge + R^*R$
10) $(PQ)^*P = P(QP)^*$
11) $(P+Q)^* = (P^*Q^*)^* = (P^* + Q^*)^*$
12) $(P+Q)R = PR + QR$
$R(P+Q) = RP + RQ$

1Q. The regular expression representing the set of string in which every zero is immediately followed by atleast two 1s is equivalent to another regular expression

$$R = \wedge + 1^*(011)^*(1^*(011)^*)^*$$

$$R_1 = (1 + 011)^*$$

$$R = \wedge + \underbrace{1^*(011)^*}_{R}(1^*(011)^*)^*$$
$$= \wedge + RR^* = R^* \quad [\text{using I9}]$$
$$= (1^*(011)^*)^* = (1^* + (011)^*)^* \quad [\text{using I11}]$$
$$= (1 + (011))^* = R_1 \quad [\text{using I11}]$$

2Q. prove that $(1+00^*1) + (1+00^*1)(0+10^*1)^*(0+10^*1) = 0^*1(0+10^*1)^*$

LHS $= (1+00^*1) + (1+00^*1)(0+10^*1)^*(0+10^*1)$
$= (1+00^*1)[1 + (0+10^*1)^*(0+10^*1)]$
$= (1+00^*1)(0+10^*1)^*$
$= [00^* + \wedge]1(0+10^*1)^*$

$= 0^*1(0+10^*1)^* = RHS$

Hence proved//

3Q. prove the following identity:

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

~~RHS = $(a+ab+ba)^* = (a^*+ab)^*+(ba)^*$~~

~~LHS = $(a^*ab+ba^*)^* a^*$~~

~~$= ((a^*ab)^*(ba^*)^*)^* a^*$~~

~~$= [(a^*+ab)^*(b^*+$~~

Let $a_1 = (a^*ab+ba^*)^*$ $a_2 = a^*$

Now, let ~~$a^* = 8$~~

we get, ~~LHS =~~ $(ab+ba)^*$ — ①

Now, let $a_1 =$

~~RHS = $a(1+Ab+Ab)^* = (a(A+b))^* =$~~

~~$= [a(1+bb^*)]^*$~~

$LHS = (a^*ab+ba^*)^* a^* = [b(a^*a+a^*)^*]^* a^*$

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

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

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

ARDEM'S Theorem

Let P and Q be two regular expression over Σ. If P doesn't contain $\wedge$ then the following equ:

$$R = Q + RP$$

has a unique solution $R = QP^*$

Assume, $R = QP^*$

Substitute the value in $R = Q + RP$, we get

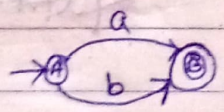
$$R = Q + (QP^*)P$$
$$= Q + QP^*P$$
$$= Q(1 + P^*P) = QP^*$$

or

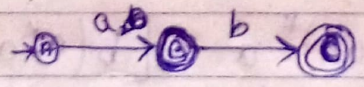
$$R = Q + (Q + RP)P$$
$$= Q + QP + RP^2 = Q[1 + P + P^2 + P^3] = QP^*$$

Regular expression to FA ?

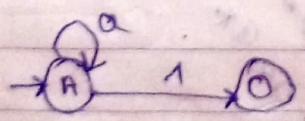
1) a+b



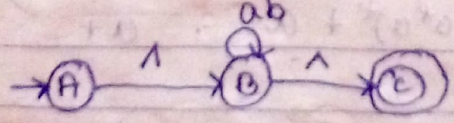
2) ab



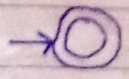
3) a^*



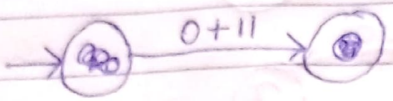
4) ~~ab~~$(a+b)^*$



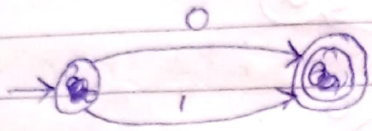
5) 1



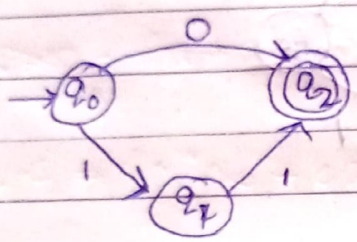
1Q. Design a FA for the regular expression
R = 0 + 11



Step 1



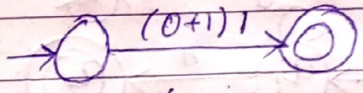
Step 2



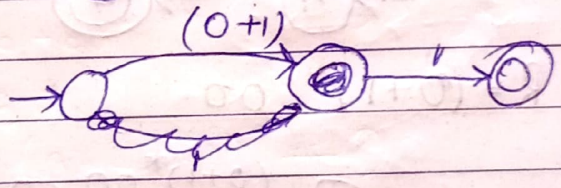
Step 3

2Q. R = (0+1)1

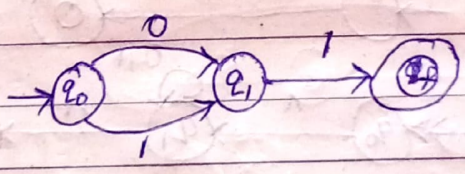
Step 1



Step 2

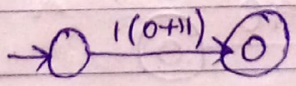


Step 3

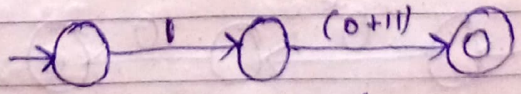


3Q. R = 1(0+11)

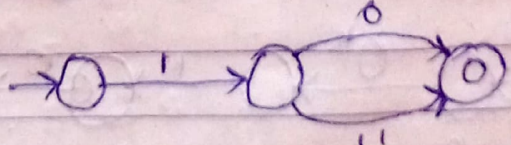
Step 1



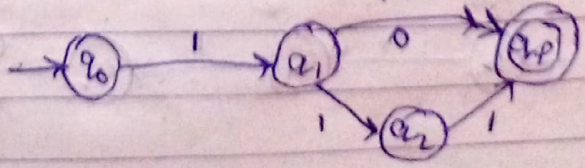
Step 2



Step 3

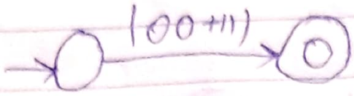


Step 4



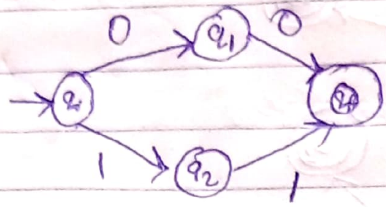
4Q. R = (00 + 11)

Step 1



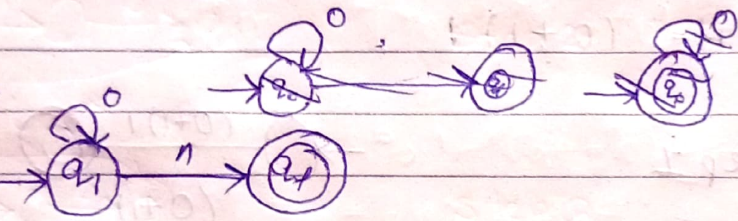
Step 2

Step 3

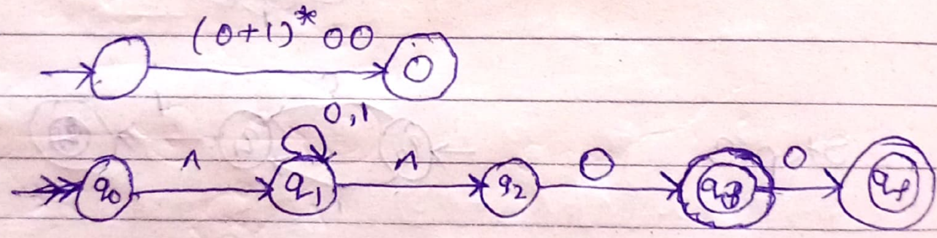


5Q. R = 0^*

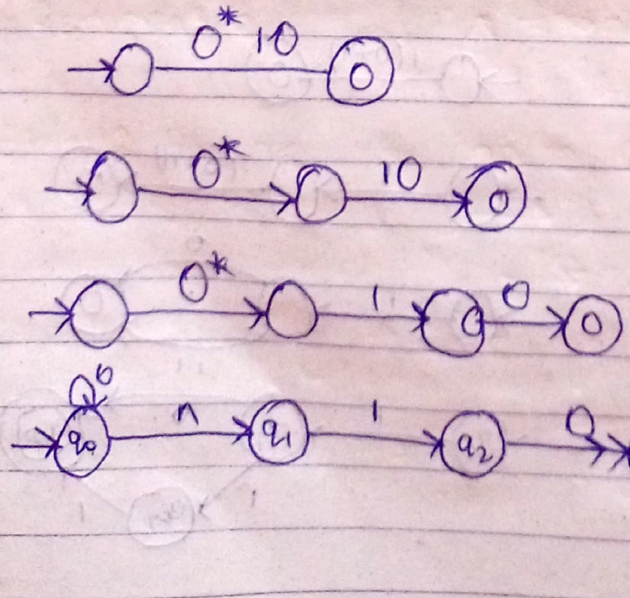
~~step 1~~



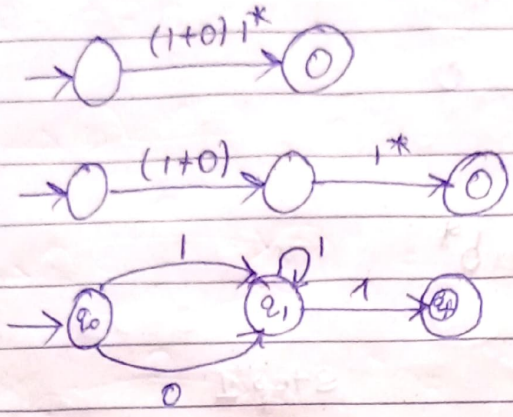
6Q. R = $(0+1)^*$ 00



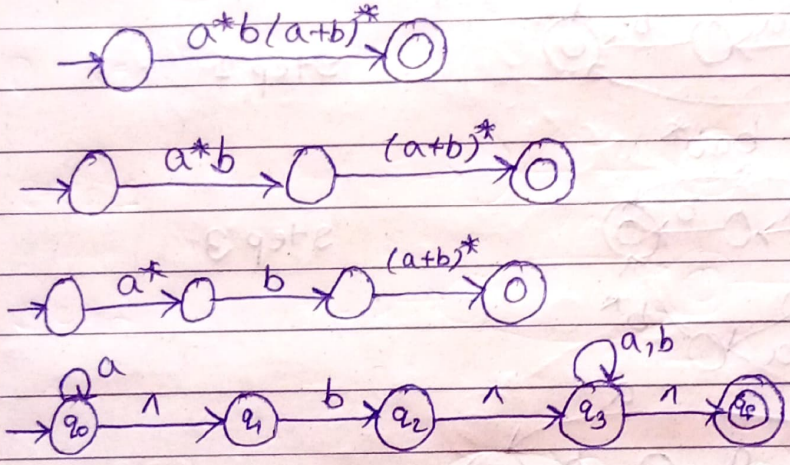
7Q. R = 0^* 1 0



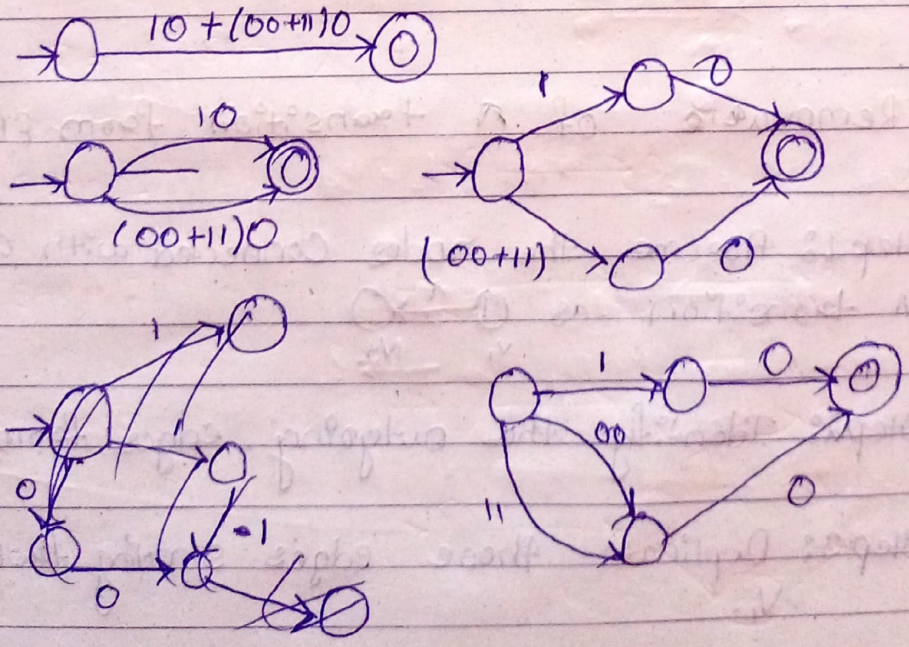
8Q. $R = (1+0)1^*$

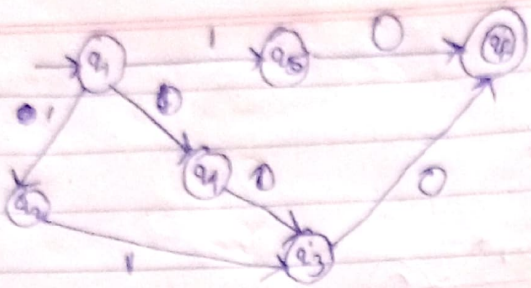


9Q. $R = a^*b(a+b)^*$

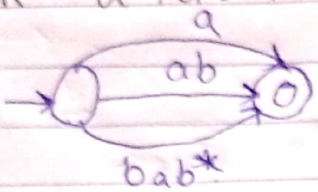


10Q. $R = 10 + (00+11)0$

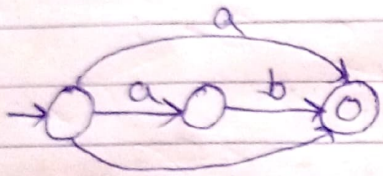




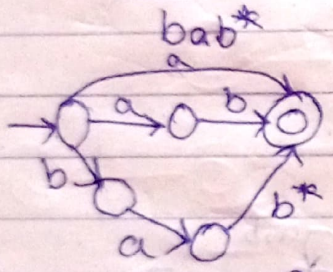
IIQ. $R = a + ab + bab^*$



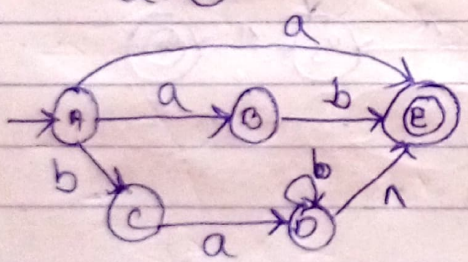
Step 1



Step 2



Step 3



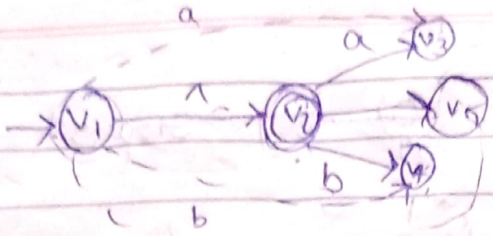
Step 4

Removable of ^ transition from FA

Step 1: Assume the nodes connected with a ^ transition as $V_1 \xrightarrow{\wedge} V_2$

Step 2: Identify the outgoing edges from V_2.

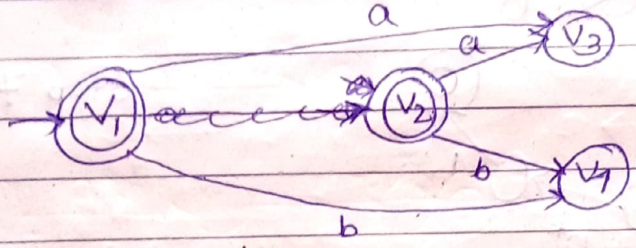
Step 3: Duplicate those edges starting from V_1.



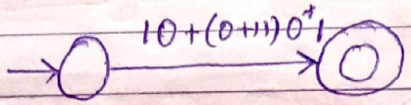
Step 4: If V_2 is final then make V_1 final.

Step 5: If V_1 is initial then make V_2 also initial.

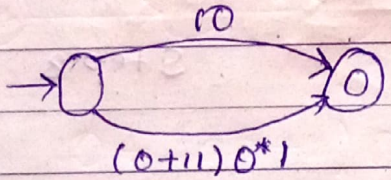
Step 6: Eliminate the $\wedge$ edges.



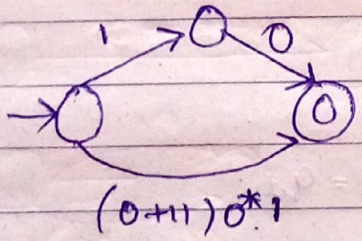
12 Q. $10 + (0+11)0^*1$



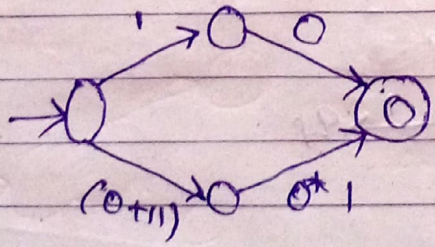
Step 1



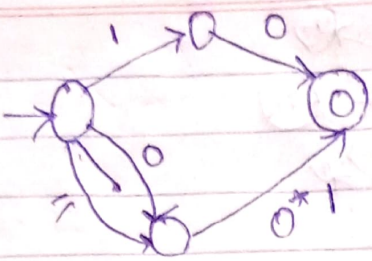
Step 2



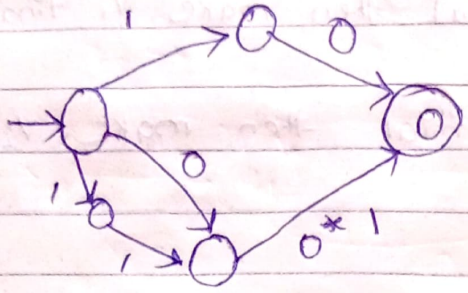
Step 3



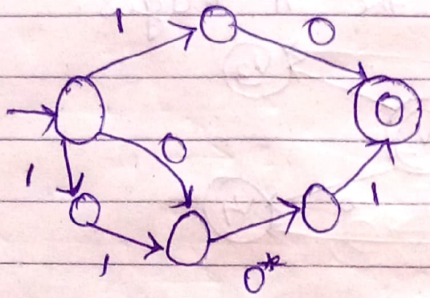
Step 4



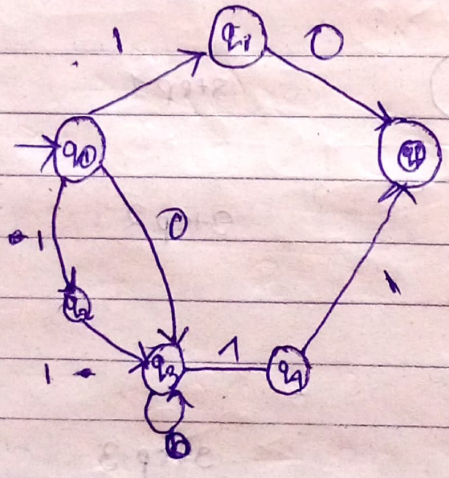
Step 5



Step 6



Step 7



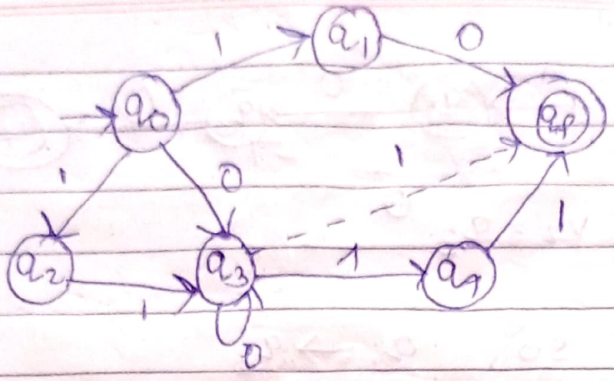
Step 8

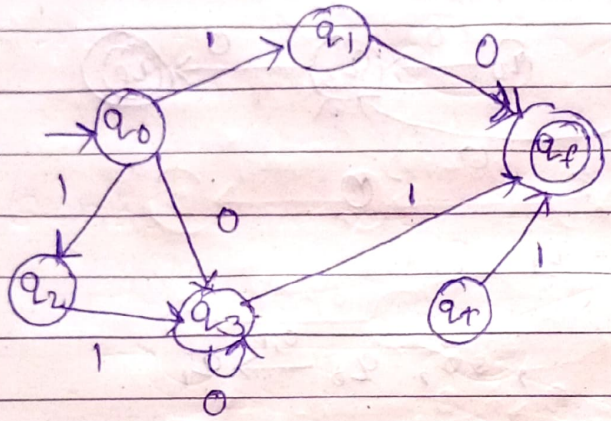
Let $V_1 = q_3$ and $V_2 = q_4$

$q_3 \xrightarrow{1} q_4$

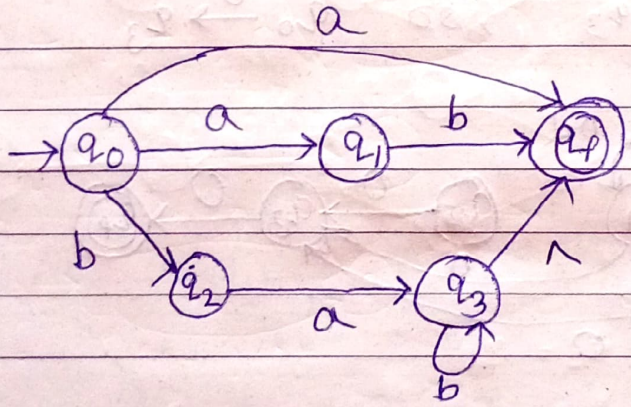
Next transition, $q_4 \xrightarrow{1} q_f$

So, $q_3 \xrightarrow{1} q_f$

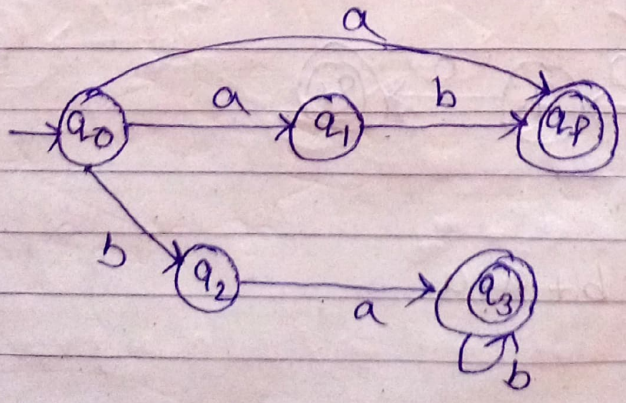




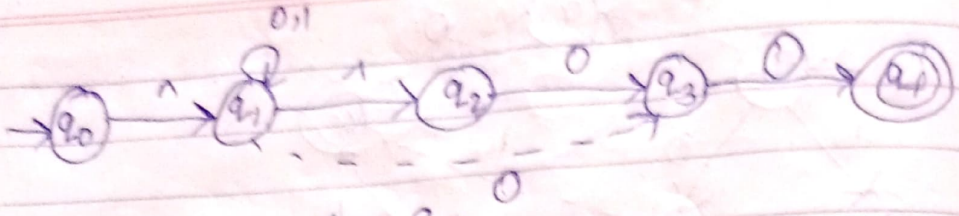
11Q



There is no outgoing edges of q_f and q_3 is not intial stage, so we make q_3 has final state.

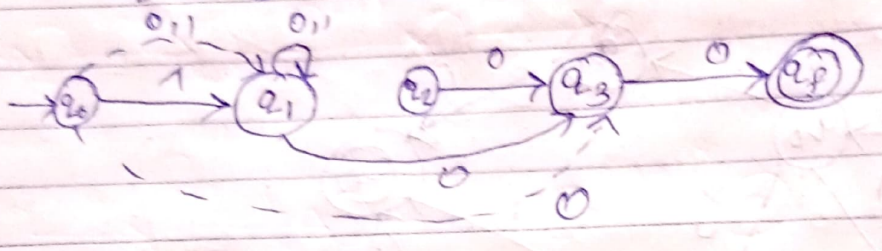


3



$V_1 = q_1$ $V_2 = q_3$

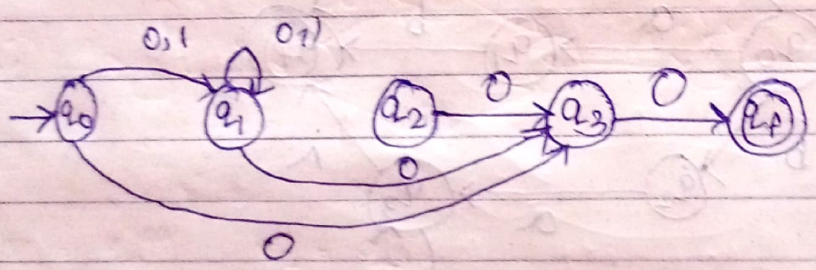
$q_2 \xrightarrow{0} q_3$ so, $q_1 \rightarrow q_3$



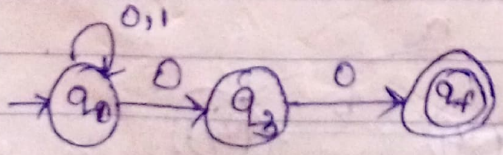
$V_1 = q_0$ $V_2 = q_1$

$q_0 \xrightarrow{0,1} q_1$, so, $q_0 \xrightarrow{0,1} q_1$

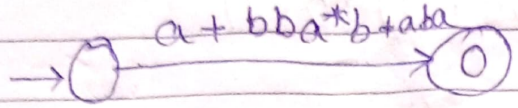
$q_1 \xrightarrow{0} q_3$, so $q_0 \xrightarrow{0} q_3$



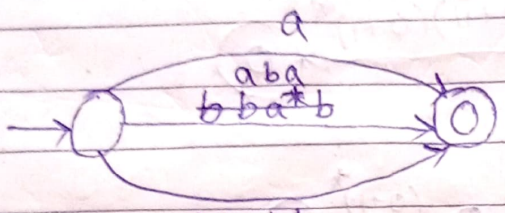
Here, q_0 and q_1 are same, so, we take q_0 and q_2 is ~~un~~ not reachable state, so q_2 is removed.



$a + bb\,a^* b + aba$

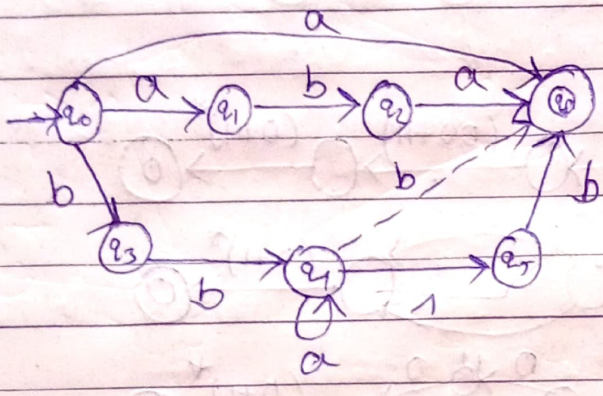


Step 1



Step 2

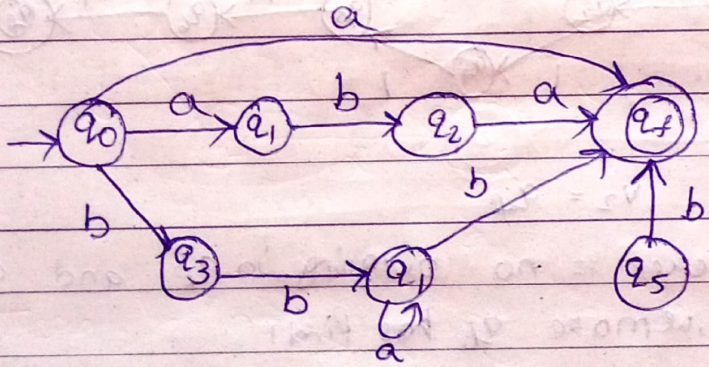
bba*b



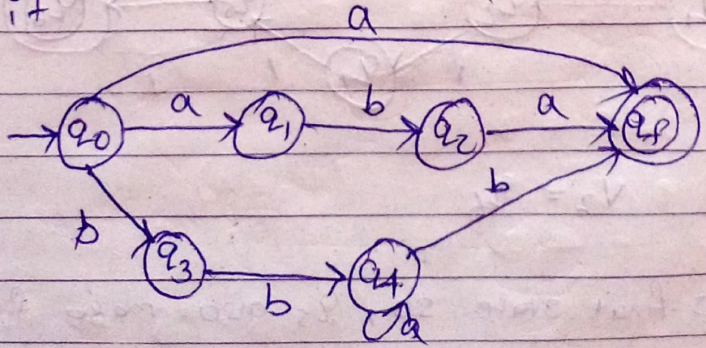
Step 3

$V_1 = q_4$ $V_2 = q_5$

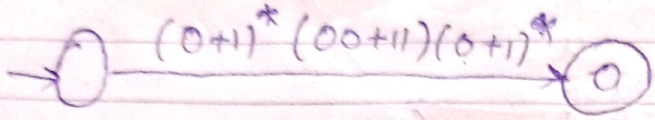
$q_5 \xrightarrow{b} q_f$ so $q_4 \xrightarrow{b} q_f$

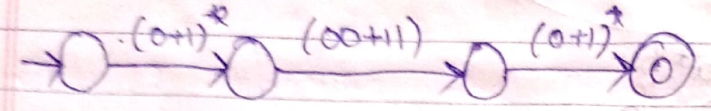


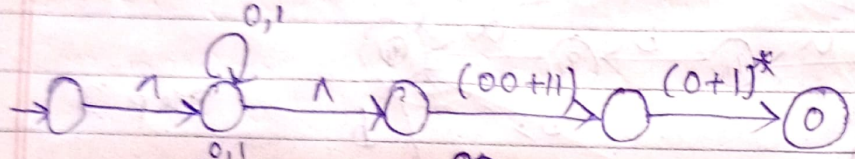
Here q_5 is unreachable state, so we removed it

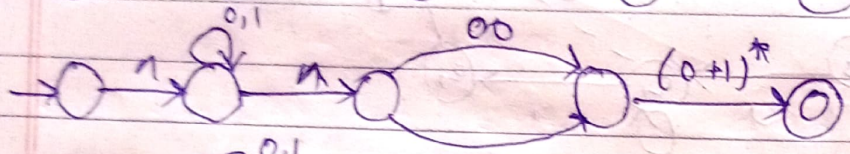


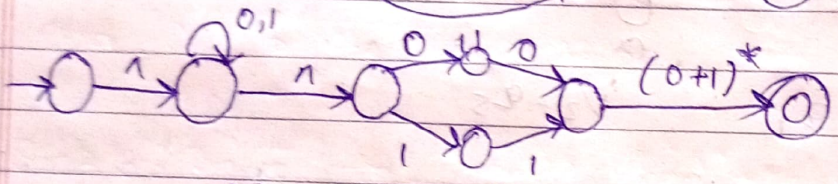
14Q. $R = (0+1)^* (00+11)(0+1)^*$

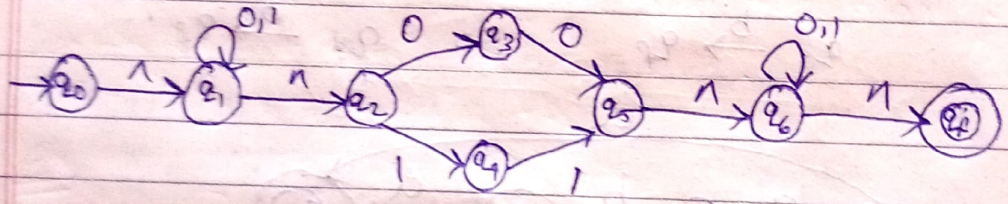






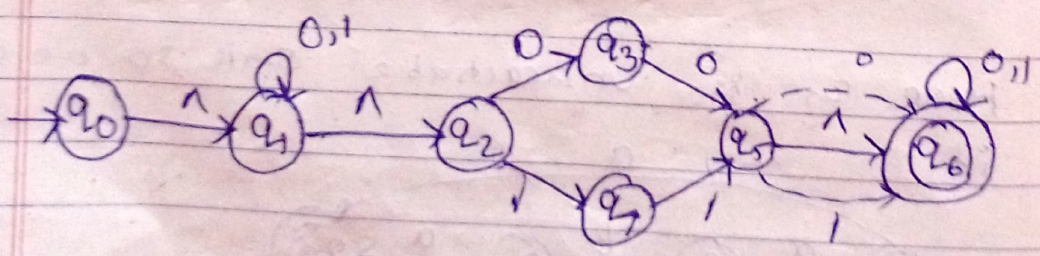






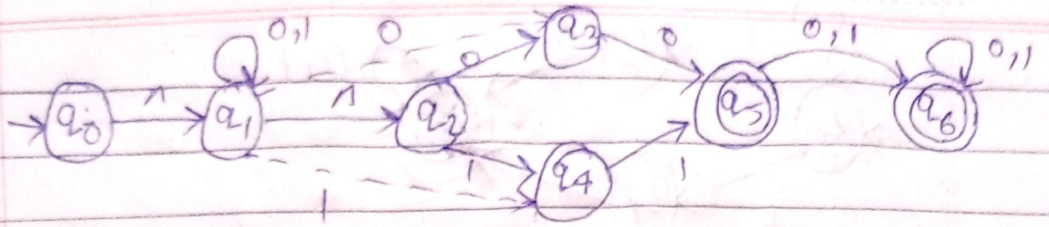
$V_1 = q_6$ $V_2 = q_f$

Here, there is no outgoing in q_f and q_6 is not initial, so, we make q_6 has final



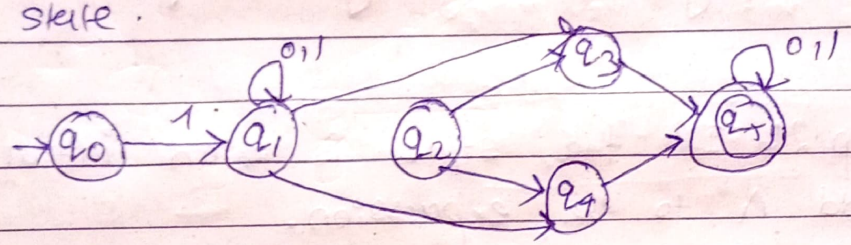
$V_1 = q_5$ $V_2 = q_6$

Here, q_6 is final state so, q_5 also make final and here is 0,1 are outgoing of q_6

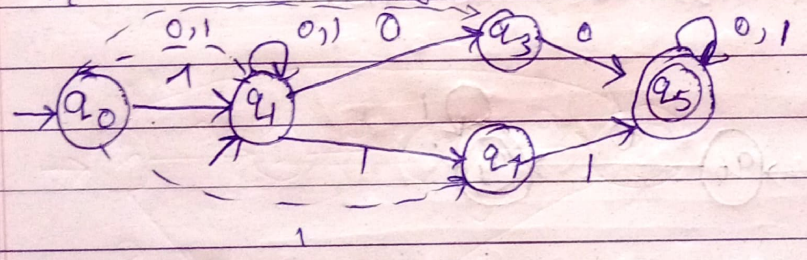


$V_1 = q_1$ $V_2 = q_2$

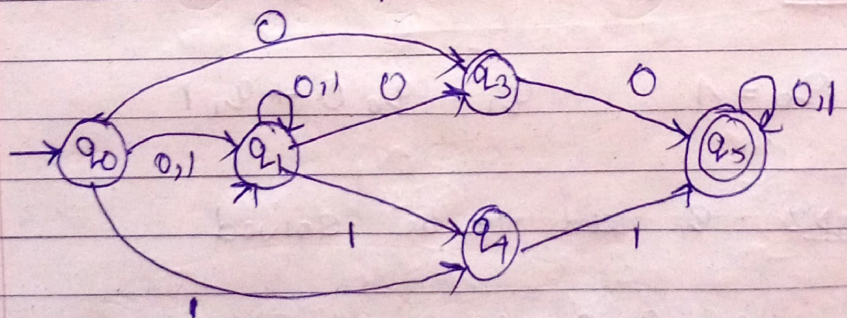
Here, we combine q_5 and q_6 has q_5 because Both have same ~~tra~~ transition state. ~~consider~~ No intial & final state, so we ~~~~ outgoing state.



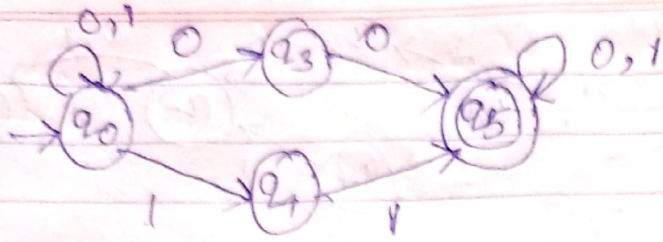
Remove unreachable state from FA. Here, q_2 is unreachable state



$V_1 = q_0$ $V_2 = q_1$



Here, q_0 and q_1 are initial state. Both have same ~~tra~~ transition state. so, we take q_0 and q_1 has q_0.



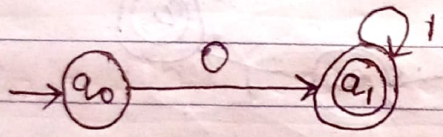
FA to Regular Expression

Step 1: Generate the expression for each and every states of the finite Automata by considering the incoming edges.

Step 2: If the state is an initial state add $\wedge$ to the expression.

Step 3: Solve the equn for the final state which will be the answer.

Ex



Step 1 equn for each state

$q_0 = 1$ $\quad q_1 = q_0 0 + q_1 1$

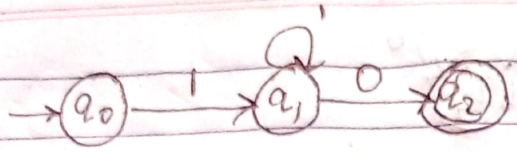
Step 2 q_1 need to be solved

$$q_1 = q_0 0 + q_1 1$$
$$= 1 0 + q_1 1 \quad (q_0 = 1)$$
$$= 0 + q_1 1$$

$\boxed{q_1 = 01^*}$ $\quad (R = Q + PR \rightarrow R = QP^*)$

$\boxed{RE = 01^*}$

1 Q.



Step 1

$q_0 = \Lambda$ $\quad q_1 = q_0 1 + q_1 1$

$q_2 = q_1 0$

Step 2

q_2 need to be solved

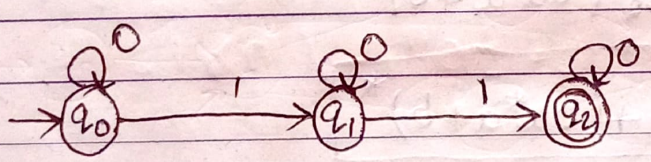
$q_2 = q_1 0$

$q_1 = q_0 1 + q_1 1 = \Lambda 1 + q_1 1 = 1 + q_1 1$

$\Rightarrow \boxed{q_1 = 11^*}$

$\Rightarrow \boxed{q_2 = 11^*0}$ $\qquad \boxed{RE = 11^*0}$

2Q.



Step 1

$q_0 = \overset{\Lambda +}{1} q_0 0$ $\quad q_1 = q_0 1 + q_1 0$

$q_2 = q_1 1 + q_2 0$

Step 2 q_2 need to be solved

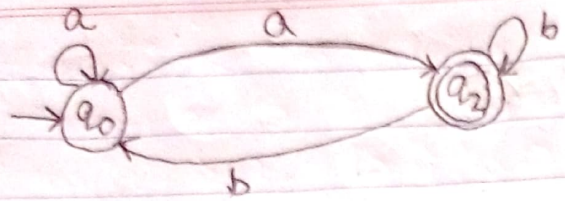
~~$q_1 = \Lambda + q_0 0) 1$~~ $\quad q_0 = \Lambda + q_0 0 \Rightarrow \boxed{q_0 = \Lambda 0^* = 0^*}$

$q_1 = 0^* 1 + q_1 0 \Rightarrow \boxed{q_1 = 0^* 1 0^*}$

$q_2 = 0^* 1 0^* 1 + q_2 0 \Rightarrow \boxed{q_2 = 0^* 1 0^* 1 0^*}$

$\boxed{RE = 0^* 1 0^* 1 0^*}$

3Q.



Step 1

$q_0 = \Lambda + q_0a + q_2b \qquad q_2 = q_0a + q_2b$

Step 2 q_2 need to be solved

$q_0 = \Lambda + q_0a + q_2b$

$q_0 = (\Lambda + q_2b) + q_0a$

$\boxed{q_0 = (\Lambda + q_2b)\,a^*}$

$q_2 = q_0a + q_2b = (\Lambda + q_2b)a^*a + q_2b \qquad (q_0 = (\Lambda + q_2b)a^*)$

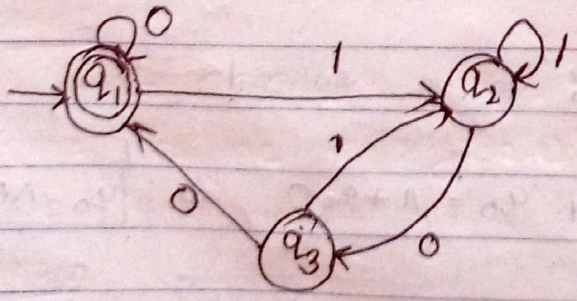
$= \Lambda a^*a + q_2ba^*a + q_2b \qquad (\Lambda = 1, \emptyset = 0)$

$= a^*a + q_2(ba^*a + b)$

$\boxed{q_2 = a^*a + (ba^*a + b)^*}$

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

4Q.



Step 1

$q_1 = \Lambda + q_10 + q_30 \qquad q_2 = q_21 + q_21 + q_31$

$q_3 = q_20$

Step 2 q_1 need to be solved

$q_2 = q_1 1 + q_2 1 + q_3 1$

$= (q_1 1 + q_2 1) + q_2 1$ $= q_1 1 + q_2 1 + q_2 01$

$= (q_1 1 1^*) + q_2 1$ $= q_1 1 + q_2 [1 + 01]$

$= q_1 1^* 1^*$ $\boxed{q_2 = q_1 1 (1+01)^*}$

~~$q_3 = q_1 1 1^* 1^* 0$~~

$q_1 = \Lambda + q_1 0 + (q_1 11^* 1^* 0)$

$= \Lambda + q_1 0 + q_1 11^* 1^* 0$

$= \Lambda + q_1 (0 + 11^* 1^* 0)$

$=$

$q_3 = q_1 1 (1+01)^* 0$

$q_1 = \Lambda + q_1 0 +$ ~~$q_1 1 (1+01)^* 0$~~ $q_1 1 (1+01)^* 00$

$= \Lambda + q_1 (0 + 1 (1+01)^* 00)$

$= \Lambda [0 + 1 (1+01)^* 00]^*$

$RE = (0 + 1 (1+01)^* 00)^*$

5m Pumping lemma

It is used to prove a set / ~~Regular~~ Language is not regular.

Step 1: Assume L is regular and ~~end~~ be ~~with~~ the no. of states in the FA that accepts L.

Step 2: Select a string 'w' = xyz such that length of w is greater than equal to n, i.e., len(w) ≥ n, |xy| ≤ n & |y| > 0

Step 3: Find out the suitable interger i such that $xy^i z$ doesn't belong to (&) L.

It Contradictes the assumation. Hence, L is not regular.

Defn:- It is used to prove a set/Language is not regular but pumping a substring from a given string and matching the pattern which won't be similar.

Q. prove that $L = \{0^n 1^n / n \geq 1\}$ is not regular.

A:- Step 1

Assume the L is regular.

Step 2

w = 000111 |w| - 6 ≥ n

and assume |xy| ≤ n |y| > 0

w - xyz - 000111

Case 1: y consists of only 0's = 0^k

Case 2: y consists of only 1's = 1^k

case 3: y consists of both 0's & 1's = $0^k 1^m$

step 3

assume $i = 2$,

case 1 $y = 0^k$

$xy^2z = xyyz = xyz\,y$

$xyz = 0^n 1^n = 0^{n-k} 0^k 1^n$

$xy^2z = 0^{n-k} 0^k 0^k 1^n = 0^{n-k+2k} 1^n$

$= 0^{n+k} 1^n \Rightarrow n+k \neq n$

$\therefore xy^2z \notin L$

case 2 $y = 1^j$

$xy^2z = xyyz = 0^n 1^j 1^j 1^{n-j}$

$xyz = 0^n 1^j 1^{n-j}$

$xy^2z = 0^n 1^j 1^j 1^{n-j} = 0^n 1^{n-j+2j} = 0^n 1^{n+j}$

$\Rightarrow n \neq n+j$

$\therefore xy^2z \notin L$

case 3 $y = 0^k 1^m$

$xy^2z = xyyz$

$xyz = 0^{n-k} 0^k 1^m 1^{n-m}$

$xy^2z = 0^{n-k} 0^k 0^k 1^m 1^m 1^{n-m}$

$= 0^{n-k+2k} 1^{n-m+2m} = 0^{n+k} 1^{n+m}$

$\Rightarrow n+k \neq n+m \quad (k \neq m)$

$\therefore xy^2z \notin L$

so, L is not regular. proved

2Q. prove that $L = \{a^n / n$ is a prime no.$\}$ is not regular.

step 1

assume the L is regular

step 2

$\omega = a^p$ / p = prime no.

$|\omega| = |xyz| = p, \quad |xy| \leq n, \quad |y| > 0$

step 3 assume $i = p+1$

$|xy^iz| = |xyz| + |y^{i-1}|$

$= p+i-1 = p+p+1-1 = 2p$

2p is not a prime number. so, $xy^iz \notin L$

$\therefore$ L is not regular. proved//

3Q. prove that $L = \{a^{n^2} / n \geq 1\}$ is not regular.

step 1

assume the L is regular.

step 2

$w = a^{n^2}$ $\quad |w| = n^2, \; |xy| \leq n, \; |y| > 0$

step 3

assume $i = n+1$

$|xy^iz| = |xyz| + |y^{i-1}|$

$= n^2 + i - 1 \quad = n^2 + n + 1 - 1$

$= n^2 + n$

n^2+n is not a perfect square, so,

$xy^iz \notin L$.

$\therefore$ L is not regular. (proved)

4Q. prove that $L = \{0^{2n} / n \geq 1\}$ is not regular

step 1 assume the L is regular

step 2

$w = 0^{2n}$

$|w| = |xyz| = 2n, \quad |xy| \leq n, \quad |y| > 0$

step 3 assume $i = 2$

$|xy^iz| = |xyz| + |y^{i-1}|$

$= 2n + 2 - 1$

$= 2n + 1$

$2n+n$ is not in even number. so, $xy^iz \notin L$

$\therefore$ L is not regular. (proved).

5Q. prove that $L = \{ww \mid w \in \{a,b\}^*\}$ is not regular.

Step 1

assume L is regular.

Step 2

$w = a^n b$ $\quad ww = a^n b a^n b$

$|ww| = 2n+2 \geq n$, $|xy| \leq n$, $|y| > 0$

Step 3

assume $i = 2$

Case 1: y consists of a's only

$y = a^k$

~~$xy^2z = xyyz =$~~ $|xy^iz| = |xyz| + |y|$

$= 2n+2+$

~~$xyz = a$~~

~~$xyz = a^k a^{n-k} b a^k b = a^{n-k} a^k b a^{n-k} a^k b$~~

$= a^{n-}$

$|xy^iz| = |xyz| + |y| = 2n+2+k$

$= 2(n+1)+k \notin L$

case 2: y consists of b's only

$y = b^j$

$|xy^iz| = |xyz| + |y| = 2n+2+j$

$= 2(n+1)+j \notin L$

case 3: y consists of both a, b

$y = a^k b^j$

$|xy^i z| = |xyz| + |y| = 2(n+1) + k + j \notin L$

∴ L is not regular. (proved)

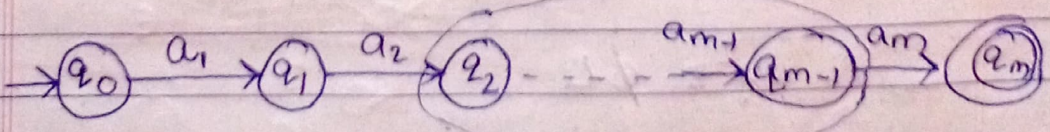
Pumping Lemma Theorem:
Let $M(Q, \Sigma, \delta, q_0, F)$ be a FA with 'n' no. of states, 'L' be the language accepted by M. and string 'w' belongs to L such that length of w i.e., $|w| \geq m$.

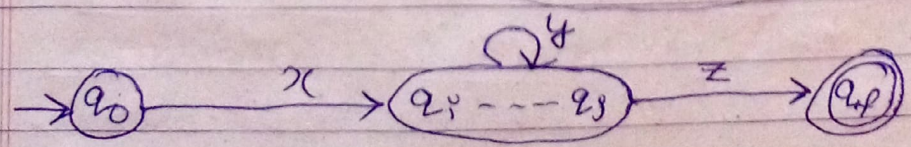
If $m \geq n$ then, there exist x, y, z such that $w = xyz$, $y \neq \wedge$ and $xy^i z \in L$ for each $i \geq 0$.

Proof Assume $w = a_1, a_2, \ldots a_m$, $m \geq n$

$\delta(q_0, a_1 a_2 \ldots a_i) = q_i$

There are only 'n' no. of distinct states that read 'm' no. of symbols that implies some states are combined.





$xyz \in L$, $xy^2z \in L$, $xy^iz \notin L$

Q. $L = \{a^n b^{2n} \mid n \geq 1\}$ is not Regular.

Step 1

assume L is regular

Step 2

$w = a^n b^{2n}$, $|xy| \leq n$, $|y| > 0$

$|w| = 3n \geq 0$

Step 3

case 1: y consists of only a's

$y = a^k$

case 2: y consists of only b's

$y = b^i$

case 3: y consists of both a's & b's

$y = a^k b^i$

Let us assume $i = 2$,

case 1

$xyz = a^{n-k} a^k b^{2n}$

$xy^2z = xyyz = a^{n-k} a^k a^k b^{2n}$

$= a^{n-k+2k} b^{2n} = a^{n+k} b^{2n}$ $k \neq 0$

$\therefore xy^2z \notin L$

case 2

$$xyz = a^n b^{n-j} b^j$$

$$xy^2z = xyyz = a^n b^j b^j b^{2n-j} = a^n b^{2n+j}, \; j \neq 0$$

$\therefore xy^2z \notin L$

case 3

$$xyz = a^{n-k} b^{2n-j} a^k b^j$$

$$xy^2z = xyyz = a^{n-k} a^k a^k b^j b^j b^{2n-j}$$
$$= a^{n+k} b^{2n+j}, \; k \neq j$$

$\therefore xy^2z \notin L$

$\therefore$ L is not regular (proved).

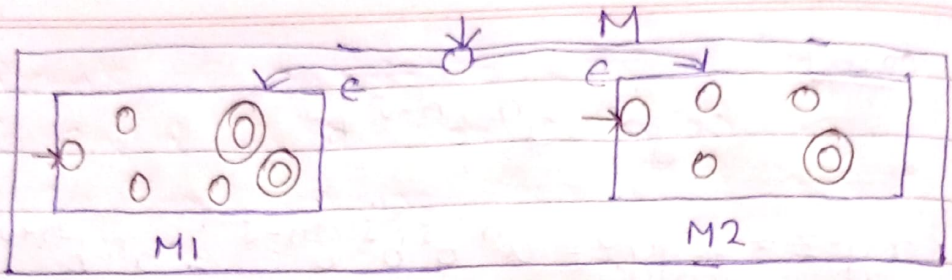
H.W Q. $L = \{a^m b^n / GCD(m,n) = 1\}$ is not regular.

Closure properties of Regular sets :-

→ union operation, concatenation, Kleen's closure, complement, set difference ~~and~~ intersection and Reverse

* Union closure :- class of regular languages or sets is closed under union operation. i.e., if L1 & L2 are two regular language then L1 ∪ L2 also regular.

Let M1 be a machine with 5 tuples (Q_1, Σ_1, δ_1, q_{0_1}, F_1) accepts L1 and M2 (Q_2, Σ_2, δ_2, q_{0_2}, F_2) accepts L2.



A machine M has been constructed that holds/accepts a language L consisting of L1 or L2.

M is represented by $M(Q, \Sigma, \delta, q_0, F)$ then

$Q = Q_1 \cup Q_2 \quad \Sigma = \Sigma_1 \cup \Sigma_2 \quad \delta = \delta_1 \cup \delta_2$

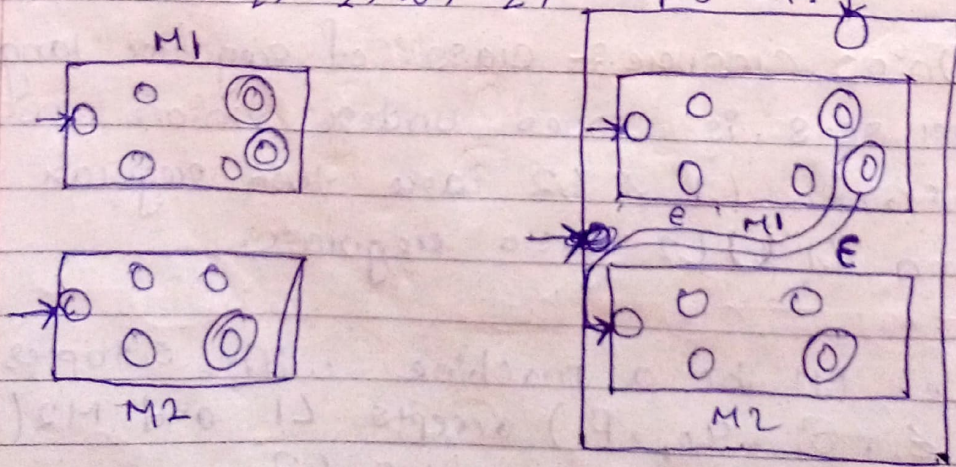
~~$q_0 = q_{0_1} \cup q_{0_2}$~~ $\quad F = F_1 \cup F_2$

$(q_0, \varepsilon) = q_0', q_0^2$

Hence, it is proved that L1 ∪ L2 is regular.

* Concatenation closure:- The class of regular languages is closed under concatenation. i.e., if L1 and L2 are two regular languages then, ~~[illegible]~~ L1L2 is also regular.

Let M1 be a machine that accepts L1, and $M2(Q_2, \Sigma_2, \delta_2, q_0^2, F_2)$ accepts L2.



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

$Q = Q_1 Q_2 \quad \Sigma = \Sigma_1 \Sigma_2 \quad \delta = \delta_1 \delta_2$

$q_0 = q_0' \quad F = F_2$

Hw

step 1

Assume L is regular

step 2

$\omega = a^m b^n$, $|\omega| = m+n \geq n = |xyz|$

$|xy| \leq n$, $|y| > 0$

step 3 assume $i = 2$,

case 1 : y consists of a's only a^k

$xyz = a^{m-k} a^k b^n$

$xy^2z = xyyz = a^{m-k} a^k a^k b^n = a^{m+k} b^n$

$GCD(m+k, n) \neq 1$ as $k \neq 0$

$\therefore xy^2z \notin L$

case 2 : y consists of b's only b^k

$xyz = a^m b^{n-k} b^k$

$xy^2z = xyyz = a^m b^k b^k b^{n-k} = a^m b^{n+k}$

$GCD(m, n+k) \neq 1$ as $k \neq 0$

$\therefore xy^2z \notin L$

case 3 : y consists of a's & b's $a^k b^j$

$xyz = a^{m-k} a^k b^{n-j} b^j$

$xy^2z = xyyz = a^{m-k} a^k b^j a^k b^j b^{n-j} = a^{m+k} b^{n+j}$

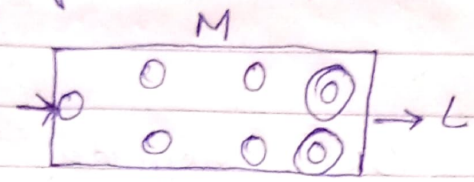
$GCD(m+k, n+j) \neq 1$

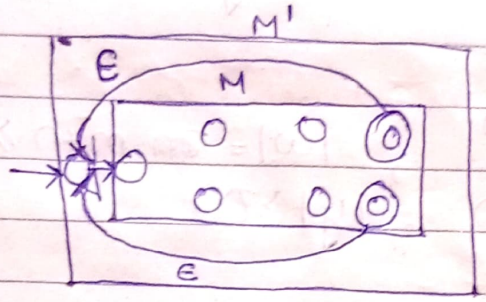
$\therefore xy^2z \notin L$

$\therefore$ L is not regular. (proved) //

* Kleen's Closure :- The class of regular language is closed under star operation. i.e., if L is a regular language then L^* is also regular.

Assume $M(Q, \Sigma, \delta, q_0, F)$ is a FA that accepts the language L.

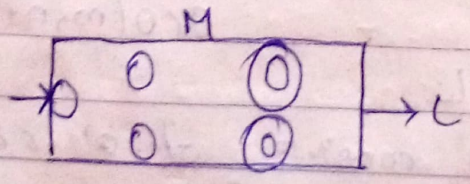


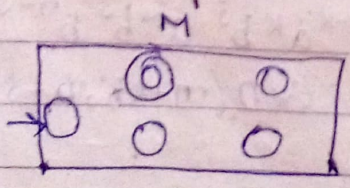


$M' = Q \cup q_0'$, $\Sigma \cup \epsilon$, $\delta \cup \delta(q_f, \epsilon) = q_0, q_0', F$

* Complement :- The class of regular language is closed under Complement operation. i.e., if L is a regular language then $\bar{L}$ is also regular.

Assume $M(Q, \Sigma, \delta, q_0, F)$ is a FA that accepts L.





① M', $Q' = Q$ $\quad \Sigma' = \Sigma$ $\quad \delta' = \delta'$ $\quad q_0' = q_0$

$F' = Q - F$

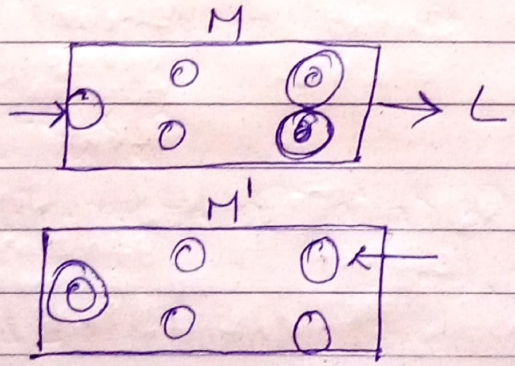
~~Hence~~ There exist a machine $M'(Q', \Sigma', \delta', q_0', F')$ such that ①, accepts $\bar{L}$.

Hence, $\bar{L}$ is also regular.

Reverse

* ~~Set difference~~ :- The class of regular languages is closed under reverse operation i.e., if L is a regular language then L^R is also regular.

Assume $M(Q, \Sigma, \delta, q_0, F)$ is a FA that accepts L.



- There exist a machine $M'(Q', \Sigma', \delta', q_0', F')$ such that

$Q' = Q$ $\quad \Sigma' = \Sigma$ $\quad \delta'$ = with the states

$\delta(ps, P_i) = ns$

$\delta'(ns, P_i) = ps$

$q_0' = F$ $\quad F' = q_0$

Hence, L^R is also Regular.

Grammer :-

It is the set of rules to derive a string from the given set of input symbols.

It is denoted by $G(V, T, P, S)$.

where V → set of variables (all capital letters)
T → terminals (except capital letters, symbol, lowercase letters, digits)
S → starting symbol
P → production (set of substitution rules)

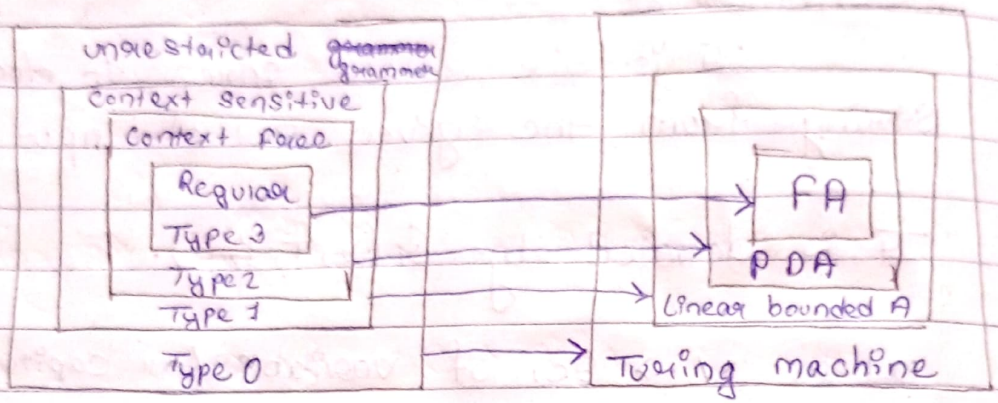
$P = \{ S \to NV , N \to i / you , V \to read / eat \}$
$V = \{ S, N, V \}$
$T = \{ i, you, read, eat \}$
$S \to \{ S \}$

$S \to NV \to you\ V \to you\ read$

10

Chomsky Hierarchy :-
Classification of grammars based on the production rules is called as chomsky hierarchy or chomsky classification.

Types of grammer :-
i) Type 0 grammer
ii) Type 1 grammer
iii) Type 2 grammer
iv) Type 3 grammer



PDA → push Down Automata

Type	Name	machine	Production
0	unrestricted grammar	Turing machine	$\alpha \to \beta$ / $\alpha, \beta \in (V \cup T)^*$ $L(G) \neq \theta$
1	Context Sensitive	Linear Bounded Automata (LBA)	$\alpha \to \beta$ / $\alpha, \beta \in (V \cup T)^*$ $L(G) \neq \theta$ & $\|\alpha\| \leq \|\beta\|$ $S \neq \epsilon$
2	Context free	push Down Automata (PDA)	$\alpha \to \beta$ / $\alpha \in V$, $\beta \in (V \cup T)^*$ $\|\alpha\| \leq \|\beta\|$
3.	Regular	finite Automata	$\alpha \to \beta$ / $\alpha \in V$, $\beta \in (V \cup T)^*$ & β → atleast 1 terminal symbols at leftmost or rightmost position

Examples

Type 0 $Aa \to Baba$, $a \to Aa$, $Aa \to B$

Type 1 $Aa \to aB$, $A \to B$, $A \to Baa$, $Aa \to aB$

Type 2 $A \to aB$, $A \to Ba$, $A \to Bb$, $A \to BB$

Type 3 $A \to Ba$, $A \to aBa$, $A \to aaa$

Regular Derivation Tree :- The substitution are represented using a tree like structure for the generation of any specific string from a given grammer is called as derivation tree.

properties :-

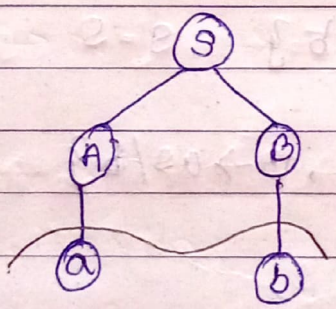
1. Every vertex/node is levled with a variable or a terminal ~~Hex~~
2. The root node is levelled with S.
3. The internal nodes are levelled with variables.
4. The terminal nodes are the terminal symbols T.
5. The terminal symbols are collected from left to right one at a time to define the string.

: $S \to AB$ $A \to a$ $B \to b$

$V = \{S, A, B\}$ $T = \{a, b\}$

$P = \{S \to AB,\ A \to a,\ B \to b\}$

$S = S$

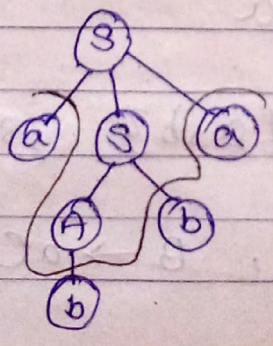


$\omega = ab$

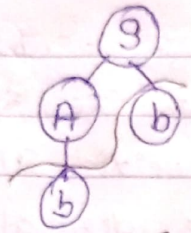
: $S \to aSa$ $S \to Ab$ $A \to b$

$P = \{S \to aSa / Ab,\ A \to b\}$

$V = \{S, A\}$ $T = \{a, b\}$ $S = S$



$\omega = abba$



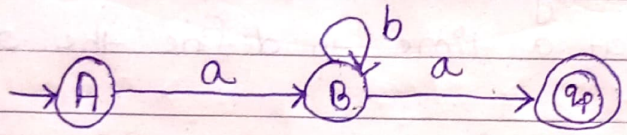
$\omega = bb$

Regular ~~Expression~~ Grammer to FA :-

Ex :- $A \to aB \quad B \to bB/a$

$P = \{ A \to aB, B \to bB/a \}$

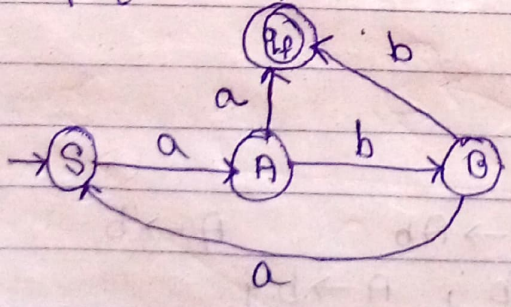
$S = A \quad T = \{a, b\} \quad V = \{A, B\}$



Q. $S \to aA \quad A \to bB \quad B \to aS \quad A \to a \quad B \to b$

$V = \{S, A, B\} \quad T = \{a, b\} \quad S = S$

$P = \{S \to aA, A \to bB/a, B \to aS/b\}$

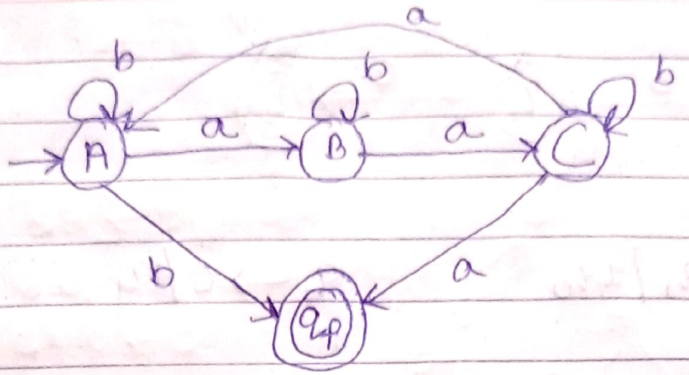


Q. $A \to aB/bA/b \quad B \to aC/bB$

$C \to aA/bC/a$

$V = \{A, B, C\} \quad T = \{a, b\} \quad S = A$

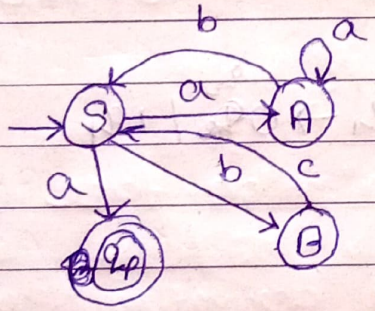
$P = \{A \to aB/bA/b, \quad B \to aC/bB, \quad C \to aA/bC/a\}$



Q. $S \to a/aA/bB \quad A \to aA/bS \quad B \to cS$

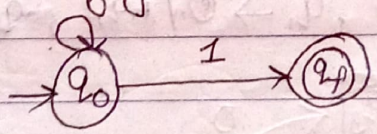
$V = \{S, A, B\} \quad T = \{a, b, c\} \quad S = S$

$P = \{S \to a/aA/bB,\ A \to aA/bS,\ B \to cS\}$



FA to Regular Grammer:-

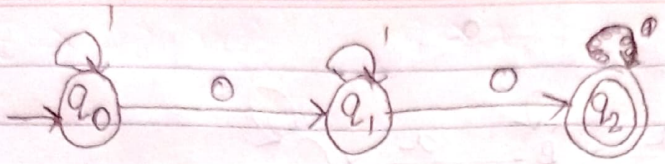
Ex:-



$q_0 \to 0q_0/1$

$V = \{q_0\} \quad T = \{0, 1\} \quad S = q_0$

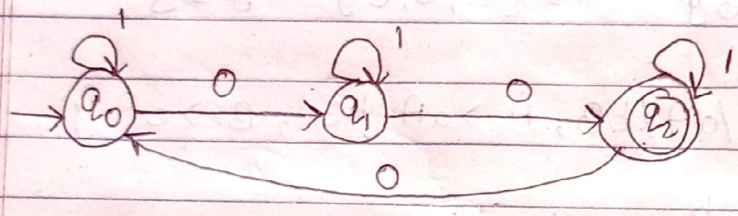
$P = \{q_0 \to 0q_0/1\}$

Q. 

$q_0 \rightarrow 0q_1 / 1q_0$ $q_1 \rightarrow 0/1q_1$

$V = \{q_0, q_1\}$ $T = \{1, 0\}$ $S = q_0$

$P = \{ q_0 \rightarrow 0q_1 / 1q_0 , 0/1q_1 \}$

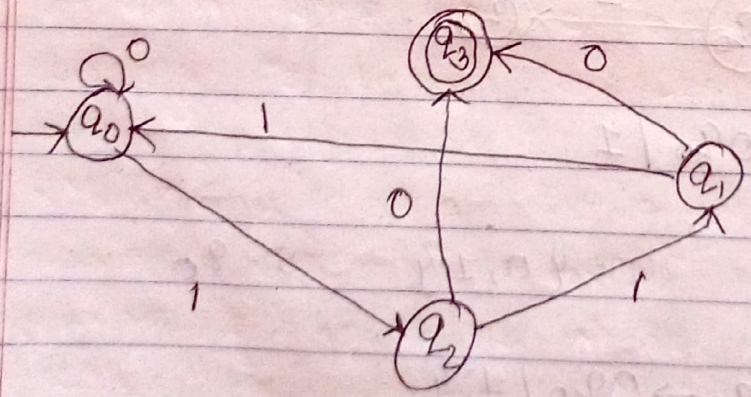
Q. 

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

$q_2 \rightarrow 1/0q_0$

$V = \{q_0, q_1, q_2\}$ $T = \{0, 1\}$ $S = q_0$

$P = \{q_0 \rightarrow 0q_1 / 1q_0 , q_1 \rightarrow 0q_2 / 1q_1 , q_2 \rightarrow 1/0q_0\}$

Q. 

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

$V = \{q_0, q_1, q_2\}$ $T = \{0, 1\}$ $S = q_0$

$P = \{q_0 \to 0q_2/1q_1,\ q_1 \to 1q_0/0,\ q_2 \to 1q_1/0\}$

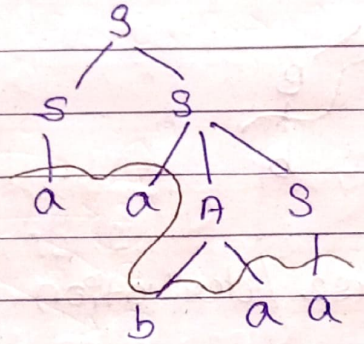
String derivation :-

Ex:- For a given grammer $G\{\{S, A\}, \{a, b\}, P, S\}$

where $P = \{S \to aAS/a/SS,\ A \to SbA/ba\}$

derive a string $\omega = aabaa$

(1)



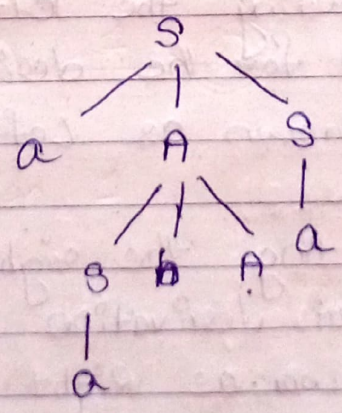
$S \to SS$ $(S \to a)$
$\to aS$ $(S \to aAS)$
$\to aaAS$ $(A \to ba)$
$\to aabaS$ $(S \to a)$
$\to aabaa$

$S \overset{*}{\Rightarrow} aabaa$

↓

Sentential form

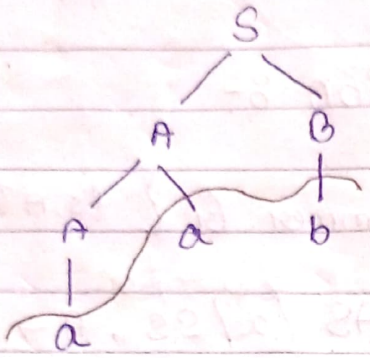
(2)



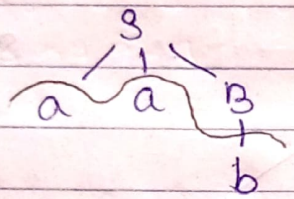
We can derive a string by this method.

Q. $S \to AB/aaB$ $A \to a/Aa$ $B \to b$

$\omega = aab$



$S \to AB$ $(A \to Aa, B \to b)$
$S \to Aab$ $(A \to a)$
$S \to aab$



$S \to aaB$ $(B \to b)$
$S \to aab$

Types of parse / derivation tree

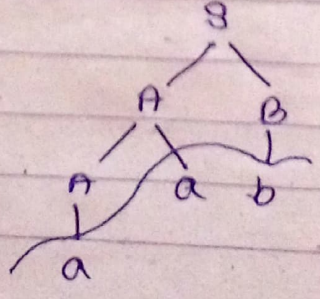
1. LMD (Left Most derivation tree)
2. RMD (Right Most derivation tree)

LMD:- Substituting the left most variable at each step for the derivation of any string is known as Left Most Derivation.

RMD:- Substituting the right most variable at each step of derivation to get the desired string is known as Right Most Derivation.

LMD

$S \to AB$ $(A \to Aa)$
$\to AaB$ $(A \to a)$
$\to aaB$ $(B \to b)$
$\to aab$



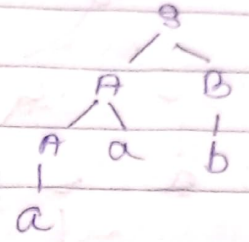
RMD

$S \to AB \ (B \to b)$

$\to Ab \ (A \to Aa)$

$\to Aab \ (A \to a)$

$\to aab$



Ambigous grammer :- The A grammer that generates two or more left most derivation tree (LMD) or Right most derivation tree (RMD) for deriving a specific string is called as ambigous grammer.

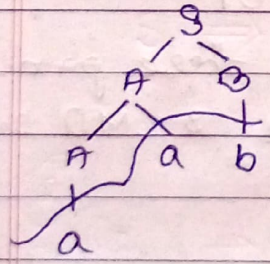
Ex :-

$S \to AB \ (A \to Aa)$

$\to AaB \ (A \to a)$

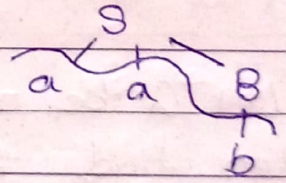
$\to aaB \ (B \to b)$

$\to aab$



$S \to aaB \ (B \to b)$

$\to aab$



$E \to E+E \ / \ a \qquad \omega = a+a+a$

LMD

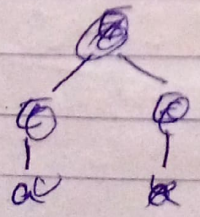
$E \to E+E \ (E \to a)$

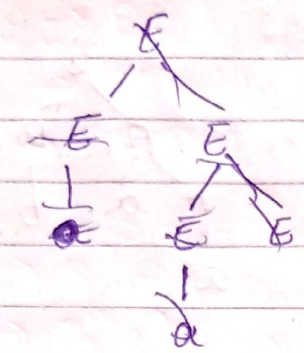
$\to a+E \ (E \to E+E)$

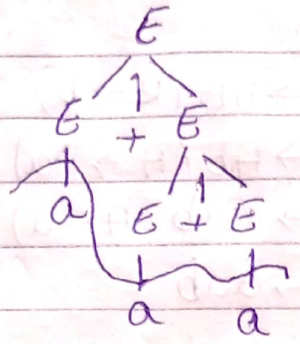
$\to a+E+E \ (E \to a)$

$\to a+a+E \ (E \to a)$

$\to a+a+a$







LMD

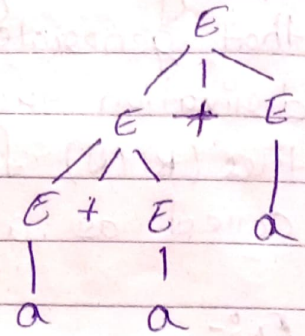
$E \to E+E \quad (E \to E+E)$

$\to E+E+E \quad (E \to a)$

$\to a+E+E \quad (E \to a)$

$\to a+a+E \quad (E \to a)$

$\to a+a+a$



∴ Hence the given grammer is ambigous grammer as we got 2 LMD tree.

Q. $S \to 0B / 1A \qquad A \to 0/0S/1AA \qquad B \to 1/1S/0BB$

$\omega = 00110101$

Let G be a grammer with these production then find the RMD sequence & LMD sequence & derivation tree.

LMD

$S \to 0B \quad (B \to 0BB)$

$\to 00BB \quad (B \to 1S)$

$\to 001SB \quad (S \to 1A)$

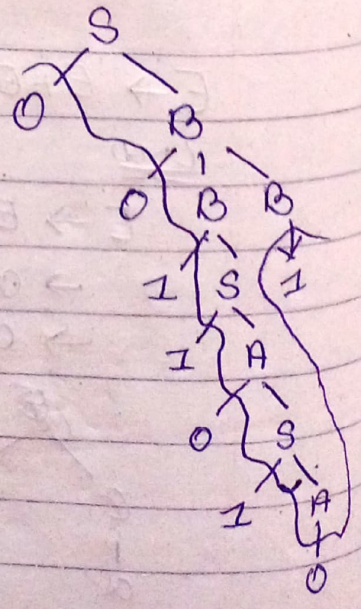
$\to 0011AB \quad (A \to 0S)$

$\to 00110SB \quad (S \to 1A)$

$\to 001101AB \quad (A \to 0)$

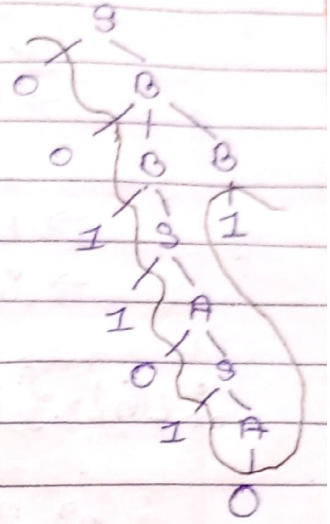
$\to 0011010B \quad (B \to 1)$

$\to 00110101$



RMD

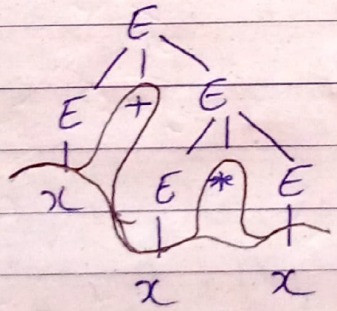
$S \to 0B$ $(B \to 0BB)$
$\to 00BB$ $(B \to 1)$
$\to 00B1$ $(B \to 1S)$
$\to 00S11$ $(S \to 1A)$
$\to 0011A1$ $(A \to 0S)$
$\to 00110S1$ $(S \to 1A)$
$\to 001101A1$ $(A \to 0)$
$\to 00110101$



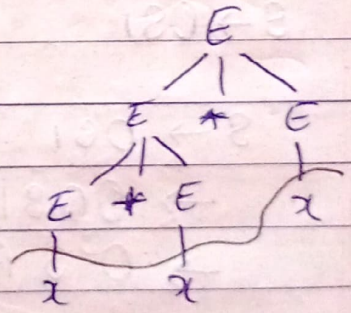
Q. $E \to E+E / E*E / x$ $\omega = x+x*x$

Check whether the grammer is ambigous or not

$E \to E+E$ $(E \to x)$
$\to x+E$ $(E \to E*E)$
$\to x+E*E$ $(E \to x)$
$\to x+x*E$ $(E \to x)$
$\to x+x*x$



$E \to E*E$ $(E \to E+E)$
$\to E+E*E$ $(E \to x)$
$\to x+E*E$ $(E \to x)$
$\to x+x*E$ $(E \to x)$
$\to x+x*x$



∴ Hence, the given grammer is ambigous grammer.

Language to Grammer and vice versa:-

Q. Find $L(G)$ for $G = \{\{S\}, \{0,1\}, S, \{S \to 0S1/1\}\}$
(V, T, S, P)

$S \to \wedge$ $S \to 0S1$

$S \to 0S1 \quad (S \to 0S1)$
$\to 00S11 \quad (S \to 0S1)$
$\to 000S111$
$\vdots$
$\to 0^n S 1^n \quad (S \to \wedge)$
$\to 0^n 1^n$

$\therefore L(G) = \{ 0^n 1^n \mid n \geq 0 \}$

$0^n 1^n$ $\quad n=0 \quad \wedge$
$\quad n=1 \quad 01$
$\quad n=2 \quad 0011$

$S \to 0S1$
$S \to \wedge$:

Q. Construct the grammer G for the given language $L(G) = \{0^n 1^n, n \geq 1\}$

$S \to 0S1 \qquad S \to 01$

$S \to 0S1 \quad (S \to 0S1)$
$\to 00S11$
$\to 0^3 S 1^3$
$\vdots$
$\to 0^{n-1} S 1^{n-1} \quad (S \to 01)$
$\to 0^{n-1} 01 1^{n-1}$
$\to 0^n 1^n$

$G = \{\{S\}, \{0,1\}, \{S \to 0S1 / 01\}, S\}$

Q. Construct a Grammer for the language $L(G) = \{a^n b a^n / n \geq 1\}$

$S \to asa$

$S \to asb$

$S \to asbaas$ $\quad S \to asa$

$\to aaasbaaas$ $\quad \to a^2sa^2$

$\to a^4$ $\quad \to a^3sa^3$

$\vdots$ $\quad (s \to b)$

$\to a^n b a^n$

$G = \{\{S\}, \{a, b\}, \{S \to asa/b\}, S\}$

Q. $G = \{\{S\}, \{a\}, \{S \to SS\}, S\}$ find $L(G)$.

$S \to SS$

$\therefore L(G)$ Ø

because there is no terminal in production.

* Q. Let L be a language consisting of all palindrome string over $\{a, b\}$. Construct a grammer G that derives $L(G)$.

Step 1 the possible palindrome strings are $\wedge$, a, b, aba, For any palindrome string x, axa, bxb.

Step 2

$P = \{S \to \wedge, S \to a, S \to b, S \to asa, S \to bsb\}$

Hence,

$G = \{\{S\}, \{a, b\}, \{S \to \wedge/a/b/asa/bsb\}, S\}$

Q. Construct a grammer generating the language $L = \{wcw^T / w \in \{a,b\}^*\}$.

$S \to A \quad S \to asa \quad S \to bsb \quad S \to c$

$S \to asa \quad (S \to asa)$
$\to aasaa \quad (S \to c)$
$\to a^2ca^2$

$S \to asa \quad (S \to bsa)$
$\to absba \quad (S \to c)$
$\to abcba$

$G = \{\{s\}, \{a,b,c\}, \{s \to c/asa/bsb\}, S\}$

Q. Find G for $L = \{\underbrace{a^n b^n}_{L_1} \underbrace{c^i}_{L_2} / n \geq 1, i \geq 0\}$

$s \to asb \quad S \to abb \quad S \to Sc$

$S \to Sc \quad (S \to Sc)$
$\to Sc^2$
$\to Sc^3$
$\vdots$
$\to Sc^i \quad (S \to asb)$
$\to asbc^i \quad (s \to ab)$
$\to a^2b^2c^i$
$\to \vdots$
$\to a^n b^n c^i$

$G = \{\{s\}, \{a,b,c\}, \{s \to asb/ab/Sc\}, S\}$

Q. Find G for $L = \{a^n b^i c^i / n \geq 0, i \geq 1\}$

$s \to aS \quad S \to bsc \quad S \to bc$

$S \to aS \to a^2S \to \cdots \to a^nS \quad (S \to bSc)$
$\to a^n bSc \quad (S \to bc)$
$\to a^n bbScc \to \cdots \to a^n b^n c^n$

$G_r = \{\{S\}, \{a, b, c\}, \{S \to bSc/bc/aS\}, S\}$

Normal form of CFG:-
1. Chomsky Normal form (CNF)
2. Greibach Normal form (GNF)

Simplification → A grammer it simplify or reduced with the following steps:

Step 1: Removal of the variabre that does not derive any terminal stone.

Step 2: Removal of the terminal symbols that don't appear in the sentential form.

Step 3: Elimination of null production.

Step 4: Elimination of unite production.

Q. $S \to AB, A \to a, B \to b, B \to C, E \to c$
Removal of unnecessary variables
*) Construction of V'

$\omega_1 = \{A \mid A \in V \text{ & } A \to \alpha \text{ & } \alpha \in T\}$

$\omega_1 = \{A, B, E\}$

$\omega_2 = \omega_1 \cup \{A \mid A \to \omega_1\}$

$\omega_2 = \{A, B, E\} \cup \{S\}$
$= \{S, A, B, E\}$

$W_3 = W_2 \cup \{A \mid A \to w_2\}$

$w_3 = \{A, B, E, S\} \cup \{D\}$

$= \{S, A, B, E\}$

*) Construction of P'

$P' = \{A \to \alpha \mid A, \alpha \in (V' \cup T)^*\}$

$P' = \{S \to AB, A \to a, B \to b, E \to c\}$

Removal of unneccessary teaminals.

*) $W_1 = \{S\}$

$W_2 = W_1 \cup \{$ all the variables derived from $W_1\}$

$= W_1 \cup \{A, B\} = \{S, A, B\}$

~~$W_3 = W_2 \cup \{E\} = \{S, A, B, E\}$~~

$W_3 = W_2 \cup \{a, b\} = \{S, A, B, a, b\}$

$V' = V \cap W_3 = \{S, A, B\}$

$T' = \{a, b\}$

$P' = \{S \to AB, A \to a, B \to b\}$

$S = S$

Elimatation of $\wedge$.

Q. $S \to aS \mid AB \quad A \to \wedge \quad B \to \wedge \quad D \to b$

$D \to b$

$S \to aS, S \to a$

$S \to AB, S \to A, S \to B$

$S \to aS$

$\to aAB$

$\to a \wedge \wedge \to a$

Elimination of unit production

Q. $S \to AB \quad A \to a \quad B \to C/b \quad C \to D$
$D \to E \quad E \to a$

$w_0(S) \Rightarrow \{S\}$ $\quad w_0(A) \Rightarrow \{A\}$
$w_0(E) \Rightarrow \{E\}$

$w_0(B) \Rightarrow \{B\}$	$w_0(C) = \{C\}$
$w_1(B) \Rightarrow \{B, C\}$	$w_1(C) = \{C, D\}$
$w_2(B) \Rightarrow \{B, C, D\}$	$w_2(C) = \{C, D, E\}$
$w_3(B) \Rightarrow \{B, C, D, E\}$	

$w_0(D) = \{D\}$
$w_1(D) = \{D, E\}$

$S \to AB, \; A \to a, \; B \to b, \; E \to a$

$B \to a \quad C \to a \quad D \to a$

Dt - 07/05/22

Q. Find the reduce grammer equivalent to the grammer G whose production are
$S \to AB / CA \quad B \to BC / AB \quad A \to a$
$C \to aB / b$

Sol) for reduction :
1) Removal of unneccessary variables
2) Removal of unneccessary terminals
3) ~~Elim~~. Removal of null production
4) Removal of unit production

① Identify the variables that derive a terminal symbol.

$A \to a \quad C \to b$

$\omega_0 = \{A, C\} \quad \omega_1 = \{S, A, C\}$

$V_1 = \{S, A, C\} \quad T_1 = \{a, b\} \quad P_1 = \{S \to CA, A \to a, C \to b\}$

② $\omega_0 = \{S\} \quad \omega_1 = \{S, A, C\}$

$\omega_2 = \{S, A, C, a, b\}$

$V_2 = \{S, A, C\} \quad T_2 = \{a, b\} \quad P_2 = P_1$

③ No null production

④ No unit production

Q. Construct a reduced grammer equivalent to the grammer G and the production are

$S \to aAa \quad A \to Sb / bCC / DaA$

$C \to abb / DD \quad D \to aDA \quad E \to aC$

Soln

① Removal of unnecessary variables

$C \to abb$

$\omega_0 = \{C\}$

$W_1 = \{ C, A, E \}$

$W_2 = \{ C, A, E, S \}$

$V_1 = \{ C, A, E, S \}$ $T_1 = \{ a, b \}$

$P_1 \rightarrow \{ C \rightarrow abb,\ S \rightarrow aAa,\ A \rightarrow bCC,\ E \rightarrow aC,\ A \rightarrow Sb \}$

(2) removal of unnecessary Terminals

$W_0 = \{ S \}$ $W_1 = \{ S, A, a \}$

$W_2 \rightarrow \{ S, A, a, b, C \}$

$W_3 \rightarrow \{ S, A, C, a, b \}$

$V_2 = \{ S, A, C \}$ $T_2 = \{ a, b \}$

$P_2 = \{ S \rightarrow aAa,\ A \rightarrow Sb,\ A \rightarrow bCC,\ C \rightarrow abb \}$

(3) No null production

(4) No unit production

Chomsky Normal form (CNF)

A Context-free grammer is said to be chomsky normal form, if every production is in the form $A \rightarrow a$ or $A \rightarrow BC$, or $S \rightarrow \Lambda$ such that S doesn't appear on right hand side.

Ex:

$A \to a$ $B \to b$ $S \to AB/\Lambda$ ✓

$S \to AC$ $A \to aB$ $B \to b$ ✗

Q. $S \to AB$ $A \to aB$ $B \to b$

(a) reduction of grammer

(1) $\omega_0 = \{B\}$ $\omega_1 = \{B, S, A\}$

$V_1 = \{S, A, B\}$ $T_1 = \{a, b\}$

$P_1 = \{S \to AB, A \to aB, B \to b\}$

(2) $\omega_0 = \{S\}$ $\omega_1 = \{S, A, B\}$

$\omega_2 = \{S, A, B, a, b\}$

$V_2 = V_1$ $T_2 = T_1$ $P_2 = P_1$

(3) NO null prod^n

(4) NO unit prod^n

(b) Conversion of CNF

$S \to AB$, $B \to b$ in CNF

we need to convert $A \to aB$ into CNF

Let's substitute $C_1 \to a$

$\Rightarrow A \to C_1 B$

$P = \{S \to AB, B \to b, A \to C_1 B, C_1 \to a\}$

$V = \{B, A, D, C_1\}$ $T = \{a, b\}$ $S = S$

Q. Reduce the grammer G_1 into CNF.

$S \to aAD$ $A \to aB / bAB$ $B \to b$ $D \to d$

(a) reduction of grammer G_1

Already reduced grammer.

(b) conversion of CNF

$B \to b$, $D \to d$ in CNF

(1) replace the lower case letter (terminals)

$S \to C_1AD$ $A \to C_1B / C_2AB$

$C_1 \to a$ $C_2 \to b$

(2) reduce the variable length to 2 of RHS

$B \to b$, $D \to d$, $C_1 \to a$, $C_2 \to b$, $A \to C_1B$ in CNF

Now, convert $S \to C_1AD$ and $A \to C_2AB$

Let's take $C_3 \to AD$ $C_4 \to AB$

$S \to C_1C_3$ $A \to C_2C_4$

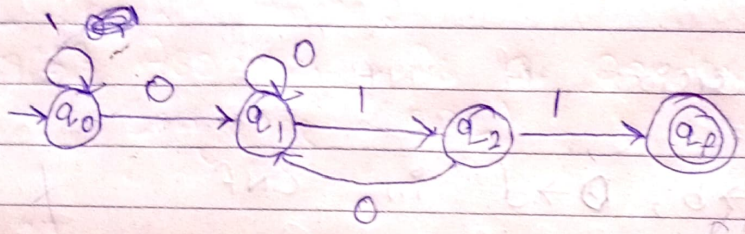
$V = \{S, B, A, D, C_1, C_2, C_3, C_4\}$ $T = \{a, b, d\}$

$P = \{B \to b, D \to d, C_1 \to a, C_2 \to b, A \to C_1B, C_3 \to AD, C_4 \to AB, S \to C_1C_3, A \to C_2C_4\}$

1) If the right side of the production contain starting symbol then generate another production that will generate the starting symbol.

2) Removal of unit production
3) Removal of null production

1Q. Draw the DFA state diagram accept all the strings over 0, 1 and ended with 011.

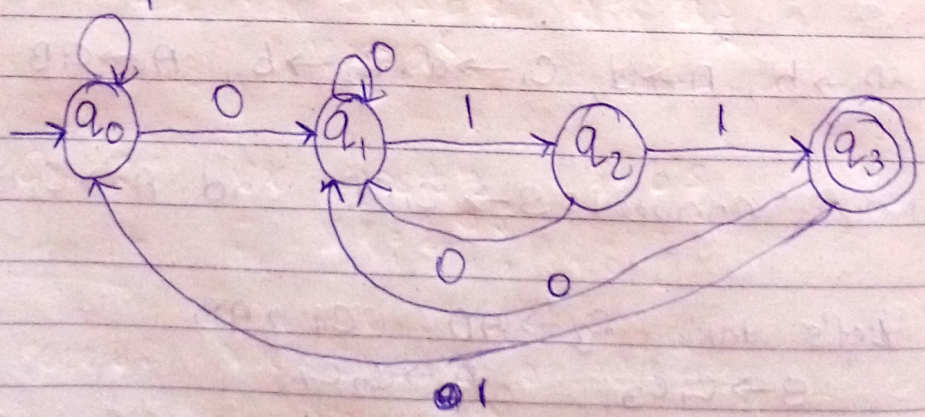


1011 ✓
0011 ✓

δ =

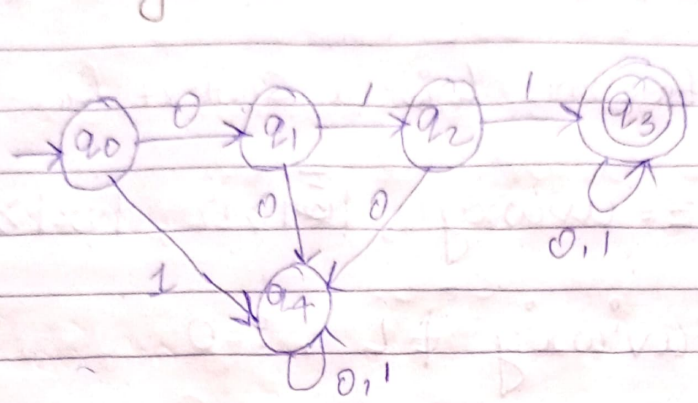
	0	1
→ ε q_0	q_1	q_0
0 q_1	q_1	q_2
01 q_2	q_1	q_3
011 (q_3)	q_1	q_0

$q_0 \rightarrow \varepsilon$
$q_1 \rightarrow 0$
$q_2 \rightarrow 01$
$q_3 \rightarrow 011$



$Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{0, 1\}$
$S = q_0$
$F = \{q_3\}$

2Q. Starting with 011



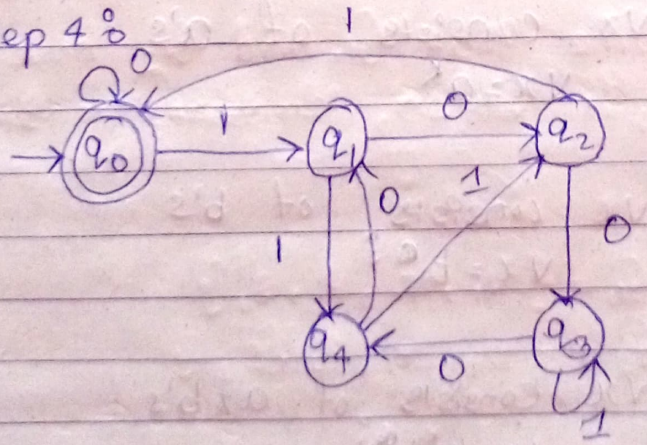
3Q. Divisible by 5

Step 1 : possible remainders are
0, 1, 2, 3, 4

Step 2 : No such string behaviour

Step 3 : No. of states
$0 \to q_0$ $1 \to q_1$ $2 \to q_2$ $3 \to q_3$
$4 \to q_4$

Step 4 :



$Q = \{q_0, q_1, q_2, q_3, q_4\}$ $S = q_0$ $F = \{q_0\}$
$\Sigma = \{0, 1\}$
$\delta = (q_0, 0) \to q_0, (q_0, 1) \to q_1,$ ~~$q_1, q_1 0$~~

pumping lemma for CFL :-

Step 1: Assume L is a Context free Language

Step 2: $z = uvwxy$, $|z| \geq n$, $|vx| \geq 1$

Step 3: $uv^i wx^i y \notin L$, $i = 0$

Q. prove that $L = \{a^m b^n \mid m = n^2\}$ is not CFL.

A:- Step 1: Assume L is a context free Language (CFL)

Step 2:

$z = a^m b^n = uvwxy$ $\quad |z| = m+n$

$= n^2 + n \geq n$

$|vx| \geq 1$

Step 3: $i = 2$

Case 1: vx consists of a's

$vx = a^k$

Case 2: vx consists of b's

$vx = b^j$

Case 3: vx consists of a & b's

$vx = a^k b^j$

$uv^2wx^2y = |uvwxy| + |vx|$

Case 1:

$uv^2wx^2y = n^2+n+K$

$\neq n^2+n$

$uv^2wx^2y \notin L$

Case 2:

$uv^2wx^2y = n^2+n+j$

$\neq n^2+n$

$uv^2wx^2y \notin L$

Case 3:

$uv^2wx^2y = n^2+n+K+j$

$\neq n^2+n$

$uv^2wx^2y \notin L$

$\therefore L$ is not CFL

Q. prove that $L = \{a^p / p$ is a prime no.$\}$ is not CFL.

A: Step 1: Assume L is a CFL

Step 2:

$z = a^n$ $\quad p$ = prime no.

$n \geq n$

$|z| = |uvwxy| = n$, $\quad |vx| \geq 1$

Step 3: Assume $i = 2$ Let $vx = a^K$

$|uv^iwx^iy| = |uv^2wx^2y|$

$= |uvvwxy| + |vx|$

$= n + K$

$n+K$ may not be a prime no. always.
Hence, $uvwxy \neq n+K$.

$\therefore$ L is not CFL.

Closure properties of CFL:-

CFL closed under:
1. Union
2. Concatenation
3. Kleen's Closure

Union Closure: If L_1 and L_2 are two context free languages then $L_1 \cup L_2$ is also context free.

Step 1: Assume L_1 & L_2 are CFL accepted by the grammer G_1 & G_2 respectively.

Step 2: Assume that G_1 and G_2 have no common variables, if ~~yes~~ so, ~~rewrite~~ rename the variables.

Step 3: $G_1 = \{V_1, T_1, P_1, S_1\}$
$G_2 = \{V_2, T_2, P_2, S_2\}$

Step 4: ~~Let us~~ $L = L_1 \cup L_2$ generated by a grammer $G = G_1 \cup G_2$.

$G_1 = \{V_1, T_1, P_1, S\}$ $\quad V = \{V_1 \cup V_2\}$
$T = \{T_1 \cup T_2\}$

$P = \{P_1 \cup P_2\}$ $S = \{S_1 \cup S_2\}$ S_1 / S_2

Ex: $\{S_1 \to aA \quad A \to b\}$ G_1

$\{S_2 \to bB \quad B \to a\}$ G_2

$S \to S_1 / S_2 = \{ab, ba\}$

Concatenation Closure: If L_1 and L_2 are 2 CFL accepted then $L_1 . L_2$ is also context free.

Step 1: Assume L_1 & L_2 are 2 CFL accepted by the grammar G_1, G_2 respectively.

Step 2: Assume that G_1 and G_2 have no common variables, if so rename the variables

Step 3: $G_1 = \{V_1, T_1, P_1, S_1\}$
$G_2 = \{V_2, T_2, P_2, S_2\}$

Step 4: $L = L_1 L_2$ generated by a grammar
$G = G_1 G_2$

$G = \{V, T, P, S\}$ $V = \{V_1 \cup V_2\}$ $T = \{T_1 \cup T_2\}$
$P = \{P_1, P_2\}$ $S = S_1 S_2$

$G_1 = \{S_1 \to aA, \; A \to b\}$
$G_2 = \{S_2 \to bB, \; B \to a\}$

$S \to S_1 S_2 \to aAS_2 \to abS_2 \to abbB \to abba$

Kleen's Closure: If L_1 is a CFL then L_1^* is also CFL. Context Free.

Step 1:

Step 1: Assume L_1 is CFL

Step 2: L_1 is accepted by $G_1\{V_1, T_1, P_1, S_1\}$

Step 3: There exist G_1 that accepts L such that $L = L_1^*$

$G_1 = \{V_1, T_1, P_1, S\}$ $V_1 = V$ $T_1 = T$ $P_1 = P$

$S \to S_1 S / \epsilon$

Ex $G_1 = \{S_1 \to aA, A \to b\}$

$S \to S_1 S \to S_1 S_1 S \to abababS \dots \to (ab)^n S$

$\to (ab)^* S \to (ab)^* \epsilon = (ab)^*$.

Greibach Normal Form (GNF)

A grammer is said to be in GNF if it is in the form $A \to a\alpha$ or $A \to a$ such that $a \in T$ and $\alpha \in (V \cup T)^*$.

Conversion into GNF:

Step 1: Eliminate the null production and unit production.

Step 2: Convert it to into CNF

Step 3: Rename all the variables in the form A_i.

Step 4: Select a production $A_i \rightarrow A_j$ $(i > j)$ and apply substitution

Step 5: Apply left recursion until the desired productions are formed.

Left recursion: ~~$A \rightarrow A\alpha/\beta$~~

$A \rightarrow A\alpha/\beta$

$A \rightarrow \beta A'$
$A' \rightarrow \alpha A'/\epsilon$

Ex $A \rightarrow Aa/b/ab$

$A \rightarrow A\alpha/\beta_1/\beta_2$ $[\alpha \rightarrow a, \beta_1 = b, \beta_2 = ab]$

$A \rightarrow \beta_1 A'/\beta_2 A'$
$A' \rightarrow \alpha A'/\epsilon$

$A \rightarrow bA'/abA'$
$A' \rightarrow \alpha A'/\epsilon$

Q. Convert the given grammar into GNF:
$S \rightarrow AB \quad A \rightarrow BS/b \quad B \rightarrow SA/a$

Soln: Step 1: No null & unit production

Step 2: Already in CNF

Step 3: Rename variables in A_i form

$S = A_1 \quad A = A_2 \quad B = A_3$

$A_1 \rightarrow A_2A_3 \quad A_2 \rightarrow A_3A_1/b \quad A_3 \rightarrow A_1A_2/a$

Step 4: $A_i \rightarrow A_j$ $(i > j)$

Select

A $A_3 \rightarrow A_1A_2/a$ and apply substitution.

$A_3 \rightarrow A_1A_2 \rightarrow A_2A_3A_2 \rightarrow A_2A_3A_3A_1$

$A_3 \rightarrow A_2A_3A_2 \rightarrow A_3A_1A_3A_2/bA_3A_2$

$A_3 \rightarrow A_2A_3A_2 \rightarrow bA_3A_2$

Step 5: left recursion

$A_3 \rightarrow A_3\alpha/\beta$ $[\alpha = A_3A_1A_2 \quad \beta = bA_3A_2]$

$A_3 \rightarrow \beta A_3'$ $A_3' \rightarrow \alpha A_3'/\varepsilon$

$A_3 \rightarrow bA_3A_2A_3'$ $A_3' \rightarrow A_3A_1A_2A_3'/\varepsilon$

The productions in GNF

$A_2 \rightarrow b$ $A_3 \rightarrow a$ $A_3 \rightarrow bA_3A_2A'/bA_3A_2$

Non-GNF are: $A_1 \rightarrow A_2A_3$, $A_2 \rightarrow A_3A_1$,
$A_3 \rightarrow A_1A_2$ $A' \rightarrow A_3A_1A_2A'/\varepsilon$

Apply value of A_3 in A_2,

$A_3 \rightarrow bA_3A_2A'A_1/bA_3A_2A_1/aA_1$

$A_1 \rightarrow bA_3A_2A'A_1A_3 / bA_3A_2A_1A_3 / aA_1A_3 / bA_3$

$A' \rightarrow bA_3A_2A'A_1A_3A_3A_2A' / bA_3A_2A_1A_3A_3A_2A' / aA_1A_3A_3A_2A' / bA_3A_3A_2A' / \varepsilon$

Q. $S \rightarrow AA / a \quad A \rightarrow SS / b$

Step 1: NO null and unit production

Step 2: Already in CNF

Step 3: Rename variables

$S \rightarrow A_1 \quad A \rightarrow A_2$

$A_1 \rightarrow A_2A_2 / a \quad A_2 \rightarrow A_1A_1 / b$

Step 4: $A_i \rightarrow A_j \quad (i > j)$

Select $A_2 \rightarrow A_1A_1 / b$ & apply substitution

$A_2 \rightarrow A_2A_2A_1 / aA_1 / b$

Step 5: left recursion

$A_2 \rightarrow A_2\alpha / \beta_1 / \beta_2 \quad [\alpha = A_2A_1, \beta_1 = aA_1, \beta_2 = b]$

$A_2 \rightarrow \beta_1A' / \beta_2A' \quad A' \rightarrow \alpha A' / \varepsilon$

$A_2 \rightarrow aA_1A' / bA' \quad A' \rightarrow A_2A_1A' / \varepsilon$

production in GNF:

✓ $A_2 \rightarrow aA_1A' / bA'$

Substitute A_2 in A_1,

$A_1 \rightarrow A_2A_2 / a$

✓ $A_1 \rightarrow aA_1A'A_2 / bA'A_2 / a$

Substitute A_1 in A',

$A' \rightarrow A_2A_1A' / \varepsilon$

✓ $A' \rightarrow aA_1A'A_1A' / bA'A_1A' / \varepsilon$

CYK parsing :- This is a membership algorithm for finding whether a given string is the member of the grammar or not.

Algorithm:

begin:

1. For i = 1 to n (n → no. of symbols(w))
2. $V_{i1} = \{A / A \rightarrow a$ is a production & $a \in T\}$
3. For j = 2 to n
4. For i = 1 to n-j+1
5. begin
6. $V_{ij} = \emptyset$
7. For K = 1 to j-1
8. $V_{ij} = V_{ij} \cup \{A / A \rightarrow BC, B \in V_{iK}, C \in V_{i+K, j-K}\}$

Ex $S \to AB/BC \quad A \to BA/a \quad B \to CC/b \quad C \to AB/a$

$w = baaba$

Soln

$\{S,A\}_{15}$				
	$\{C\}_{25}$			
$\{S,A\}_{13}$	$\{S,C\}_{24}$	$\{B\}_{35}$		
$\{B\}_{12}$	$\{B\}_{23}$	$\{S,C\}_{34}$	$\{S,A\}_{45}$	
$\{B\}_{11}$	$\{A,C\}_{22}$	$\{A,C\}_{33}$	$\{B\}_{44}$	$\{A,C\}_{55}$
b	a	a	b	a

2nd row

$V_{ij} = \{V_{ii}, V_{i+1,j}\}$

3rd row

~~$V_{ij} = \{V_{ii}, V_{i+1,j}\}$~~

4th row

~~$V_{ij} = \{V_{ii}, V_{i+3,j}\}$~~

2nd

$V_{23} = V_{22}, V_{3,3} = \{AC\}, \{AC\} = \{AA, AC, CA, CC\}$ (B)

$34 = (33)(44) = \{A,C\}\{B\} = \{AB, CB\}$ (S, C) X

$45 = (44)(55) = \{B\}\{AC\} = \{BA, BC\}$ (A) (S)

3rd

$13 = (1,1)(3,3) = \{B\}\{A,C\} = \{S,A\}$

$24 = (2,2)(4,4) = \{A,C\}\{B\} = \{S,C\}$

$35 = (3,3)(55) = \{A,C\}\{A,C\} = \{B\}$

4th

$14 = (1,1)(4,4) = \{B\}\{B\} = \{BB\}$ X

$25 = (2,2)(5,5) = \{A,C\}\{A,C\} = \{B\}$

5th

$15 = (1,1)(5,5) = \{B\}\{A,C\} = \{S,A\}$

$15 = (11, 25)(12, 35)(13, 45)(14, 55)$

$= (B, SAC)(S, A, B)(\theta, SA)(\theta, AC)$

$= (BS, BA, BC, SB, AB, SA, AC)$

$\times \quad A \quad \theta \quad \times \quad S, C \quad \times \quad \times$

$= \{A, S, C\} = \{S, A, C\}$

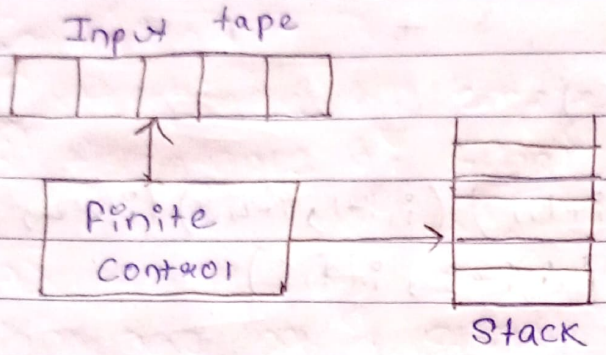
∴ As we were getting the ~~String is a member~~ starting Symbol S on the last entry of the passing table, the given string is a member of the given grammer.

pushdown Automata (PDA)

A PDA is defined as a finite automata with memory. It is also called as a push down stack.

A stack data structure is used to store the symbols of a ~~pda~~ PDA.

A PDA generates the CFL.



PDA

Types of PDA

1) Deterministic PDA :- for each input symbol if a single next state is obtained then the PDA is called deteaministic PDA

2) Non-Deterministic PDA :- for each input symbol if we are getting more than one next state then it is called as Non-deterministic PDA.

Representation of PDA

$PDA = (Q, \Sigma, \delta, q_0, z_0, \Gamma, F)$

where z_0 = inital stack symbol
Γ = stack symbols

$\delta : Q * \Sigma * \Gamma \longrightarrow Q * \Gamma$

Instantaneous Description (ID) :-

Each and every transition in PDA is

represented using the present state, present input symbol and top of the stack to indicate the next state and stack content. This representation is called as ID.

$$(q_p, x, \alpha) \vdash (q_n, \beta)$$

where

$q_p \rightarrow$ present state

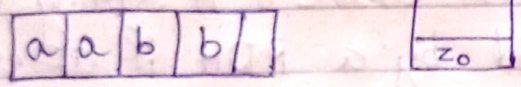
$x \rightarrow$ present input

$\alpha \rightarrow$ top of the stack

$q_n \rightarrow$ next state

$\beta \rightarrow$ stack content

$\omega = aabb$

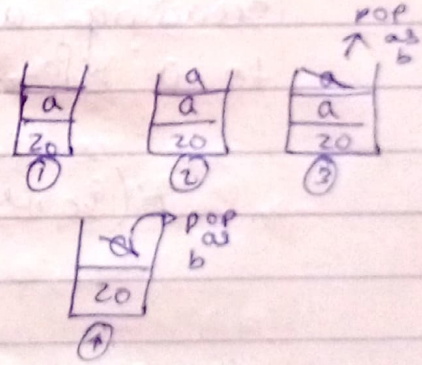


① $(q_0, a, Z_0) \vdash (q_1, aZ_0)$

② $(q_1, a, aZ_0) \vdash (q_1, aa)$

③ $(q_1, b, aa) \vdash (q_2, aZ_0)$

④ $(q_2, b, aZ_0) \vdash (q_3, Z_0)$



Short Questions (possible)

① Define a PDA with all the tuples

② Differentiate b/w finite Automata and PDA.

③ Describe the different types of PDA

④ Describe the components of a PDA.

⑤ Define an ID

⑥ Define the moves of a PDA with an example

Acceptance by PDA :-

Condition :
1. Acceptance by empty stack
2. Acceptance by final state

→ Acceptance by empty stack : A given language or a string is accepted by a PDA if after reading all the i/p symbols the stack is empty.

$$\delta(q_0, w, z_0) \vdash^* (p, \epsilon, z_0)$$

where $q_0, p \in Q$.

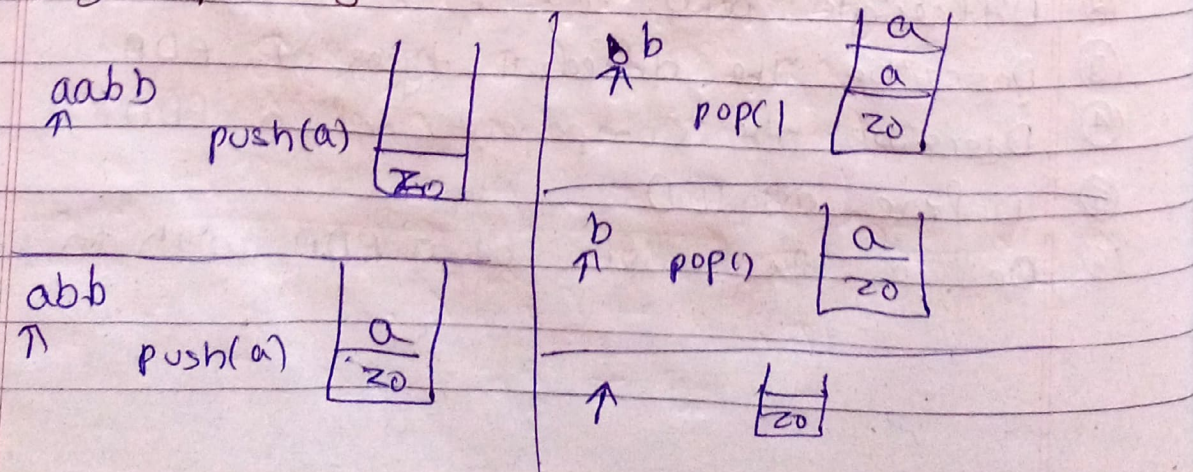
→ Acceptance by final state : A string w is said to be accepted by a PDA if after reading all the i/p symbols the PDA reaches to a final state.

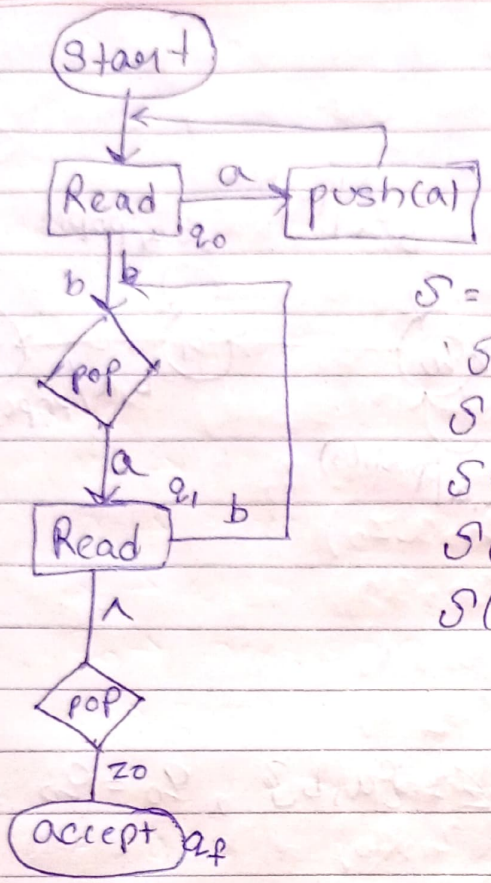
$$\delta(q_0, w, z_0) \vdash^* (p, \epsilon, \lambda)$$

where $q_0 \in Q$, $p \in F$, $\lambda \in V$

Design of PDA :

Q. Construct a PDA that accepts the language $L = \{a^n b^n / n \geq 1\}$





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

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

$q_0 = q_0, z_0, \sqrt{}(z_0, a)$

$\delta =$

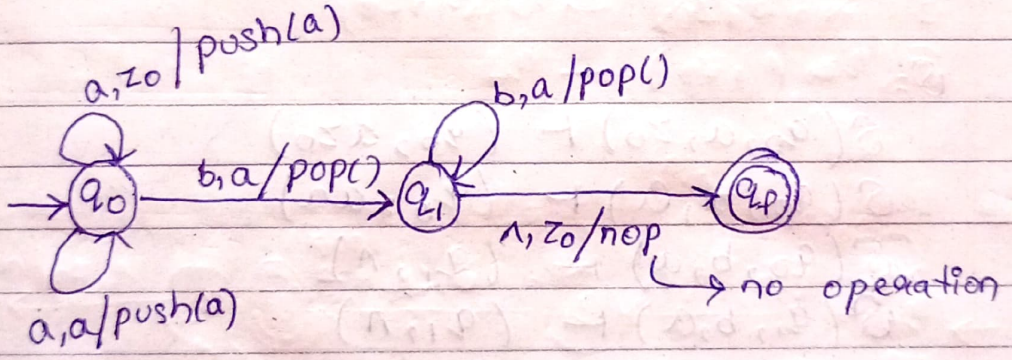
$\delta(q_0, a, z_0) \vdash (q_0, az_0)$

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

$\delta(q_0, b, a) \vdash (q_1, \wedge)$

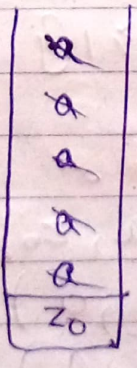
$\delta(q_1, b, a) \vdash (q_1, \wedge)$

$\delta(q_1, \wedge, z_0) \vdash (q_f)$



Q. $L = \{a^5 b^5\}$

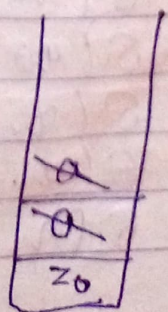
aaaaabbbbb

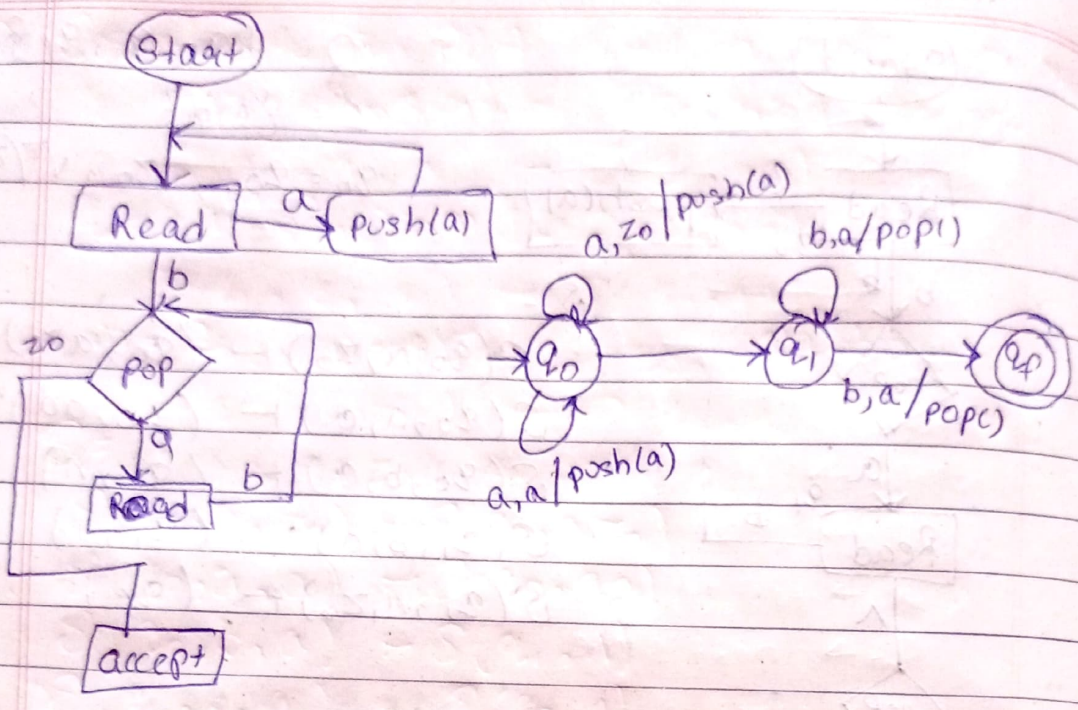


Q. $L = \{a^2 b^3\}$

$L = \{a^n a^{n+1}, n \geq 1\}$

aabbb





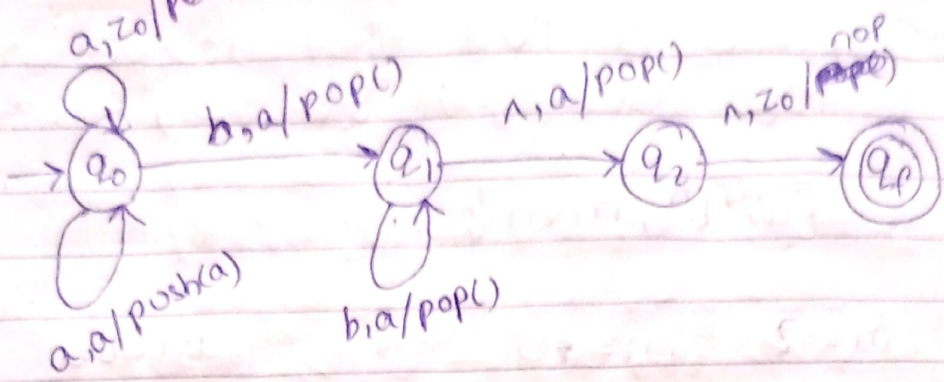
$Q = \{q_0, q_1, q_f\}$, $\Sigma = \{a, b\}$, $q_0 = q_0$, z_0
$\wedge(z_0, a)$.

$\delta =$

$\delta(q_0, a, z_0) \vdash (q_0, az_0)$
$\delta(q_0, a, a) \vdash (q_0, aa)$
$\delta(q_0, b, a) \vdash (q_1, \wedge)$
$\delta(q_1, b, a) \vdash (q_1, \wedge)$
$\delta(q_1, b, z_0) \vdash q_f$

Q. $L = \{a^{n+1} b^n \mid n \geq 1\}$

$\delta =$ $\delta(q_0, a, z_0) \vdash (q_0, az_0)$
$\delta(q_0, a, a) \vdash (q_0, aa)$
$\delta(q_0, a, a) \vdash (q_0, aa)$
$\delta(q_0, b, a) \vdash (q_1, \wedge)$
$\delta(q_1, b, a) \vdash (q_1, \wedge)$
$\delta(q_1, \wedge, a) \vdash (q_2, \wedge)$
$\delta(q_2, \wedge, z_0) \vdash q_f$



$Q = \{q_0, q_1, q_2, q_f\}$, $\Sigma = \{a, b\}$, $q_0 = q_0$, z_0, $V(z_0, a)$

Q. $L = \{a^n b^{2n} / n \geq 1\}$ $\quad a^2 a^4 = aabbbb$

$\delta =$ $\delta(q_0, a, z_0) \vdash \delta(q_0, a z_0)$

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

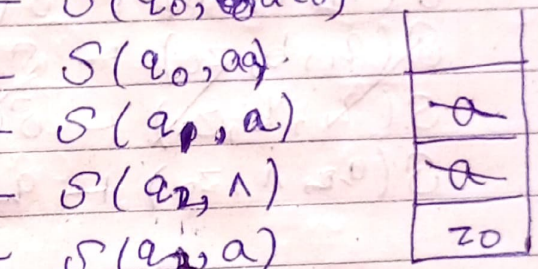
$\delta(q_0, b, a) \vdash \delta(q_1, a)$

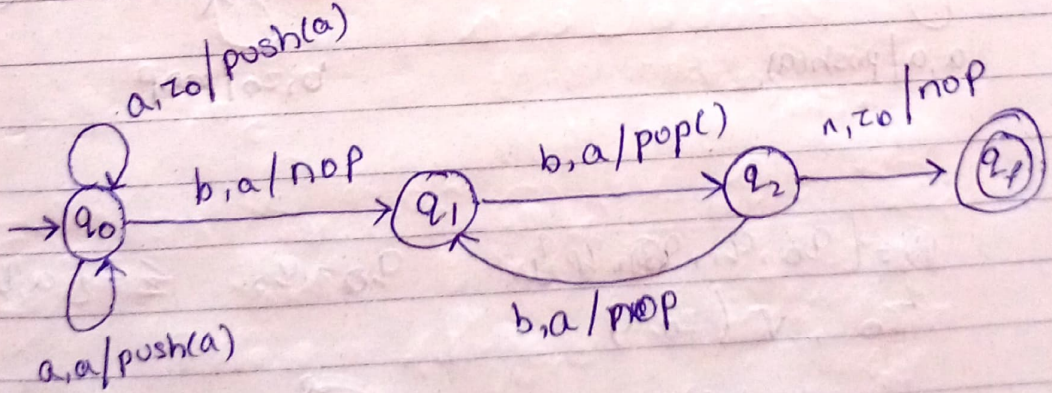
$\delta(q_1, b, a) \vdash \delta(q_2, \lambda)$

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

~~$\delta(q_1, b, a) \vdash \delta(q_2, \lambda)$~~

$\delta(q_2, \lambda, z_0) \vdash q_f$





$Q = \{q_0, q_1, q_2, q_f\}$, $\Sigma = \{a, b\}$, $q_0 = q_0$, z_0, $V(z_0, a)$

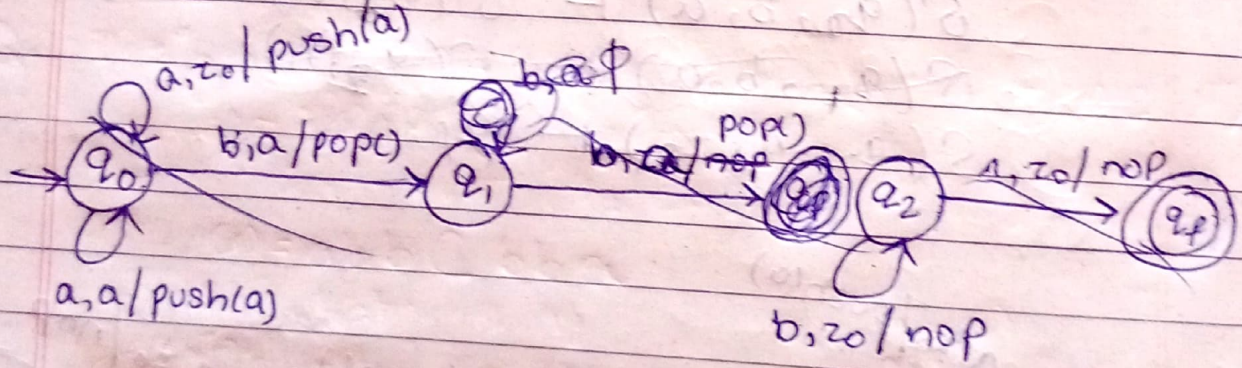
Q. Construct a PDA that accepts no. of a's is less than no. of b's in a string.

$L = \{a^m b^n, m<n\}$

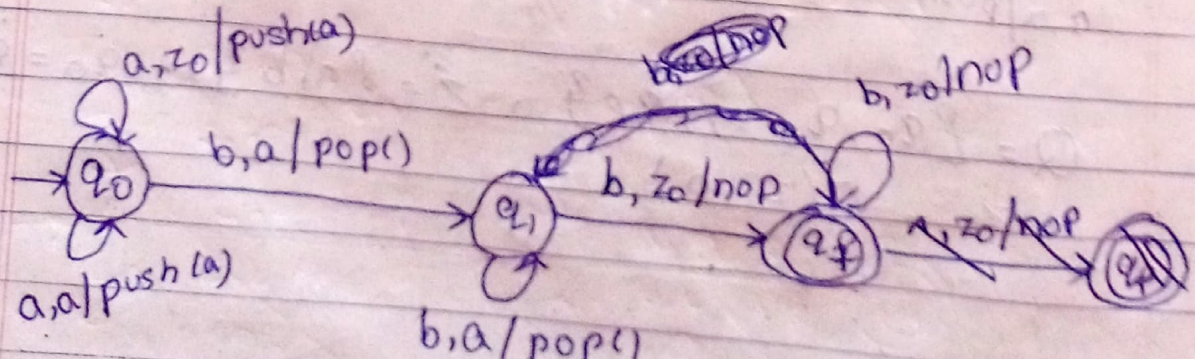
$m = 2 \quad n = 5 \quad aabbb$

$\delta =$

$\delta(q_0, a, z_0) \vdash \delta(q_0, a z_0)$
$\delta(q_0, a, a) \vdash \delta(q_0, aa)$
$\delta(q_0, b, a) \vdash \delta(q_1, \wedge)$
$\delta(q_1, b, a) \vdash \delta(q_2, \wedge)$
$\delta(q_2, b, z_0) \vdash \delta(q_f, z_0)$
$\delta(q_1, b, z_0) \vdash \delta(q_1, z_0)$
$\delta(q_1, b, z_0) \vdash \delta(q_1, z_0)$
$\delta(q_2, \wedge, z_0) \vdash q_f$

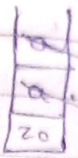


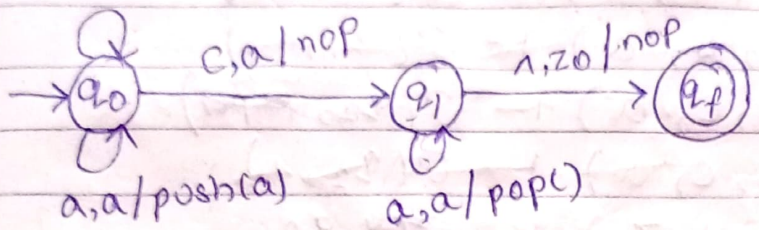
$Q = \{q_0, q_1, q_2, q_f\} \quad q_0 = q_0 \quad \Sigma = \{a, b\}$
$z_0, V(z_0, a)$



Q. $L = \{a^n c a^n / n \geq 1\}$

$\omega = aacaaa$





$Q = \{q_0, q_1, q_f\}$, $\Sigma = \{a, c\}$, $q_0 = q_0$, z_0

$V(z_0, a)$

$\delta =$ $\delta(q_0, a, z_0) \vdash \delta(q_0, aa)$

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

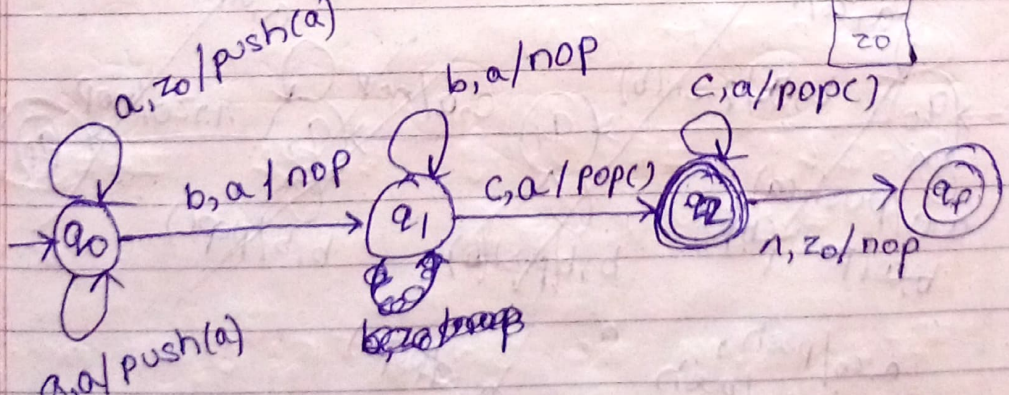
$\delta(q_0, c, aa) \vdash \delta(q_1, a)$

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

$\delta(q_1, \lambda, z_0) \vdash q_f$

Q. Design a PDA that accepts all the string of the form $a^n b^m c^n$, $m, n \geq 1$

$\omega = aabbbcc$



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

z_0, $V(z_0, a)$

δ =

$\delta(q_0, a, z_0) \vdash \delta(q_0, az_0)$

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

$\delta(q_0, b, a) \vdash \delta(q_1, aa)$

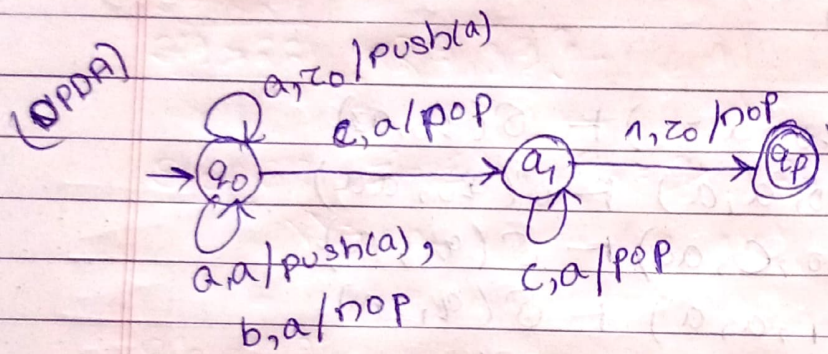
~~$\delta(q_1, b, a) \vdash \delta(q_1, aa)$~~

$\delta(q_1, c, a) \vdash \delta(q_2, pop(a))$

~~$\delta(q_2, c, a) \vdash \delta(q_2, \wedge)$~~

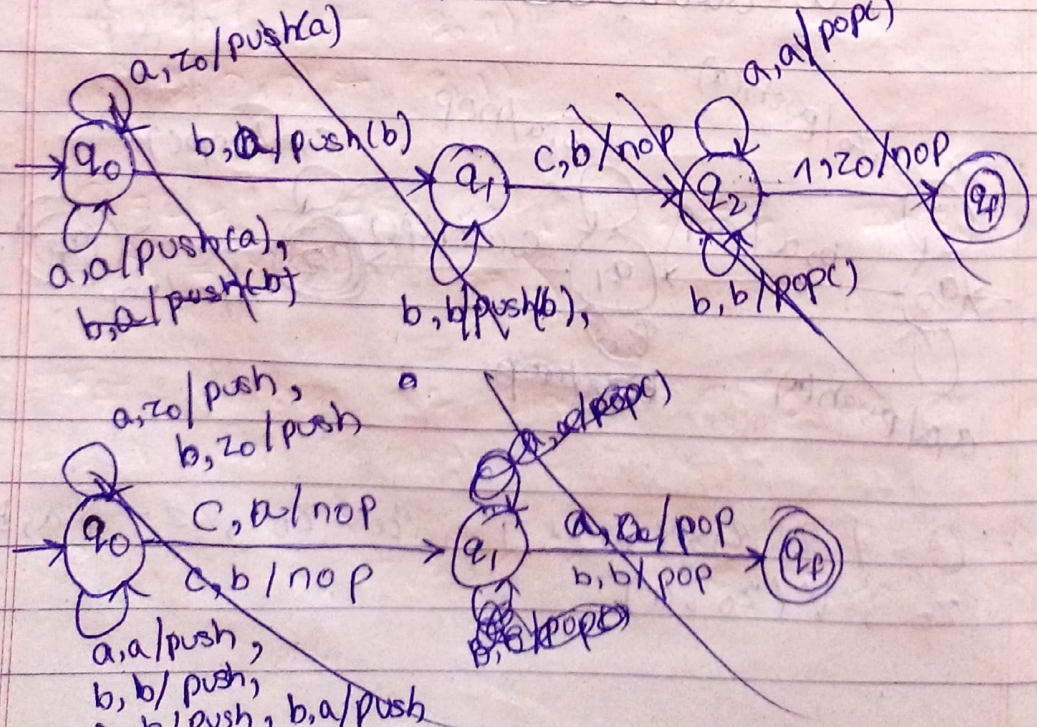
$\delta(q_2, \wedge, z_0) \vdash$ ~~$\delta(q_2, \wedge)$~~ q_f

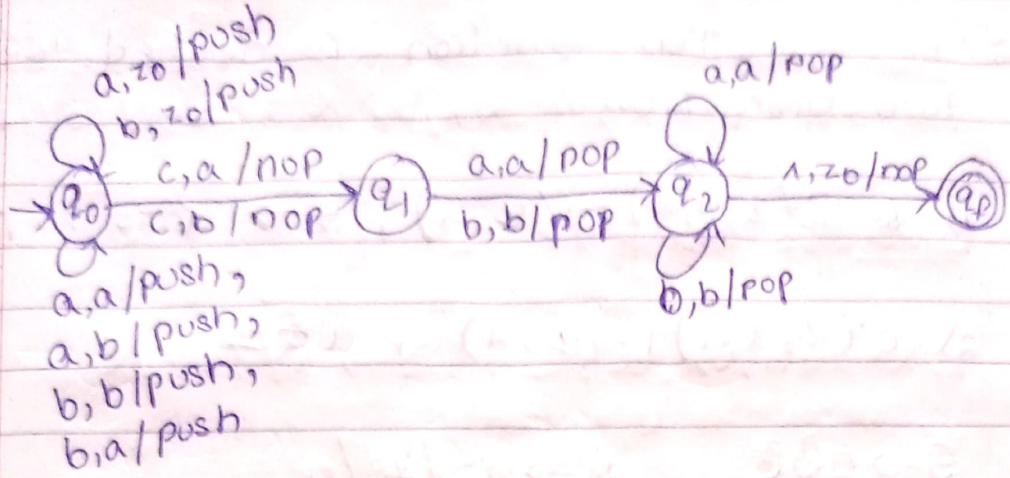
Q. $a^n b^m c^n$ / $m \geq 0, n \geq 1$



Q. Design a PDA that accepts $\{L = \omega c \omega^T / \omega \in \{a,b\}^* \}$

$\omega = abbCbba$





$Q = \{q_0, q_1, q_2, q_f\}$, $\Sigma = \{a, b, c\}$, $q_0 = q_0$, z_0 , $\Gamma(z_0, a, b)$

$\delta =$ $\delta(q_0, a, z_0) \vdash \delta(q_0, az_0)$
$\delta(q_0, a, a) \vdash \delta(q_0, aa)$
$\delta(q_0, a, b) \vdash \delta(q_0, b)$
$\delta(q_0, b, z_0) \vdash \delta(q_0, b, z_0)$
$\delta(q_0, b, b) \vdash \delta(q_0, bb)$
$\delta(q_0, b, a) \vdash \delta(q_0, ba)$
$\delta(q_0, c, a) \vdash \delta(q_1, nop)$
$\delta(q_1, a, a) \vdash \delta(q_2, 1)$
$\delta(q_1, b, b) \vdash \delta(q_2, 1)$
~~$\delta(q_2, b, b) \vdash \delta(q_2,$~~
$\delta(q_2, 1, z_0) \vdash q_f$

CFG to PDA :-

Step 1: Convert the grammer into GNF

Step 2: Construct the set of states $Q = \{q\}$

Step 3: The input symbol are as ~~above~~ $\Sigma = \Sigma$ then the initial state is $q_0 = q$ final state $F = \emptyset$, $z_0 = s$, $\Gamma = (V \cup \Sigma)$

Step4: The transition function is defined using two rules. i.e.,

1) $\delta(q, \wedge, A) \vdash (q, \alpha)$, $A \to \alpha$ is in Production

2) $\delta(q, a, a) \vdash (q, \wedge)$, $a \in \Sigma$

Ex:- $S \to 0BB$ $\quad B \to 0S/1S/0$

Step 1: Grammer is already in GNF.

Step 2: $Q = \{q\}$

Step 3: $\Sigma = \Sigma$ $\quad q_0 = q$ $\quad F = \emptyset$ $\quad Z_0 = S$

$V =$

Step 4:

$\delta =$

$\delta(q, \wedge, S) \vdash (q, 0BB)$

$\delta(q, 1, B) \vdash (q, 0S), (q, 1S), (q, 0)$

$\delta(q, 1, 1) \vdash (q, \wedge)$

$\delta(q, 0, 0) \vdash (q, \wedge)$

$Q = \{q\}$ $\quad \Sigma = \{0, 1\}$ $\quad q_0 = q$ $\quad Z_0 = S$

$V = (S, B, 0, 1)$ $\quad F = \emptyset$

Q. Test whether the string 01^40 is accepted by a PDA or not.

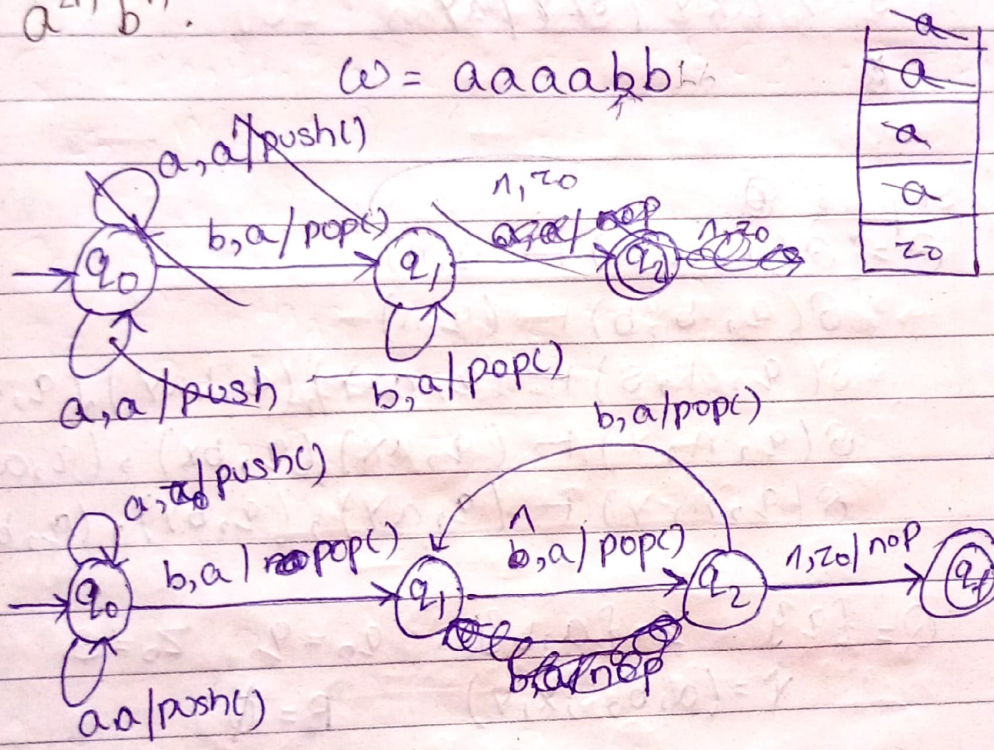
$\delta(q, 01^40, S) \vdash (q, 01^40, 0BB) \vdash (q, 1^40, BB)$

$\vdash (q, i^9 0, 1SB) \vdash (q, i^3 0, BBBB) \vdash (q, i^2 0, 1SB)$
$\vdash (q, i 0, 1SB) \vdash (q, 0, SB) \vdash$

∴ Hence the given string is not accepted by the PDA.

Q. Design a PDA that accepts the string $a^{2n} b^n$.

$\omega = aaaabb$



$Q = \{q_0, q_1, q_2, q_f\}$, $\Sigma = \{a, b\}$, $q_0 = q_0$, z_0, $\forall (z_0, a)$. $F = q_f$

δ: $\delta(q_0, a, z_0) \vdash (q_0, a z_0)$
$\delta(q_0, a, a) \vdash (q_0, aa)$
$\delta(q_0, b, a) \vdash (q_1, 1)$
$\delta(q_1, b, a) \vdash (q_2, 1)$
$\delta(q_2, b, a) \vdash (q_1, 1)$
$\delta(q_2, \Lambda, z_0) \vdash q_f$

CFG

$S \to xY$, $X \to aX/bx/a$ $Y \to aY/bY/b$

Step 1: The GNF is

$X \to aX/bx/a$ $Y \to aY/bY/b$

$S \to axY/bxY/aY$

Step 2:

$\delta = \{\delta(q,a,a) \vdash (q,n)$

$\delta(q,b,b) \vdash (q,n)$

$\delta(q,1,S) \vdash (q,axY),(q,bxY),(q,aY)$

$\delta(q,n,x) \vdash (q,ax),(q,bx),(q,a)$

$\delta[q,n,Y) \vdash (q,aY),(q,bY),(q,b)\}$

$Q = \{q\}$ $\Sigma = \{a,b\}$ $q_0 = q$ $Z_0 = S$

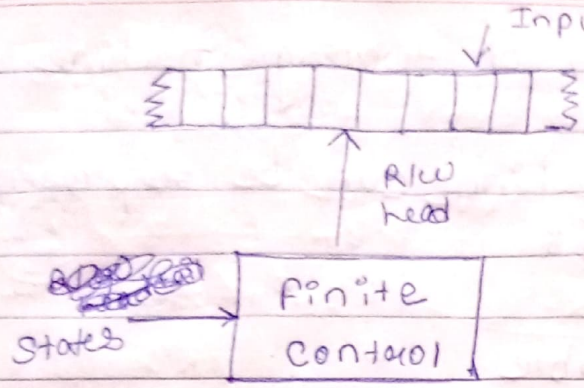
$\Gamma = (a,b,S,X,Y)$ $F = \phi$

Turing Machine :-

It is a simple mathematical model of general purpose computers to implement the computing power.

The turing machine accepts all the grammers or languages specifically the unrestricted grammer or type-0 language.

It is considered as a finite Automata with infinite tape and two way read write head.



Components are :
Input tape
R/W head (Read/write)
Finite Control

R/W head :- It indicates the present input symbol.
- A symbol under the read/write head can be changed to a new symbol.
- It moves in both the direction i.e., left or right which is indicated in the transition functions.

Tuples of a Turing Machine :

It is represented by 7 tuples. Those are :-

$TM = (Q, \Sigma, \delta, q_0, \nabla, b, F)$ → finite state

- Q : finite Set of States
- Σ : set of i/p Symbols
- δ : transition function
- q_0 : initial state
- ∇ : tape Symbols
- b : blank

$$\delta = Q \times \nabla \rightarrow Q \times \nabla \times \{L, R\}$$

- Q : present state
- ∇ : present tape
- Q : next state
- ∇ : new tape symbol
- $\{L, R\}$: R/W head

Representation of turing machine

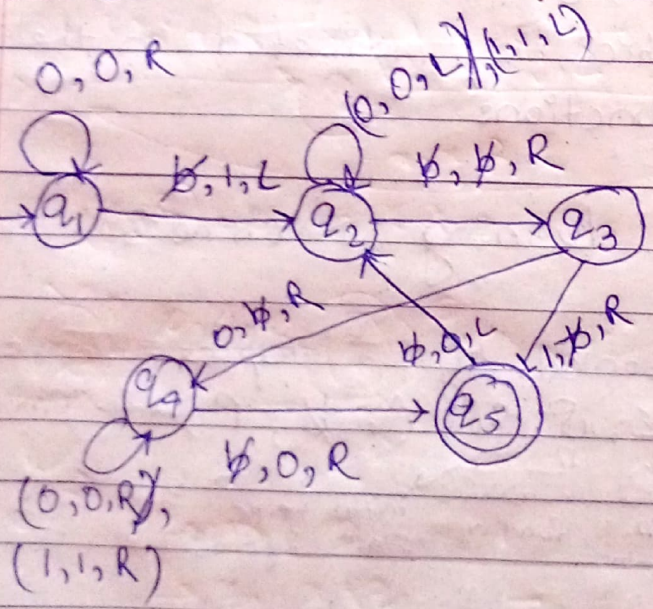
A turing machine can be represented in three ways:

i) Transition table
ii) Transition Diagram
iii) Instaneous Description (ID)

i)

Pstate	P. tape symbol		
	⊔	0	1
$\rightarrow q_1$	$1Lq_2$	$0Rq_1$	–
q_2	$\sqcup Rq_3$	$0Lq_2$	$1Lq_2$
q_3	–	$\sqcup Rq_4$	$\sqcup Rq_5$
q_4	$0Rq_5$	$0Rq_4$	$1Rq_4$
q_5	$0Lq_2$	–	–

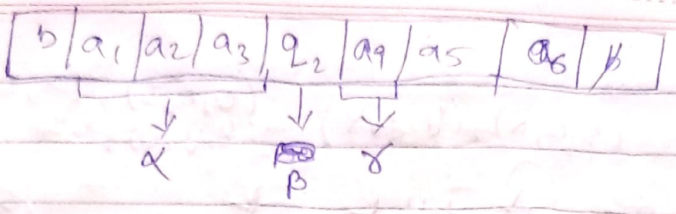
ii)



$Q = \{q_1, q_2, q_3, q_4, q_5\}$
$\Sigma = \{\sqcup, 0, 1\}$
$q_0 = q_1$
$F = \{q_5\}$

Q.

iii) An ID of a turing machine is a string in the form of α, β, γ where β is present state, γ is present symbol and α is path of the string which is already read.



$a_1 a_2 a_3 \cdots a_i\ q\ a_{i+1}\ a_{i+2} \cdots a_n$

Moves in a TM

The changes in the ID of a Turing machine is called as transition or move of a TM.

$\delta(q, x_i) \vdash (p, y, L/R)$

present state — present symbol

$\delta(q, x_i) \vdash (p, y, R)$

ID

$x_1 x_2 \cdots x_{i-1}\ q\ x_i\ x_{i+1} \cdots x_n$

$\vdash x_1 x_2 \cdots x_{i+1}\ y\ p\ x_{i+1} \cdots x_n$

Q. Design a TM that accepts the language $L = \{0^n 1^n\}$

$\not b\ 0011 \vdash x\ 011 \vdash x0\ 11 \vdash x0y\ 1 \vdash x0y1$

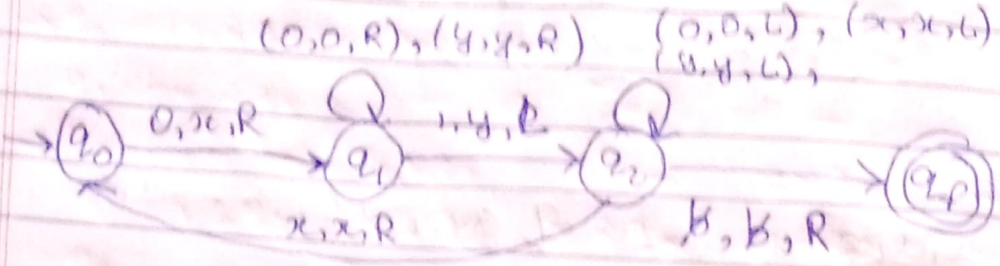
$q_0 \quad q_1 \quad q_1 \quad q_2$

$\vdash x0y1 \vdash x0y1 \vdash xxy1 \vdash xxy1$

$q_2 \quad q_0 \quad q_1 \quad q_1$

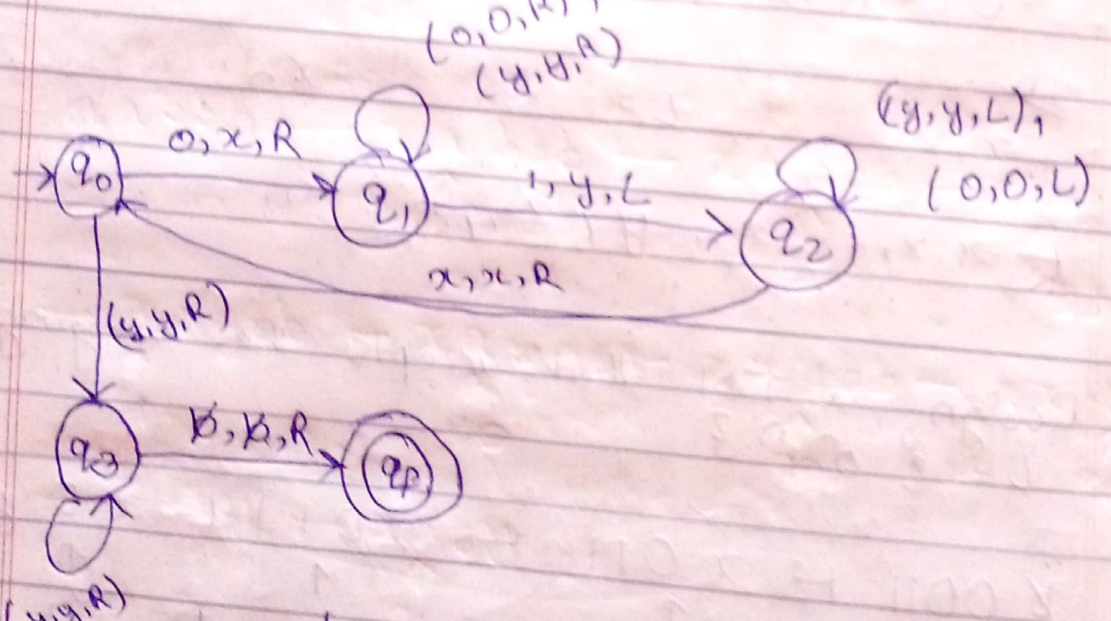
$\vdash xxyy \vdash xxyy \vdash xxyy \vdash \not b\ xxyy$

$q_2 \quad q_2 \vdash q_f \quad q_2$



Next State, I/P Symbol

PStates	0	1	B	x	y
	~~xRq1~~	—			
→q0	xRq1	—	—	—	—
q1	0Rq1	yLq2	—	—	yRq1
q2	0Lq2		BRqf	xRq0, xLq2	yLq2
(qf)	—	—	—	—	—



	0	1	B	x	y
→q0	xRq1	—	—	—	y,R,q3
q1	0Rq1	yLq2	—	—	yRq1
q2	0Lq2	—	—	xRq0	yLq2
q3	—	—	BRqf	—	yRq3
(qf)	—	—	—	—	—

Q. $L = \{ a^{2^n} b^n \mid n \geq 1 \}$

$aaaabb \vdash xaaabb \vdash xaaabb \vdash xxaabb$

$q_0 \quad q_1 \quad q_2 \quad q_2$

$\vdash xxaabb \quad xxaabb \vdash xxaaybb$

$q_2 \quad q_2 \quad q_3$

$\vdash xxaaybb \vdash xxaaybb \vdash xxaaybb$

$q_3 \quad q_3 \quad q_0$

$\vdash xxxaybb \vdash xxxxybb \vdash xxxxybb$

$q_1 \quad q_2 \quad q_3$

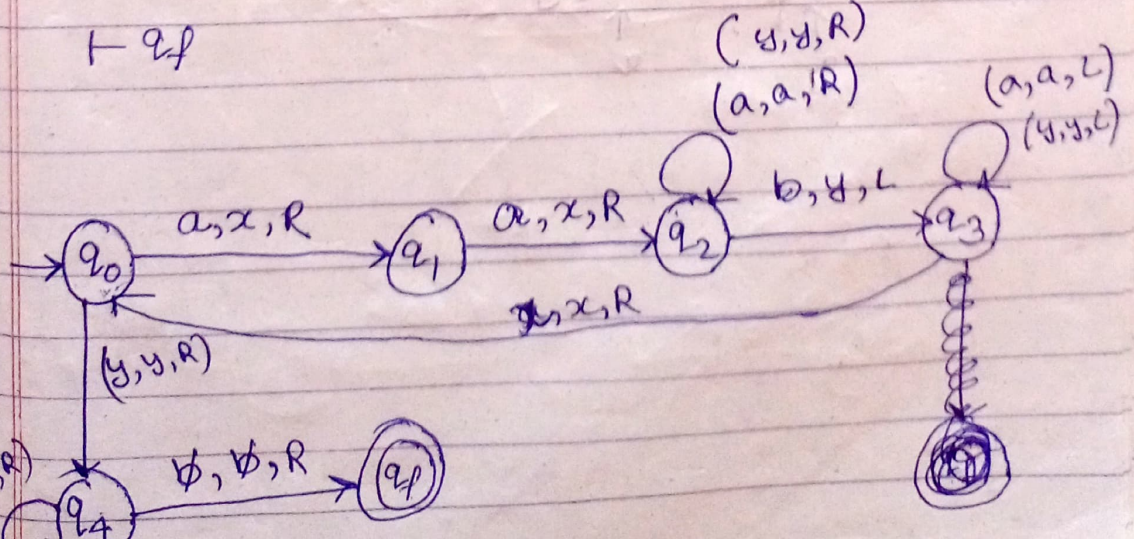
$\vdash xxxxyyb \quad xxxxyyb \vdash xxxxyyb$

$q_3 \quad q_0 \quad q_4$

$\vdash xxxxyyb \vdash xxxxyyb \vdash xxxxyyb$

$q_4 \quad q_4 \quad q_f$

$\vdash q_f$



δ =	a	b	x	y	⊔
$\rightarrow q_0$	x, R, q_1	–	–	y, R, q_4	–
q_1	$x R q_2$	–	–	–	–
q_2	$a R q_2$	y, L, q_3	–	$y R q_2$	–
q_3	$a L q_3$	~~$y L q_3$~~	$x R q_0$	$y L q_3$	–
q_4	–	–	–	$y R q_4$	$\sqcup R q_f$
(q_f)	–	–	–	–	–

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

$\Sigma = \{a, b, x, y, \sqcup\}$

$q_0 = q_0$ $\quad F = \{q_f\}$ $\quad \Gamma = \{a, b, x, y, \sqcup\}$

$\delta \; \sqcup = \sqcup$

Q. $L = \{a^n b^{n+1} / n \geq 1\}$

$L = aabbb$

$aabbb \vdash xabbb \vdash xabbb \vdash xaybb$

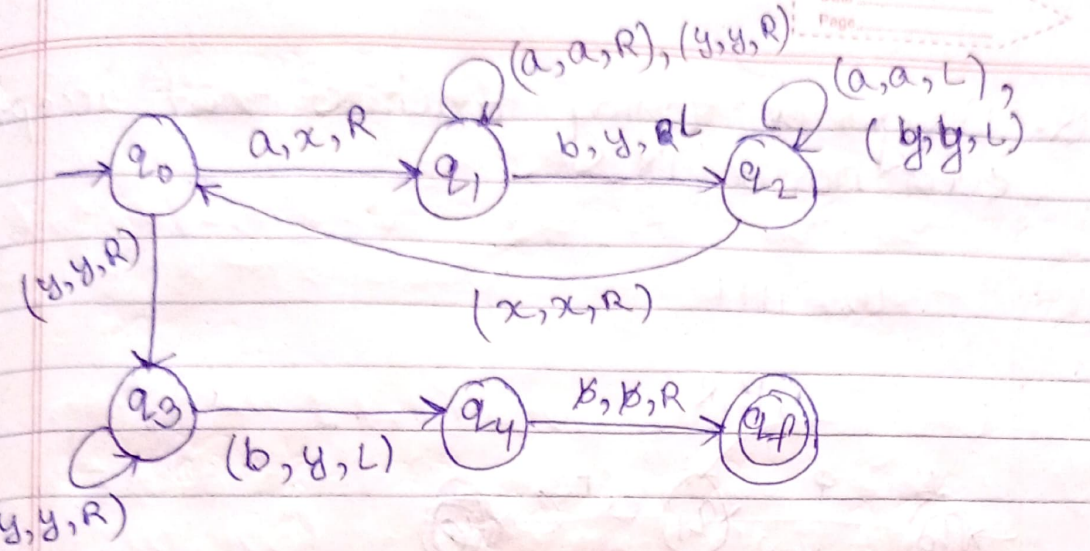
$q_0 \quad q_1 \quad q_1 \quad q_1$

$\vdash xaybb \vdash xabbb \vdash xxybb \vdash xxybb$

$q_1 \quad q_0 \quad q_1 \quad q_0$

$\vdash xxyy \vdash xxyy$

$q_1 \quad q_2$



$abb \vdash xbb \vdash xyb \vdash xyb \vdash xyb\not{b}$

$\vdash xyy\not{b} \vdash$ ~~$xyy\not{b}q_f$~~

$\delta =$

	a	b	x	y	$\not{b}$
$\to q_0$	xRq_1	–	–	yRq_3	–
q_1	aRq_1	yLq_2	–	yRq_1	–
q_2	aLq_2	–	xRq_0	yLq_2	–
q_3	–	yLq_4	–	yRq_3	–
q_4	–	–	–	–	$\not{b}Rq_f$
(q_f)	–	–	–	–	–

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

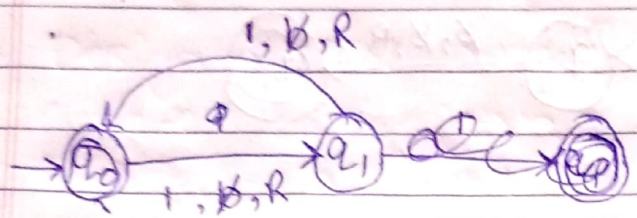
$\Sigma = \{a, b, x, y, \not{b}\}$

$q_0 = q_0 \quad F = \{q_f\} \quad \Gamma = \{a, b, x, y, \not{b}\}$

$\not{b} = \not{b}$

Q. Design a turing machine that accepts even no. of 1's.

$w = 1111$

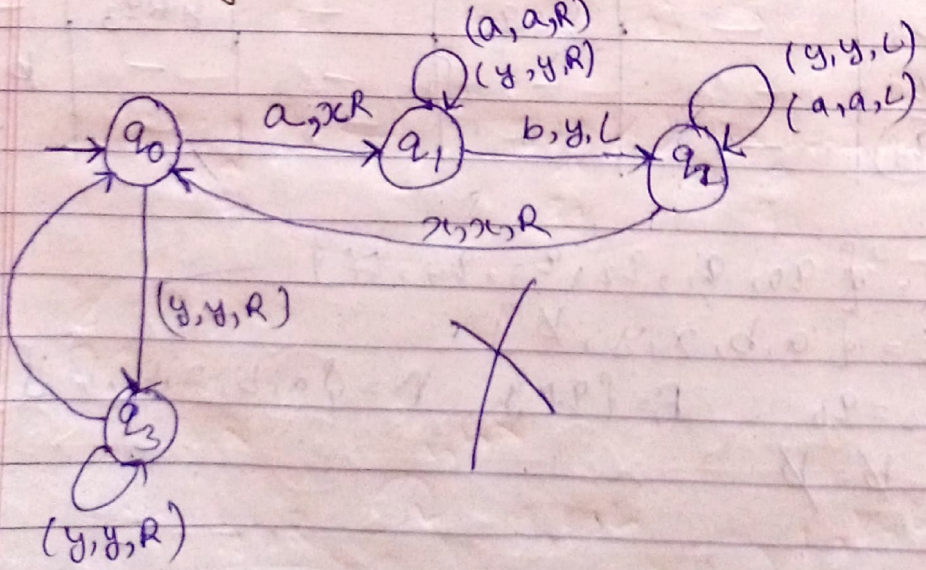


$1111 \vdash B111 \vdash BB11 \vdash BBB1 \vdash BBBB$

$q_0 \quad q_1 \quad q_0 \quad q_1 \quad q_2$

$\delta =$	1
$\rightarrow q_0$	BRq_1
q_1	BRq_0

Q. Design a TM that accepts all the strings consisting of equal no. of a's & b's

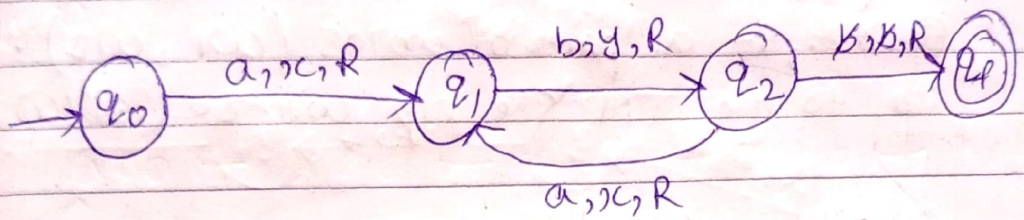


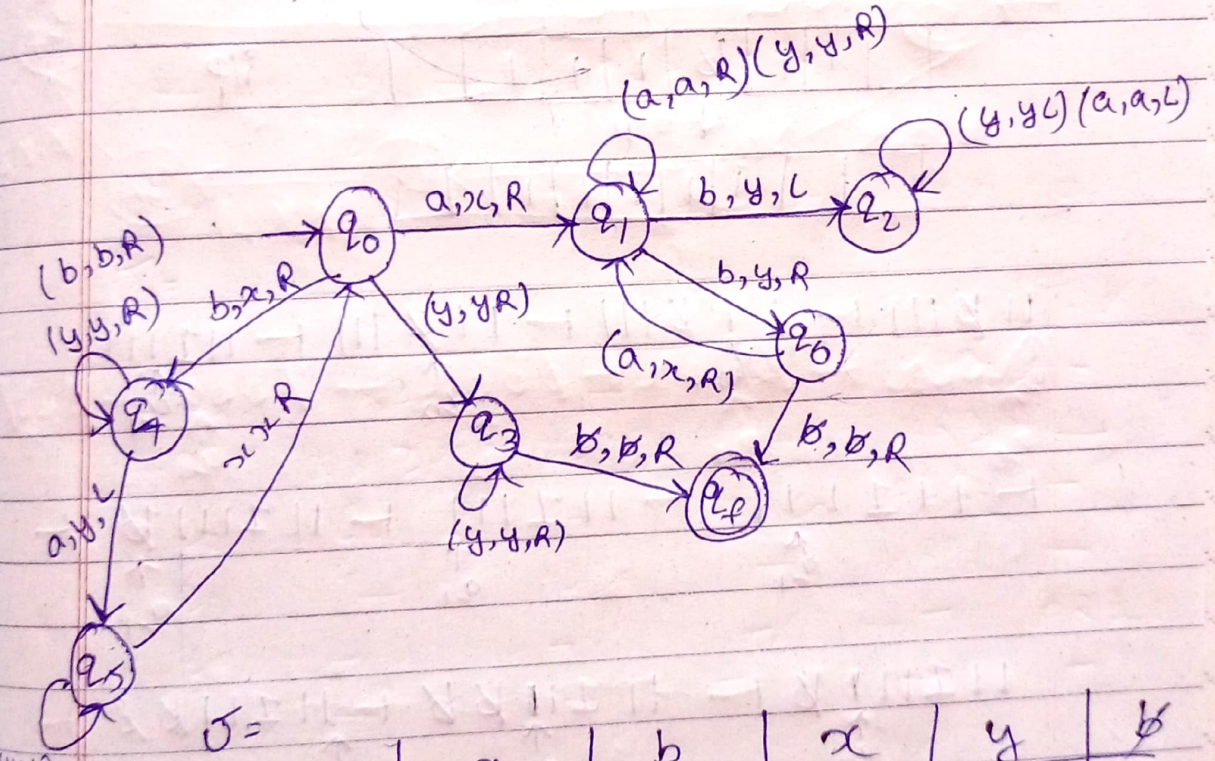
$abab \vdash xbab \vdash xyab \vdash xyab$

$q_0 \quad q_1 \quad q_2 \quad q_2$

$\vdash xyab \vdash xyab \vdash xyab \vdash xyxb$

$q_2 \quad q_0 \quad q_3 \quad q_f$





$\delta =$

	a	b	x	y	b
$\to q_0$	xRq_1	xRq_4		yRq_3	
q_1	aRq_1	yLq_2		yRq_1	
q_2					
q_3					
q_4					
q_5					
q_6					
q_f					

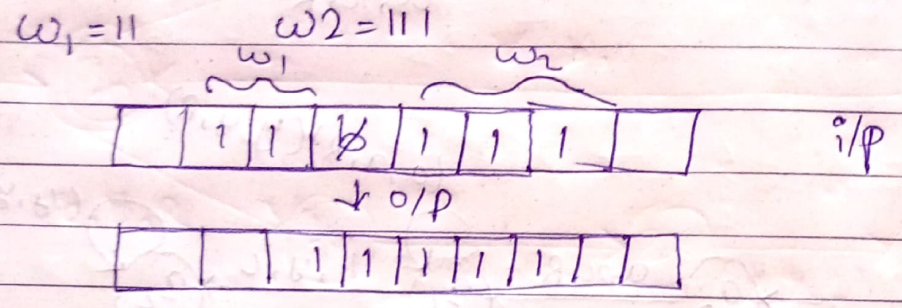
$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_f\}$

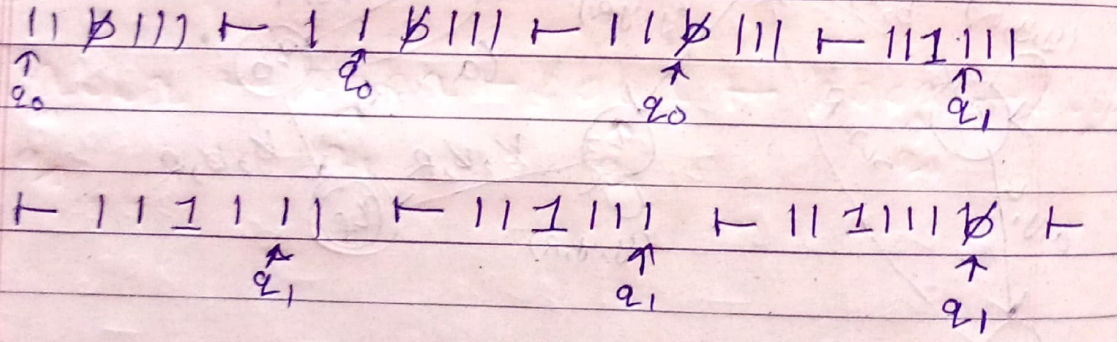
$\Sigma = \{a, b, x, y, \not b\}$

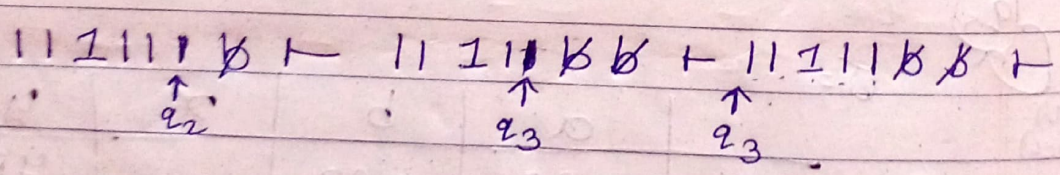
$q_0 = q_0 \quad F = \{q_f\} \quad \Gamma = \{a, b, x, y, \not b\}$

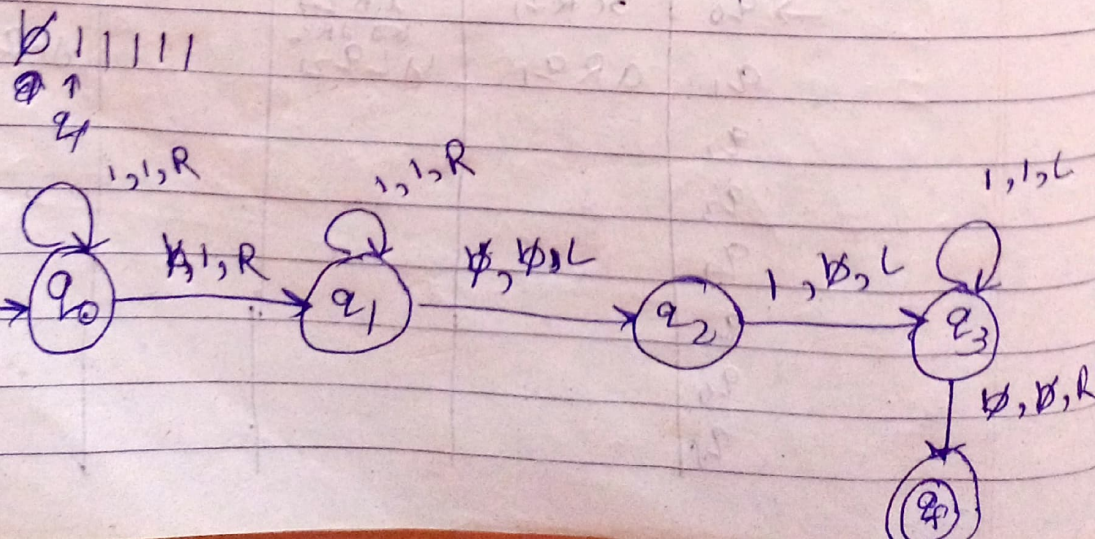
$\not b = \not b$

Q. Design a TM over $\{1, \not b\}$ which can compute concatenation function over $\Sigma = \{1\}$ i.e., if a pair of words (w_1, w_2) is the i/p to the TM the o/p will be $w_1 w_2$.









δ =

	1	$\not b$
$\to q_0$	$1Rq_0$	$1Rq_1$
q_1	$1Rq_1$	$\not b Lq_2$
q_2	$\not b Lq_3$	-
q_3	$1Lq_3$	$\not b Rq_4$
(q_4)	-	-

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

$\Sigma = \{1, \not b\}$

$q_0 = q_0 \quad F = \{q_4\}$

$\Gamma = \{1, \not b\}$

1Q. Construct a TM that can accepts the set of all even palindrome's over the set $\Sigma = \{a, b\}$.

2Q. Construct a TM that accepts all strings of the form $1^n 2^n 3^n$.