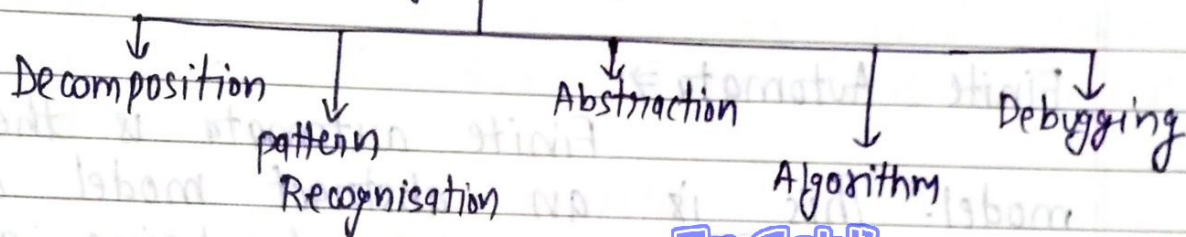


Theory of Computation:—

The theory of computation is a branch of computer science that deals with how efficiently a problem can be solved on a model of computation using an algorithm. The field is divided into 3 major branches:—

- (i) Autodata theory & language
- (ii) Computability theory



- (iii) Complexity theory

Er Sahil
Ka
Gyan



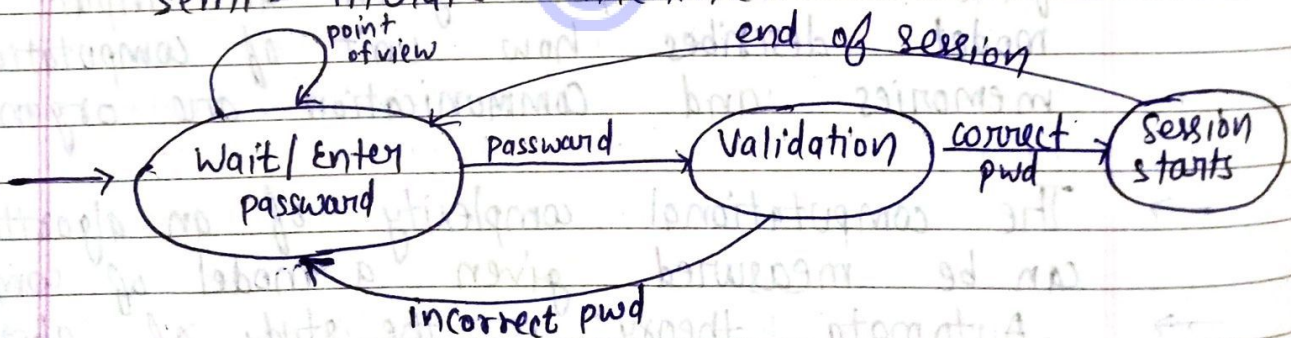
A model of computation is a model which describes how an output of a mathematical function is computed given an input. A model describes how unit of computation, memories and communication are organised.

- The computational complexity of an algorithm can be measured given a model of computation.
- Automata theory is the study of abstract machine and problem which they are able to solve. A typical abstract machine consist of a defination in term of input, output and set of allowable operations used to turn the former into the later.

→ Computation thinking allow us to take a complex problem then understand what the problem is and develop possible solution. We can then present these solution in a way that the computer, consumer and both can understand. In English we have letters, words and sentences. Not all collection of letters form a valid word, Not all collection of words form a valid sentence.

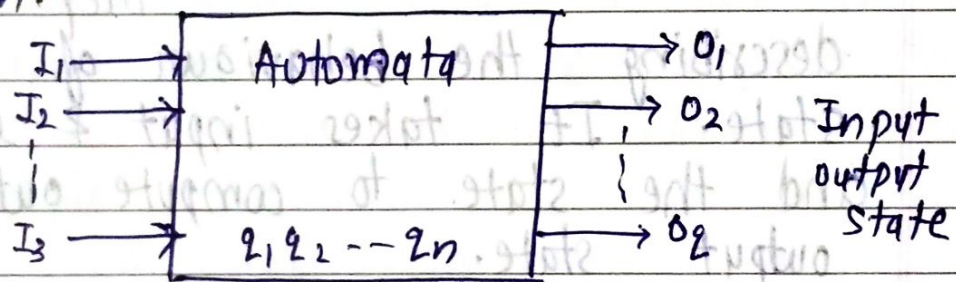
✶ Finite Automata ⇒

Finite automata is the simplest model. This is an abstract model of digital computer. The meaning of being abstract is that theoretically it is closed to the computation system and formally describe the behavior of the computer system. It is nearly a one way semi-literate machine.



Automaton definition ⇒ An automaton is self operating machine. The word automaton is derived from greek word "Acting of ones own will". It is more often used to describe non

electronic moving machines especially those that have been made to assemble human or animal actions. It is the study of Abstract machine and the problems which they are able to solve these abstract machines are called Automata languages are accepted by an automaton when it accept all of the string in the language & no other. The most restricted family of automata are finite automata consisting of only a finite no of state & a read only tape containing the I/P to be read in one direction.



Characteristics of Automaton \Rightarrow

- (i) Input at each of the discrete instant of time t_1, t_2, \dots, t_n the input value I_1, I_2, \dots, I_p each of which can take a finite no. of fixed values from the input alphabet Σ .
- (ii) Output O_1, O_2, \dots, O_q are the outputs of the model each of which can take a finite no. of fixed values.
- (iii) States at any instants of time the automaton can be a one of the states

q_1, q_2, \dots, q_n

(iv) States relation - the next state of an automaton at any instant of time is determined by the present state & present input.

(v) Output relation - output is related to either state only or to both the input and the state.

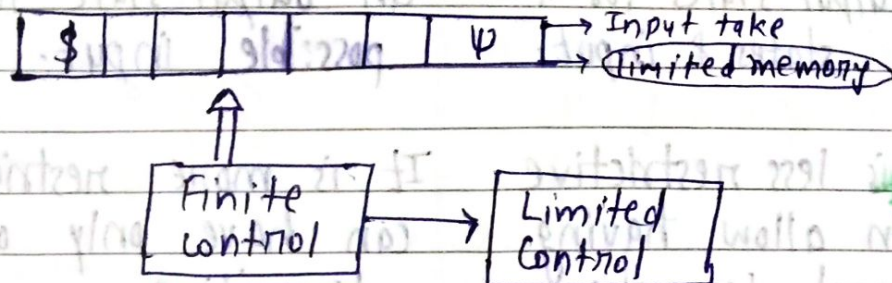
Finite Automaton:-

They are one way of describing the behaviour of a circle with state. It takes input & uses those inputs and the state to compute output string & output state.

The finite automaton is mathematical model of a machine that has fixed a finite capacity. It accepts a particular language over sets. It can be used as language recognizer that is why it is called an acceptor or language recognizer.

A finite automaton has an input tape, read only head & a finite control. It receives its input as a string & delivers to an input tape without delivering any output. This means it is not capable of writing so it can't write the output by any means. This is the major limitation.

of this model. This model is used for recognizing purpose only not for computing purpose.



Mathematical Definition:—

An automata with a set of state and its control moves from state to state in response to external inputs is called a finite Automata. It is a collection of 5 elements to form a quintuple.

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

Q = Finite non empty set of state

Σ = Finite non empty set of inputs called as input Alphabets.

δ = Transition function that makes $Q \times \Sigma \rightarrow Q$ to determine the next state.

q_0 = Initial State i.e. $q_0 \in Q$

F = Final state set i.e. $F \subseteq Q$

Types of Automata:—

There are 2 type of Automata

- (i) Deterministic FA
- (ii) Non - Deterministic FA

NDEFA

DFA

(i) It may have multiple no. of output states for a given states & input.

It has exactly one state as an output state for each possible input.

(ii) It is less restrictive as an allow having several transition.

It is more restrictive as it can have only one transition

(iii) It is more powerful.

It is less powerful.

* Deterministic Finite Automata:-

The FA is called DFA if there is only one path for a specific input from current state to next state.
DFA is described by 5 tuple i.e.
 $M = (Q, \Sigma, \delta, q_0, F)$

Q = Finite ~~state~~ non empty set of state

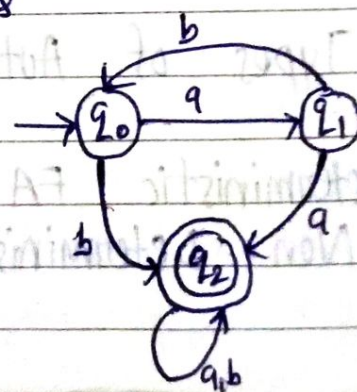
Σ = Finite non empty inputs called as input Alphabet.

δ = Transition function that makes $Q \times \Sigma \rightarrow Q$

q_0 = Initial state i.e. $q_0 \in Q$

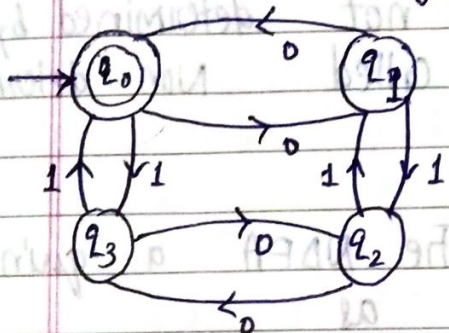
F = Final state $F \subseteq Q$

State	Input	
	a	b
q_0	q_1	q_2
q_1	q_2	q_0
q_2	q_2	q_2



Acceptability of a string by Finite Automata \Rightarrow

Consider the finite automata & check the acceptability of a string.



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

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

$$q_0, F = q_1$$

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

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

$$\delta(q_1, 0) = q_0, \delta(q_3, 0) = q_0$$

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

States	Input	
	0	1
$\rightarrow q_0$	q_1	q_3
q_1	q_0	q_2
q_2	q_3	q_1
q_3	q_2	q_0

Acceptability of 0101 string

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

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

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

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



Acceptability of 0111 string

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

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

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

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

Not acceptable string



Non-Deterministic Finite Automata:-

(N DFA)

Non-deterministic

means a choice of moves for an automaton rather than prescribing a unique move in each situation. We allow a set of possible move. We achieve this by defining the transition function so that its range is

a set of possible states. So we can say that in NFA. When an input is read the automata each step may choose the ~~both~~ go to any of possible several states. Since this choice is not determined by anything therefore it is called Non-deterministic.

Mathematical definition:-

The NFA a quintuple and can be defined as

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

Q = Finite non-empty set of state

Σ = Finite non-empty set of input

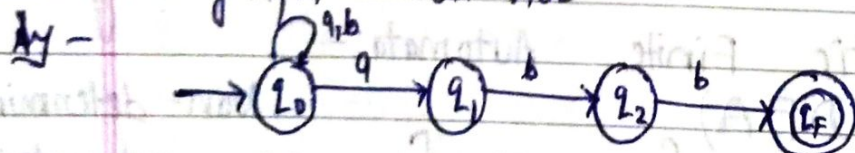
q_0 = Initial state i.e. $q_0 \in Q$

F = Final state i.e. $F \subseteq Q$

δ = Transition function that takes a state s in Q and an input symbol in Σ as an argument and return a subset of Q . i.e.

$$Q * \Sigma \rightarrow 2^Q$$

Q. check ^{whether} the strings are accepted, by transition graph on not.



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

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

$$I.S. = q_0$$

$$F = q_f$$

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

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

$$\begin{aligned} \delta(q_1, a) &= \emptyset, & \delta(q_2, a) &= \emptyset \\ \delta(q_1, b) &= q_2, & \delta(q_2, b) &= q_F \\ \delta(q_F, a) &= \emptyset \\ \delta(q_F, b) &= \emptyset \end{aligned}$$

States	Input	
	a	b
$\rightarrow q_0$	q_0, q_1	q_0
q_1	\emptyset	q_2
q_2	\emptyset	q_F
(q_F)	\emptyset	\emptyset

Er Sahil
Ka
Gyan

Acceptability of abb string

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

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

$$\delta(q_0, b) = q_0 \text{ or } \pi$$

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

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

Acceptability of abbabb string:—

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

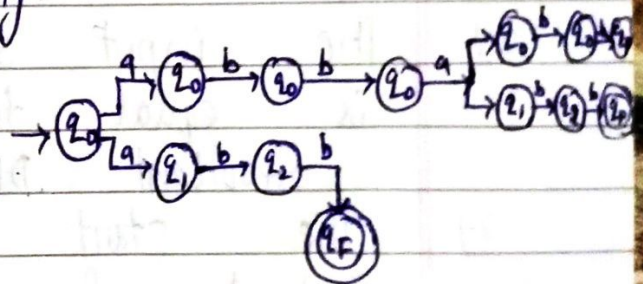
$$\delta(q_0, b) = q_0$$

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

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

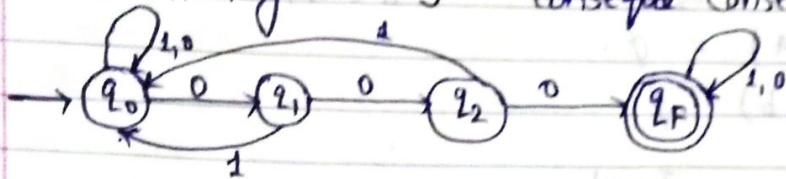
$$\delta(q_1, b) = q_2$$

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



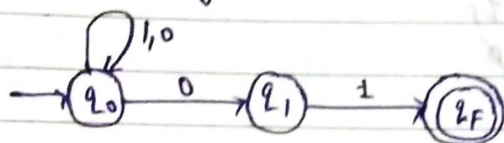
Q. Construct a NDEFA to accept those strings containing 3 consecutive consecutive zeros.

Ans -



Q. Construct a NDEFA to accept all strings terminating zero, one.

Ans -



Conversion of NDEFA to DFA \Rightarrow

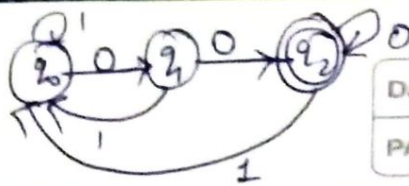
The conversion method has following steps: -

$M = (Q, \Sigma, \delta, q_0, F)$ is a NDEFA and $M' = (Q', \Sigma', \delta', q'_0, F')$ is the resultant DFA.

1.) The input symbol of the given NDEFA is equal to input symbol for the resultant DFA

2.) The start state of NDEFA is equal to start of the DFA then find the transition from this start state.

3.) To find the states of the DFA, we will make set of all states of given NDEFA and from these states we can find out the states of the DFA which are the subset of the given NDEFA



DATE : / /

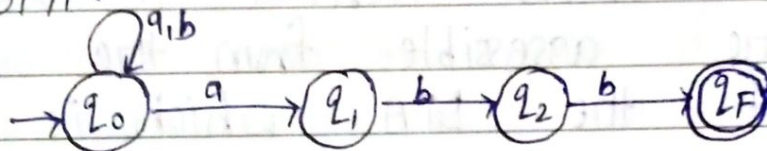
PAGE NO. :

i.e. if NFA has n state then DFA has 2^n state.

4.] Now from the states of the DFA, we have to select only assessible state.

5.] In the search for the final states for the resultant DFA, we make all assessible state as the final state which content atleast one final state of NFA.

Q. Consider the NFA and find its equivalent DFA.



Ans- Here the state transition diagram is given so it must be converted into state transition table.

State	Input	
	a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	q_0
q_1	\emptyset	q_2
q_2	\emptyset	q_F
(q_F)	\emptyset	\emptyset

(i) Input symbol for the resultant DFA is equal to input symbol for the given NFA = $\{a, b\}$
 $\Sigma = \Sigma'$

(ii) Start state of the resultant DFA = set of start state of given NFA = q_0

Step-3 Since there are 4 states in the given NFA
So the no. of states in resultant DFA
= 24

All the states of the resultant DFA
= $\emptyset, q_0, q_1, q_2, q_F, [q_0, q_1], [q_0, q_2],$
 $[q_0, q_F], [q_1, q_2], [q_1, q_F], [q_2, q_F],$
 $[q_0, q_1, q_2], [q_0, q_1, q_F], [q_0, q_2, q_F]$
 $[q_1, q_2, q_F], [q_0, q_1, q_2, q_F]$ (16)

Step-4 Now we will find the accessible states
from the states derived in step-3 i.e.
which are accessible from the start
state of the DFA which is q_0 in
this case.

$$\delta(q_0, a) = [q_0, q_1]$$

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

$$\delta([q_0, q_1], a) = \delta(q_0, a) \cup \delta(q_1, a) = [q_0, q_1] \cup \emptyset$$

$$\delta([q_0, q_1], b) = [q_0, q_1]$$

$$\delta([q_0, q_1], b) = \delta(q_0, b) \cup \delta(q_1, b) = q_0 \cup q_2$$

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

States	Input	
	a	b
$\rightarrow q_0$	$[q_0, q_1]$	q_0
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_2]$

$$\delta([q_0, q_2], a) = \delta(q_0, a) \cup \delta(q_2, a) = [q_0, q_1]$$

$$\delta([q_0, q_2], b) = \delta(q_0, b) \cup \delta(q_2, b) = [q_0, q_1] \cup [q_F]$$

States	Input	
	a	b
$\rightarrow q_0$	$[q_0, q_1]$	q_0
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_2]$
$[q_0, q_2]$	$[q_0, q_1]$	$[q_0, q_F]$

$$\delta([q_0, q_1], a) = \delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_F, a)$$

$$= [q_0, q_1] \cup \emptyset = [q_0, q_1]$$

$$\delta([q_0, q_F], b) = \delta(q_0, b) \cup \delta(q_F, b)$$

$$= q_0 \cup \emptyset = q_0$$

State	Input	
	a	b
$\rightarrow q_0$	$[q_0, q_1]$	q_0
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_2]$
$[q_0, q_2]$	$[q_0, q_1]$	$[q_0, q_F]$
$[q_0, q_F]$	$[q_0, q_1]$	q_0

Now we have no choice of the next step to be considered as we don't get any new state and we stop our processing.

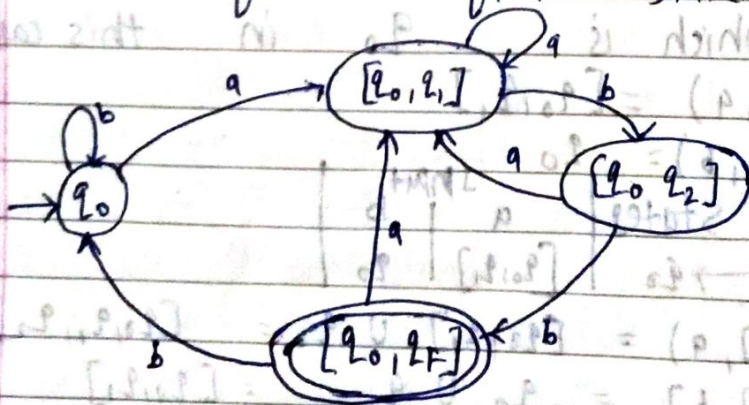
$$Q' = \{[q_0], [q_0, q_1], [q_0, q_2], [q_0, q_F]\}$$

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

$$I.S. = q_0$$

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

Step-5 Final step for the resultant DFA is those accessible state that consists at least one of the final state of the given NFA.



Q.

States	Input	
	a	b
$\rightarrow q_0$	$\{q_0, q_1, q_3\}$	q_0
q_1	q_2	q_1
q_2	q_3	q_3
(q_3)	\emptyset	q_2

Find equivalent
DFA.

OR

CONVERT NFA to DFA

Soln:- Step-① Input symbol for the resultant DFA = Input symbol for given NFA = $\{a, b\}$

Step-② Start state of DFA = set of start state of NFA = q_0

Step-③ Since there are 4 state in the given NFA
so no. of state in resultant DFA = 2^4

All the states of resultant DFA
= $\emptyset, q_0, q_1, q_2, q_3, [q_0, q_1], [q_0, q_2], [q_0, q_3],$
 $[q_0, q_1, q_2], [q_0, q_1, q_3], [q_0, q_2, q_3], [q_1, q_2, q_3]$
 $[q_1, q_2], [q_1, q_3], [q_2, q_3], [q_0, q_1, q_2, q_3]$

Step-④ Now we will find the assessible states from the states derived in step ③ i.e. which are assessible from the start state of DFA which is q_0 in this case

$$\delta(q_0, a) = [q_0, q_1]$$

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

States	Input	
	a	b
$\rightarrow q_0$	$[q_0, q_1]$	q_0

$$\delta([q_0, q_1], a) = [q_0, q_1] \cup q_2 = [q_0, q_1, q_2]$$

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

States	Input	
	a	b
$\rightarrow q_0$	$[q_0, q_1]$	q_0
$[q_0, q_1]$	$[q_0, q_1, q_2]$	$[q_0, q_1]$

$$\delta([q_0, q_1, q_2], a) = [q_0, q_1] \cup q_2 \cup q_3 = [q_0, q_1, q_2, q_3]$$

$$\delta([q_0, q_1, q_2], b) = q_0 \cup q_1 \cup q_3 = [q_0, q_1, q_3]$$

States	Input	
	a	b
$\rightarrow q_0$	$[q_0, q_1]$	q_0
$[q_0, q_1]$	$[q_0, q_1, q_2]$	$[q_0, q_1]$
$[q_0, q_1, q_2]$	$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_3]$

Case-I

$$\delta([q_0, q_1, q_2, q_3], a) = [q_0, q_1] \cup q_2 \cup q_3 \cup \emptyset = [q_0, q_1, q_2, q_3]$$

$$\delta([q_0, q_1, q_2, q_3], b) = q_0 \cup q_1 \cup q_3 \cup q_2 = [q_0, q_1, q_2, q_3]$$

States	Input	
	a	b
$\rightarrow q_0$	$[q_0, q_1]$	q_0
$[q_0, q_1]$	$[q_0, q_1, q_2]$	$[q_0, q_1]$
$[q_0, q_1, q_2]$	$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_3]$
$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_2, q_3]$

Case-II

$$\delta([q_0, q_1, q_3], a) = [q_0, q_1] \cup q_2 \cup \emptyset = [q_0, q_1, q_2]$$

$$\delta([q_0, q_1, q_3], b) = q_0 \cup q_1 \cup q_2 = [q_0, q_1, q_2]$$

States	Input	
	a	b
$\rightarrow q_0$	$[q_0, q_1]$	q_0
$[q_0, q_1]$	$[q_0, q_1, q_2]$	$[q_0, q_1]$
$[q_0, q_1, q_2]$	$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_3]$
$[q_0, q_1, q_3]$	$[q_0, q_1, q_2]$	$[q_0, q_1, q_2]$

Step-5

States $\rightarrow q_0$	Input	
	a	b
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_1, q_2]$	$[q_0, q_1, q_2]$
$[q_0, q_1, q_2]$	$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_2]$
$[q_0, q_1, q_2]$	$[q_0, q_1, q_2]$	$[q_0, q_1, q_2]$
$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_2, q_3]$

NDFA with NULL(ϵ) moves \Rightarrow

A NDFA can be enhanced to accept NULL string it means that a transition function of a NDFA $\text{move}(Q \times \Sigma^* \rightarrow Q^S)$

Method for converting NDFA with NULL to DFA \Rightarrow

Step-1 ϵ -closure(q) :- Set of states which are reachable from state q on NULL input including state q . It is equivalent to one state of equivalent DFA.

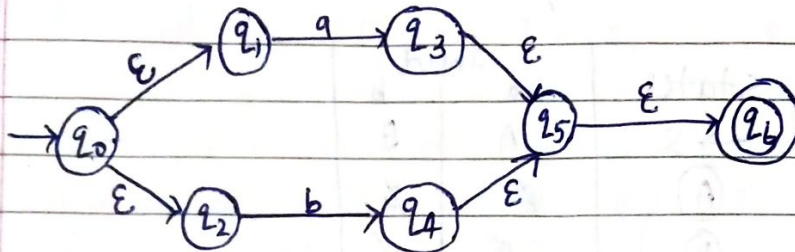
Step-2 We will obtain transition on each input move $(q, a) =$ Set of reachable states on input a from state q .

ϵ -closure($\text{move}(q, a)$) \equiv Next state from state q on input a

Step-3 Final state is obtained from those sets which have atleast one final state of the given

NDFA.

Q. Convert the following NDFA with NULL to equivalent DFA.



Ans - $M = \{Q, \Sigma, \delta, s, F\}$

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

Step-2 $\delta(\delta, a) = \epsilon\text{-closure}(\text{move}(\{q_0, q_1, q_2\}, a))$
 $\delta(\delta, a) = \epsilon\text{-closure}(q_3)$
 Let $\delta(s, a) = A = \{q_3, q_5, q_6\}$

Now $\delta(s, b) = \epsilon\text{-closure}(\text{move}(\{q_0, q_1, q_2\}, b))$
 $= \epsilon\text{-closure}(q_4)$
 let $B = \{q_4, q_5, q_6\}$

States	Input	
	a	b
$\rightarrow s$	A	B

$A = \epsilon\text{-closure}(\{q_3, q_5, q_6\}) =$

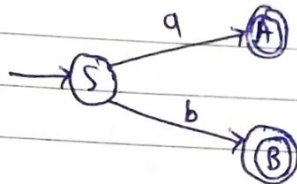
Now $\delta(A, a) = \epsilon\text{-closure}(\text{move}(\{q_3, q_5, q_6\}, a))$
 $= \epsilon\text{-closure}(\emptyset) = \emptyset$

$\delta(A, b) = \epsilon\text{-closure}(\text{move}(\{q_3, q_5, q_6\}, b))$
 $\delta(A, b) = \emptyset$

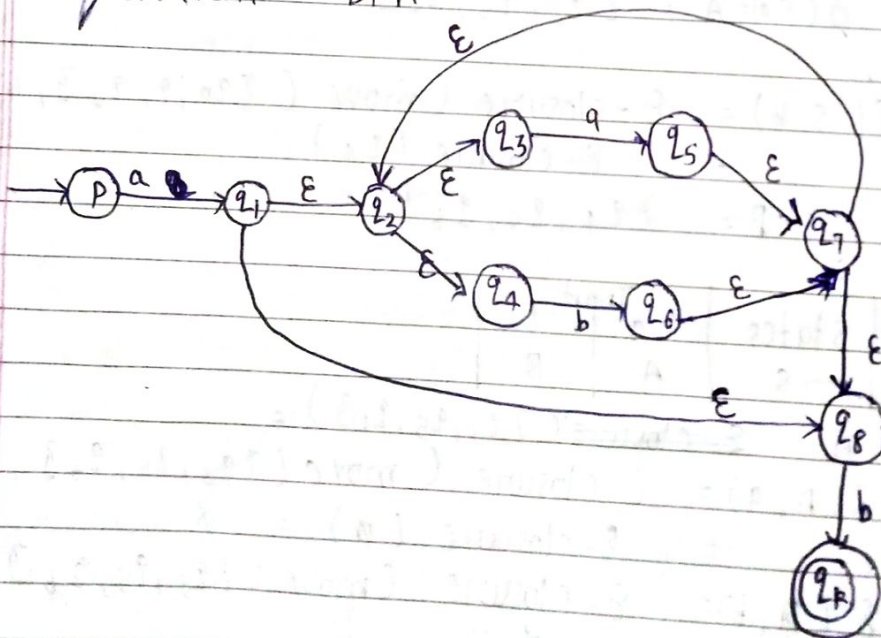
Now $\delta(B, a) = \epsilon\text{-closure}(\text{move}(\{q_4, q_5, q_6\}, a))$
 $= \emptyset$
 $\delta(B, b) = \epsilon\text{-closure}(\text{move}(\{q_4, q_5, q_6\}, b))$
 $= \emptyset$

States	Input	
	a	b
$\rightarrow S$	A	B
(A)	\emptyset	\emptyset
(B)	\emptyset	\emptyset

Step ③



Q. ~~Consider~~ Convert the NFA with NULL to
 ★ Equivalent DFA



Ans - Step-① $S = \epsilon\text{-closure}(P)$

$$S = \{P\}$$

Step-② $\delta(S, a) = \epsilon\text{-closure}(\text{move}(P, a))$

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

$$A = \delta(S, a) = \{q_1, q_2, q_3, q_4, q_8\}$$

$$\delta(S, b) = \epsilon\text{-closure}(\text{move}(P, b))$$

$$B = \delta(S, b) = \epsilon\text{-closure}(\emptyset) = \emptyset$$

states	Input	
	a	b
$\rightarrow S$	A	\emptyset

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

$$= \epsilon\text{-closure}(\text{move}(\{q_1, q_2, q_3, q_4, q_8\}, a))$$

$$= \epsilon\text{-closure}(q_5)$$

$$B = \delta(A, a) = \{q_5, q_7, q_8, q_2, q_3, q_4\}$$

$$\delta(A, b) = \epsilon\text{-closure}(\text{move}(\{q_1, q_2, q_3, q_4, q_8\}, b))$$

$$= \epsilon\text{-closure}(q_6)$$

$$C = \delta(A, b) = \{q_6, q_2, q_7, q_8, q_5, q_4\}$$

states	Input	
	a	b
$\rightarrow S$	A	\emptyset
A	B	C

$$\delta(B, a) = (\{q_2, q_3, q_5, q_7, q_8\}, a)$$

$$\delta(B, a) = \epsilon\text{-closure}(\text{move}(\{q_2, q_3, q_5, q_7, q_8\}, a))$$

$$= \epsilon\text{-closure}(q_5)$$

$$= \{q_5, q_7, q_8, q_2, q_3, q_4\} = B$$

$$\delta(B, b) = \epsilon\text{-closure}(\text{move}(\{q_2, q_3, q_4, q_5, q_7, q_8\}, b))$$

$$= \epsilon\text{-closure}(q_6)$$

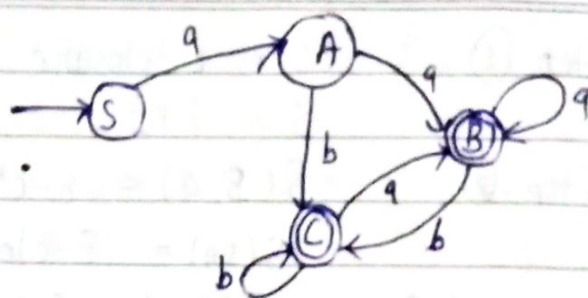
$$= \{q_6, q_7, q_8, q_2, q_3, q_4\} = C$$

$$\delta(C, a) = \epsilon\text{-closure}(\text{move}(q_6, q_2, q_7, q_8, q_3, q_4, a))$$

$$= \epsilon\text{-closure}(q_5) = \{q_5, q_7, q_8, q_2, q_3, q_4\} = B$$

$$\delta(C, b) = C$$

states	input	
	a	b
→ S	A	∅
A	B	C
(B)	B	C
(C)	B	C



Minimization of Finite Automata \Rightarrow

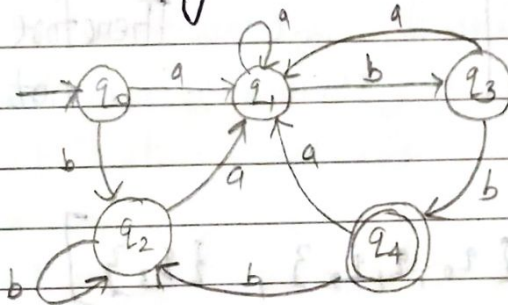
Method for construction of a finite automata ^{min-state}
:-

1. Construct the transition table from the given state transition diagram (STD)
2. Now we apply the process for partitioning the set of states where partition are denoted by Π_0, Π_1, Π_2 .
We can obtain Π_0 by grouping final states in one set and non-final state in other set.
3. We have 2 set of states here first states is final state & another is non-final state. Now leave the state containing final state as such and partition other non-final state (NFS) in such a way that the equivalent states are group together in separate set. This is Π_1 .
4. The step 3 is repeated for Π_2, Π_3 and so on until we get $\Pi_k = \Pi_{k+1}$ where k is any integer.

5. The set obtained in the Π_{k+1} are considered to be as one state and remaining of state transition in the state transition table goes according to the original state transition table.

6. New state transition diagram is constructed from state transition table.

Q. Construct a min. state automata for the following DFA.



Minimization
of
DFA

Ans- Step-1 Construct the state transition table from STD.

states	Input	
	a	b
→ q ₀	q ₁	q ₂
q ₁	q ₁	q ₃
q ₂	q ₁	q ₂
q ₃	q ₁	q ₄
(q ₄)	q ₁	q ₂

Step-2 Now Π_0 is obtained by grouping final state q₄ in one set and remaining non-final state in a separate set.

$$\Pi_0 = [\{q_4\}, \{q_0, q_1, q_2, q_3\}]$$

Step-3

Now partitioning the NFA as follows

States	Input a	Input b	
→ q ₀	q ₁	q ₂	Both belongs to same set
q ₁	q ₁	q ₃	

Both are equal

Therefore q₀ & q₁ are equivalent.

States	Input a	Input b	
→ q ₀	q ₁	q ₂	Both are equal
q ₂	q ₁	q ₂	

Therefore q₀ & q₂ are equivalent

States	Input a	Input b	
→ q ₀	q ₁	q ₂	Both are equal
q ₃	q ₁	q ₄	

are not Both belongs to same set

Therefore q₀ & q₃ are not equivalent

$$\text{Now } \Pi_1 = [\{q_4\}, \{q_0, q_1, q_2\}, \{q_3\}]$$

States	Input a	Input b		state	Input a	Input b	
→ q ₀	q ₁	q ₂	Both are equal	→ q ₀	q ₁	q ₂	Both are equal
q ₁	q ₁	q ₃		q ₂	q ₁	q ₂	

Both are not belongs to same set

Therefore q₀ & q₂ are equivalent but q₀ & q₁ are not equivalent.

$$\text{Now } \Pi_2 = [\{q_4\}, \{q_0, q_2\}, \{q_1\}, \{q_3\}]$$

Step-4

By applying the process again we get $\Pi_2 = \Pi_3$ therefore we stop here & Π_3 is a final solution.

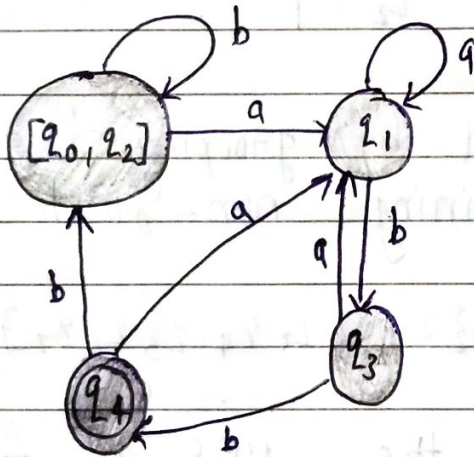
Step-5

$$\delta([q_0, q_2], a) = \delta(q_0, a) \cup \delta(q_2, a) \\ = \{q_1\} \cup \{q_1\} = \{q_1\}$$

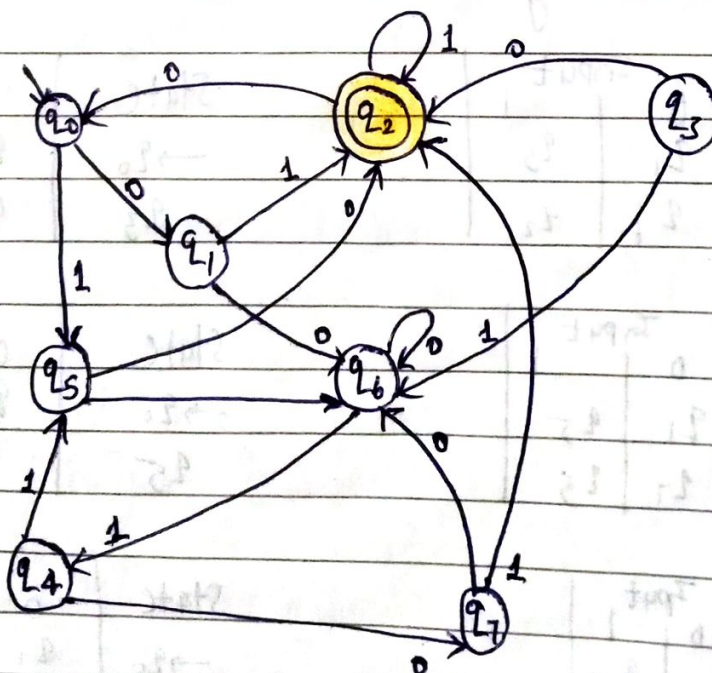
$$\delta([q_0, q_2], b) = \{q_2\} \cup \{q_2\} = \{q_2\}$$

Step-6

States	Input	
	a	b
$[q_0, q_2]$	q_1	$[q_0, q_2]$
q_1	q_1	q_3
q_3	q_1	q_4
(q_4)	q_1	$[q_0, q_2]$



Q.

Minimization
of
DFA

A₁ - Step-1 Construct the STT from STD

States	Input	
	a=0	b=1
→ q ₀	q ₁	q ₅
q ₁	q ₆	q ₂
(q ₂)	q ₀	q ₂
q ₃	q ₂	q ₆
q ₄	q ₇	q ₅
q ₅	q ₂	q ₆
q ₆	q ₆	q ₄
q ₇	q ₆	q ₂

Step-2 Now Π_0 is obtained by grouping final state q₂ in one set and remaining non-final state in a separate set.

$$\Pi_0 = \{ \{q_2\}, \{q_0, q_1, q_3, q_4, q_5, q_6, q_7\} \}$$

Step-3 Now partitioning the NFS as follow

state	Input	
	0	1
→ q ₀	q ₁	q ₅
q ₁	q ₆	q ₂

Both are
belong
to same
set

not belong
same

state	Input	
	0	1
→ q ₀	q ₁	q ₅
q ₄	q ₇	q ₅

Both are
belong to
same set

state	Input	
	0	1
→ q ₀	q ₁	q ₅
q ₆	q ₆	q ₄

Both are
belong to same set

state	Input	
	0	1
→ q ₀	q ₁	q ₅
q ₃	q ₂	q ₆

Both are
not belong
to same
set

state	Input	
	0	1
→ q ₀	q ₁	q ₅
q ₅	q ₂	q ₆

Both
are
not belong
to same
set

state	Input	
	0	1
→ q ₀	q ₁	q ₅
q ₇	q ₆	q ₂

Both are
not belong
to same
set

Therefore q_0, q_3, q_5, q_7 are not equivalent
but q_1, q_4, q_6 are equivalent.

$$\Pi_1 = [\{q_2\}, \{q_0\}, \{q_3\}, \{q_5\}, \{q_7\}, \{q_1\}, \{q_4\}, \{q_6\}]$$

$$\Pi_1 = [\{q_2\}, \{q_1, q_4, q_6\}, \{q_0\}, \{q_3\}, \{q_5\}, \{q_7\}]$$

States	Input 0	Input 1
q_1	q_6	q_2
q_4	q_7	q_5

not belong
same set

State	Input 0	Input 1
q_1	q_6	q_2
q_6	q_6	q_4

not same
set

$$\Pi_3 = [\{q_2\}, \{q_0, q_4\}, \{q_6\}, \{q_1, q_7\}, \{q_3, q_5\}]$$

$$\Pi_3 = [[q_2], [q_0, q_4], [q_6], [q_1, q_7], [q_3, q_5]]$$

$$\delta([q_0, q_4], a) = \{q_1\} \cup \{q_7\} = [q_1, q_7]$$

$$\delta([q_0, q_4], b) = \{q_5\} \cup \{q_5\} = [q_5]$$

$$\delta([q_1, q_7], a) = \{q_6\} \cup \{q_6\} = [q_6]$$

$$\delta([q_1, q_7], b) = \{q_2\} \cup \{q_2\} = [q_2]$$

$$\delta([q_3, q_5], a) = \{q_2\} \cup \{q_2\} = [q_2]$$

$$\delta([q_3, q_5], b) = \{q_6\} \cup \{q_6\} = [q_6]$$

	a	b
$\rightarrow [q_0, q_4]$	$[q_1, q_7]$	q_5
$[q_1, q_7]$	q_6	q_2
$[q_3, q_5]$	q_2	q_6
$[q_2]$	q_6	$[q_0, q_4]$
q_2	$[q_0, q_4]$	q_2

Equivalence of DFA & NFA \Rightarrow

Steps for the comparison of two finite automata
 Let us consider two finite automata M_1 & M_2 over some given set of input alphabet Σ

Step-1 If the name of states are same in automata M_1 & M_2 then change the name of states of M_2 which is similar to state of M_1 .

Step-2 We start with taking the initial state of finite automata M_1 & M_2 . Let the initial state of automata M_1 is q_0 & for the automata M_2 initial state is q'_0 . Then pair of initial state is (q_0, q'_0) . Then for the given inputs we make transition from the initial states of both automata M_1 & M_2 .

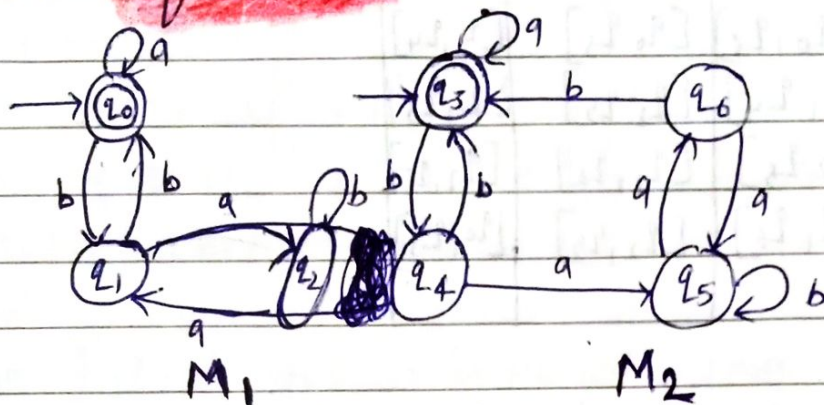
Step-3 Repeat the step-1 for every new entry.

Step-4 We repeat the step-1 for each new entry under the following cases :-
Case-1 Construction is terminated as soon as we reach the pair (q_F, q'_F) such that q_F is the final state of M_1 and q'_F is a non-final state of M_2 and vice-versa. In this case, the automata M_1 & M_2 are not equivalent.

Case-2 If non-new element appears in further column then the construction is terminated and then automata M_1 & M_2 are equivalent.

Q. Consider the following 2 finite automata M_1 & M_2 over $\Sigma = \{a, b\}$. Find out whether they are equivalent or not.

Ans -



Step-① Here no state is common in M_1 & M_2 so no need to change the name of state of M_2 .

Step-② Hence starting with the initial state q_0 & q_3 of M_1 & M_2 respectively for the input $\{a, b\}$

$$\begin{aligned} \delta(q_0, a) &= q_0 & , & & \delta(q_3, a) &= q_3 \\ \delta(q_0, b) &= q_1 & , & & \delta(q_3, b) &= q_4 \end{aligned}$$

States	Input	
	a	b
$\rightarrow [q_0, q_3]$	$[q_0, q_3]$	$[q_1, q_4]$

Step-③

$$\begin{aligned} \delta([q_0, q_4], a) &= q_0 \cup q_5 = [q_0, q_5] \\ \delta([q_1, q_4], b) &= q_1 \cup q_3 = [q_1, q_3] \\ \delta([q_2, q_5], a) &= q_2 \cup q_6 = [q_2, q_6] \\ \delta([q_2, q_5], b) &= q_2 \cup q_5 = [q_2, q_5] \end{aligned}$$

state	Input	
	a	b
$\rightarrow [q_0, q_3]$	$[q_0, q_3]$	$[q_1, q_4]$
$[q_1, q_4]$	$[q_2, q_5]$	$[q_0, q_3]$
$[q_2, q_5]$	$[q_1, q_4]$	$[q_2, q_5]$

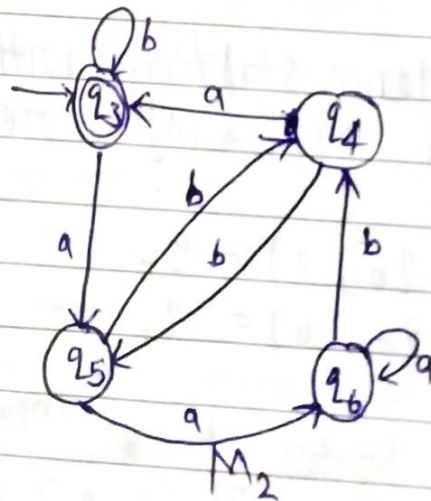
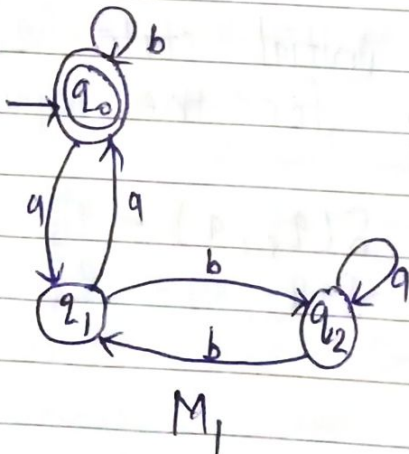
$$\delta([q_1, q_6], a) = \{q_2\} \cup \{q_5\} = [q_2, q_5]$$

$$\delta([q_1, q_6], b) = \{q_0\} \cup \{q_3\} = [q_0, q_3]$$

State	Input	
	a	b
$\rightarrow [q_0, q_3]$	$[q_0, q_3]$	$[q_1, q_4]$
$[q_1, q_4]$	$[q_2, q_5]$	$[q_0, q_3]$
$[q_2, q_5]$	$[q_1, q_4]$	$[q_2, q_5]$
$[q_1, q_6]$	$[q_2, q_5]$	$[q_0, q_3]$

According to table there is no new element entry so the M_1 & M_2 Automata are equivalent due to the case-2.

Q.



Ans - Step-1 Here no state is common in M_1 & M_2 so no need to change the name of states of M_2 .

Step-2 Hence starting with initial state q_0 & q_3 of M_1 & M_2 respectively for the input $[a, b]$

$$\begin{aligned} \delta(q_0, a) &= q_1, & \delta(q_3, a) &= q_5 \\ \delta(q_0, b) &= q_0, & \delta(q_3, b) &= q_3 \end{aligned}$$

Step ③ $\delta([q_1, q_5], a) = q_0 \cup q_6 = [q_0, q_6]$
 $\delta([q_1, q_5], b) = q_2 \cup q_4 = [q_2, q_4]$

States	Input	
	a	b
$[q_0, q_3]$	$[q_1, q_5]$	$[q_0, q_3]$
$[q_1, q_5]$	$[q_0, q_6]$	$[q_2, q_4]$

In the above table, we can see the $[q_0, q_6]$ pair that is the q_0 is initial or final state but q_6 is not a final or initial state of automata M_2 so we can quite from here. Hence both the automata are not equivalent.

★ Finite Automata with Output \Rightarrow The automata is like a finite automata except that the set of accepting states is replaced by an output set as an output function. There are 2 types of finite automata with output where output depends on either present input and present state or only present states.

- (i) Moore Machine
- (ii) Mealy Machine

The Generalised Model is the mealy machine and the restricted model is the moore machine.

Present state state in which resides at any instant of time. Present input input the machine takes to reach the present state.

Moore machine :- It is the finite state machine in which next state is decided by current state and is independent of current input symbol. The output at a given time depends only on the present state of the machine.

A moore machine can be described by 6 tuple $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ by

Where

Q = The finite & non-empty set of state

Σ = input alphabet

Δ = output alphabet

δ = Transition function which makes present state & input symbol on to the next state on $Q \times \Sigma \rightarrow Q$.

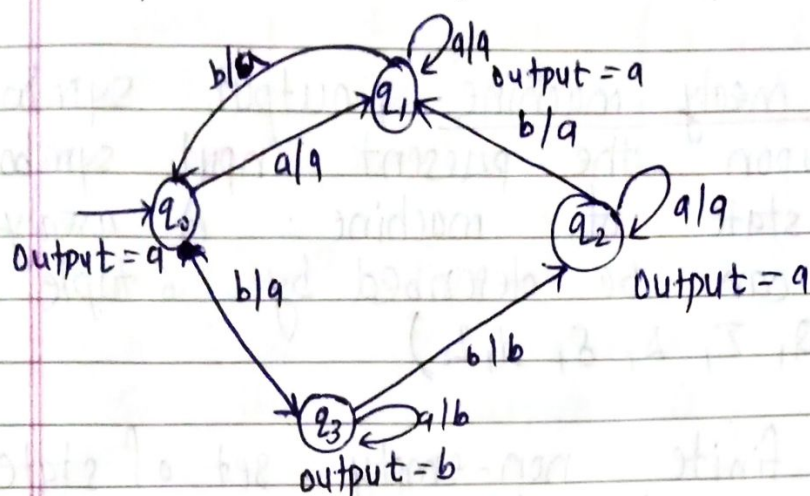
λ = Output function which makes $Q \rightarrow \Delta$ where Q is present state & Δ is output.

q_0 = q_0 belongs to Q is the initial state.

If $z(t)$, $q(t)$ are the output and present state respectively at time t then

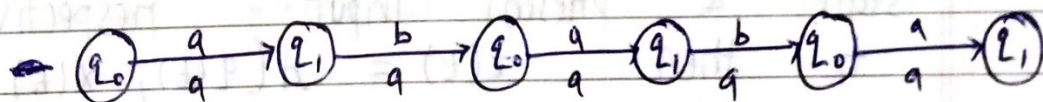
$$z(t) = \lambda(q(t))$$

Q. Consider the moore machine and what is the output for input ababa



States	Input		Output
	a	b	
→ q ₀	q ₁	q ₃	a
q ₁	q ₁	q ₀	a
q ₂	q ₂	q ₁	a
q ₃	q ₃	q ₂	b

At input ababa



So the output is aaaaa

The output of Moore machine doesn't depends on the input therefore the first ~~output~~ output signal is additional from the initial state without reading the input that is null input. And the output length is one greater than input length but it is not included in the above output.

The input is important for the transition & doesn't

effect the output.

(ii) In the mealy machine, output symbol depends upon the present input symbol & present state of machine. A mealy machine can be described by 6 tuple
 $M = (Q, \Sigma, \Delta, \delta, d, q_0)$

Q = the finite non-empty set of state

Σ = Input alphabet

Δ = Output alphabet

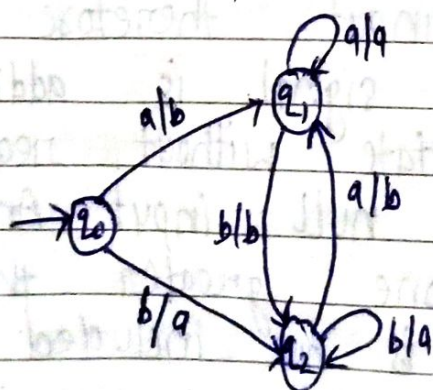
δ = transition function which makes $Q \times \Sigma \rightarrow Q$

d = output function which makes $Q \times \Sigma \rightarrow \Delta$
~~where Q is p.s. & Δ is output~~

$q_0 = q_0 \in Q$ is initial state.

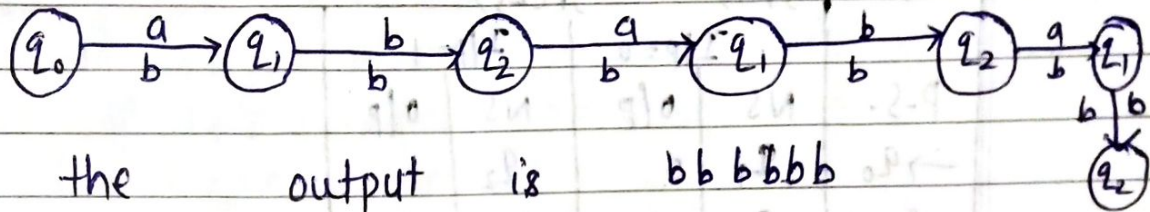
If $z(t)$, $q(t)$ & $x(t)$ are output, present state & present input respectively at time t then $z(t) = d(q(t), x(t))$

Q. Consider the mealy machine & find the output for input ababab



Ans -

States	Input = a		Input = b	
	Next state	Output	Next state	Output
$\rightarrow q_0$	q_1	b	q_2	a
q_1	q_1	a	q_2	b
q_2	q_1	b	q_2	a



So the output is b b b b b b

In this output length is equal to the input length because it is dependent on input also therefore the no. of transition for the input decide the length of the output.

Conversion of Moore machine to Mealy Machine



The output function λ can be $\lambda'(z, a) = \lambda(\delta(z, a))$

Q. Convert the following moore machine into mealy machine.

Present state	IP=0	IP=1	O/P
	NS	NS	
$\rightarrow q_0$	q_1	q_2	1
q_1	q_3	q_2	0
q_2	q_2	q_1	1
q_3	q_0	q_3	1

Ans -

First state q_0 , $\lambda'(q_0, 0) = \lambda(\delta(q_0, 0))$

$$= \lambda(q_1) = 0$$

$$\lambda(q_0, 1) = \lambda(q_2) = 1$$

$$\lambda'(q_1, 0) = \lambda(q_3) = 1$$

$$\lambda'(q_1, 1) = \lambda(q_2) = 1$$

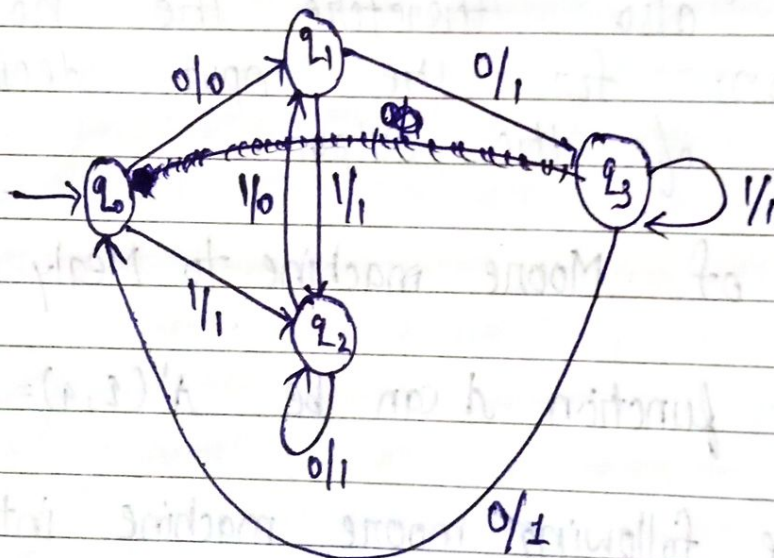
$$\lambda'(q_2, 0) = \lambda(q_2) = 1$$

$$\lambda'(q_2, 1) = \lambda(q_1) = 0$$

$$\lambda'(q_3, 0) = \lambda(q_0) = 1$$

$$\lambda'(q_3, 1) = \lambda(q_3) = 1$$

P.S.	I/p=0		I/p=1	
	NS	O/p	NS	O/p
$\rightarrow q_0$	q_1	0	q_2	1
q_1	q_3	1	q_2	1
q_2	q_2	1	q_1	0
q_3	q_0	1	q_3	1



Conversion of Mealy to Moore machine \Rightarrow

Q.

P.S.	I/p=0		I/p=1	
	NS	O/p	NS	O/p
$\rightarrow q_1$	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

A4 - Output of $q_1 = 1$ when $i/p = 0$
 Output of $q_1 = 1$ when $i/p = 1$
 Output of $q_2 = 1$ when $i/p = 0$
 Output of $q_2 = 0$ when $i/p = 1$
 Output of $q_3 = 0$ when $i/p = 0$
 Output of $q_3 = 0$ when $i/p = 1$
 Output of $q_4 = 1$ when $i/p = 0$
 Output of $q_4 = 0$ when $i/p = 1$

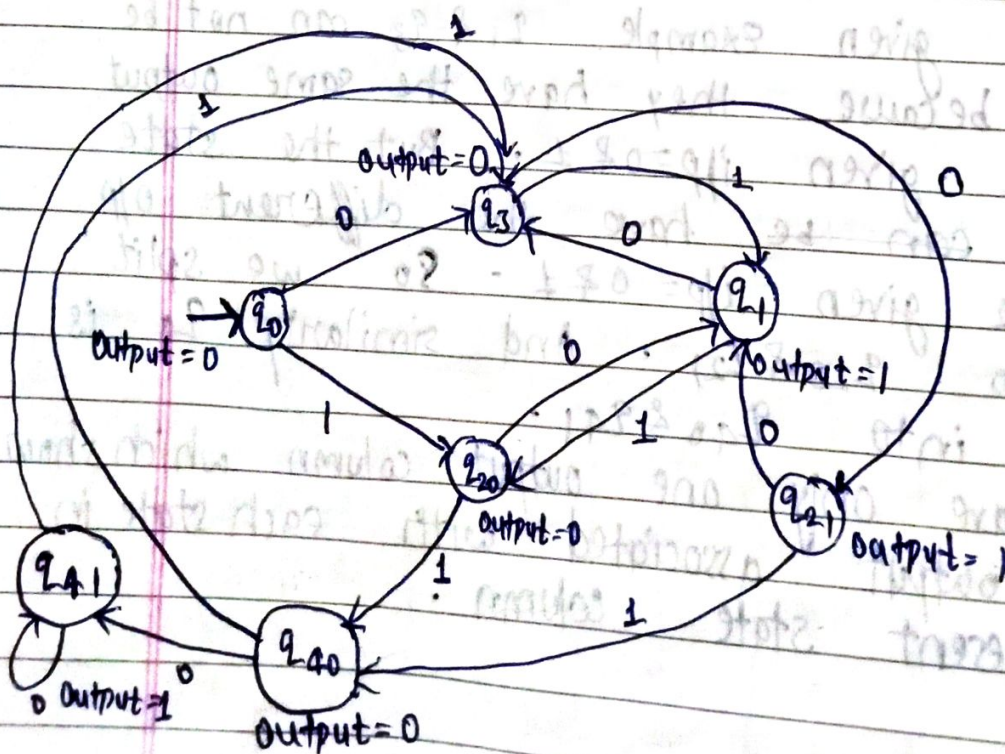
P.S.	I/p = 0		I/p = 1	
	NS	O/p	NS	O/p
$\rightarrow q_1$	q_3	0	q_{20}	0
q_{20}	q_1	1	q_{40}	0
q_{21}	q_1	1	q_{40}	0
q_3	q_{21}	1	q_1	1
q_{40}	q_{41}	1	q_3	0
q_{41}	q_{41}	1	q_3	0

In the given example q_1 & q_3 can not be split because they have the same output for the given $i/p = 0$ & 1 . But the state q_2 & q_4 can be have the different o/p for the given $i/p = 0$ & 1 . So we split q_2 into q_{20} & q_{21} . And similarly q_4 is splitted into q_{40} & q_{41} .

So we have only one output column which show single output associated with each state in the present state column.

Present state	Output
q₀	
→ q ₁	1
q ₂₀	0
q ₂₁	1
q ₃	0
q ₄₀	0
q ₄₁	1

PS	I/P=0 NS	I/P=1 NS	O/P
→ q ₀	q ₃	q ₂₀	0
q ₁	q ₃	q ₂₀	1
q ₂₀	q ₁	q ₄₀	0
q ₂₁	q ₁	q ₄₀	1
q ₃	q ₂₁	q ₁	0
q ₄₀	q ₄₁	q ₃	0
q ₄₁	q ₄₁	q ₃	1



Languages And Grammar \Rightarrow

Each language has its own well defined representation ~~failed~~ construct. Construct of a language means well define ~~mean~~ rule over some fix notation & these rule over some fix notations are called Grammar. So we can say that a language is the collection of words arranged in well define fashion.

A formal language is a set of words that is finite strings of letters, symbols or tokens. A ~~to~~ set from which these letters are taken is called the alphabet over which the language is defined or in other words we can say that a formal language L over an alphabet submission (Σ) is just a subset of submission (Σ^*) that is a set of words over the ~~a~~ alphabet.

An alphabet is a set of symbols. A string over an alphabet is the sequence of symbol taken from that alphabet.

A formal language over an alphabet is a set of strings. This set may be empty, non-empty, finite or infinite.

$L(M)$ is the notation for a language defined by a machine. The Machine M accepts a certain set of string thus are a language. $L(G)$ is the notation for a language defined by a grammar G . The grammar G recognises a certain

set of string thus are a language.
 $M(L)$ is the notation for a machine that accepts a language and
 $G(L)$ is the notation for grammar that recognises a language.

→ For example consider a english sentence

Ram talks loudly.

Let S be a variable denoting a sentence.
 Now we can form the following rules to generate the above given sentence:—

$\langle \text{sentence} \rangle \rightarrow \langle \text{noun-phrase} \rangle \langle \text{predicate} \rangle$

$\langle \text{noun-phrase} \rangle \rightarrow \langle \text{article} \rangle \langle \text{noun} \rangle$

$\langle \text{predicate} \rangle \rightarrow \langle \text{verb} \rangle$

Each arrow represent the rule and the word on the right side arrow can replace the word on the left side arrow.

Let P denotes the collection of rules or can be called as production.

Definition of Grammar:—

A grammar expresses language. A Grammar is defined by 4 terms

$$G = (V_N, \Sigma, P, S)$$

When V_N = Finite non empty set of variable or non-terminal. AB

Σ = Finite non empty set of terminals.

P = Finite non-empty set of production rules.
 s = S belongs to V_N and is known as start symbol.

Here $V_N \cap \Sigma = \emptyset$ and P includes the production rule like $\alpha \rightarrow \beta$ read as α derives β where $\alpha, \beta \in (V_N \cup \Sigma)$

An α must have a symbol from V_N and β has no restriction.

It can be either from variable or terminals or both.

Q. Which of the following follows the production rules for the following grammar where

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

$$\Sigma = \{a, b\} \text{ and start symbol is } S.$$

Ans-

$$a \rightarrow b \quad \times$$

$$S \rightarrow SAB \quad \checkmark$$

$$b \rightarrow SA \quad \times$$

$$SA \rightarrow ab \quad \checkmark$$

$$SB \rightarrow a \quad \checkmark$$

$$Sb \rightarrow ac \quad \times$$

$$S \rightarrow d \quad \checkmark$$

(symbol)

Language Generated By the Grammar \Rightarrow

Q. If a grammar $G = (V_N, \Sigma, P, S)$ where

$$V_N = \{S\}, \quad \Sigma = \{a, b\}, \quad P = \{S \rightarrow aS, S \rightarrow \epsilon\}$$

Ans- Let the first production rule

$$P_1: S \rightarrow aS \quad \text{and}$$

$$P_2: S \rightarrow \epsilon$$

As ϵ in the right hand side which has no variable so ϵ is in $L(G)$.

Using production P_1 recursively, we find

$$\begin{aligned} S &\rightarrow aS && \{ \text{one step derivation} \} \\ S &\rightarrow aas && \{ \text{two step derivation} \} \\ S &\rightarrow aaas && \{ \text{Three step derivation} \} \\ &\vdots \\ S &\rightarrow a^n s && \{ n^{\text{th}} \text{ step derivation} \} \end{aligned}$$

So the following can be derived
 $a, aa, aaa, aaa \dots$

$$L(G) = \{ \epsilon, a, aa, aaa \dots \}$$

$$L(G) = a^* \text{ or } \{ a^n : n = 0, 1, 2, 3, \dots \}$$

Q. If a grammar $G = (V_N, \Sigma, P, S)$

$$V_N = \{s\}$$

$$\Sigma = \{a\}$$

$$P = \{ S \rightarrow SS, S \rightarrow a \}$$

Ans:- Let

$$P_1 = S \rightarrow SS$$

$$P_2 = S \rightarrow a$$

$$S \rightarrow SS$$

$$S \rightarrow SSS$$

$$S \rightarrow aa$$

$$S \rightarrow aaq$$

$$L(G) = \{ a, aa, aaa \dots \}$$

$$\Rightarrow L(G) = a^+$$

$$L(G) = \{ a^n \mid n = 1, 2, 3, \dots \}$$

Q. If $P_1 = S \rightarrow aCa$

$$P_2 = C \rightarrow aCa$$

$$P_3 = C \rightarrow b$$

find the value of L = ?

Aj - $V_N = \{S, c\}$
 $\Sigma = \{a, b\}$
 $S = S$

$S \rightarrow aca$, $c \rightarrow aca$, $c \rightarrow b$
 $S \rightarrow aba$
 $S \rightarrow aa caa$, $S \rightarrow aa a caa a$
 $S \rightarrow aab a a$, $S \rightarrow aaab a a a$

$L = \{aba, aabaa, aaabaaa, \dots\}$

$L(G) = \{a^n b^n a^n \mid n = 1, 2, 3, \dots\} \text{ on } a^+ b a^+$

★ Grammar generated by Language \Rightarrow

Q. Find the grammar which generate $L = \{a^n b^n \mid n \geq 1\}$

Aj - $L(G) = \{ab, aabb, aaabbb, \dots\}$

$P_1: S \rightarrow aSb$
 $S \rightarrow aabb$
 $S \rightarrow aaSbb$
 $S \rightarrow aaabbb$

$P_2: S \rightarrow ab$

$V_N = \{S\}$

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

$P = \{S \rightarrow aSb, S \rightarrow ab\}$
 starting state = S

$G = \{V_N = \{S\}, \Sigma = \{a, b\}, P = \{S \rightarrow aSb, S \rightarrow ab\}, S\}$

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

Aj -

~~$L = \{ab, aabbc, aaabbbcc, \dots\}$~~

If $i = 0$

$i = 1$

$L_1 = \{ab, aabb, aaabbb, \dots\}$

$L_2 = \{abc, aabbbc, \dots\}$

$$P_1: A \rightarrow aAb$$

$$P_2: A \rightarrow ab$$

$$S \rightarrow Sc/A$$

$$A \rightarrow aAb/ab$$

$$S \rightarrow AC \text{ (Ist)}$$

$$S \rightarrow abc$$

$$S \rightarrow Sc$$

$$S \rightarrow ScC$$

$$S \rightarrow aabbcc \text{ (IInd)}$$

$$V_N = \{S, A\}$$

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

$$P_1 = A \rightarrow aAb, P_2 = A \rightarrow ab$$

$$P_3 = S \rightarrow Sc/A, P_4 = A \rightarrow aAb/ab$$

$$G = \{ V_N = \{S, A\}, \Sigma = \{a, b, c\}, A \rightarrow aAb, A \rightarrow ab, S \rightarrow Sc/A, A \rightarrow aAb/ab, S \}$$

Regular Expression / Language and Regular Grammar \Rightarrow

Regular languages are the simplest of the formal language expression and regular expression is one way to describe such language.

A regular grammar is the context free grammar in which all rules have no more than 2 symbol on the RHS and non terminal if they occur either always occur as the first symbol or the last symbol.

Regular Expression :- The language recognised by finite automata are regular

languages and these languages are described by simple expression called regular expression. RE are used to represent certain set of string in some algebraic manner.

Q. Write the Regular Expression for the language accepting (a) all the combination of the a over the set $\Sigma = \{a\}$

Ans - $L = \{ \epsilon, a, aa, aaa, \dots \}$
 $= a^*$

(b) All the combination of a except the null string over the set $\Sigma = \{a\}$

Ans - $L = \{ a, aa, aaa, \dots \} = a^+$

(c) All the string having any no. of a & b.

Ans - $L = \{ \epsilon, a, b, aa, bb, ab, ba, \dots \} = (a+b)^*$

(d) All the string having a & b except the null string.

Ans - $L = \{ a, b, aa, bb, ab, ba, \dots \} = (a+b)^+$

(e) All the string which are ending with 00 over the set $\Sigma = \{0, 1\}$

Ans - $L = \{ 00, 100, 000, \dots \}$

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

(f) The string which are starting with 1 & ending with 0 over set $\Sigma = \{0, 1\}$

Ans - $L = \{ 10, 100, 10101, \dots \}$

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

(g) Which accept all the string which begin or end with 00 or 11.

Ans- $L = \{ 00, 11, 0011, 1100, 00000, 00100, 11011, \dots \}$

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

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

(h) In which any no. of a is being followed by any no. of b is being followed by any no. of c over the set $\Sigma = \{a, b, c\}$

Ans-

$L = \{ \epsilon, a, b, c, ab, bc, aabc, \dots \}$

$$L = a^*b^*c^*$$

(i) For the string in which atleast a is followed by atleast b is followed by atleast c followed by over the set $\Sigma = \{a, b, c\}$

Ans-

$L = \{ abc, aabbcc, aaabbbccc, \dots \}$

$$L = a^+b^+c^+$$

(j) Consisting all words having aa as a substring over the $\Sigma = \{a, b\}$

Ans- $L = \{ aa, aab, baq, aabaa, \dots \}$

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

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

(k) Which accept all the string with atleast 2 b over the set $\Sigma = \{a, b\}$.

Ans - $L = \{ b b, a b b, a b a b, b b b, b b a, a b b b, \dots \}$

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

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

(l) All words over the $\Sigma = \{a, b\}$ such that no 3 character from right hand of the is always a.

Ans -

$$L = \{ a b b, a a a, \dots \}$$

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

(m) For the following over the set $\Sigma = \{a, b, c\}$ all string that have exactly one a.

Ans -

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

(n) All the string that have no more than 3 a.

Ans -

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

$$(b+c)^* (\epsilon + a) (b+c)^* (\epsilon + a) (b+c)^* (\epsilon + a) (b+c)^*$$

(o) That have no more than 2a and a will be placed simultaneously.

Ans -

$$(b+c)^* (\epsilon + a + a a) (b+c)^*$$

(P) On the string having atmost one pair of zero and on atmost pair of 1 over the set 0,1.

Ans -

$$L = \{ \epsilon^* (\epsilon + 00) (\epsilon + 11) \}$$

$$L = \{ \epsilon^* (\epsilon + 01)^* (\epsilon + 00)^* \}$$

$$L = (1+01)^* + (1+01)^* 00 (1+01)^* + (0+10)^* + (0+10)^* 11 (0+10)^*$$

(Q.) $L = \{ a^{2n} b^{2m+1} \mid n \geq 0, m \geq 0 \}$

Ans - $L = \{ (b + b^3 + b^5 + \dots + a^2 b + a^4 b + \dots) \}$

$$L = a^0 (b) a^2 (b) a^4 (b) \dots$$

$$L =$$

$$(aa)^* (bb)^* b$$

(R) $L = \{ a^n b^m \mid n \geq 4, m \leq 3 \}$

$$L = aaaaa^* (\epsilon + b + bb + bbb)$$

(S) $L = \{ a^n b^m : (n+m) \text{ is even} \}$

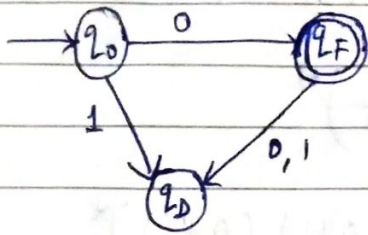
$$L = (aa)^* (bb)^* + a(aa)^* b(bb)^*$$

(T) $L = \{ a^n b^m \mid n \geq 3, m \in (a+b)^+ \}$

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

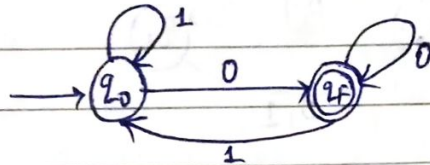
Q. DFA accepts 0 only.

Ans-



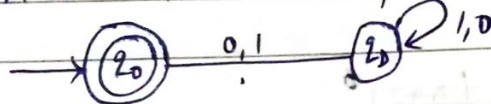
Q. DFA for $(0+1)^*0$

Ans-



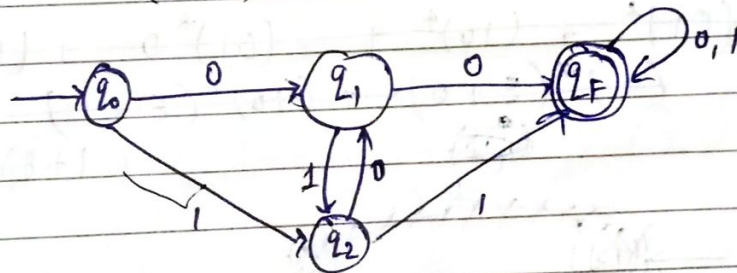
Q. DFA that accepts nothing.

Ans



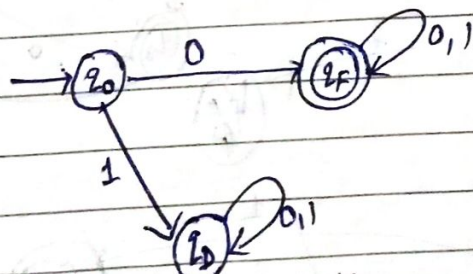
Q. DFA that accepts doublet somewhere $(0+1)^*(00+11)(0+1)^*$

Ans-



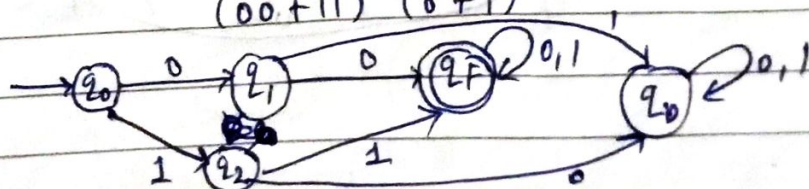
Q. DFA with all the words start with zero. $0(0+1)^*$

Ans-



Q. DFA for word with start with doublet $(00+11)(0+1)^*$

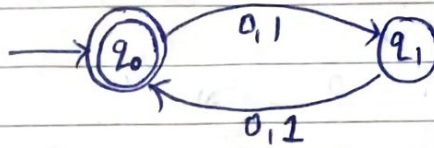
Ans-



Q. DFA that accepts the word ~~length~~ length is even.

$$\overline{(00+11)^*}$$

$$(00+11) \quad [(0+1)(0+1)]^*$$



Q. DFA that do not contain doublet.

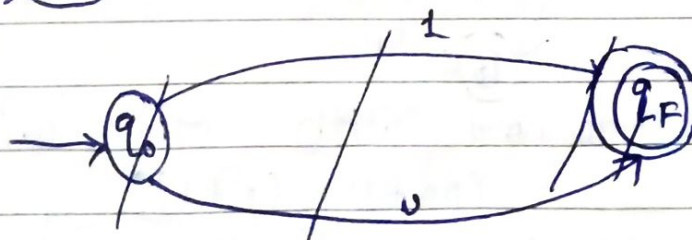
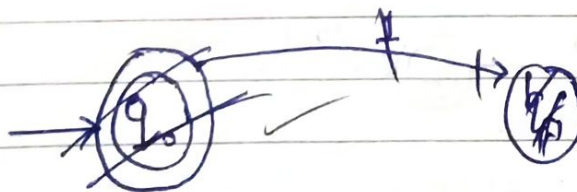
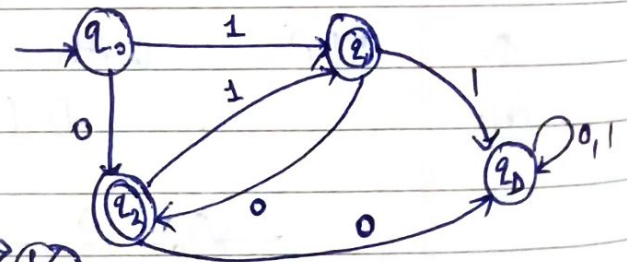
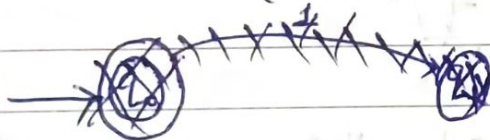
$$\overline{(0+1)(0+1)^*}$$

$$\overline{(0+1)(00+11)^*}$$

$$\overline{(0000)^*}$$

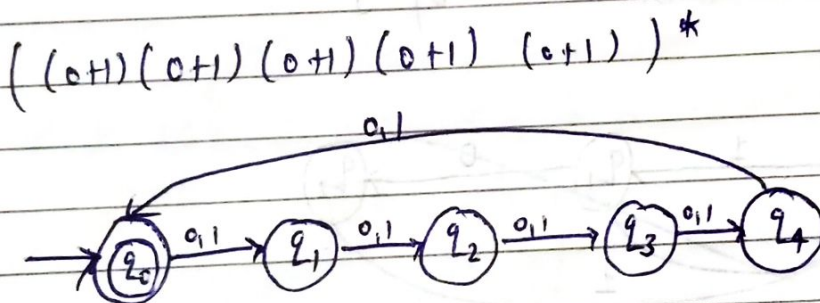
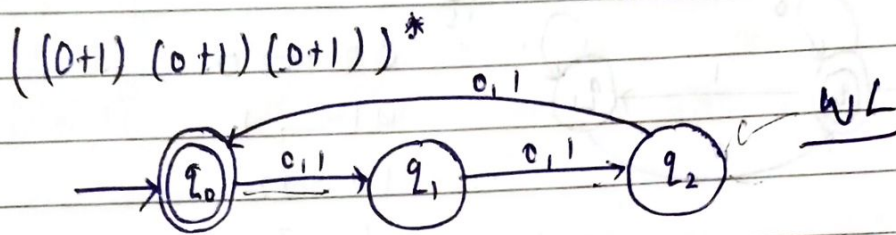
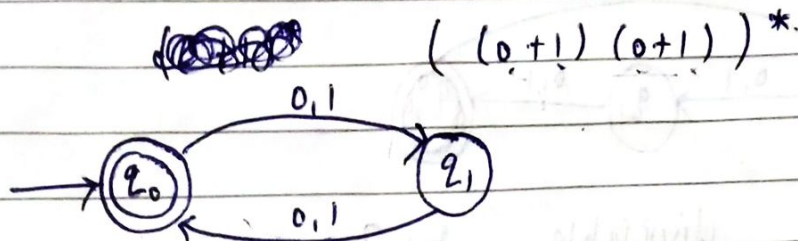
$$\overline{(01)^* + (10)^* + (01)^*0 + (10)^*1}$$

$$= \overline{(01)^*(\epsilon + 0) + (10)^*(\epsilon + 1)} = (1+\epsilon)(01)^*(\epsilon + 0)$$



DFAWL

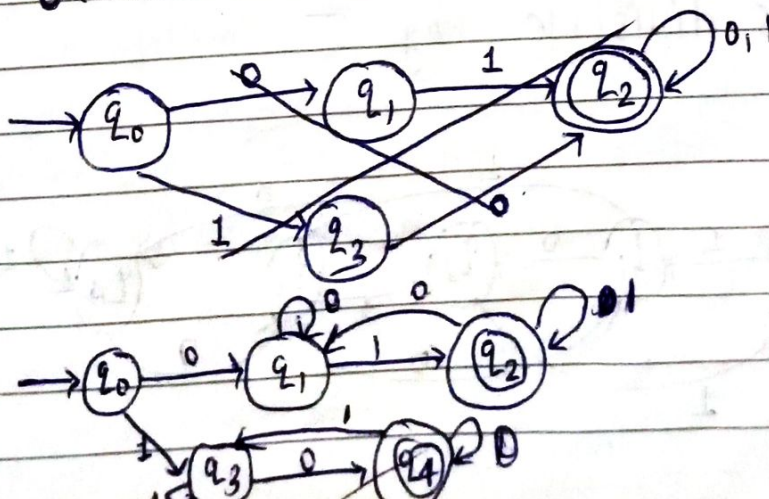
Q. Word length is divisible by 2, ~~DFA~~ is divisible by 5 and ~~word length~~ is divisible by 3. Make DFA. ~~DFA~~ WL Word length



Q. Construct the DFA on which word start and end with different character.

Ans -

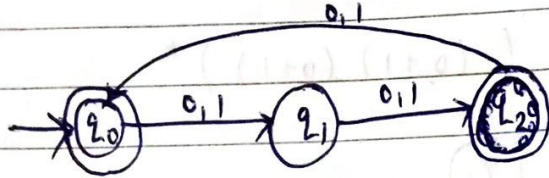
$$(0+1)(0+1)^*(0+1) + 0(0+1)^*1 + 1(0+1)^*0$$



Q. Word length is divisible by 3.

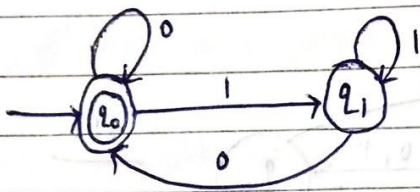
Ans -

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



Q. DFA is divisible by 2.

Ans -



$$00 \rightarrow 0$$

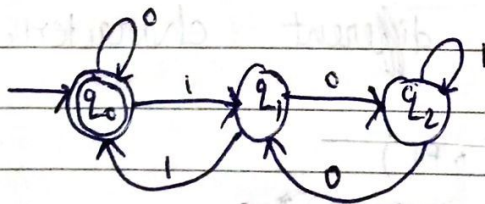
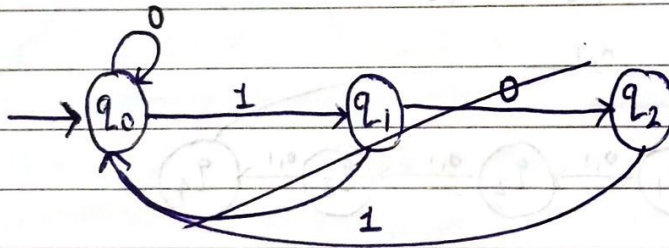
$$01 \rightarrow 1$$

$$10 \rightarrow 2$$

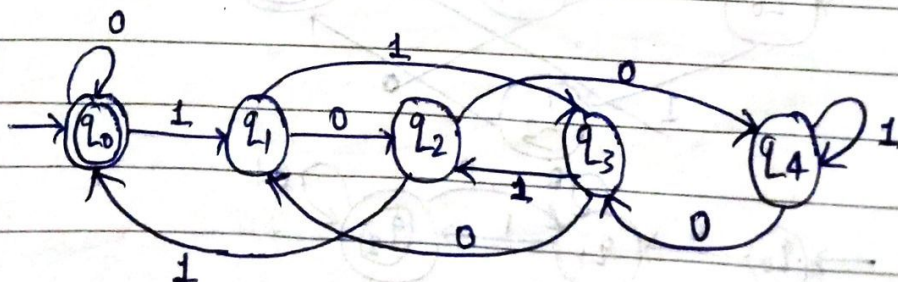
$$11 \rightarrow 3$$

Q. DFA is divisible by 3

$$\begin{array}{r} 000 \\ 011 \\ \hline 3 \\ \hline 0 \\ 001 \\ \hline 3 \\ \hline 010 \\ \hline 011 \\ \hline 3 \\ \hline 100 \end{array}$$



Q. DFA is divisible by 5



Er Sahil
Ka
Gyan

11000 $\rightarrow 2m$
11001 $\rightarrow 2m+1$

DATE: / /

PAGE NO.:

I/p $m=0$

$2(3m)$

$2(3m+1)$

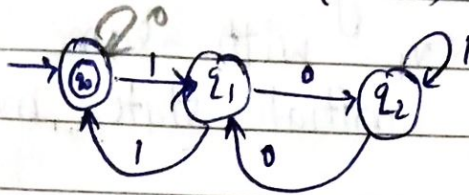
$2(3m+2)$

I/p $m=1$

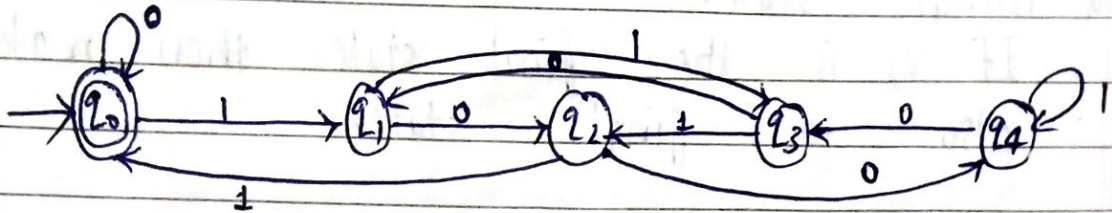
$2(3m)+1$

$2(3m+1)+1$

$2(3m+2)+1$



Divisible by 5



$m=0$

$2(5m)$

$2(5m+1)$

$2(5m+2)$

$2(5m+3)$

$2(5m+4)$

$m=1$

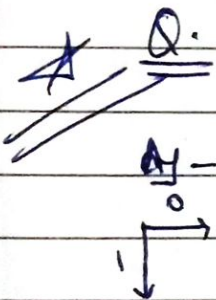
$2(5m)+1$

$2(5m+1)+1$

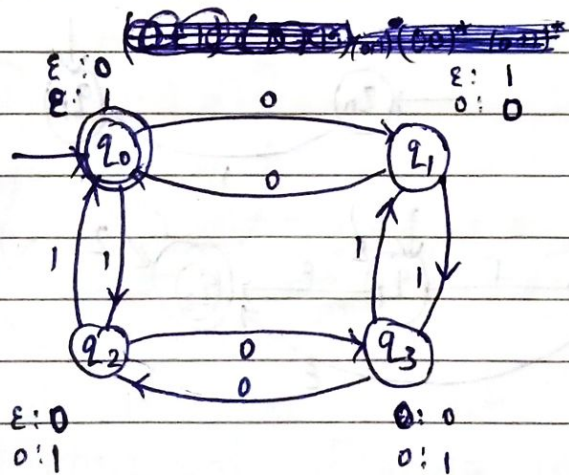
$2(5m+2)+1$

$2(5m+3)+1$

$2(5m+4)+1$



DFA for even no. of 0 and even no. of 1



Final state

$q_0 \rightarrow \epsilon=0, 1$

$q_1 \rightarrow \epsilon=1, \text{odd}=0$

$q_2 \rightarrow \epsilon=0, \text{odd}=1$

$q_3 \rightarrow \text{odd}=0, \text{odd}=1$

Elimination of Null moves from the transition system \Rightarrow

Suppose we want to replace a ϵ from vertex V_1 to vertex V_2 then the following procedure is applied:—

Step-1

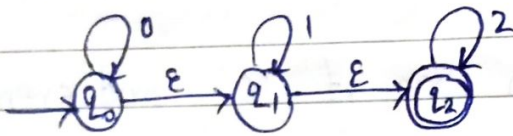
Find all the edges starting from V_2

Step-2 Delete the ϵ transition V_2 from V_1 and label the transition exactly same as in the next transition from V_1 with same label.

Step-3 If V_1 is the initial state make V_2 also a initial state.

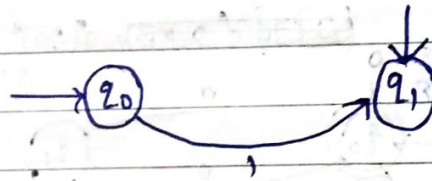
Step-4 If V_2 is the final state then make V_1 also a final state.

Q. A finite automata with ϵ move is given convert it into an equivalent automata without ϵ moves.

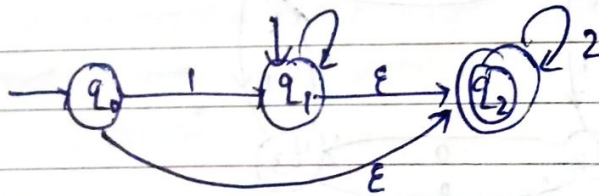


dy -

Step - 1, 2



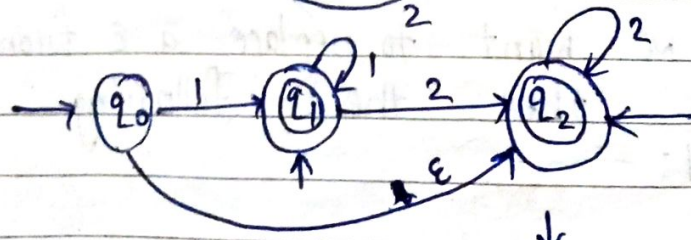
Now



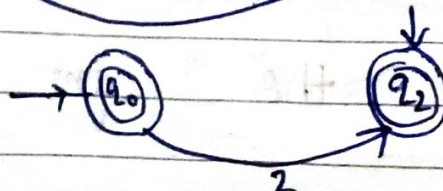
Step - 1, 2, 3, 4

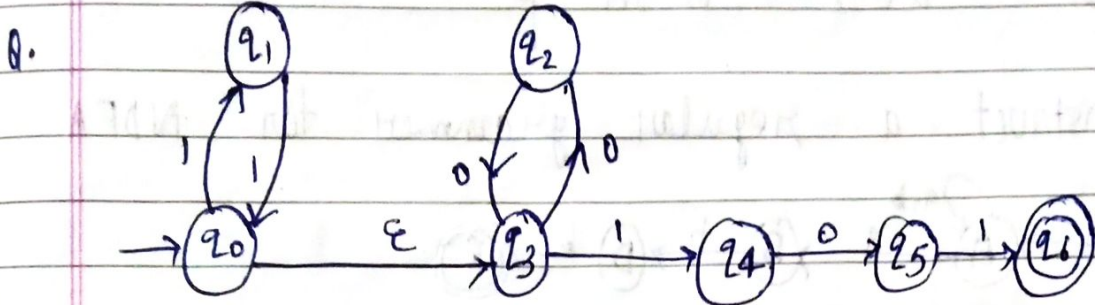
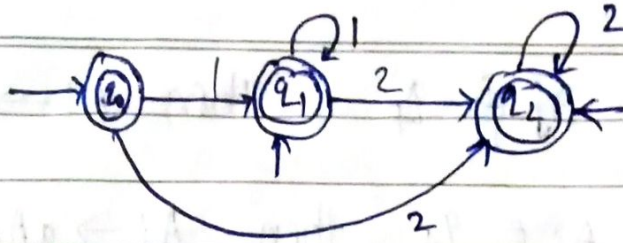


Now

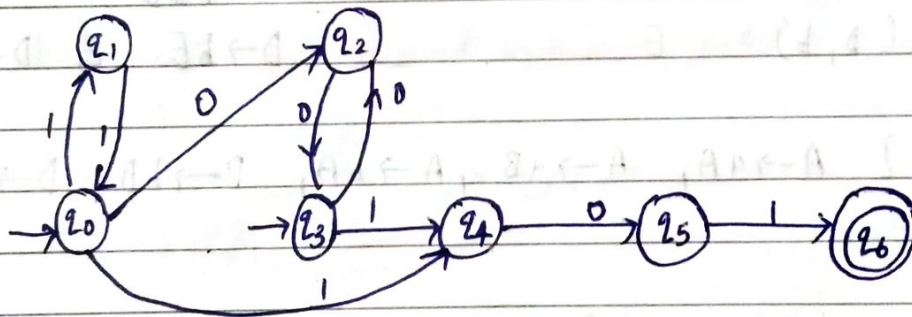
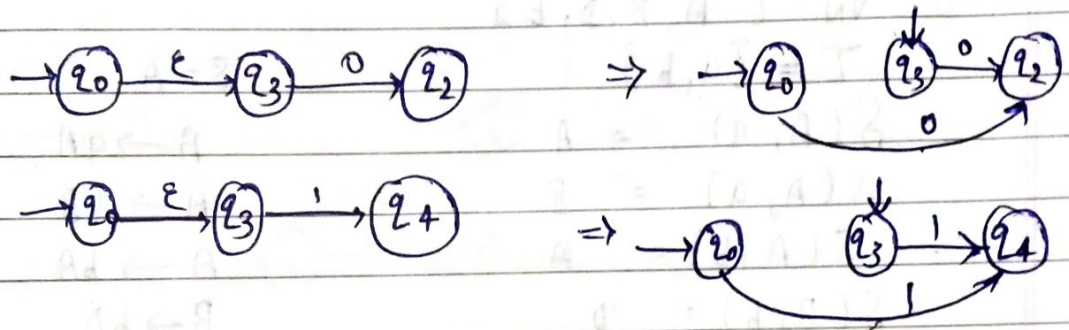


Again





Ans -



Construction of Regular Grammar for a given DFA \Rightarrow

Let $M = (Q, \Sigma, \delta, A_0, q_F)$ where

$Q = \{A_0, A_1, A_2, \dots, A_n\}$

For each state we assume variable from Q

let $G = (V_N, \Sigma, P, S)$

$V_N = \{A_0, A_1, A_2, \dots, A_n\}$

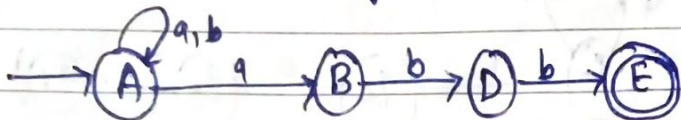
Σ is same for both M & G

P includes following production rule

case-1 If $\delta(A_i, a) = A_j \notin Q_f$ then $A_i \rightarrow aA_j$
 for $a \in \Sigma$

case-2 If $\delta(A_i, a) = A_j \in Q_f$ then $A_i \rightarrow aA_j$ & $A_i \rightarrow a$
 for $a \in \Sigma$ is in P .

Q. Construct a regular grammar for NDFA



$$V_N = \{A, B, D, E\}$$

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

$$S = A$$

$$\delta(A, a) = A$$

$$A \rightarrow aA$$

$$\delta(A, a) = B$$

$$A \rightarrow aB$$

$$\delta(A, b) = A$$

$$A \rightarrow bA$$

$$\delta(B, b) = D$$

$$B \rightarrow bD$$

$$\delta(D, b) = E$$

$$D \rightarrow bE \text{ \& } D \rightarrow b$$

$$P = \{A \rightarrow aA, A \rightarrow aB, A \rightarrow bA, B \rightarrow bD, D \rightarrow bE, D \rightarrow b\}$$

Construct of finite automata for regular
Grammar \Rightarrow

Let $G = (V_N, \Sigma, P, A_0)$ where

$$V_N = \{A_0, A_1, \dots, A_N\}$$

Let $M = (Q, \Sigma, \delta, q_0, Q_f)$ where

$$Q = \{A_0, A_1, \dots, A_N\}$$

Σ is same for both M & G then

δ include the following transition

case=1 $\delta(A_i, a) = A_j \notin f$ if production rule
 $A_i \rightarrow aA_j$ for $a \in \Sigma$ is in P

Ques-2 $\delta(A_i, a) = A_j \in f$ if production rule ~~$A_i \rightarrow a A_j$~~ $A_i \rightarrow a$ for $a \in \Sigma$ is in P .
 A_0 is the initial state
 $\&$ A_f is the final state.

Q. $G = (\{S\}, \{a, b\}, \{S \rightarrow aS, b\}, \{S\})$

Ans-

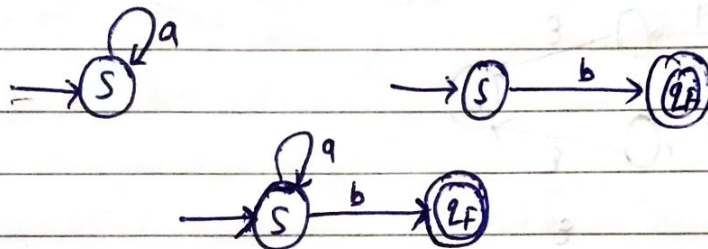
$$Q = \{S, q_f\}$$

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

$$q_0 = S$$

$$S \rightarrow aS, A_j \notin f$$

$$S \rightarrow b, A_j \in f$$



Q. Let $G = (\{S, A\}, \{a, b\}, P, \{S\})$ where
 P is $\{S \rightarrow aA, A \rightarrow bA, A \rightarrow a, A \rightarrow bS\}$

Ans-

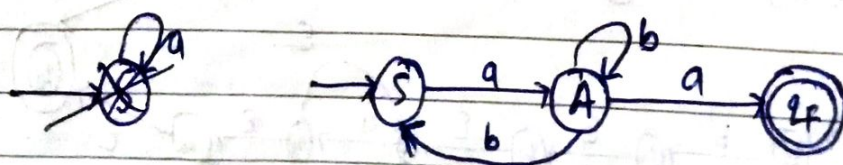
$$Q = \{S, A, q_f\}$$

$$\Rightarrow S \rightarrow aA \quad :- \quad \text{Diagram showing } S \xrightarrow{a} A$$

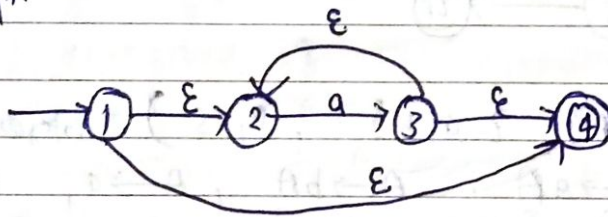
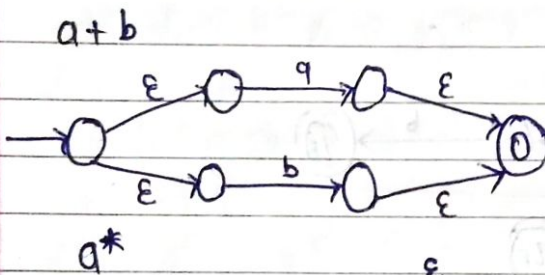
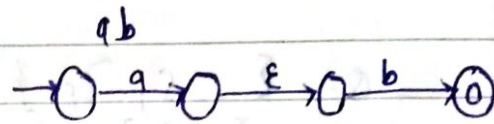
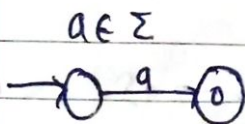
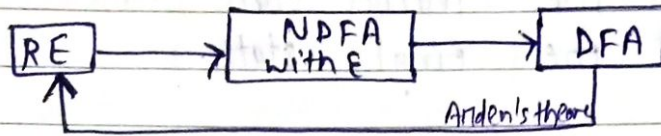
$$\Rightarrow A \rightarrow bA \quad :- \quad \text{Diagram showing } A \xrightarrow{b} A$$

$$\Rightarrow A \rightarrow a \quad :- \quad \text{Diagram showing } A \xrightarrow{a} q_f$$

$$\Rightarrow A \rightarrow bS \quad :- \quad \text{Diagram showing } A \xrightarrow{b} S$$



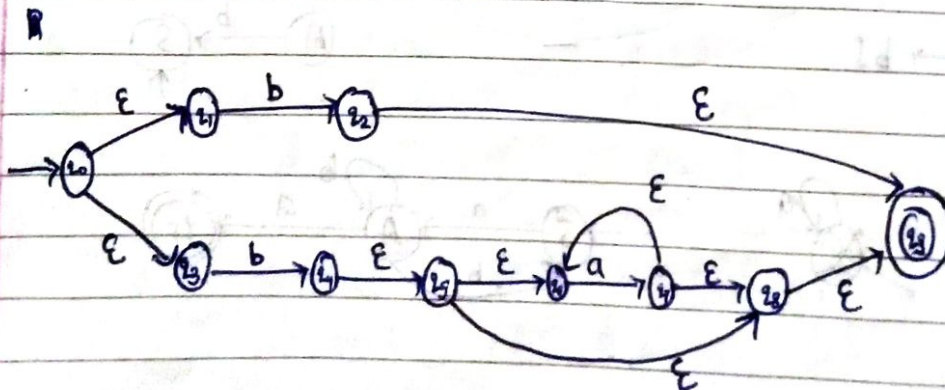
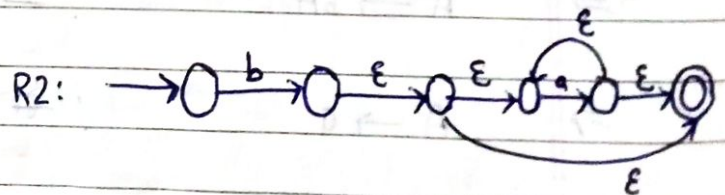
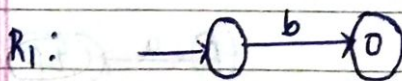
★ Conversion of Regular Expression to NDEA



Q. Construct a NDFA for the RE $b+ba^*$

Ans- $R = b + ba^*$

$R = R_1 + R_2$

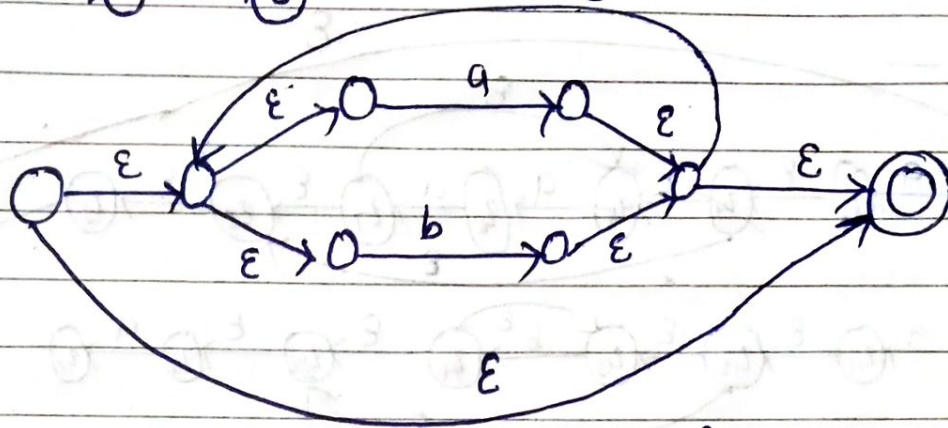


Q. Construct N DFA for RE $a(a+b)^*ab$

Ans - $R = a(a+b)^*ab$
 $R = R_1 \times R_2 \times R_3$

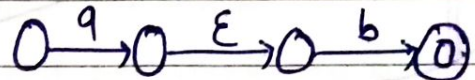
R_1 : 

R_2 :

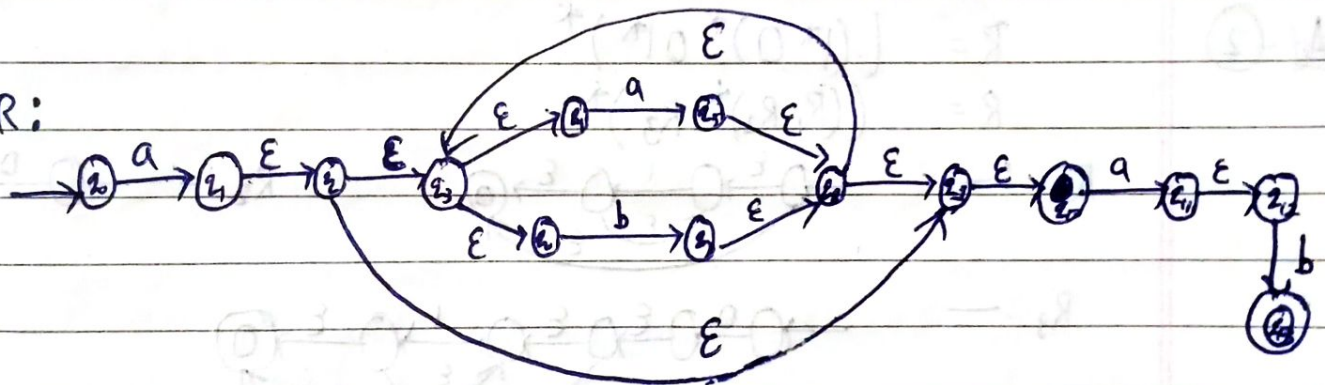


R_3 :

ab

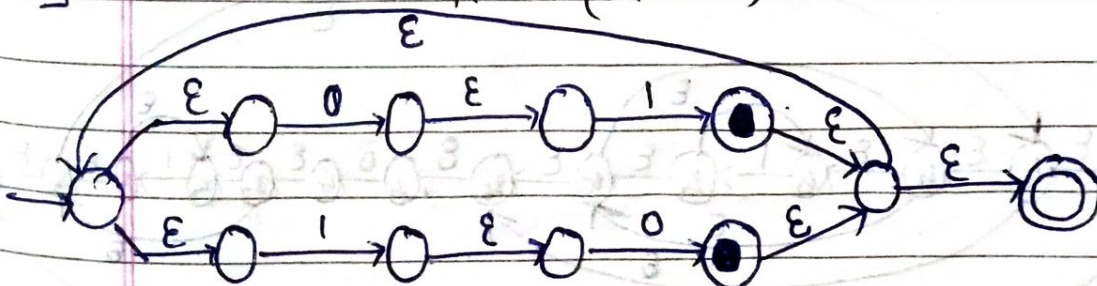


R :



Q. Construct the N DFA for RE $(01+10)^+$

Ans - $R = (01+10)^+$

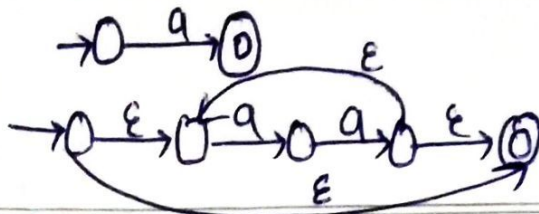


Q. 1 $(a(aa)^*b + ab^*a)^*$

Q. 2 $((1^*0)^*01^*)^*$

Ans - 1 $R = (a(aa)^*b + ab^*a)^*$
 $R = (R_1 R_2 R_3 + R_4 R_5)^*$

R_2 :-



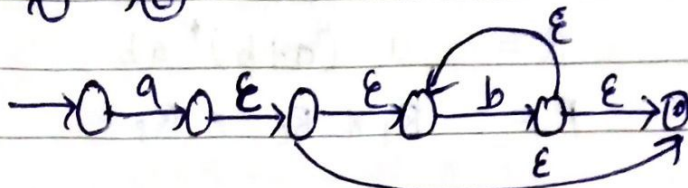
DATE : / /

PAGE NO. :

$R_3 : -$



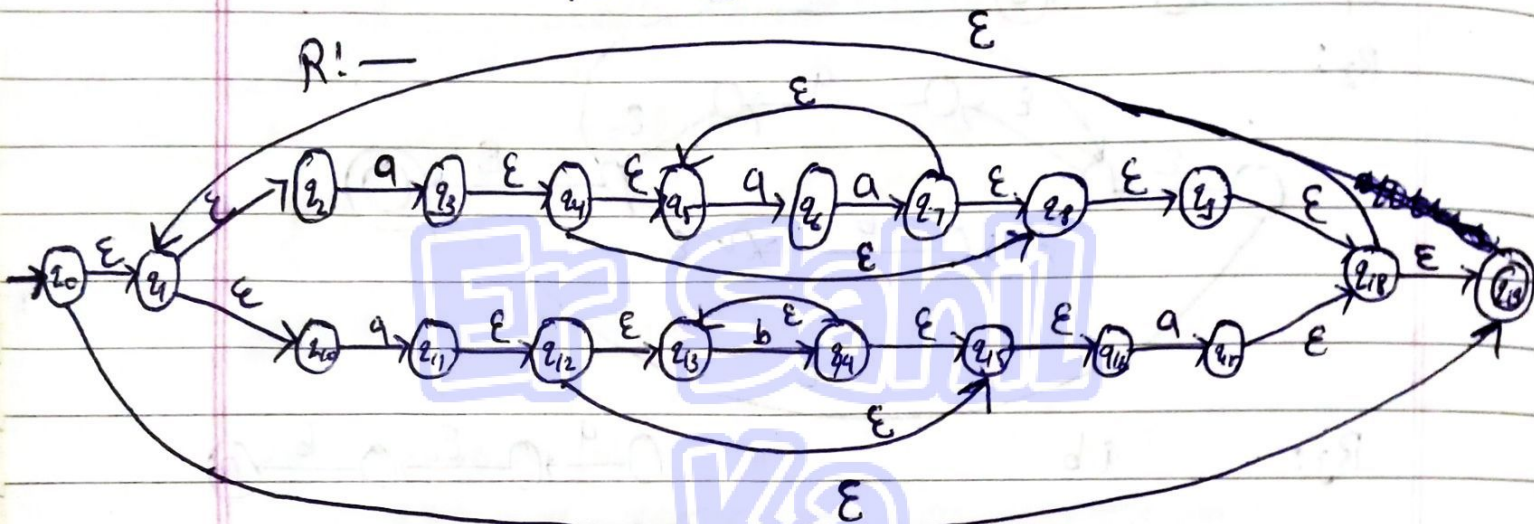
Ry: —



R5: —



R! —



Ex-2

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

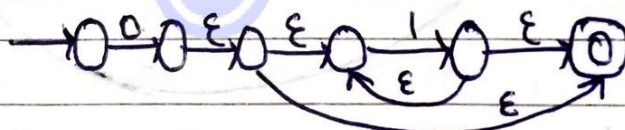
$$R = ((R_1 R_2)^* R_3)^*$$

R₁ :-

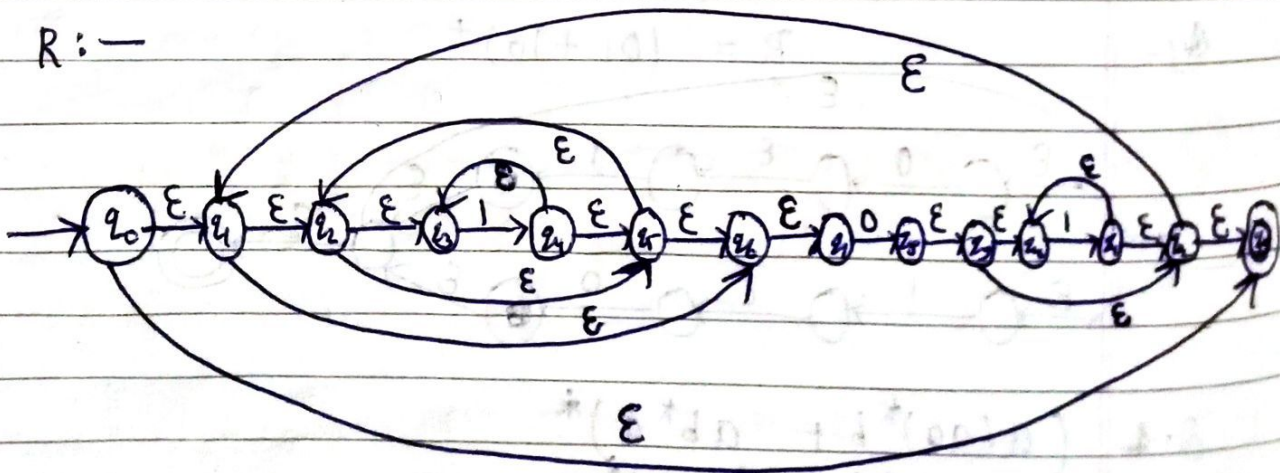


$R_2: \rightarrow \textcircled{0} \xrightarrow{0} \textcircled{0}$

R_3 :-



R: —



Construction of finite automata equivalent to RE

Step-1 Construction a transition diagram^(T.D.) equivalent to the given regular expression with null move.

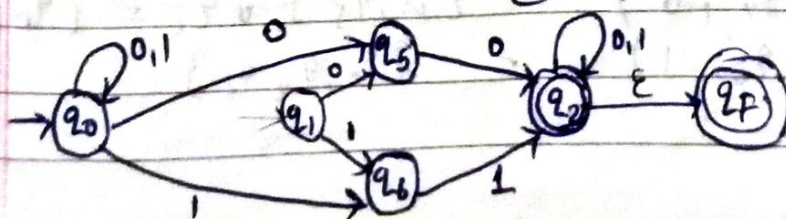
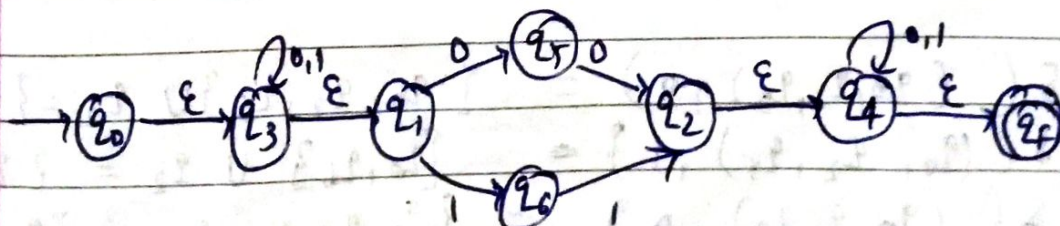
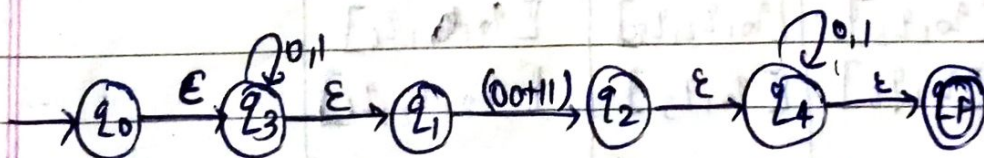
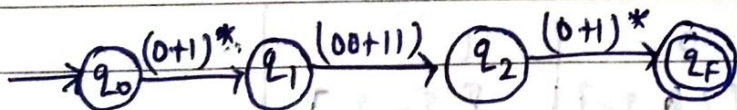
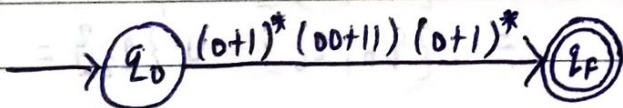
Step-2 Convert the NDFA with null to the NDFA without null move

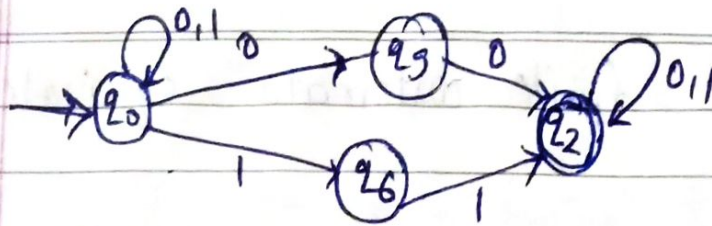
Step-3 Construct the transition table for T.D obtain in step 2

Step-4 Convert the obtained NDFA into DFA.

Q. Design a finite automata to the RE $(0+1)^* (00+11) (0+1)^*$

Ans





Now convert NFA to DFA

State	Input	
	0	1
→ q ₀	{q ₀ , q ₅ }	{q ₀ , q ₆ }
q ₅	q ₂	∅
q ₆	∅	q ₂
(q ₂)	q ₂	q ₂

$$\begin{aligned} \delta((q_0, q_5), 0) &= \delta(q_0, 0) \cup \delta(q_5, 0) \\ &= \{q_0, q_5\} \cup q_2 \\ &= \{q_0, q_2, q_5\} \end{aligned}$$

$$\begin{aligned} \delta((q_0, q_5), 1) &= \{q_0, q_6\} \cup \emptyset \\ &= \{q_0, q_6\} \end{aligned}$$

$$\delta((q_0, q_6), 0) = \{q_0, q_5\} \cup \emptyset = \{q_0, q_5\}$$

$$\delta((q_0, q_6), 1) = \{q_0, q_6\} \cup \{q_2\} = \{q_0, q_2, q_6\}$$

State	0	1
→ q₀	[q ₀ , q ₅]	[q ₀ , q ₆]
[q ₀ , q ₅]	[q ₀ , q ₂ , q ₅]	[q ₀ , q ₆]
[q ₀ , q ₆]	[q ₀ , q ₅]	[q ₀ , q ₂ , q ₆]

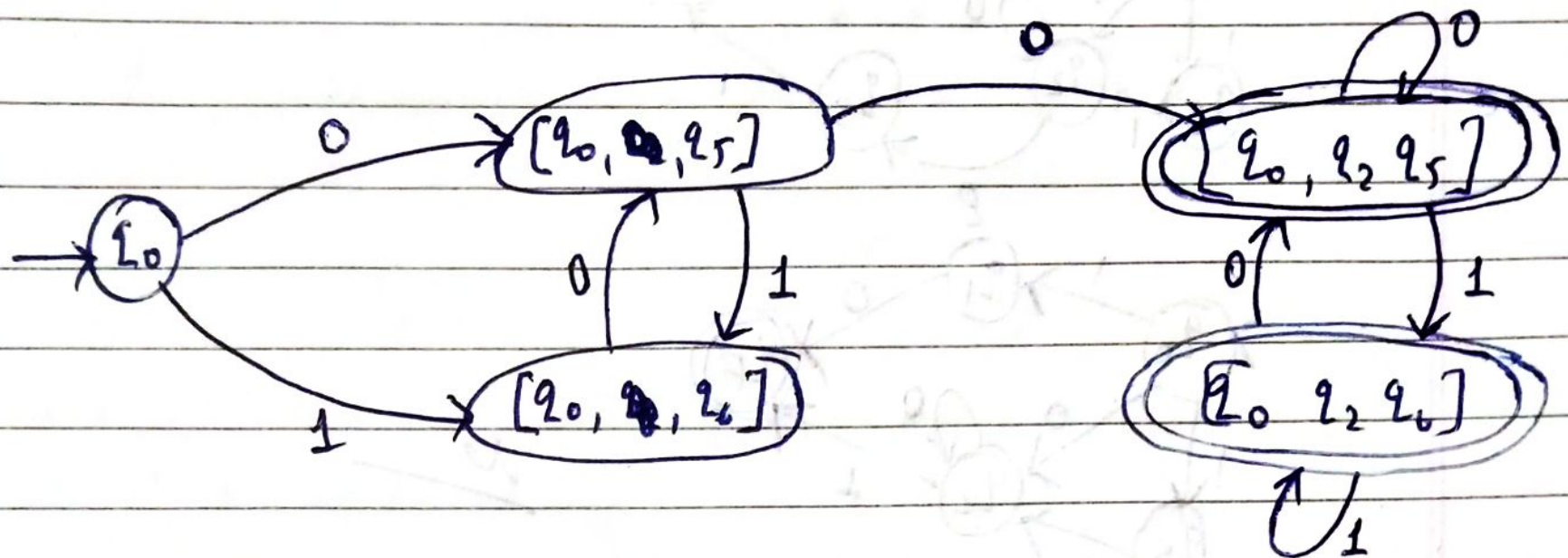
$$\delta((q_0, q_2, q_5), 0) = \{q_0, q_2, q_5\} \cup q_2 = \{q_0, q_2, q_5\}$$

$$\delta((q_0, q_2, q_5), 1) = \{q_0, q_6\} \cup q_2 = \{q_0, q_2, q_6\}$$

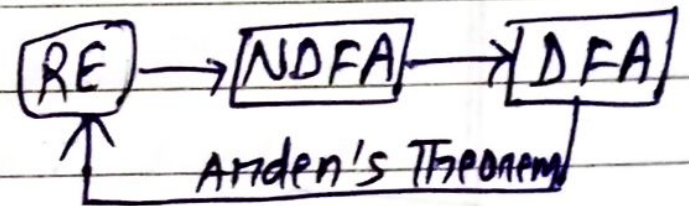
$$\delta((q_0, q_2, q_6), 0) = \{q_0, q_5\} \cup q_2 = \{q_0, q_2, q_5\}$$

$$\delta((q_0, q_2, q_6), 1) = \{q_0, q_2, q_6\}$$

State	0	1
$\rightarrow q_0$	$[q_0, q_5]$	$[q_0, q_6]$
$[q_0, q_5]$	$[q_0, q_5]$	$[q_0, q_6]$
$[q_0, q_6]$	$[q_0, q_5]$	$[q_0, q_2, q_6]$
$[q_0, q_2, q_5]$	$[q_0, q_2, q_5]$	$[q_0, q_2, q_6]$
$[q_0, q_2, q_6]$	$[q_0, q_2, q_5]$	$[q_0, q_6, q_2]$



Conversion of finite automata into RE \Rightarrow

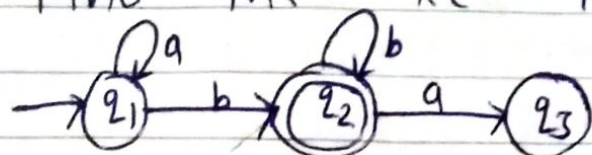


- (i) $\emptyset + \alpha = \alpha$
- (ii) $\emptyset \alpha = \alpha \emptyset = \emptyset$
- (iii) $\epsilon \alpha = \alpha \epsilon = \alpha$
- (iv) $\epsilon^* = \epsilon$ or $\emptyset^* = \epsilon$
- (v) $\alpha + \alpha = \alpha$
- (vi) $\alpha^* \alpha^* = \alpha^*$
- (vii) $\alpha \alpha^* = \alpha^* \alpha = \alpha^+$
- (viii) $(\alpha^*)^* = \alpha^*$
- (ix) $(PQ)^* P = P(QP)^*$
- (x) $(P+Q)^* = (P^* Q^*)^* = (P^* + Q^*)^*$
- (xi) $(P+Q)R = PR + QR$

Arden's Theorem:- If P & Q are two RE over an alphabet S such that P doesn't represent the ϵ string then $R = Q + RP$ has a unique solⁿ $R = QP^*$

solution i.e. $R = QP^*$

Q. Find the RE for the T.D. given:-



Ans -

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

$$q_2 = q_1 b + q_2 b \quad \text{--- (2)}$$

$$q_3 = q_2 a \quad \text{--- (3)}$$

From eqⁿ - (1)

$$q_1 = \epsilon + q_1 a$$

$$\therefore R = Q + RP \quad \text{then} \quad R = QP^*$$

$$q_1 = \epsilon a^*$$

where $R = q_1$
 $Q = \epsilon$
 $P = a$
 $R = q_1$

$$q_1 = a^* \quad \text{--- (4)}$$

From eqⁿ - (2)

$$q_2 = q_1 b + q_2 b$$

$$q_2 = a^* b + q_2 b$$

$$\therefore R = Q + RP \quad \text{then} \quad R = QP^*$$

$$q_2 = a^* b b^*$$

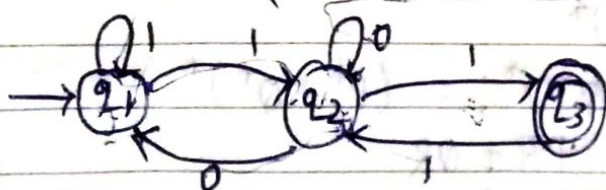
$$q_2 = a^* b^+$$

$$a^+ = a^*$$

Ans

Q. prove the string recognized by given automata is:

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



$$q_1 = \epsilon + q_1 1 + q_2 0 \quad \text{--- (1)}$$

$$q_2 = q_1 1 + q_2 0 + q_3 1 \quad \text{--- (2)}$$

$$q_3 = q_2 1 \quad \text{--- (3)}$$

Ans -

In eqⁿ - (2)

$$L_2 = q_1 1 + q_2 0 + q_2 11$$

$$q_2 = q_1 1 + q_2 (0+11)$$

$$\therefore R = Q + RP \text{ then } R = QP^*$$

$$q_2 = q_1 1 (0+11)^* \text{ --- (4)}$$

In eqⁿ - (1)

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

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

$$\therefore R = Q + RP \text{ then } R = QP^*$$

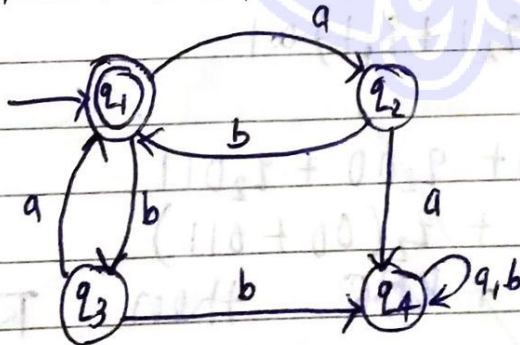
$$q_1 = \epsilon (1 + 1(0+11)^* 0)^*$$

$$q_2 = \epsilon (1 + 1(0+11)^* 0)^* 1 (0+11)^*$$

So now in eqⁿ - (3)

$$L_3 = (\epsilon (1 + 1(0+11)^* 0)^* 1 (0+11)^*)^*$$

Q. Find the RE for finite automata



Ans -

$$q_1 = \epsilon + q_2 b + q_3 a \text{ --- (1)}$$

$$q_2 = q_1 a \text{ --- (2)}$$

$$q_3 = q_1 b \text{ --- (3)}$$

$$q_4 = q_4 a + q_4 b + q_3 b + q_2 a \text{ --- (4)}$$

In eqⁿ - (1)

$$q_1 = \epsilon + q_1 ab + q_1 ba$$

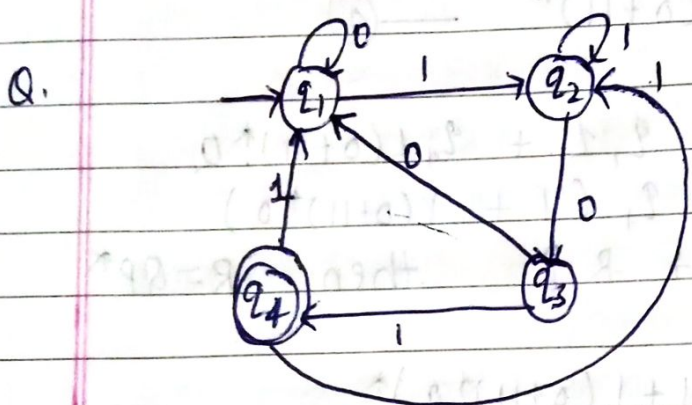
$$q_1 = \epsilon + q_1(ab+ba)$$

$$\therefore R = \emptyset + RP \text{ then}$$

$$R = \emptyset P^*$$

$$q_1 = \epsilon (ab+ba)^*$$

$$q_1 = (ab+ba)^*$$



Ans-

$$q_1 = \epsilon + q_1 0 + q_3 0 + q_4 1 \quad \text{--- (1)}$$

$$q_2 = q_1 1 + q_2 1 + q_4 1 \quad \text{--- (2)}$$

$$q_3 = q_2 0 \quad \text{--- (3)}$$

$$q_4 = q_3 1 \quad \text{--- (4)}$$

From (2), (4) & (3)

$$q_4 = q_2 0 1$$

$$q_4 = (q_1 1 + q_2 1 + q_4 1) 0 1$$

In eqⁿ (1)

$$q_1 = \epsilon + q_1 0 + q_2 0 0 + q_2 0 1 1$$

$$q_1 = q_1 0 + q_2 (0 0 + 0 1 1)$$

$$R = RP + QP \text{ then } R = \emptyset P^*$$

$$q_1 = q_2 (0 0 + 0 1 1) 0^*$$

In eqⁿ (2)

$$q_2 = q_2 (0 0 + 0 1 1) 0^* 1 + q_2 1 + q_2 0 1 1$$

$$q_2 = q_2 [(0 0 + 0 1 1) 0^* 1 + 1 + 0 1 1]$$

$$q_2 = q_1 1 + q_2 1 + q_4 1$$

$$q_2 = q_1 1 + q_2 1 + q_2 0 1 1$$

$$q_2 = q_1 1 + q_2 (1 + 0 1 1)$$

$$\therefore R = Q + RP \text{ then } R = QP^*$$

$$q_2 = q_1 1 (1 + 0 1 1)^* \quad \text{--- (5)}$$

$$q_3 = q_1 1 (1 + 0 1 1)^* 0 \quad \text{--- (6)}$$

$$q_4 = q_1 1 (1 + 0 1 1)^* 0 1 \quad \text{--- (7)}$$

$$q_4 = (\epsilon + q_1 0 + q_3 0 + q_4 1) 1 (1 + 0 1 1)^* 0 1$$

Now

$$q_1 = \epsilon + q_1 0 + q_1 1 (1 + 0 1 1)^* 0 0 + q_1 1 (1 + 0 1 1)^* 0 1 1$$

$$q_1 = \epsilon + q_1 (0 + 1 (1 + 0 1 1)^* 0 0 + 1 (1 + 0 1 1)^* 0 1 1)$$

$$R = Q + RP \text{ then } R = QP^*$$

$$q_1 = \epsilon (0 + 1 (1 + 0 1 1)^* 0 0 + 1 (1 + 0 1 1)^* 0 1 1)^*$$

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

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

So

$$q_4 = (0 + 1 (1 + 0 1 1)^* 0 0 0 + 1 (1 + 0 1 1)^* 0 1 1) ($$

$$q_4 = (0 + 1 (1 + 0 1 1)^* 0 0 + 1 (1 + 0 1 1)^* 0 1 1)^* 1 (1 + 0 1 1)^* 0 1$$

$$q_4 = (0 + 1 (1 + 0 1 1)^* 0 (0 + 1 1))^* 1 (1 + 0 1 1)^* 0 1$$

Equivalence of 2 RE :-

equivalence of RE P & Q To prove the following steps are involved for the

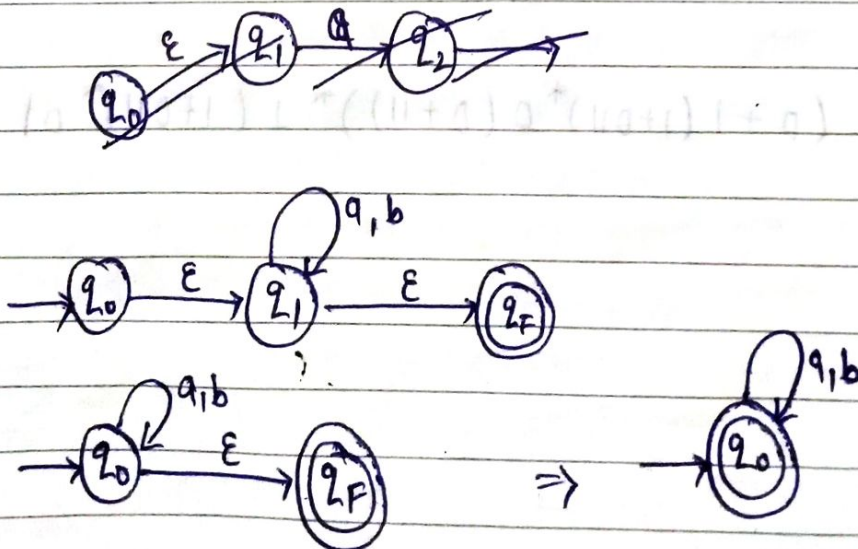
Case-(i) For the RE P & Q if we find the string that is in the one set but not in another set then this ^{2 exp} ~~is not~~ equivalence. If they are in same set then this are equivalent.

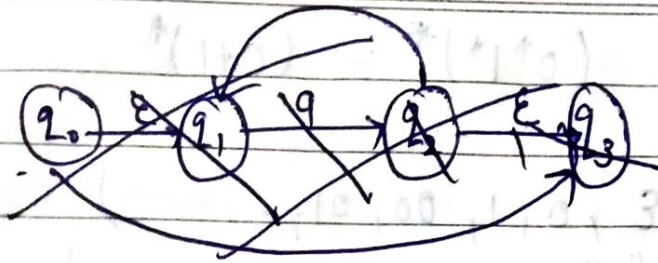
Case-(ii) We can use the identities to prove the equivalence of the RE P & Q .

Case-(iii) We construct the corresponding finite automata M & M' for the RE P & Q and prove that they are equal & if they are not then they are not equivalent.

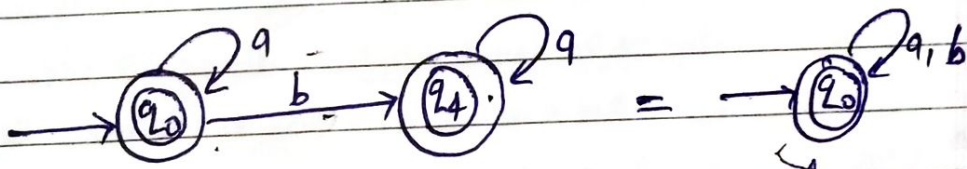
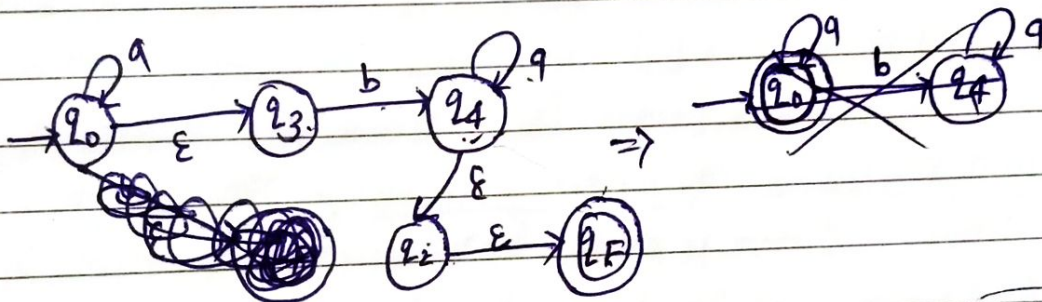
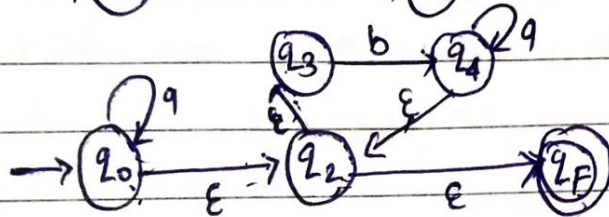
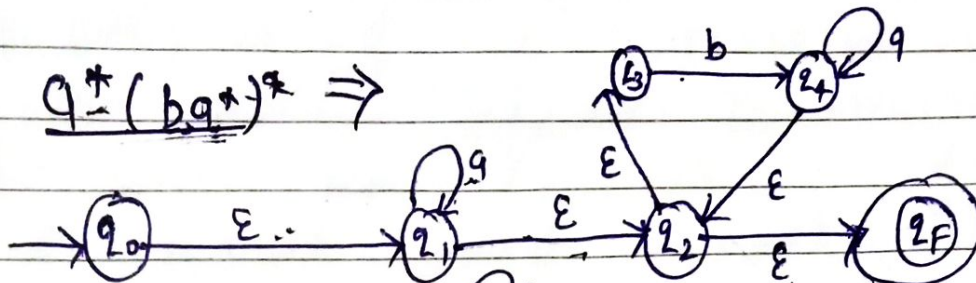
Q.
Ans.

$(a+b)^* = a^*(ba^*)^*$ prove it is equivalent or not





$$q^*(bq^*)^*$$



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

$$\underline{\text{Ans}} - \text{LHS} (1 + 00^*1)(\epsilon + (0 + 10^*1)^*(0 + 10^*1)) \quad (0 + 10^*1)^* = \epsilon + 10^*1 + 10^*10^*1 + \dots$$

$$\begin{aligned} & (1 + 00^*1)(0 + 10^*1)^* \\ & (\epsilon + 00^*1)(0 + 10^*1)^* \\ & 0^*1(0 + 10^*1)^* = \text{RHS} \end{aligned}$$

Context Free Grammar (CFG) \Rightarrow

A context free Grammar (CFG) consists of ^{production} rules. Each production has an abstract symbol called a non-terminal as its LHS and the sequence of one or more terminal and terminal symbol as its RHS. A Grammar $G = (V_N, P, \Sigma, S)$ is said to be CFG if the production rule of G is in the form

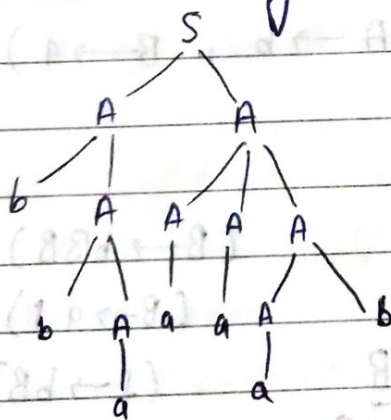
$$A \rightarrow \alpha$$

$$A \rightarrow BC \quad \alpha \in (V_N \cup \Sigma)^*$$

Left most derivation & Right most derivation free :-

Q. Given CFG is $S \rightarrow AA, A \rightarrow AAA/bA/Ab/a$ and string is $w = \underline{bbaaqab}$

Ans -



$$S \rightarrow \underline{AA}$$

$$S \rightarrow \underline{bAA}$$

$$S \rightarrow \underline{bbAA}$$

$$S \rightarrow \underline{bbAa}$$

$$S \rightarrow \underline{bbqAAA}$$

$$S \rightarrow \underline{bbaaqAb}$$

$$\boxed{S \rightarrow bbaaqab}$$

Q. Consider the CFG is $S \rightarrow aB/bA, A \rightarrow a/aS/bAA, B \rightarrow b/bS/aBB$ find the parse tree for the string $w = \underline{aabbabba}$

Ans -

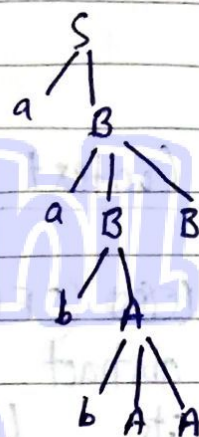
$w = aabbabba$

$S \rightarrow aB$

$S \rightarrow aaBB$

$S \rightarrow aabAAS$

$S \rightarrow aabbAAba$



Q. Consider the following CFG for the production

$S \rightarrow bB/aA$, $A \rightarrow bs/b/aAA$

$B \rightarrow a/as/bBB$

$w = bbaababba$

Ans - LMD

$S \rightarrow bB$

$S \rightarrow bbBB$ ($B \rightarrow bBB$)

$S \rightarrow bbaSB$ ($B \rightarrow aS$)

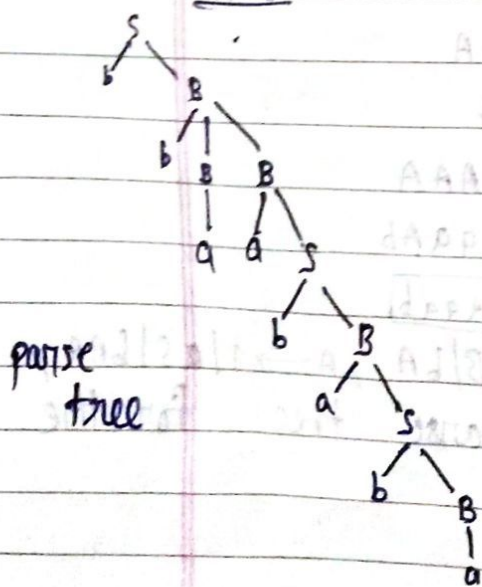
$S \rightarrow bbaqAB$ ($S \rightarrow aA$)

$S \rightarrow bbaqbSB$ ($A \rightarrow bS$)

$S \rightarrow bbaqbaAB$ ($S \rightarrow aA$)

$S \rightarrow bbaqbaba$ ($A \rightarrow b$, $B \rightarrow a$)

RMD



parse
tree

$S \rightarrow bB$

$S \rightarrow bbBB$ ($B \rightarrow bBB$)

$S \rightarrow bbBaS$ ($B \rightarrow aS$)

$S \rightarrow bbBabB$ ($S \rightarrow bB$)

$S \rightarrow bbBabaS$ ($B \rightarrow aS$)

$S \rightarrow bbBababB$ ($S \rightarrow bB$)

$S \rightarrow bbBababba$ ($B \rightarrow a$)

$S \rightarrow bbaababba$

Context free Language :-

CFL are the set of language that are accepted by non-deterministic push down automata to provide us with some information. The properties of CFL are

- (i) The CFL are closed under union. $L = L_1 \cup L_2$
- (ii) CFL are ~~not~~ closed under concatenation. $L = L_1 L_2$
- (iii) The CFL are closed ~~place~~ closer. $L = L_1^*$
- (iv) The CFL are not closed under ~~section~~ intersection. $L \neq L_1 \cap L_2$ ~~+++~~
- (v) The CFL are not closed under complement. ~~+++~~ $L \neq \bar{L}$

Ambiguity and CFG :-

The parse tree produced by the grammar is not unique whether the derivation left most or right most.

(a) Ambiguous Grammar :-

A Grammar that produces more than one parse tree is called ambiguous grammar. Ambiguous grammar are undesirable because multiple tree provide multiple information about a certain tree. Ambiguity is the negative property of a grammar.

(b) Unambiguous grammar :-

If a parse tree produced by the grammar is unique i.e. only one parse tree is generated for the given string

- either through LMD or through RMD that is only one and single output is generated.

Q. show that the following grammar are ambiguous

$$S \rightarrow SS | a | b$$

Ans -

LMD:-

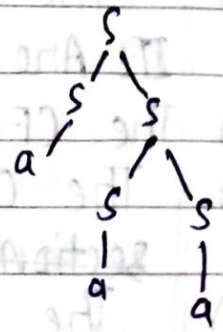
$$S \rightarrow SS$$

$$S \rightarrow as$$

$$S \rightarrow aSS$$

$$S \rightarrow aas$$

$$S \rightarrow aqa \text{ or } S \rightarrow bbb$$



RMD:-

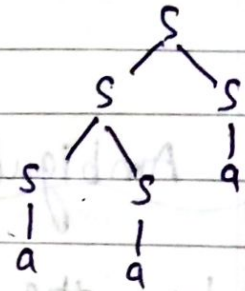
$$S \rightarrow SS$$

$$S \rightarrow Sa$$

$$S \rightarrow SSa$$

$$S \rightarrow Saa$$

$$S \rightarrow aqa$$



So the grammar is ambiguous.

Q. show that the following grammar are ambiguous

$$S \rightarrow A | B | b$$

$$A \rightarrow aAB | ab$$

$$B \rightarrow abB / \epsilon$$

Ans -

LMD :-

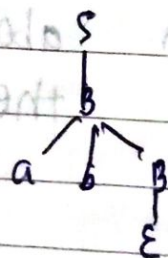
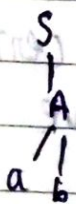
$$S \rightarrow A$$

$$S \rightarrow ab$$

$$S \rightarrow B$$

$$S \rightarrow abB$$

$$S \rightarrow ab$$



There are multiple parse tree so grammar is ambiguous.

Simplification of CFG \Rightarrow

The reduction of

CFG involves the following steps:-

- (i) We have to eliminate ^{useless} symbol ~~or~~ those variable or terminal that do not appear in any derivation of terminal string from the start symbol.
- (ii) We have to eliminate null production those of the form $A \rightarrow \epsilon$ for some variable A .
- (iii) We have to eliminate unit production those of the form $A \rightarrow B$ and $B \rightarrow A$ for variable A & B can be rewritten as $A \rightarrow a$



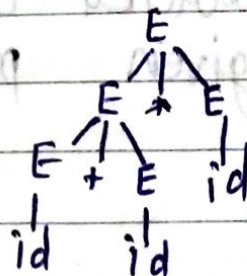
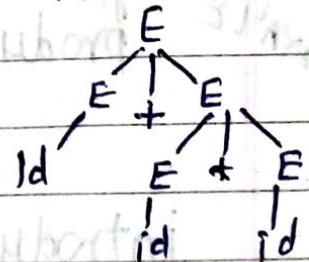
Eliminating Ambiguity :- As we know ambiguous grammar is not acceptable

so to make it acceptable grammar has to be ~~re-written~~ so as to ~~reduce~~ ^{remove} ambiguity.

Ex- For example, the given grammar as
 $E \rightarrow E + E$, $E \rightarrow E * E$, $E \rightarrow id$ and
 we want to produce the $w = id + id * id$

Ans-
 $E \rightarrow E + E$
 $E \rightarrow id + E$
 $E \rightarrow id + E * E$
 $E \rightarrow id + id * id$

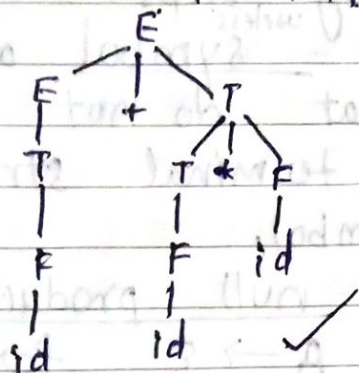
or
 $E \rightarrow E * E$
 $E \rightarrow E + E * E$
 $E \rightarrow id + E * E$
 $E \rightarrow id + id * E$
 $E \rightarrow id + id * id$



$$E \rightarrow E + T / T$$

$$T \rightarrow T * F / F$$

$$F \rightarrow id$$



$$E \rightarrow E + T$$

$$E \rightarrow E + T * F$$

$$E \rightarrow T + T * F$$

$$E \rightarrow F + F * F$$

$$E \rightarrow id + id * id$$

$$E \rightarrow E + (E) \quad T \rightarrow F$$

$$E \rightarrow (E) \quad E \rightarrow E * (E) F$$

★
2

Elimination of Left recursion \Rightarrow

A grammar is left recursive if the left most symbol on the right side of production is same as known non-terminal on the left side of production.

In other words, production is of the form $A \rightarrow A\alpha$ for some string α if grammar is left recursive. Suppose the grammar is in the following form

$$A \rightarrow A\alpha / \beta$$

to remove the immediate left recursion from the grammar of above type we will convert it into following productions

$$A \rightarrow A\alpha / \beta$$

$$A \rightarrow \beta A'$$

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

$$A \rightarrow \beta A'$$

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

thus we have introduced another non-terminal A' in the given production.

Q. Given production for some grammar

$$E \rightarrow E+T / T$$

$$T \rightarrow T * F / F$$

$$F \rightarrow (E) / id$$

recursion in case if it is present

Ans-

$$E \rightarrow E+T / T$$

$$T \rightarrow T * F / F$$

$$(A \rightarrow A\alpha / \beta)$$

$$A \rightarrow \beta A'$$

$$A' \rightarrow A' / \epsilon$$

$$E \rightarrow A$$

$$+T \rightarrow \alpha$$

$$T \rightarrow \beta$$

$$E \rightarrow TE'$$

$$E' \rightarrow +TE' / \epsilon$$

$$T \rightarrow FT'$$

$$T' \rightarrow *FT' / \epsilon$$

Now the obtained productions are

$$E \rightarrow TE'$$

$$T \rightarrow FT'$$

$$E' \rightarrow +TE' / \epsilon$$

$$T' \rightarrow *FT' / \epsilon, F \rightarrow (E) / id$$

* (3)

Left factoring :- Sometimes more than one production

rule is given for same non-terminal i.e. we have more than one choices and it is difficult to decide which one to apply. The productions are generally of the form

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

Here α is some string

till we have not seen complete α string i.e. we have no idea which of the ~~prod~~ A production is used. So the production after left factoring are

$$A \rightarrow \alpha A'$$

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

where A' is new non-terminal.

Q. Consider the following grammar applying left factoring to it.

$$A \rightarrow xByA / xByAZA / a$$

$$B \rightarrow b$$

Ans -

$$A \rightarrow \frac{xByA}{\alpha} \frac{\epsilon}{\beta_1} / \frac{xByA}{\alpha} \frac{zA}{\beta_2}$$

$$\therefore A \rightarrow \alpha A'$$

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

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

↓

Now

$$A \rightarrow \alpha A'$$

$$A \rightarrow xByAA' / \alpha$$

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

$$A' \rightarrow \epsilon / zA$$

$$B \rightarrow b$$

4

Elimination of Useless Symbol \Rightarrow

The symbol that can be used in the derivation due to there unavailability in the production rule or inability in deriving the terminals are known as useless symbol.

It means that the symbols are not reachable from start states and hence it can be removed.

Q.(i) Eliminate the useless symbol from the given CFG.

$$S \rightarrow \overline{AB} / CA$$

$$\times B \rightarrow BC / AB$$

$$C \rightarrow \overline{B} / b$$

$$A \rightarrow a$$

Ans -

Then

$$S \rightarrow CA$$

$$C \rightarrow b$$

$$A \rightarrow a$$

(ii)

$$S \rightarrow aA / a / Bb / cC$$

$$A \rightarrow aB$$

$$B \rightarrow a / Aa$$

$$C \rightarrow aCC$$

$$D \rightarrow ddd$$

dy -

$$A \rightarrow aB \quad (\because B \rightarrow a)$$

$$A \rightarrow aA, B \rightarrow a/Aa$$

$$C \rightarrow cCD \quad (\because D \rightarrow ddd)$$

$$X C \rightarrow cCddd$$

~~After~~ Elimination $S \rightarrow aA / a / Bb / \boxed{cC}^X$

$$B \rightarrow a/Aa$$

$$A \rightarrow aB, X C \rightarrow cCddd, D \rightarrow ddd$$

After Elimination

$$S \rightarrow aA / a / Bb, A \rightarrow aB$$

$$B \rightarrow a/Aa$$

$$D \rightarrow ddd$$

★

5

Elimination of Null Production \Rightarrow

A production rule $A \rightarrow \epsilon$ is called Null production where A is a variable and if a variable B having production rule such that $B^* \Rightarrow \epsilon$ is called a Nullable variable

Q. Consider the following production rule and remove NULL production.

$$S \rightarrow a / Xb / aYa$$

$$X \rightarrow Y / \epsilon$$

$$Y \rightarrow b / X$$

dy -

As we know

$$X \rightarrow \epsilon$$

then

$$\boxed{S \rightarrow b}$$

And $Y \rightarrow X$ then $Y \rightarrow \epsilon$

~~then~~ So

$$\boxed{S \rightarrow aA}$$

So

$$S \rightarrow a / b / aA$$

$$X \rightarrow Y$$

$$Y \rightarrow b / X$$

Q. Eliminate the NULL CFG $S \rightarrow aS/AB$

Ans - $A \rightarrow \epsilon$ $B \rightarrow \epsilon$ $D \rightarrow b$
 $S \rightarrow aS$ ($\because S \rightarrow AB$)
 $S \rightarrow aAB$ ($\because A \rightarrow \epsilon, B \rightarrow \epsilon$)
 $S \rightarrow a$

If $S \rightarrow AB$ ($\because A \rightarrow \epsilon$)

$S \rightarrow B$

If $S \rightarrow AB$ ($\because B \rightarrow \epsilon$)

$S \rightarrow A$

After Elimination the NULL CFG is

$S \rightarrow aS/a/B/A/AB$

$D \rightarrow b$

Q. Eliminate the NULL production from following CFG

$S \rightarrow XYX$

$X \rightarrow 0X/\epsilon$

$Y \rightarrow 1Y/\epsilon$

Ans - $S \rightarrow XYX$ ($\because X \rightarrow \epsilon$)

$S \rightarrow Y$ ($\because Y \rightarrow \epsilon$)

If $S \rightarrow XYX$ ($\because Y \rightarrow \epsilon$)

$S \rightarrow XX$

$S \rightarrow XY,$

$S \rightarrow YX$

If $S \rightarrow Y$ ($\because Y \rightarrow 1Y$)

$S \rightarrow 1Y$ ($\because Y \rightarrow \epsilon$)

~~$S \rightarrow 1$~~

$S \rightarrow XX$

$S \rightarrow 0X0X$ ($\because X \rightarrow \epsilon$)

~~$S \rightarrow 00$~~

$X \rightarrow 0$ ($\because X \rightarrow \epsilon$)

$Y \rightarrow 1$ ($\because Y \rightarrow \epsilon$)

After elimination the NP

$$S \rightarrow \epsilon$$

$$S \rightarrow x y x / x / y / x y / y x / x x$$

$$x \rightarrow 0$$

$$y \rightarrow 1$$

6. Elimination of Unit production \Rightarrow

A production of type $A \rightarrow B$ where A, B are variable is called UP. Rule that create indirection and result into requirement of extra step which involved in deriving the terminal.

Q. Consider the CFG and remove the UNIT PRODUCTION

$$S \rightarrow AB$$

$$A \rightarrow a$$

$$B \rightarrow c/b$$

$$C \rightarrow D$$

$$B \rightarrow c$$

$$D \rightarrow E$$

$$E \rightarrow a$$

$$C \rightarrow D$$

then

$$A \rightarrow a$$

$$B \rightarrow a$$

$$C \rightarrow a$$

$$D \rightarrow a$$

$$E \rightarrow a$$

$$E \rightarrow a$$

There is no use of C, D & E in starting S so

obtained CFG: $S \rightarrow AB$

$$A \rightarrow a$$

$$B \rightarrow a$$

Q. Identify unit production $S \rightarrow A/bb, A \rightarrow B/b$
 $B \rightarrow S/a$

Unit production:- $S \rightarrow A/bb$

$$A \rightarrow B/b$$

$$B \rightarrow S/a$$

After Elimination of unit production:-

$$S \rightarrow bb/b/a$$

$$A \rightarrow b/a/bb$$

$$B \rightarrow a/bb/b$$

NORMAL FORMS FOR CFG \Rightarrow

We basically have to represent the grammar in some specifically format. There are 2 popular and important normal form :-

- ★ (i) Chomsky Normal form (CNF)
- ★ (ii) Greibach Normal form (GNF)

(i) Chomsky Normal Form:- A CFG is in CNF if every production is of the form $A \rightarrow a$, $A \rightarrow BC$ & $S \rightarrow \epsilon$ is in G if $\epsilon \in L(G)$. We assume that S does not appear on the RHS of any production. For example consider G whose production are - $S \rightarrow AB/E$, $A \rightarrow a$ & $B \rightarrow b$ then G is in CNF.

Method for converting CFG to CNF :-

- Step-① Eliminate the null production from the given CFG
- Step-② Eliminate the unit production from the given CFG
- Step-③ Eliminate the useless production from CFG
- Step-④ Put the grammar into CNF.

Step ④ Put the grammar into CNF

Q. Convert the following grammar to CNF

$$S \rightarrow bA/aB$$

$$A \rightarrow bAA/aS/a$$

$$B \rightarrow aBB/bS/a$$

Ans - ① There is no null production in the given grammar

② There is no unit production in given grammar.

③ There is no useless symbol in given grammar.

④ The production that are in the CNF are

$$A \rightarrow a$$

$$B \rightarrow a$$

The production that are not in CNF are

$$S \rightarrow bA/aB$$

$$A \rightarrow bAA/aS/a$$

$$B \rightarrow aBB/bS$$

We replace the terminal b by P_2 and a by P_1
Now obtain grammar are

$$S \rightarrow P_2A/P_1B$$

$$A \rightarrow P_2AA/P_1S/a$$

$$B \rightarrow P_1BB/P_2S/a$$

$$P_1 \rightarrow a$$

$$P_2 \rightarrow b$$

The production that are not in CNF are

$$A \rightarrow P_2AA$$

$$B \rightarrow P_1BB$$

We replace the AA by P_3 and BB by P_4 hence the grammar becomes

$$S \rightarrow P_2A/P_1B, A \rightarrow P_2P_3/P_1S/a$$

$$B \rightarrow P_1P_4/P_2S/a, P_1 \rightarrow a, P_2 \rightarrow b$$

$$P_3 \rightarrow AA, P_4 \rightarrow BB$$

Q. Convert the following grammar to CNF

$$\begin{aligned} S &\rightarrow abSb / a / aAb \\ A &\rightarrow bS / aAAb \end{aligned}$$

Ans-

(i), (ii), (iii) There is no null production, unit & useless symbol. The production that are in CNF are $S \rightarrow a$

(iv) The production that aren't in CNF are

$$S \rightarrow abSb / aAb$$

$$A \rightarrow bS / aAAb$$

~~One~~ replace $P_1 \rightarrow a$ & $P_2 \rightarrow b$

$$S \rightarrow P_1 P_2 S P_2 / P_1 A P_2$$

$$A \rightarrow P_2 S / P_1 A A P_2$$

Replace $P_3 \rightarrow P_1 A$

$$S \rightarrow P_1 P_2 S P_2 / P_3 P_2$$

$$A \rightarrow P_2 S / P_3 A P_2$$

Replace $P_4 \rightarrow S P_2$ & $P_5 \rightarrow P_1 P_2$, $P_6 \rightarrow A P_2$

Hence grammar becomes

$$S \rightarrow P_5 P_4 / P_5 P_2$$

$$A \rightarrow P_2 S / P_3 P_6$$

$$P_1 \rightarrow a$$

$$P_2 \rightarrow b$$

$$P_3 \rightarrow P_1 A$$

$$P_4 \rightarrow S P_2$$

$$P_5 \rightarrow P_1 P_2$$

$$P_6 \rightarrow A P_2$$

Q. Convert the following grammar to CNF

$$S \rightarrow s+s / s * s$$

$$S \rightarrow a/b$$

Ans

(i), (ii), (iii) \rightarrow

(iv) The production that are in CNF

$$S \rightarrow a/b$$

production that are not in CNF are

$$S \rightarrow S + S / S * S$$

replace $P_1 \rightarrow +$, $P_2 \rightarrow *$

$$S \rightarrow S P_1 S / S P_2 S$$

$$P_3 \rightarrow P_1 S, P_4 \rightarrow P_2 S$$

$$S = S P_3 / S P_4$$

Then

$$S \rightarrow S P_3 / S P_4 / a / b$$

$$P_1 \rightarrow +, P_2 \rightarrow *$$

$$P_3 \rightarrow P_1 S, P_4 \rightarrow P_2 S$$

* (iii) Cornbach normal form (CNF) \Rightarrow

A CFG is in CNF if every production is in the form $A \rightarrow a\alpha$ where $\alpha \in V_N^*$ and $a \in \Sigma$

$a \in \Sigma$ (α may be ϵ) and $S \rightarrow \epsilon$ is in G if $\epsilon \in L(G)$. We assume that S does not appear on the RHS of any production.

Method of conversion of CFG into CNF! —

- ① Every CFG must be in the form of CNF. If it is not then convert it in CNF.
- ② Rename all the variables of grammar by A, B, \dots where A_1 is the start symbol.
- ③ Now separate the variable that are already in the form of CNF. The proper form of CNF is $A \rightarrow a$ & $A \rightarrow a\alpha$
- ④ There are 3 cases for the separation of the productions that are not in CNF. These are
 - Case-I Production of the form $A_i \rightarrow A_j \alpha$ where $j > i$, production is in the proper form and leave it now such as.

Case-② Production of the form $A_j \rightarrow A_j \alpha$ where $j < i$, the production is not in proper form so to convert in proper form we apply left factoring to production & get the production in the form $A_i \rightarrow A_j \alpha$ where $i = j$

Case-③ Production in the form of $A_i \rightarrow A_j \alpha$ where $i = j$ now we apply left recursion by ~~introducing~~ introducing the new variable B

⑤ Now apply the left recursion in an production in the CNF every left symbol of the Right side of any production must start with terminal.

⑥ Every production is of the form $A_i \rightarrow a \alpha$ & $A_i \rightarrow A_j A_k$. Now again in production B_i , all productions have RHS beginning with terminal or variable of A_i .

$B \rightarrow a_i$ & $B_i \rightarrow a_i B_i$ finally we apply the substitution ~~rule~~ ~~substitution~~ rule and get the grammar which is in CNF.

Ex Q. Convert the following grammar into CNF

$$S \rightarrow AA/a$$

$$A \rightarrow SS/b$$

Ans-Step ① The grammar is in CNF

Step ② Rename the variable S & A to A_1 & A_2 respectively.

~~Now~~ Now the given grammar is

$$A_1 \rightarrow A_2 A_2 / a$$

$$A_2 \rightarrow A_1 A_1 / b$$

Step ③ The production that are in the form of CNF are
 $A_1 \rightarrow A_2 A_2 / a$ (jxi), $A_2 \rightarrow b$

The production that are not in CNF
 $A_2 \rightarrow A_1 A_1$ (jxi)

Step ④ By applying a left factoring in the production

$$A_2 \rightarrow A_2 A_2 A_1 / a A_1$$

Hence the grammar becomes

$$A_1 \rightarrow A_2 A_2 / a$$

$$A_2 \rightarrow A_2 A_2 A_1 / a A_1 / b$$

Left Recursion
 by introduction
 new variable B_2

Consider the production $A_2 \rightarrow A_2 A_2 A_1 / a A_1 / b$

Note :-

$A_i \rightarrow B_i$	$B_i \rightarrow \alpha_i$	$B_i \rightarrow \alpha_i B_i$
$A_i \rightarrow B_i B_i$	$B_i \rightarrow \alpha_i$	$B_i \rightarrow \alpha_i B_i$

$A_2 \rightarrow a A_1 B_2, A_2 \rightarrow b B_2$
 $B_2 \rightarrow A_2 A_1 B_2$
 $B_2 \rightarrow A_2 A_1$

$\therefore A \rightarrow B A'$
 $A' \rightarrow \alpha A' / \epsilon$

$A_2 \rightarrow a A_1 B_2$
 $A_2 \rightarrow b B_2$
 $B_2 \rightarrow A_2 A_1 B_2$
 $B_2 \rightarrow A_2 A_1$

Hence the grammar becomes

$$A_1 \rightarrow A_2 A_2 / a$$

$$A_2 \rightarrow a A_1 / b / a A_1 B_2 / b B_2 / b B_2$$

So the grammar

$$A_1 \rightarrow A_2 A_2 / a$$

$$A_2 \rightarrow a A_1 / b / a A_1 B_2 / b B_2 / b B_2$$

$$B_2 \rightarrow A_2 A_1 / A_2 A_1 B_2$$

Step ⑤ By applying left recursion on $A_1 \rightarrow A_2 A_2$
 $A_1 \rightarrow a A_1 A_2$ ($A_2 \rightarrow a A_1$)
 $A_1 \rightarrow b A_2$ ($A_2 \rightarrow b$)

$$A_1 \rightarrow aA_1B_2A_2$$

$$A_1 \rightarrow bB_2A_2$$

$$(\because A_2 \rightarrow aA_1B_2)$$

$$(\because A_2 \rightarrow bB_2)$$

Thus the resultant grammar is

$$A_1 \rightarrow aA_1A_2 / bA_2 / aA_1B_2A_2 / bB_2A_2 / a$$

$$A_2 \rightarrow aA_1 / b / aA_1B_2 / bB_2$$

$$B_2 \rightarrow A_2A_1 / A_2A_1B_2$$

Step (6) By applying substitution on $B_2 \rightarrow A_2A_1 / A_2A_1B_2$

$$B_2 \rightarrow aA_1A_1$$

$$B_2 \rightarrow aA_1A_1B_2$$

$$B_2 \rightarrow bA_1$$

$$B_2 \rightarrow bA_1B_2$$

$$B_2 \rightarrow aA_1B_2A_1$$

$$B_2 \rightarrow aA_1B_2A_1B_2$$

$$B_2 \rightarrow bB_2A_1$$

$$B_2 \rightarrow bB_2A_1B_2$$

Thus the grammar becomes

$$A_1 \rightarrow aA_1A_2 / bA_2 / aA_1B_2A_2 / bB_2A_2 / a$$

$$A_2 \rightarrow aA_1 / b / aA_1B_2 / bB_2$$

$$B_2 \rightarrow aA_1A_1 / bA_1 / aA_1B_2A_1 / bB_2A_1 / aA_1A_2B_2 / bA_1B_2 /$$

$$aA_1B_2A_1B_2 / bB_2A_1B_2$$

* Pumping Lemma for regular Expression ^{Set} \Rightarrow

Pumping lemma is useful tool to prove that the language is not regular.

The no. of states in a finite Automata is finite. Let us take it as n and it can recognise all the words of length less than n without any loop.

Let us suppose a regular language L has infinite no. of words and the length of these words may or may not be equal to n . It is not possible for the finite automata to recognise L without any loop. So finite automata can recognise L having some loop whenever the length of a given word is greater than or equal to n . Where n is a no. of state of finite automata. So when a string w has a bigger length then we break the string into 3 parts say x, y, z (y should not be null string). Here we can say this regular language can be pumped which means that the middle part of the string can be repeated many no. of times to produce a new string which may be accepted by the same language. The following steps are needed for proving that given set is not regular.

Step-① Assume that L is regular. Let n be the no. of states in the corresponding finite automata.

Step-② Now choose the a string w such that $|w| \geq n$. Using pumping lemma we can write $w = xyz$ and $|y| > 0$.

Step-③ Now find the suitable integer i $xy^iz \in L$. This can contradict our assumption hence L is not regular.

* Q. Show that language $L = \{a^n b^n \mid n \geq 1\}$ is not regular.

Ans - (1) Suppose $L = \{a^n b^n \mid n \geq 1\}$ is regular.
Let n be the no. of state in finite automata accepting L .

$$\begin{array}{ll} L = ab & n=1 \\ L = aabb & n=2 \\ \vdots & \vdots \\ L = a^n b^n & n=n \quad |w| = 2n \end{array}$$

(2)

Now we assume $|w| = |a^n b^n|$ which is equal to $2n$.

$$\text{So } |w| = 2n > n$$

With the help of the pumping lemma we can write $w = xyz$ with $|xy| \leq n$ and $|y| > 0$

(3) Now consider $w = xy^iz$ so the string y can be in the following forms

$$\text{let } w = \underbrace{a}_{x} \underbrace{ab}_{y} \underbrace{b}_{z} \quad \text{where } x=a, y=ab, z=b$$

$$\text{For } i=2 \quad w = xy^2z \\ w = a(ab)^2b = aababbb$$

$$w = aababbb \notin L$$

$$\text{Therefore } w = xy^2z \notin L$$

So there is a contradict that's why $L = \{a^n b^n \mid n \geq 1\}$ is not regular.

Q. Show that language $L = \{a^p \mid p \text{ is prime no.}\}$ is not regular.

Step-① Suppose L is regular. Let n be the no. of state in finite automata accepting L

$$w = a^p$$

$$|w| = |a^p| = p \quad p > n$$

Step-② Now we assume $w = a^p$ ~~which~~ and $|w| = p$ which is greater than n with the help of pumping lemma we can write $w = xyz$ with $|xy| \leq n$ and $|y| > 0$

$$w = \underbrace{a a a \dots a}_x \underbrace{\quad\quad\quad}_y \underbrace{a}_z$$

\xrightarrow{m}
 \xleftarrow{p}

$$|y| = |a^m| = m$$

Step-③ Now consider $w = xy^i z$

$$w = |xy^i z| = |xyz| + |y(i-1)|$$

$$w = p + m(i-1)$$

let suppose $i = p+1$

$$w = p + m(p+1-1)$$

$$w = p + mp = p(1+m)$$

$$w = p(1+m) \notin L$$

\therefore prime no. can not be in ~~factor~~ product form.

Therefore $w = xy^i z \notin L$

So there is a contradict that's why

$L = \{a^p \mid p \text{ is prime no.}\}$ is not regular

Q. Show that language $L = \{ a^{i^2} \mid i \geq 1 \}$ is not regular.

Ans-

Step-① Suppose L is regular. Let n be the no. of state in finite automata accepting L .

$$w = a^{i^2}$$
$$|w| = |a^{i^2}|$$

$$\text{if } i=1, w=a$$

$$\text{if } i=2, w=aaaa = a^{2^2}$$

$$\text{if } i=3, w=a^{3^2}$$

$$\vdots$$
$$\text{if } i=n, w=a^{n^2}$$

$$|w| = |a^{i^2}| = n^2$$

Step-②

Now we assume $w = a^{n^2} \neq |w| = n^2$ which is greater than n with the help of pumping lemma we can write $w = xyz$ with $|xy| \leq n$ and $|y| > 0$

Step-③

$$w = xyz$$

$$|w| = |x| + |y| + |z| = n^2$$

$$\text{if } i=2$$

$$w = xy^2z$$

$$|w| = |x| + 2|y| + |z|$$

$$\Rightarrow n^2 < |x| + 2|y| + |z|$$

$$n^2 < |x| + |y| + |z| + |y| < n^2 + n$$

$$\Rightarrow n^2 < |xy^2z| < n^2 + n + n + 1$$

$$n^2 < |xy^2z| < (n+1)^2$$

The words lie b/w the consecutive square

Therefore $|w| = |xy^2z| \notin L$ it means

w is not a consecutive square so

there is contradict that's why $L = \{ a^{i^2} \mid i \geq 1 \}$ is not regular.

Context free grammar $\left\{ \begin{array}{l} NT \rightarrow NTNT \\ NT \rightarrow T \end{array} \right.$

DATE: / /

PAGE NO.:

$NT \rightarrow$ non terminal

$T \rightarrow$ Terminal

* Pumping Lemma for context free grammar \Rightarrow

A pumping lemma is theorem used to show that if certain string belongs to a language then certain other strings must also belong to the language. If L is a CFL then strings of L that are atleast n symbol long can be pumped to produce additional strings. Suppose L is be a infinite CFL then there is some integer m such that if s is a string of L of length atleast m then $s = uvwxy$ with $|vwx| \leq m$ and $|vx| \geq 1$ and $uv^iwx^iy \in L$ for all non-negative value of i .

ep-① Assume L is context free. let n be the natural no. obtained by using the pumping lemma.

ep-② $z \in L$ so that $|z| \geq n$ write $z = uvwxy$ using pumping lemma.

ep-③ Find the suitable k so that $uv^kwx^ky \in L$ this is a contradiction so L is not context free.

Q. Show that $L = \{ a^n b^n c^n \mid n \geq 1 \}$

Ans- Step-①

Assume L is context free language and let n be the constant for L .

if $n=1$, $L = abc$

if $n=2$, $L = aabbcc$

if $n = n$ $L = a^n b^n c^n$
 $|z| = |a^n b^n c^n| = 3n > n$

Step-(2)

Let $z = a^n b^n c^n$ which is equal to $3n$ greater than n using pumping lemma
 $z = uvwxy$ with $|vwx| \leq m$ and $|vx| \geq 1$

Step-(3)

$$z = uv^k wx^k y$$

if $k = 2$

$$z = uv^2 wx^2 y$$

Let $z = aqbbcc$

$u = a$ $v = a$ $w = bb$ $x = c$ $y = c$

At $k = 2$

$$z = aqaabbcc$$

$$z = a^3 b^2 c^3 \notin L$$

Let

$$z = \underbrace{aa}_{u} \underbrace{ab}_{v} \underbrace{bb}_{w} \underbrace{cc}_{x} \underbrace{c}_{y}$$

At $k = 2$

$$z = aqaababcbcc = a^3 b a b^2 c b c^3 \notin L$$

So

there is contradiction that's why
 $L = \{ a^n b^n c^n \mid n \geq 1 \}$ is not context free grammar.

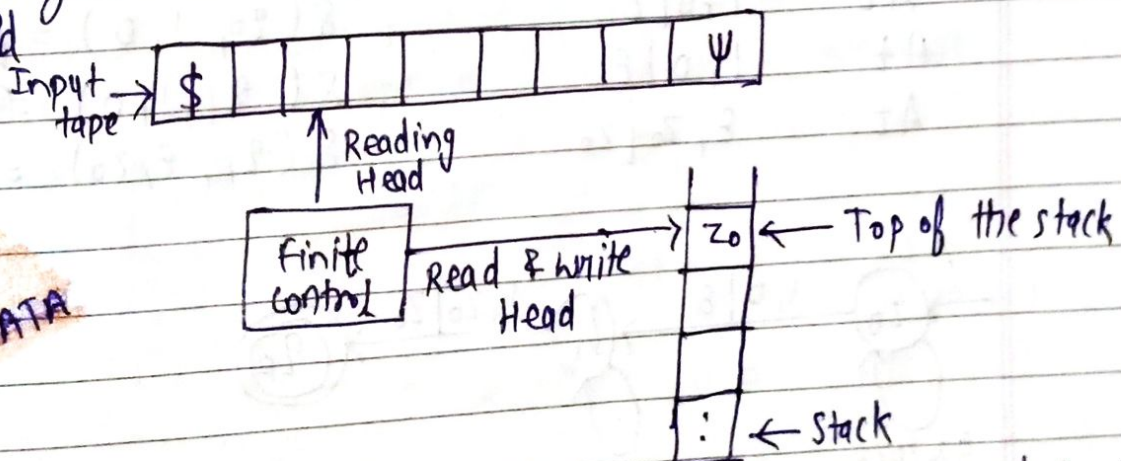
Chapter (3)

* PUSH DOWN AUTOMATA \Rightarrow

PDA are enhance version of finite automata. A finite automata with a stack memory can be called as pushdown automata. Additional stack memory enhances the capability of PDA as compare to finite automata.

A PDA consists of a finite ^{stack} ~~tape~~, a reading head which can read from the stack, a stack memory which is for read write operation and operations are performed in LIFO fashion. Like regular languages which are accepted by finite automata, context free languages are also accepted by automata but not finite automata. They need a little more complex automata called PDA. This automata behaves like a finite automata except the following two:—

- (i) Its next state is determined not only by the input symbol read. but also by the symbol at the top of ~~step~~ stack.
- (ii) The content of the stack can also be changed everytime and input symbol is read



A pushdown automata system is described by 7 tuples

$$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$$

Where Q = Finite non empty set of state
 Σ = Finite non empty ^{set of} input alphabet
 Γ = A finite non empty set of pushdown symbols

$\delta =$ Transition function mapping:
 $Q \times (Z \cup \{\epsilon\}) \times \Gamma$ to finite subset of $Q \times \Gamma$

$q_0 =$ Initial state

$z_0 =$ It is starting stack symbol

$F =$ It is the set of finite states
 i.e. $F \subseteq Q$ & sometimes $F = \emptyset$

* Q. Design PDA for the language that accept string with $L = \{0^n 1^n \mid n \geq 1\}$

Ans -

Let $n=3$

$L = 000111$

\$	0	0	0	1	1	1	ψ
----	---	---	---	---	---	---	--------

At $0, z_0 \mid 0z_0$

$\delta(q_0, 0, z_0) = \delta(q_0, 0z_0)$

At $0, 0 \mid 00$

$\delta(q_0, 0, 0) = \delta(q_0, 00)$

~~At $0, 0 \mid 00$~~

At $1, 0 \mid \epsilon$

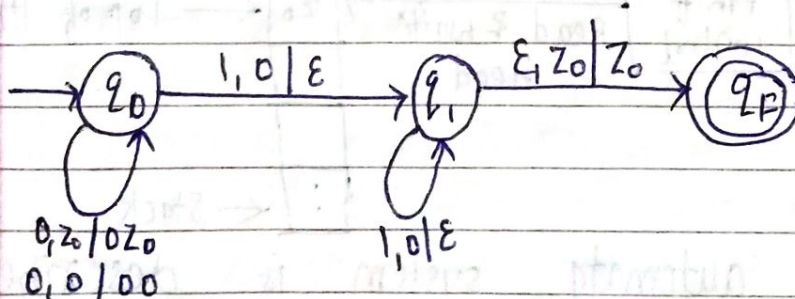
$\delta(q_0, 1, 0) = \delta(q_1, \epsilon)$

At $1, 0 \mid \epsilon$

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

At $\epsilon, z_0 \mid z_0$

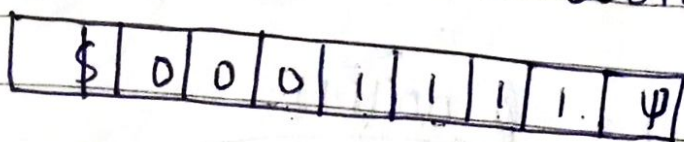
$\delta(q_1, \epsilon, z_0) = \delta(q_f, z_0)$



Q. Design PDA for language that accept string with $L = \{0^n 1^{n+1} \mid n \geq 1\}$

$n=3$

000111

Let $n=3$ $L = 0001111$ At $0, z_0 | 0z_0$ At $0, 0 | 00$ At $1, 0 | 0$ At $1, 0 | \epsilon$ At $\epsilon, z_0 | z_0$

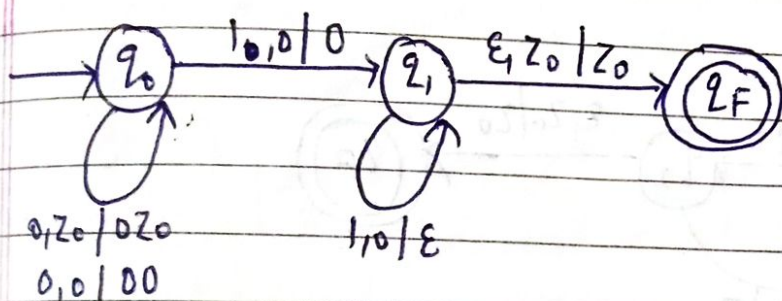
$$\delta(q_0, 0, z_0) = (q_0, 0z_0)$$

$$\delta(q_0, 0, 0) = (q_0, 00)$$

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

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

$$\delta(q_1, \epsilon, z_0) = (q_F, z_0)$$

Q. Design PDA for $L = \{0^n 1^m \mid n < m, n \neq m \geq 1\}$ Ay- Let $n=2, m=4$ $L = 001111$  001111 At $0, z_0 | 0z_0$ At $0, 0 | 00$ At $1, 0 | \epsilon$ At $1, 0 | \epsilon$ At $1, z_0 | z_0$ At $1, z_0 | z_0$

$$\delta(q_0, 0, z_0) = (q_0, 0z_0)$$

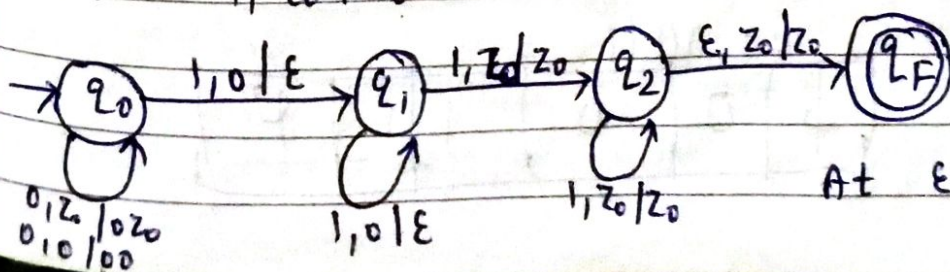
$$\delta(q_0, 0, 0) = (q_0, 00)$$

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

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

$$\delta(q_1, 1, z_0) = (q_2, z_0)$$

$$\delta(q_2, 1, z_0) = (q_F, z_0)$$



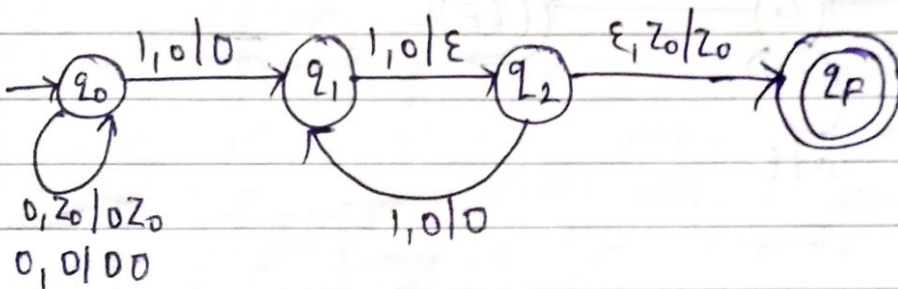
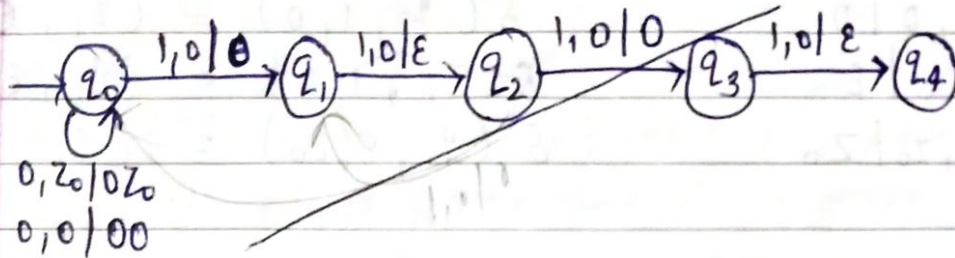
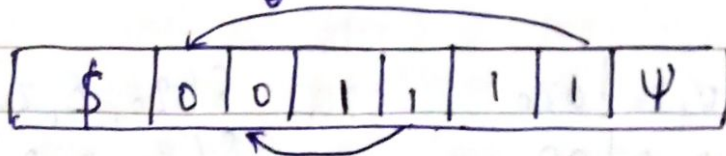
$$\text{At } \epsilon, z_0 | z_0 \quad \delta(q_2, \epsilon, z_0) = (q_F, z_0)$$

Q. $L = \{ 0^n 1^{2n} \mid n \geq 1 \}$

let $n=2$

$L = 001111$

let $n=3$



At $0, z_0 \mid 0z_0$

At $0, 0 \mid 00$

At $1, 0 \mid 0$

At $1, 0 \mid \epsilon$

At $1, 0 \mid 0$

At $\epsilon, z_0 \mid z_0$

$\delta(q_0, 0, z_0) = (q_0, 0z_0)$

$\delta(q_0, 0, 0) = (q_0, 00)$

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

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

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

$\delta(q_2, \epsilon, z_0) = (q_F, z_0)$

Q.

$L = \{ 0^n 1^n \mid n \geq 0 \}$

Ans -

$n=0$

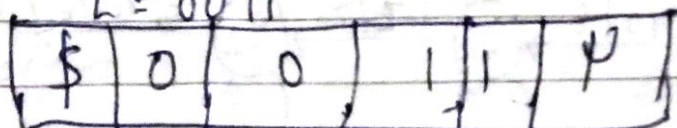
$L = \text{Nothing}$

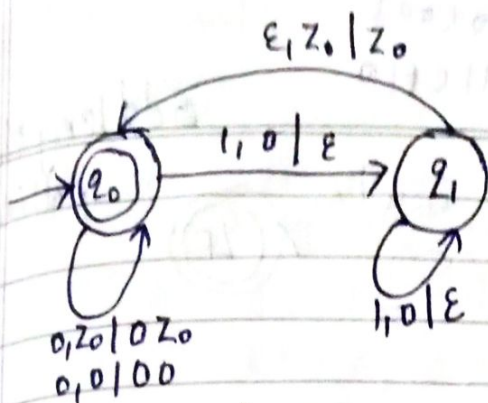
$n=1$

$L = 01$

$n=2$

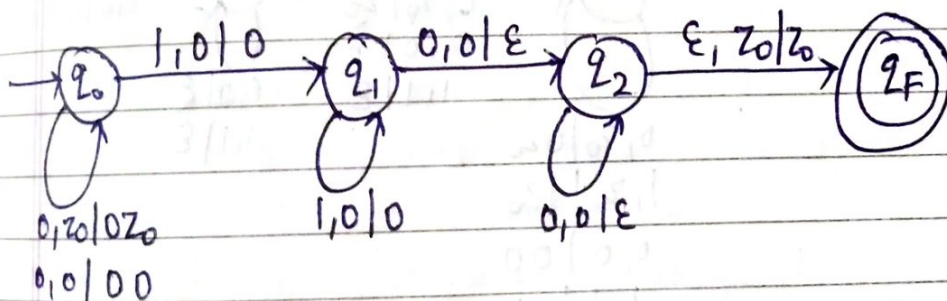
$L = 0011$




 $0, z_0 | 0z_0$
 $0, 0 | 00$
 $1, 0 | \epsilon$
 $1, 0 | \epsilon$
 $\epsilon, z_0 | z_0$
 $\delta(q_0, 0, z_0) = (q_0, 0z_0)$
 $\delta(q_0, 0, 0) = (q_0, 00)$
 $\delta(q_0, 1, 0) = (q_1, \epsilon)$
 $\delta(q_1, 1, 0) = (q_1, \epsilon)$
 $\delta(q_1, \epsilon, z_0) = (q_0, z_0)$

$$L = \{ 0^n 1^m 0^n \mid n < m, n, m \geq 1 \}$$

At $n=2, m=3$


 $0, z_0 | 0z_0$
 $0, 0 | 00$
 $1, 0 | 0$
 $1, 0 | 0$
 $0, 0 | \epsilon$
 $0, 0 | \epsilon$
 $\epsilon, z_0 | z_0$
 $\delta(q_0, 0, z_0) = (q_0, 0z_0)$
 $\delta(q_0, 0, 0) = (q_0, 00)$
 $\delta(q_0, 1, 0) = (q_1, 0)$
 $\delta(q_1, 1, 0) = (q_1, 0)$
 $\delta(q_1, 0, 0) = (q_2, \epsilon)$
 $\delta(q_2, 0, 0) = (q_2, \epsilon)$
 $\delta(q_2, \epsilon, z_0) = (q_F, z_0)$

$$L = \{ w c w^R \mid w \in (0+1)^* \} \quad \text{Odd length palindrome}$$

0110110 or

1001001

1000 0 0 0 1
0 1 1 1

1000 0 0 1
0 1 1 1 0

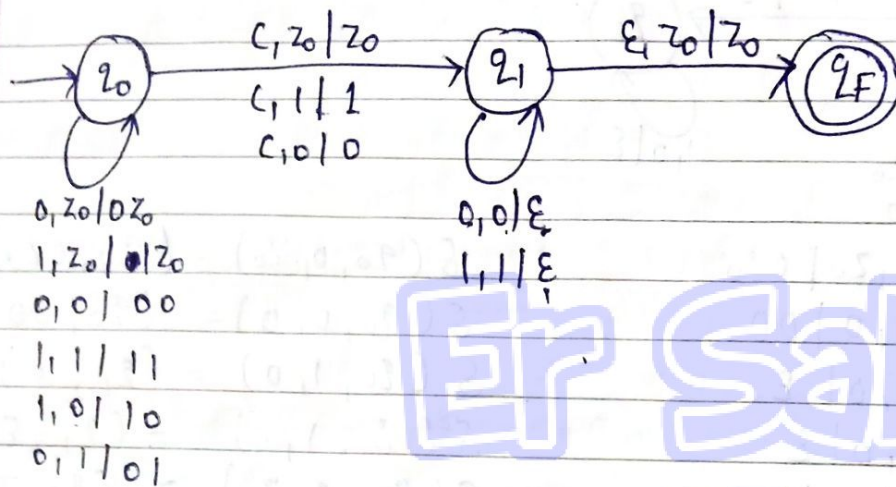
DATE: / /

PAGE NO:

odd length

palindrome

★



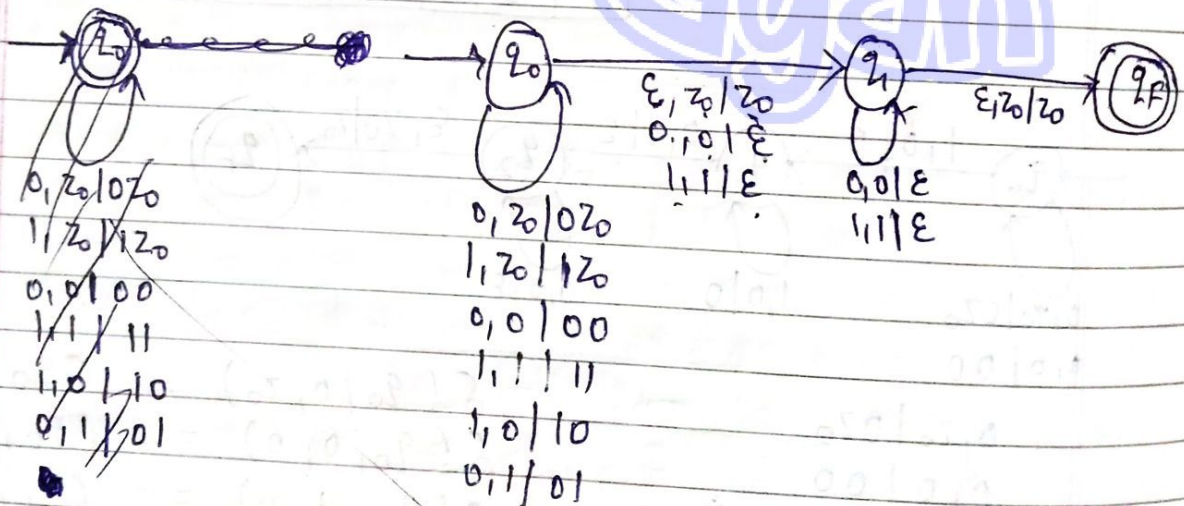
★

$$L = \{ WWR \mid W \in \{0+1\}^* \}$$

even length palindrome



NPDA

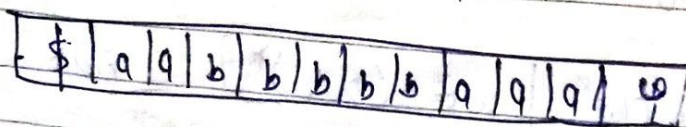


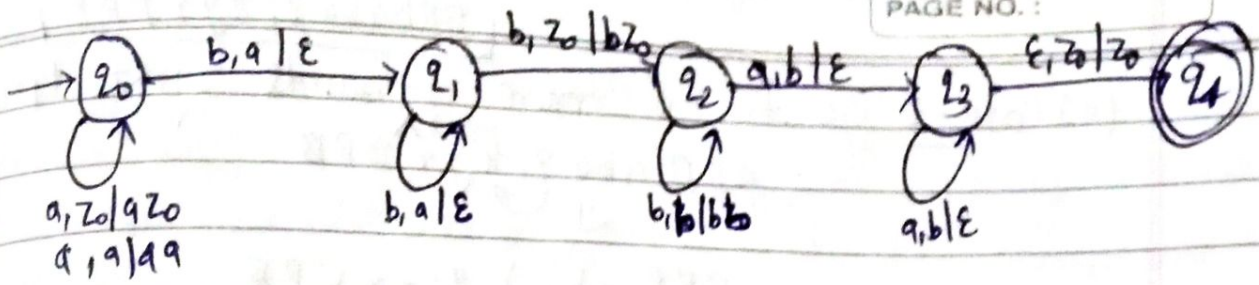
Q.

$$L = \{ a^n b^{n+m} a^m \mid n, m \geq 1 \}$$

$$n=2, m=3$$

$$L = aabbabbaa$$

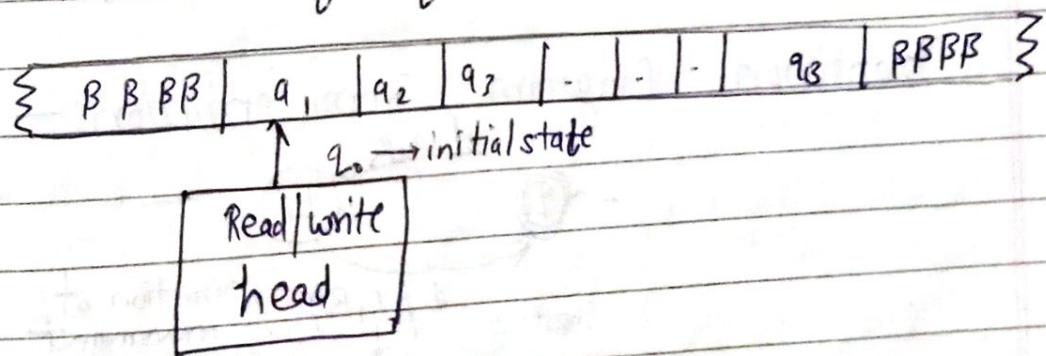




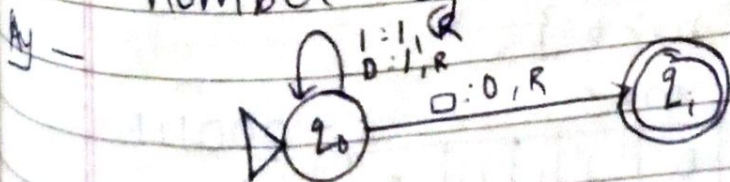
* Turing Machine \Rightarrow It is an advance version of PDA. Turing machine has a unlimited & unrestricted memory known as tape & a tape head to access the memory. Turing machine may also be said to be theoretical model of today's computer.

Formal Definition of Turing machine:—

$M = (Q, \Sigma, \Gamma, \delta, q_0, \beta, F)$ is 7-tuple
 $\Gamma \rightarrow$ set of tape symbol $\Gamma = \Sigma \cup \{\beta\}$
 q_0 is state in which R/W head initially exists.
 β is blank symbol.
 F is set of final states $F \subseteq Q$



Q. Design a Turing machine that multiplies a binary number by 2.



There are 3 schemes for representation of Turing machine

(i) Instantaneous description: —

DATE: / /
PAGE NO.:

(a) Case-I If we move towards right direction
 $\beta\beta\beta a b a q_i x y x \beta\beta\beta$
 $\downarrow \quad \rightarrow$

$\beta\beta\beta a b a X q_i y x \beta\beta\beta$

(b) Case-II If we move towards left direction.
 $\beta\beta\beta a b a q_i x y x \beta\beta\beta$
 $\downarrow \quad \leftarrow$ (This is current symbol)

$\beta\beta\beta a b a X q_i y x \beta\beta\beta$

(ii) Transition table Representation: —

$$\delta(q, x) = (\pi, \phi, \psi)$$

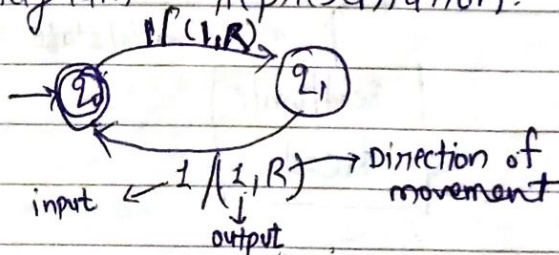
\downarrow R/w head \downarrow next state \downarrow output symbol \rightarrow Direction of movement

Then in q row & ϕ column we write $\phi \psi \pi$ ✓

Eg - If $L = \{1^n \mid n \text{ is even}\}$ TT = ?

Current state	Input symbol		
	1	β	
$\rightarrow q_0$	1Rq ₁	—	$\delta(q_0, 1) = (q_1, 1, R)$
q_1	1Rq ₀	—	$\delta(q_1, 1) = (q_0, 1, R)$

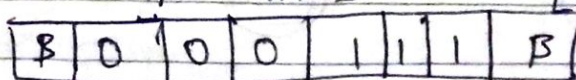
(iii) Transition Diagram Representation: —



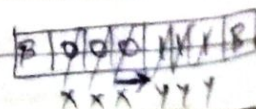
Q. Construct a Turing machine over input symbol $\Sigma = \{0, 1\}$ to accept lang.

$L = \{0^m 1^m \mid m > 0\}$

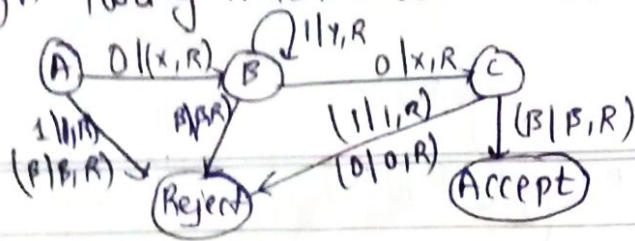
Ans - Let if $m = 3$ $L = 000111$



read 0 \rightarrow X write
 read 1 \rightarrow Y write



Q. Design Turing machine for $L = 01^*0$



DATE: / /

PAGE NO.:

Q. ~~A~~ $L = \{ w \mid w \text{ is palindrome} \}$ or $\{ w w^R \}$
 A1 - let

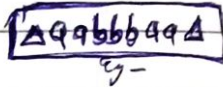
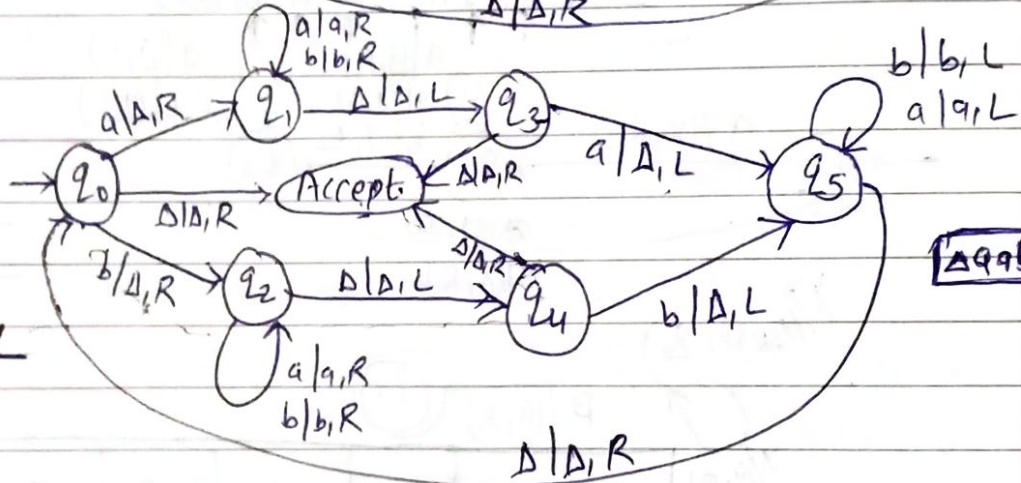
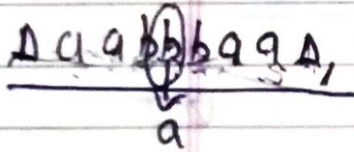


where Δ = Blank

For Even



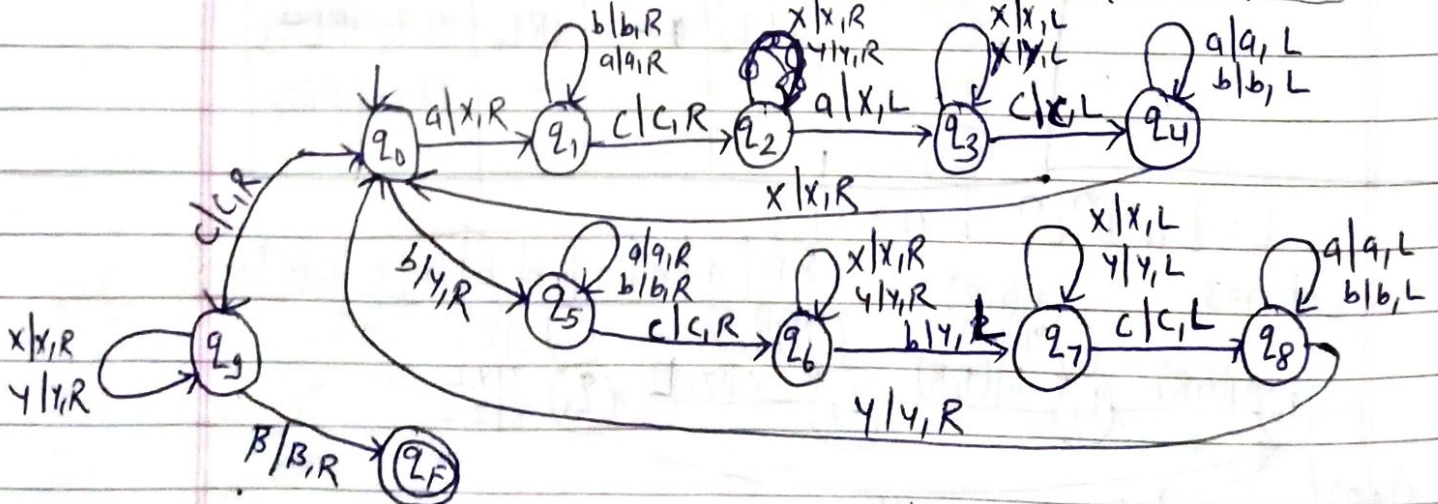
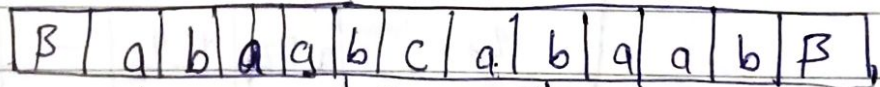
For Odd



Q.

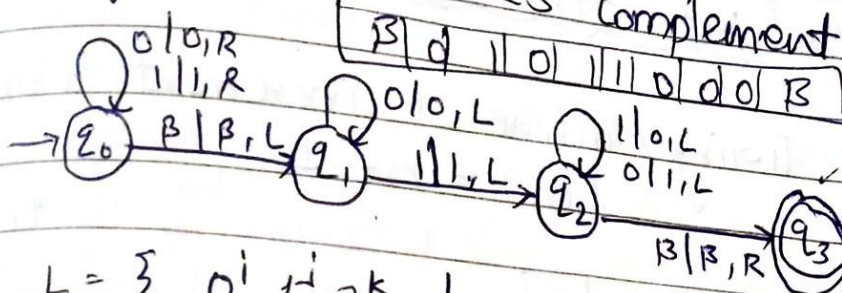
$L = \{ w c w \mid w \in (a, b)^* \}$

A1 - Let



Q. Turing machine for 2's complement

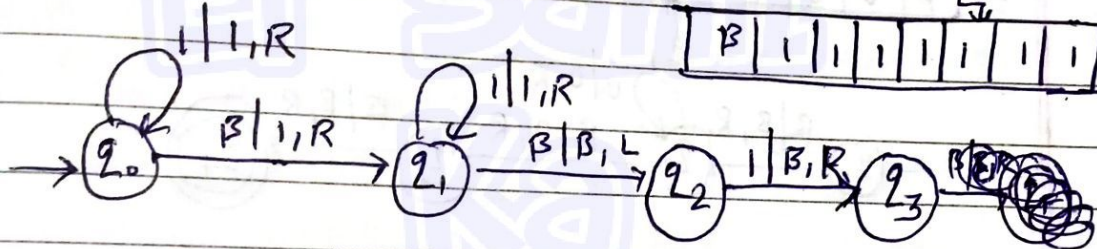
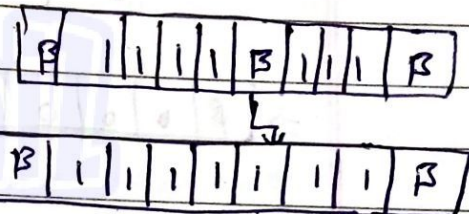
$w = 01011000$



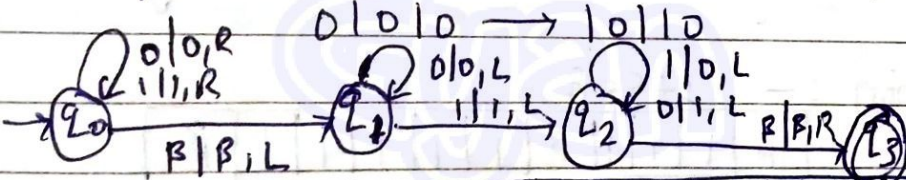
$10101000 \rightarrow 2's \text{ complement}$

Q. (9) $L = \{ 0^i 1^j 2^k \mid j = i + k, i, j, k \geq 0 \}$ T.M. = ?
 for Unary addition design Turing machine.
 Ans - Let

$$\begin{array}{r} 1111 \quad (4) \\ + 111 \quad (3) \\ \hline 111111 \quad (7) \end{array}$$

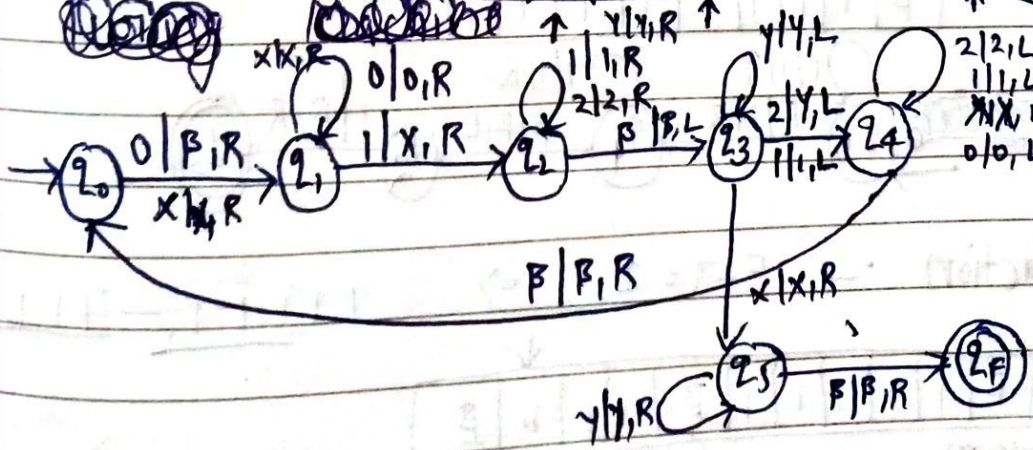


Q. To design a T.M. for 2's complement $w = 01010$



(a) $B \ 0 \ 0 \ 1 \ 1 \ 1 \ 1 \ 2 \ 2 \ B$

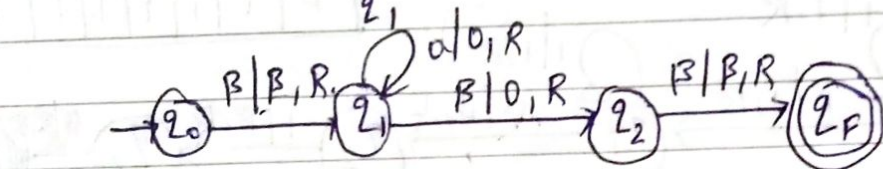
$0 \rightarrow B$
 $1 \rightarrow X$
 $2 \rightarrow Y$



$i = 2$
 $j = 4$
 $k = 2$

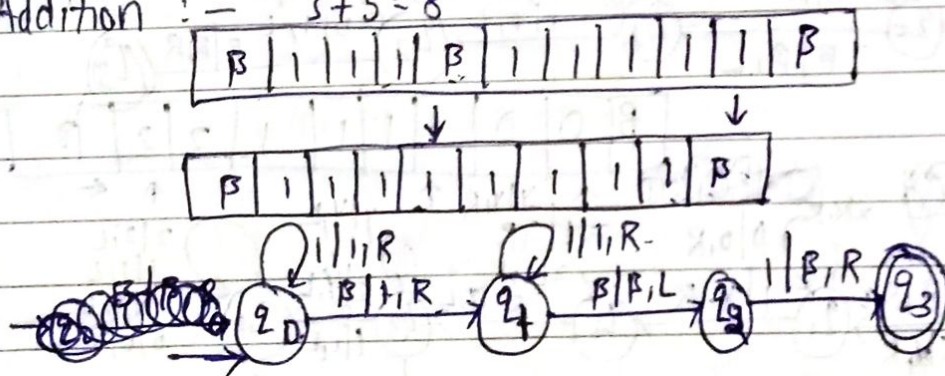
PAGE NO. :

Let $n=3$

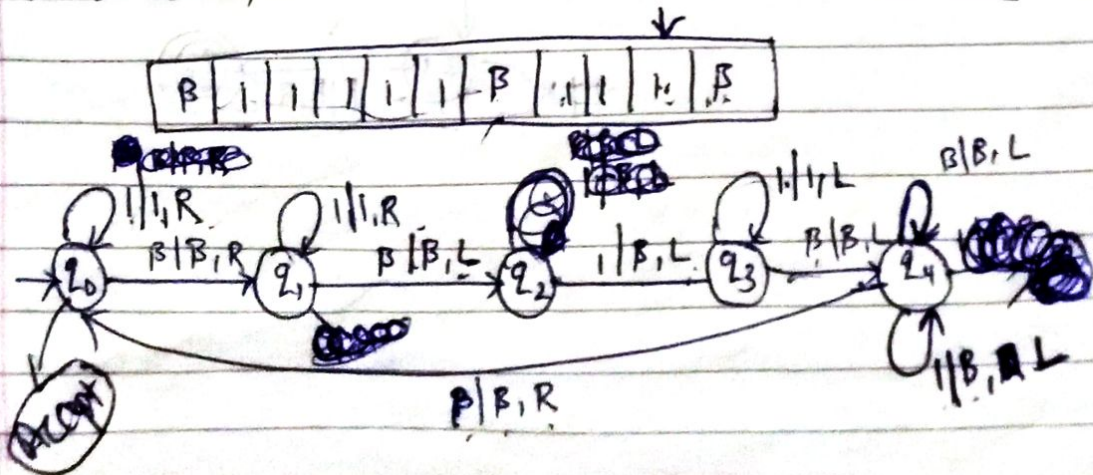


Addition & Subtraction in Turing machine:—

Addition :- $3+5=8$



Subtraction :- $5 - 3 = 2 \Rightarrow \boxed{11111} - \boxed{111} = \underline{\underline{11}}$



Q.1 Explain Turing Machine & its different types in detail?

Ans- Turing machine is used for recursive enumerable language. Turing machine have an unlimited & unrestricted memory known as tape and tape head to access the memory. Turing machines are capable of doing everything that a computer can do. The tape head has the capability to move left and right and it may also remains stationary after reading a symbol. Initially the tape contains the input string and blank symbol.

Formal definition of Turing Machine:—

A Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, \beta, F)$ is a 7-tuple structure where

Q is finite non empty set of states.

Σ is the character set of language.

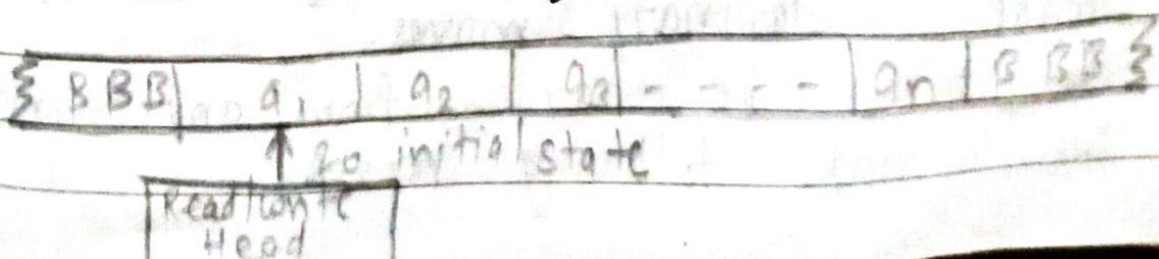
Γ is set of tape symbol $\Gamma = \Sigma \cup \{\beta\}$

δ is set of transition function

q_0 is state in which R/W head initially exists

β is blank symbol.

F is set of final states $F \subseteq Q$

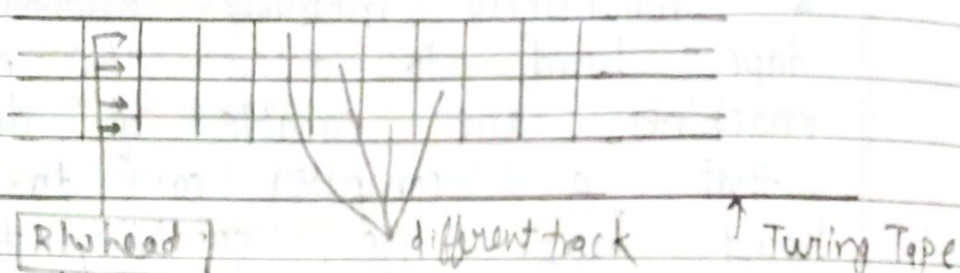


Different types of Turing Machine:—

(i) Multitrack Turing machine \Rightarrow

A multitrack Turing tape consist of multiple track on a single tape.

Fig: Multitrack TM



In multitrack Turing Machine, all the track reside on the same Turing tape and move in same direction simultaneously i.e. The direction of motion of Turing tape. The R/w head is capable of reading and writing from all the cells simultaneously which lie in same column.

(ii) Multitape Turing Machine \Rightarrow

Multitape TM is another notification of TM. In the multitrack TM all the track reside on same direction. In multitape TM, each track reside on a different tape and can move in desired direction independent of others. If movement is not required then we make stationary moves. So it can be say, from any instant a head can make following moves.

(a) In right side (b) In left side (c) In stationary mode

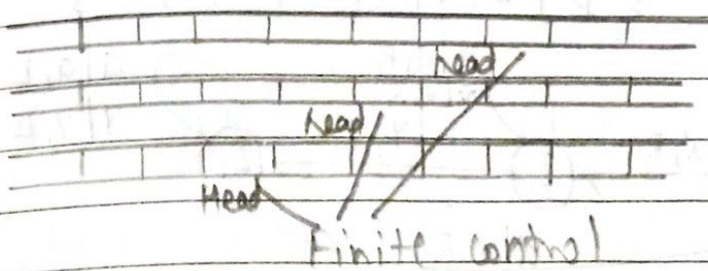


Fig - Multitape TM

Finite control is a center point where all head located. The transition function of Multitape TM is similar to multitrack TM. But in multitape TM the input is read from tape instead of track.

(ii) Multidimensional TM : -

In this TM the input tape can be viewed as extending infinitely in more than one direction (dimension). The move for multidimensional are formally defined as follows

$$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times (L, R, U, D, \text{stop})$$

Here L = Left R = Right U = Upper D = Down
Stop = stationary move.

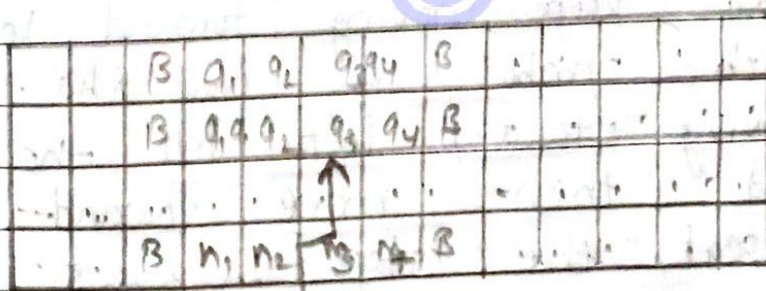


Fig - Multidimensional TM

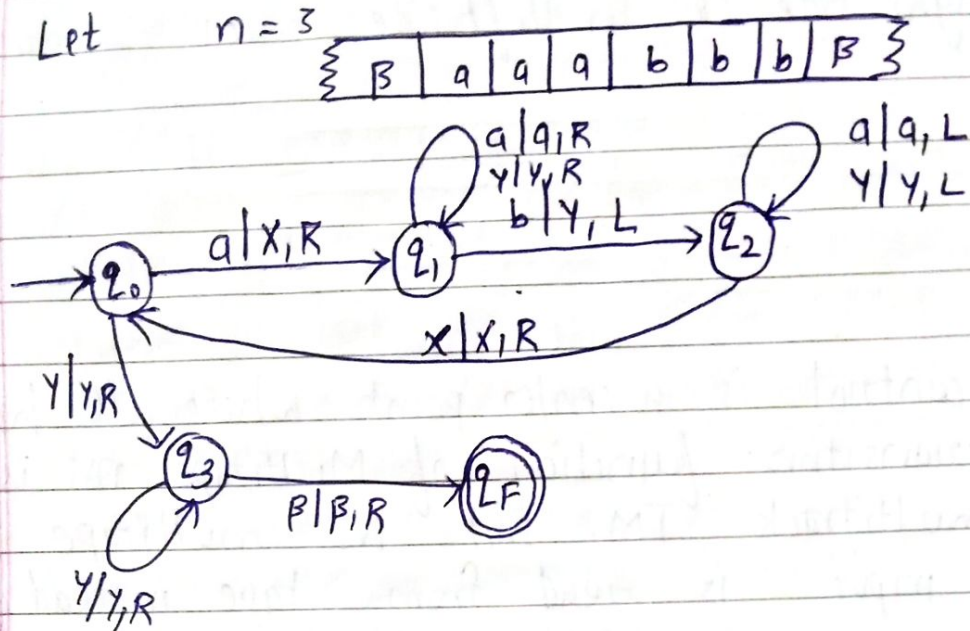
Q.2 Construct a TM which recognizes the language $L = \{x^n y^n\}$ where $n \geq 1$.

$$L = \{x^n y^n \mid n \geq 1\}$$

let $x = a$

$y = b$

$$L = \{a^n b^n \mid n \geq 1\}$$

Let $n = 3$ 

Initially the R/w head points to first symbol of input string in state q_0 .

1. State q_0 read the first symbol a and convert this a into x .
2. The R/w head keeps moving toward right till the input string is crossed and second symbol b is reached.
3. From the b , b is converted into y and R/w head keeps moving toward left one cell and reach symbol a . Now R/w head keeps moving toward left till the x is reached and then move toward right.
4. Repeat the step 1, 2 & 3 till the all a & b is converted into x & y respectively.
5. Now R/w head on x and no more a can be converted into x so
6. R/w head keeps toward right till the input string is crossed and first blank

Symbol β is reached.

Current state	Input symbol				
	a	b	x	y	β
$\rightarrow q_0$	XRq_1	-	-	YRq_3	-
q_1	qRq_1	YLq_2	-	YRq_1	-
q_2	qLq_2	-	XRq_0	YLq_2	βLq_3
q_3	-	-	-	YRq_3	βRq_F
$\textcircled{q_F}$	-	-	-	-	-

For instantaneous representation

string:- $qaabb\beta$

$\beta \mid a \mid a \mid a \mid b \mid b \mid b \mid \beta$

$\beta q_0 a a a b b b \beta$

$\beta \mid x \mid a \mid a \mid b \mid b \mid b \mid \beta$

$\beta x q_1 a a b b b \beta$

$\beta \mid x \mid a \mid a \mid b \mid b \mid b \mid \beta$

$\beta x a q_1 a b b b b \beta$

$\beta \mid x \mid a \mid a \mid b \mid b \mid b \mid \beta$

$\beta x a a q_2 b b b b \beta$

$\beta \mid x \mid a \mid a \mid y \mid b \mid b \mid \beta$

$\beta x a q_2 a y b b b \beta$

$\beta \mid x \mid a \mid a \mid y \mid b \mid b \mid \beta$

$\beta x q_2 a a y b b b \beta$

$\beta \mid x \mid a \mid a \mid y \mid b \mid b \mid \beta$

$\beta q_2 x a a y b b b \beta$

$\beta \mid x \mid a \mid a \mid y \mid b \mid b \mid \beta$

$\beta x q_0 a a y b b b \beta$

$\beta \mid x \mid x \mid a \mid y \mid b \mid b \mid \beta$

$\beta x x q_1 a y b b b \beta$

$\beta \mid x \mid x \mid a \mid y \mid b \mid b \mid \beta$

$\beta x x a q_2 y b b b \beta$

β	x	x	a	y	b	b	β
---	---	---	---	---	---	---	---

βxxayz,bbβ

β	x	x	a	y	y	b	β
---	---	---	---	---	---	---	---

βxxazzybβ

β	x	x	a	y	y	b	β
---	---	---	---	---	---	---	---

βxxazzybβ

β	x	x	a	y	y	b	β
---	---	---	---	---	---	---	---

βxzaxzybβ

β	x	x	a	y	y	b	β
---	---	---	---	---	---	---	---

βxxazzybβ

β	x	x	x	y	y	b	β
---	---	---	---	---	---	---	---

βxxxzyybbβ

β	x	x	x	y	y	b	β
---	---	---	---	---	---	---	---

βxxxzyybbβ

β	x	x	x	y	y	b	β
---	---	---	---	---	---	---	---

βxxxzyybbβ

β	x	x	x	y	y	y	β
---	---	---	---	---	---	---	---

βxxxzyyyβ

β	x	x	x	y	y	y	β
---	---	---	---	---	---	---	---

βxxxzyyyyβ

β	x	x	x	y	y	y	β
---	---	---	---	---	---	---	---

βxxzxyyyyβ

β	x	x	x	y	y	y	β
---	---	---	---	---	---	---	---

βxxxzyyyyβ

β	x	x	x	y	y	y	β
---	---	---	---	---	---	---	---

βxxxzyyyyβ

β	x	x	x	y	y	y	β	β
---	---	---	---	---	---	---	---	---

βxxxzyyyzββ

β	x	x	x	y	y	y	β	β
---	---	---	---	---	---	---	---	---

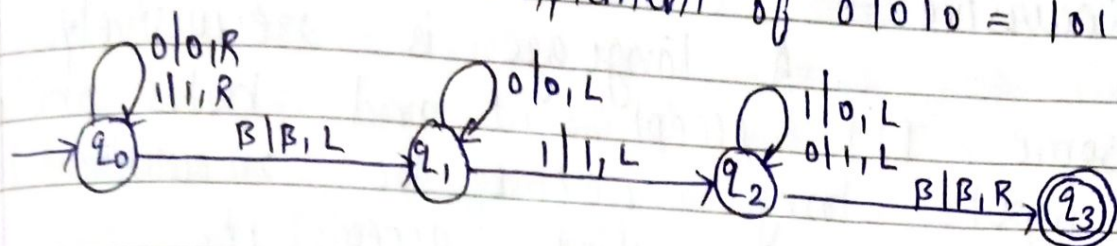
βxxxzyyyzyββ

β	x	x	x	y	y	y	β	β
---	---	---	---	---	---	---	---	---

βxxxzyyyzyfβ

Er Sahil
Ka
Gyan

Q.3 Construct the TM to find the 2's complement of a binary number. $w = 01010$
 2's complement of $01010 = 10110$



current state	Input symbol		
	0	1	B
$\rightarrow q_0$	0R q_0	1R q_0	B L q_1
q_1	0 L q_1	1 L q_2	-
q_2	1 L q_2	0 L q_2	B R q_3
q_3	-	-	-

Q.4 Explain the following

(i) Undecidability

In real world, there are several problems & most of the problems are solved by humans and machines. Those problem have a proper algorithm to be followed to solve them. There are certain problems that are algorithmically unsolvable.

Even though computer appear to be more powerful. There are some common problem that can't be solved by computer too.

One such unsolvable problem is "designing a TM to solve membership of TM". That is, it's not possible to design a TM M_2 that will accept, when given TM M_1 accepts the given string w & reject otherwise.

(ii) Recursive and Recursive enumerable language

Recursive:—

A language is recursively if some TM accept it and halts on any input string. Let L be recursive language & M is TM that accepts it.

For string w .

If $w \in L$ then M halts in final state

If $w \notin L$ then M halts in nonfinal state.

A recursive language never causes for a loop on TM.

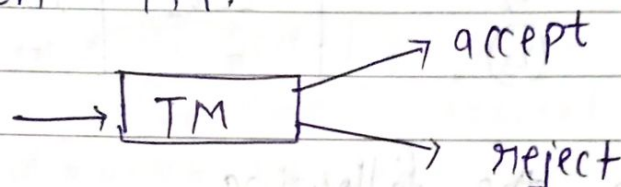


fig:— TM for recursive language

Recursive enumerable language:—

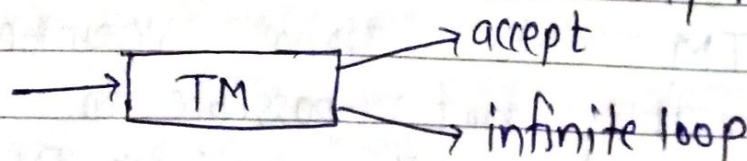
A language is recursively enumerable if some TM accept it. Let L be recursively enumerable language & M is TM that accept it.

For string w

If $w \in L$ then M halts in final state

If $w \notin L$ then M loops forever.

A RL is REL but REL may not be RL.



Rice Theorem : —

(iii) A problem that can be answered yes or no is said to be a decision problem.

Rice theorem state that Every non-trivial [answer is not known] problem on recursively enumerable language is undecidable.

Eg - If a language recursive enumerable's complement will be recursive enumerable or not is undecidable.

Halting Problem: —

(iv) Halting means that the problem on certain input will accept it & halt or reject it and halt and it would never go into infinite loop. Basically halting means terminology.

So, we can have an algorithm out will that the given program will halt ~~in~~.

→ It can be shown that halting problem is not decidable, hence unsolvable.

→ Halting problem lets us reason about the relative difficulty of algorithms. It lets us know that, there are some algorithms that don't exist, that sometimes, all we can do is guess at a problem and never know if we have solved it.

Q.5 Explain the Chomsky hierarchy of Grammar?

Ans - Noam Chomsky, the founder of formal languages provided a classification for grammars, based on their type of production rules as given below:

- Type 0 - Unrestricted grammar
 Type 1 - Context sensitive grammar
 Type 2 - Context free grammar
 Type 3 - Regular grammar

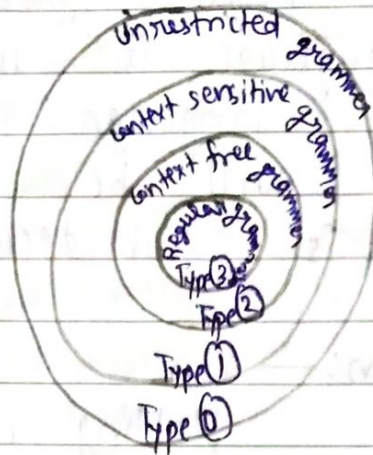


Fig: Chomsky hierarchy of Grammars

①. Type 3 Grammar [Regular Grammar]:—

A grammar $G = \{V_n, \Sigma, P, s\}$ is called type-3 or regular grammar if each production $\alpha \rightarrow \beta$ satisfies the following conditions.

$$\alpha \rightarrow \beta$$

$$|\alpha| = 1$$

$$a \in V_n$$

$$|\beta| \leq 2$$

$$\Rightarrow$$

$$|\beta| = 0$$

$$\text{then } \beta = \Lambda$$

$$|\beta| = 1$$

$$\text{then } \beta \in \Sigma$$

$$|\beta| = 2$$

$$\text{then}$$

$$\beta \in (V_n \cup \Sigma)$$

It is clear that LHS of production contains only one single Non-terminal & RHS contains maximum of 2 symbols that can be.

Null, one terminal or one terminal & one nonterminal
 for example consider the following production rules

$$P_1: S \rightarrow \Lambda, \quad P_2: S \rightarrow aA, \quad P_3: S \rightarrow Ab$$

$$P_4: S \rightarrow aa, \quad P_5: S \rightarrow bb, \quad P_6: S \rightarrow AA$$

$P_7: S \rightarrow a$

Here P_1, P_2, P_3, P_7 are in regular grammar while P_4, P_5, P_6 are not

(2) Type 2 (Context free grammar) \Rightarrow A grammar $G = (V_n, P, \Sigma, S)$ is called context free if each production $\alpha \rightarrow \beta$ satisfies the following conditions

In production

$\alpha \rightarrow \beta$

$$|\alpha| = 1$$

$$|\beta| = \text{infinite}$$

$$|\beta| \geq 0$$

$$\alpha \in V_n$$

$$\beta \in (V_n \cup \Sigma)^*$$

It is clear that LHS of production contains only one single non-terminal & RHS of production contains any combination of terminal and non terminal including null or can say RHS of ~~any~~ production have no any restriction.

Eg - $P_1: A \rightarrow abB$, $P_2: A \rightarrow \Lambda$
 $P_3: SA \rightarrow B$, $P_4: A \rightarrow Bx$
 $P_5: A \rightarrow cde$, $P_6: B \rightarrow A \cup$

Here production P_1, P_2, P_5, P_6 are in CFG while P_3 & P_4 are not.

(3) Type - 1 (Context sensitive Grammar): —

A grammar $G = (V_n, P, S, \Sigma)$ is called context sensitive if each production $\alpha \rightarrow \beta$ satisfies following condition:

$$\alpha \rightarrow \beta$$

$$|\alpha| \leq |\beta|$$

$$\alpha, \beta \in (V_n \cup \Sigma)^*$$

It is clear that both side of production can contains any combination of terminal & nonterminal.

The only restriction is length of left HS is less than RHS or equal to length of Right side.

eg -

$P_1: S \rightarrow E$

$P_2: S \rightarrow \Lambda$

$P_3: E \rightarrow aADC$

$P_4: D \rightarrow EFg$

$P_5: Dx \rightarrow C$

$P_6: EFS \rightarrow t$

$P_7: TF \rightarrow AB$

Here production P_1, P_2, P_3, P_4, P_7 are in CSN while P_5 & P_6 are not in CSN.

④ Type - 0 (Unrestricted Grammar) :- A grammar $G = (V_N, \Sigma, P, S)$

is called type-0 if each production $\alpha \rightarrow \beta$ have at least one non terminal symbol in α or can say any valid production is always in type-0 grammar.

The language corresponding to unrestricted grammar is recursively enumerable language & machine which accept recursively enumerable language Turing machine.

————— x —————

Q.1 Define P, NP, NP-hard & NP-completeness. Explain P problem \Rightarrow

This class of problem can be solved in polynomial time. "P" stands for "polynomial". The complexity class P is the set of all decision problems that can be solved in worst case polynomial time. In other words, problem can be solved in time $O(n^k)$ for some constant k , where n is size of input to the problem. But the definition of P doesn't comment on running time for output of "no".

NP problem \Rightarrow This stands for "Non deterministic polynomial time" where Non deterministic is just a fancy way of talking about guessing a solution. Till now we have focused on problem those were deterministic. Discussing non-deterministic algorithms requires 3 new functions:

- (a) Choice(s): arbitrarily choose one of element of set S .
- (b) Failure(): signals an unsuccessful completion
- (c) Success(): signals a successful completion

NP-hard \Rightarrow NP hard therefore means "at least as hard as any NP problem", although it might, in fact, be harder. If an NP-hard problem can be solved in polynomial time, then all NP-complete problems can be solved in polynomial time.

NP-completeness:— NP completeness is a solution to the practical problem of applying complexity theory to individual problem.

NP-complete problems are defined in a precise sense as the hardest problems in P.

All NP-complete problems are NP-hard. But not all NP-hard problems are NP-complete.

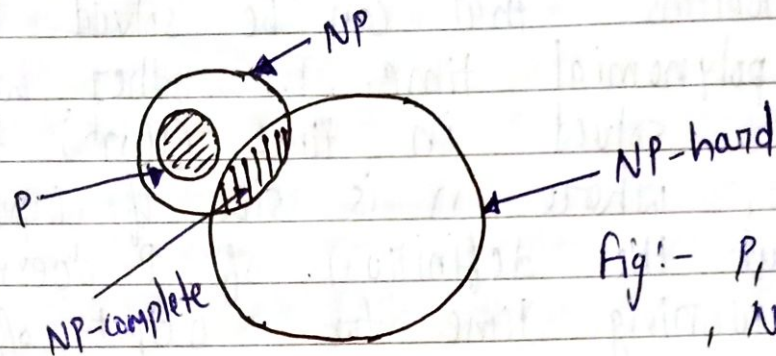


Fig:- P, NP, NP-complete, NP-hard problem reduction.

~~Q~~ (2) Explain Vertex cover ~~& set cover~~ problems with example.

Ans— Vertex Cover Problem:—

Satisfiability is a circuit simulation problem which has been proved to be NP-complete. A vertex cover of an undirected $G = (V, E)$ is subset $V' \subseteq V$.

A vertex cover problem is to find a vertex cover of minimum size in graph.

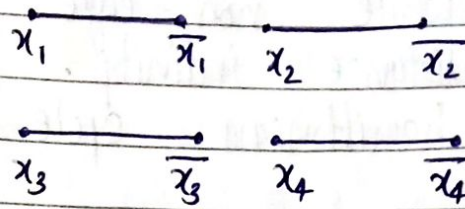
To prove vertex cover problem is NP-complete,

(i) Certificate is a vertex cover $V' \subseteq V$ itself. The no. of vertices in V' should be equal to integer k . Then if for each edge which can be checked in polynomial time, then vertex cover is in NP.

(ii) $3\text{-SAT} \leq_p \text{Vertex cover}$. If we can reduce 3-SAT problem in polynomial time then we can say that

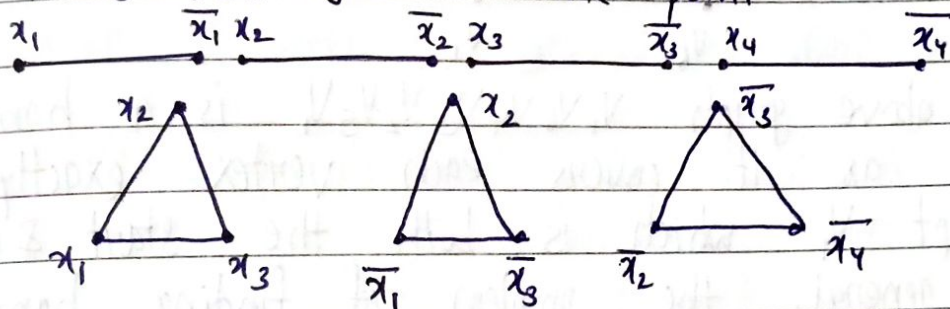
vertex cover is NP-complete.

Eg- Graph G

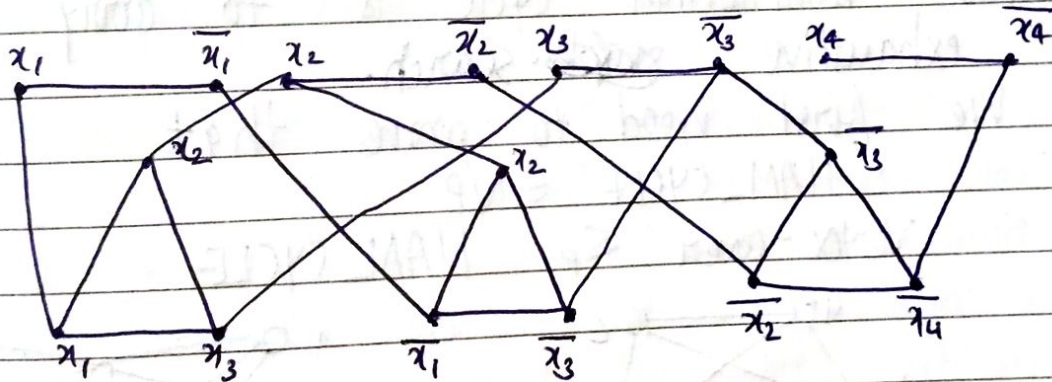


$$S = (x_1 + x_2 + x_3) \cdot (\bar{x}_1 + x_2 + \bar{x}_3) \cdot (\bar{x}_2 + \bar{x}_3 + \bar{x}_4)$$

It contains 4 variables. so, graph has 4 edges & 8 vertices till the point.



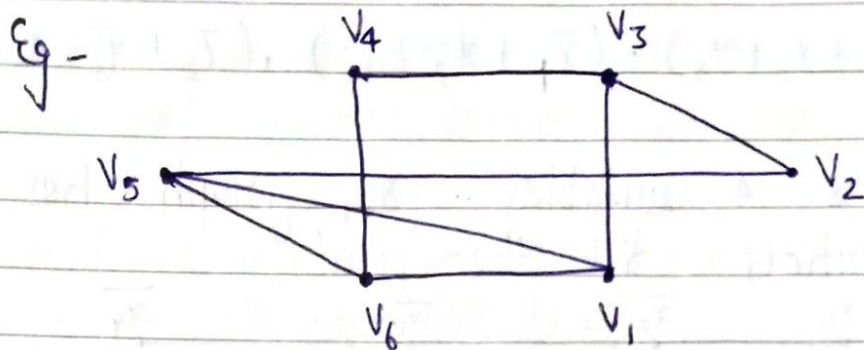
vertex cover will include at least 2 vertices per triangle.



So vertex cover for this graph would be $n+2C$ where n is no. variables in S & C is no. of clauses.

Q(3) Explain Hamiltonian path problem with example.
 A Hamiltonian cycle is cycle through a graph that visits each node exactly once and also returns to starting vertex. This problem is special case of Travelling Salesman problem where each pair

of vertices with an edge b/w has distance 1, while non-edge vertex pairs are separated by distance infinity. A graph that contains a hamiltonian cycle is called Hamiltonian graph.



In above graph $V_1, V_6, V_4, V_3, V_2, V_5, V_1$ is a hamiltonian cycle as it covers each vertex exactly once except V_1 which is both the start & end vertex.

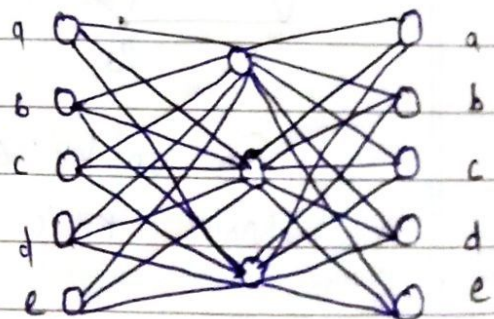
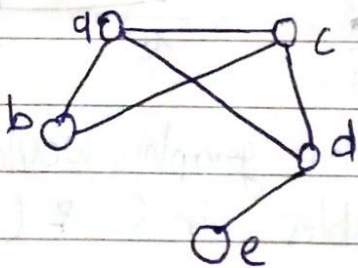
In general, the problem of finding hamiltonian cycle in a graph is NP-completeness & the only way to find whether a given graph has hamiltonian cycle is to carry out exhaustive ~~search~~ search.

We first need to prove that

(i)
(ii)

HAM-CYCLE \in NP

Vertex-cover \leq_p HAM-CYCLE



Graph with Hamiltonian Cycle.

Q.4 Explain Travelling Salesman Problem (TSP) with an example.

A- The TSP is similar to Hamiltonian cycle problem. In TSP, a salesman has to visit all the cities exactly once and finally has to reach the city he started from. The salesman has to make the tour such that total cost of his tour is minimum. Cost of tour is sum of costs of visiting from one city to another. Modeling the problem mathematically, given a complete graph G with n vertices, problem is to visit all vertices exactly once & coming back to start vertex.

$$\text{TSP} = \{ (G, c, k) : G = (V, E) \text{ is complete graph, } c \text{ is cost function from } V \times V \rightarrow \mathbb{Z}, k \text{ is element of } \mathbb{Z} \}$$

Thus salesman has to make a tour with cost at most k . To prove TSP is NP-complete.

We first need to ~~find~~ prove that $\text{TSP} \in \text{NP}$

(i) To show $\text{TSP} \in \text{NP}$: Certificate is the tour of vertices in given graph. This tour should be such that each vertex is visited exactly once and the sum of edge costs comes out to be most integer k . And as this process can be done in polynomial time so, $\text{TSP} \in \text{NP}$.

(ii) To prove $\text{HAM_CYCLE} \leq_p \text{TSP}$: We now need to prove that the HAM_CYCLE problem can be reduced to TSP in polynomial time.

$$c(i, j) = \begin{cases} 0 & \text{if } (i, j) \in E \\ 1 & \text{if } (i, j) \in E' \end{cases}$$

If and only if graph G has hamiltonian cycle ~~is~~, the tour would be of minimum cost with cost function n .

We have proved G has HC. Concluding HAM CYCLE is reducible to TSP, TSP \in NP and thus Travelling salesman problem is NP-complete.

—————X—————